基于层次黑板模型的多 Agent 系统研究

Research of Multi – Agent System Based on Hierarchical Blackboard Model

蒋丽娟　刘卫国　（中南大学信息科学与工程学院 湖南长沙 410075）

摘　要： 本文提出了一种层次黑板模型的多 Agent 系统，将系统依据任务分解构造成树形结构，并划分为多区域。讨论在区域中对黑板模型的控制节点采用冗余和负载均衡，从而实现系统的稳定性。最后本文讨论了设计该系统的关键技术。

关键词： 黑板模型 树形结构 多 Agent 系统

1　引言

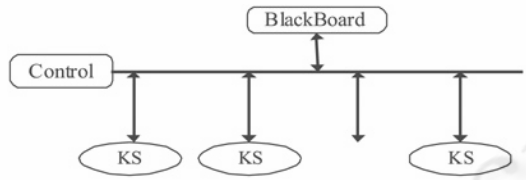
多 Agent[1]（Multi – agent System ，MAS）技术是今年来分布式人工智能（DAI）技术发展的产物，主要研究一组自治的 Agent 在分布式开放动态的环境下，通过交互、合作、竞争、协商等行为完成复杂的控制任务或任务分解。采用多 Agent 技术有利于系统管理、任务分解和模块化管理，因此在应用系统中得到广泛的应用。不过，由于在多 Agent 系统中，Agent 的角色和相对重要性可能会在计算、交互与合作的过程中发生很大的变化，并且在不同阶段，新的 agent 可以加入或者离开应用系统，系统规模越大，系统越不稳定，因此稳定性是开发多 Agent 系统重点考虑的问题之一。

黑板模型结构[2]（Blackboard Architecure ，BBA）是一种平行信息共享数据结构，它能解决分布式人工智能中多个计算实体的协作问题，实现多信息源的交流与共享。黑板结构具有层次性、内在并行性等优点，是多 Agent 系统考虑采用的框架之一。然而传统的黑板模型在串行硬件环境下实现，强调在控制器的作用下，知识源串行执行，因此不利于应用在多 Agent 系统中。

本文采用一种对传统黑板模型改进的结构 – 分布式黑板模型，并主要针对系统的稳定和执行效率考虑，提出一种树形结构，分区域，双节点控制的系统框架模型。

2　黑板模型结构

黑板模型是为了解决在不同物理环境下多个实体协作完成任务的计算模型。该模型能够实现异构知识源的集成，为多 Agent 提供通信支持。图 1 黑板模型结构图。



BlackBoard：黑板，Control：控制单元，KS：知识源

图 1　为黑板模型结构图

控制单元是黑板模型的核心，它主要负责以下几个方面的工作：(1) 对知识源的激活和分配工作；(2) 对黑板上信息的更新工作，例如删除过期的消息；(3) 负责通知知识源读取黑板上的消息。

知识源是黑板模型中的被服务的主体，它们需要在控制单元中进行注册，才能在控制单元的协调下与黑板进行信息交互，从而完成信息和知识的共享。

控制策略是控制单元中对于消息的处理和分配方法，传统的控制策略基本上是先到先的串行控制策略。

黑板模型的关键在于其控制单元的控制策略。传统黑板模型的控制单元是在串行硬件环境下实现的，不能为多 Agent 下的任务分配和调度提供必要的基础设施。因此当多 Agent 对任务分配的实时性有很高要求时，串行的任务分配会使紧急的消息滞后，从而严重影响任务分配的实时性。分布式黑板模型可以解决此缺陷。

3　黑板模型的消息格式

为了实现多 Agent 间的信息交互,各个 Agent 间必须遵循统一的信息交互的格式,所以定义黑板模型的消息格式如表 1 所示。

表 1　黑板的消息格式

报文标识	源 Agent 标识	目的 Agent 标识	许可权	生命周期	报文或消息	消息优先级	信息已读否

其中:消息标识是 Agent 讨论区中消息的序列号;许可权的取值如下:r 是只读权限;d 是删除权限;生命周期是对于设置为只读权限的黑板中的消息,在黑板中停留的时间的限定;消息生命周期随着时间的推移按照斐波那切数列递减,当计数减到零时由控制单元将该条消息删除。消息优先级确定消息被控制器收集的先后次序。

4　基于层次黑板模型的多 Agent 系统

4.1　基于熟人集的多 Agent 系统任务分配机制

任务分配[3]主要有集中式和分布式两种方法,在多 Agent 中可采用熟人集分配方式。在熟人方式中,每个 Agent 都有一个其所知道的 Agent 的能力表,形式如下:{C1:[A_{11},,,A_{1k}],,CN:[A_{n1},,,A_{nk}]},Ci 为执行任务 Ti 所需技能,而 Ai 为知道如何进行工作的 Agent,当进行任务分解时,Agent 可根据此功能表进行分配。

熟人方法通过局部查找进行任务分解。一般情况下,每个 Agent 只有 MAS 系统中部分 Agent 的功能表,假设 A、B、C、D、E 为 MAS 系统中的 Agent,则它们存储的一种可能的功能表可用矩阵表示,如图 2 所示。

$$\begin{array}{c} \\ C_1 \\ C_2 \\ C_3 \end{array}\begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \end{array}\quad \text{Agent A}$$

$$\begin{array}{c} \\ C_1 \\ C_2 \\ C_3 \end{array}\begin{array}{c} \begin{array}{ccc} A & C & D \end{array} \\ \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix} \end{array}\quad \text{Agent C}$$

$$\begin{array}{c} \\ C_1 \\ C_2 \\ C_3 \end{array}\begin{array}{c} \begin{array}{ccc} A & C & E \end{array} \\ \begin{bmatrix} 0 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \end{array}\quad \text{Agent E}$$

图 2　熟人方式的查找矩阵

图中,Agent A、Agent C、Agent E 的功能表只包含它所知道的 Agent 的功能,0 表示 Agent 没有完成任务的能力,1 表示 Agent 有此能力。当 Agent A 面临任务 C2 时,虽然本身没有能力完成任务,但通过功能表知道 Agent C 有能力完成,因此可通过与 Agent C 的通信协商使 Agent C 接受 Agent A 的请求完成任务。当功能表中没有 Agent 可以完成任务时,Agent A 可要求其相识的 Agent 用 Agent A 同样的办法来寻求任务的解决,如通过 Agent C 查看 Agent D 是否可以完成任务,直到 MAS 系统中有 Agent 同意接受此任务,或者整个系统网络没有应答,则表明任务不可完成。因此与其它方式的分解机制相比,熟人方法的机制具有实时性好、整个系统的通信量较小等优点。因此本文所提出的多 Agent 系统将采用熟人机制进行任务分配工作。

4.2　基于层次黑板通信的多 Agent 系统整体框架图

4.2.1　系统结构

我们根据任务分解,建立树形结构的多 Agent 系统模型,由任务分解粒度大小确定树的层次划分。每层被划分为一个或者多个区域,具有多个控制节点,控制 Agent 节点与其所控制的执行任务 Agent 和一黑板组成一个区域。该系统的黑板模型具有层次性,如图 3 所示。

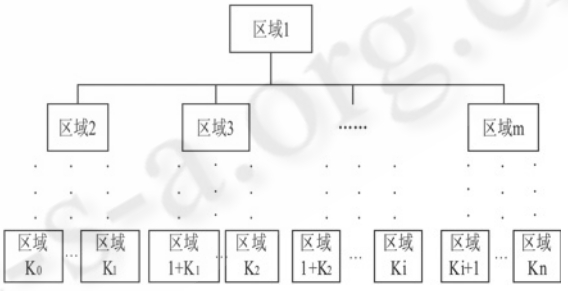


图 3　系统框架图

图中越上层任务粒度越大,越下层任务粒度越小。对系统的整体控制可细划为对区域的控制。

4.2.2　区域内部结构

区域内部是典型的黑板模型结构,如图 4 所示。

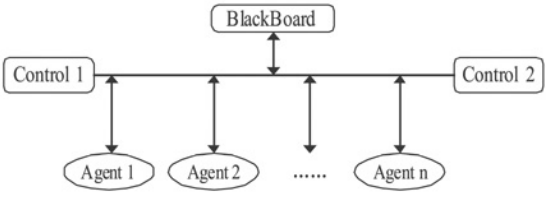


图 4　系统黑板模型结构图

与以往黑板模型结构不同的是,该区域由两个控制 Agent 节点、一个黑板和多个任务 Agent 组成。区域内的控制节点是整个区域的‘ 司令部 ”,起着控制各 Agent 执行、协调各 Agent 交互的作用。在树形结构多 Agent 系统中,上 、下层控制 agent 节点之间是管理和被管理的关系。由于控制 Agent 节点作为整个系统的连接器非常重要,一旦失效,将会影响到整个区域乃至系统的稳定。本文在每个区域中建立双控制节点 Agent,以节点的冗余实现区域稳定,这样当一个控制节点失效时,另一个控制节点可以接受其任务 ;同时,双控制 agent 节点可并行控制任务 Agent 的执行,彼此之间实现负载均衡,使该系统任务调度效率更高。控制 Agent 节点之间的通信不采用黑板通信方式,而是消息传递方式,以此加快通信效率。

控制 Agent 节点结构相同、功能相同,负责监控收集黑板上的信息,并依据信息激活相应的任务 Agent 处理。任务 Agent 将处理后的结果状态信息传给黑板。在处理信息的过程中,任务调度信息的优先级最高。如果在任务执行过程中,控制 Agent 发现本身所在的区域无法解决任务,可向下层控制节点发送任务请求,进一步进行任务分解,降低任务粒度。系统任务分配方面是采用熟人集任务分配方式,控制 Agent 节点向其他控制 Agent 节点寻找合适的能完成该任务的 Agent,计算出结果 ;否则返回无法完成任务的信息。

5 实现系统的关键技术

5.1 负载均衡

从系统整体来看,除了树的最上层次节点和最下层节点,其他的控制 Agent 节点均同时为管理节点和被管理节点。管理控制 Agent 节点执行负载均衡分配,运行负载均衡算法,将相关任务和负载调整策略通过通讯模块传递给下层被其管理的控制 Agent 节点上。

所有控制 Agent 节点装有状态时钟(ST),时间间隔分别为 Tl。ST 时钟控制各节点以周期 Tl 向管理节点发送状态消息、表明其正常工作和负载状况 ;控制 Agent 节点的负载与 CPU 、内存、硬盘系统、网络等资源利用率有关,定义 T 时刻控制节点处理能力 C(t)为 :

$$C(t) = ((1 - a(t)) * P + ((1 - b(t)) * R + (1 - c(t)) * D + (1 - d(t)) * N) ZP + R + D + N = 1$$

其中 a(t)、b(t)、c(t)、d(t)分别表示 t 时刻 CPU、内存、硬盘系统、网络的利用率,P、R、D、N 表示权值 ;Z 表示控制节点总的处理能力,则可以用 C(t)近似表示服务器在[t,t + △t]时间内的处理能力,负载均衡分配节点在[t,t + △t]内根据下层被管理的控制 Agent 节点的 C(t)进行负载分配 ;当需要进行负载分配时,总是分配给 C(t)值最大的节点。并且所有的控制 Agent 节点均有一个阀值,当 C(t) < T 时,表明该控制节点负载过重,应该进行调整。控制节点在 ST 时钟控制下周期性发送 C(t)给运行负载均衡算法的上层管理节点,处理负载突发的情况。

5.2 并发控制

由于多个 Agent 在并发执行时有可能对同一黑板数据进行引用,而这种引用是不可预测的,因此需要有一种机制来维护黑板数据存取的一致性,这就是并发控制的任务。系统的并发控制采用的是锁机制,读锁和写锁两种类型。如果黑板的某数据项被一个知识源加上了写锁,那么其它 Agent 就不能对该数据项进行读或写操作,而加锁的 Agent 可以对该数据项做任何操作。如果某数据项被一个 Agent 加上了读锁,那么其它 Agent 都不能对该数据项进行写操作,但可以做读操作。每当 Agent 要操作一个数据项时,必须首先获得一个适当的锁。当一个 Agent 试图获得一个已被其它 Agent 占有的锁时,该 Agent 的执行只能被挂起,等待这个锁被释放。某 Agent 所加的全部的锁一直维持到该 Agent 执行结束时再释放。很显然,这种并发机制是通过保证写操作的串行化而确保数据的一致性的。

采用这种锁机制必然引起死锁问题。例如,几个并发执行的 Agent 因互相等待对方释放锁而使自己的当前活动被挂起,系统就进入了死锁状态。该系统采用死锁预防法来解决死锁问题。即利用 Agent 的优先权,规定 :优先权高的 Agent 允许等待优先权低的 Agent 释放锁,否则就终止 Agent 的执行。这样,在锁的等待途中就不可能形成一个环,避免了死锁的发生,而且也保证了优先权高的 Agent 的执行不会被终止,加快了问题求解速度。

但这种办法带来的一个问题是优先权低的 Agent

很可能长时间得不到锁,不停地被启动或终止。这主要是由于优先权是预先给定的、静态不变的。改进的办法是采用时间邮戳,即一个 Agent 开始执行的时间。规定:时间邮戳小的 Agent 允许等待时间邮戳大的 Agent 释放锁,否则就终止 Agent 的执行。这样,后执行的 Agent 的时间邮戳总是比先执行 Agent 的时间邮戳小,总被允许等待得到锁。有一点要注意,一个 Agent 终止执行后又重新启动,不能给它赋新的时间邮戳。这样避免了某个 Agent 一直处于重新启动或终止的状态,保证了每个 Agent 均衡得到锁。

5.3 故障检测

系统专门设置一个检测 Agent 为系统中所有的控制节点 Agent 进行检测。在该检测 Agent 中设置状态区,用于系统内部所有 Agent 的状态,其每一项的结构如下。

表 2 检测 Agent 的状态表

Agent D	Agent 名称	Agent 对象引用	Agent 对外通讯计数	心跳计数

每个 Agent 以固定时间间隔 Ta 将检测 Agent 状态区中的相应项目的心跳计数值设置为 1,而检测 Agent 以固定间隔时间 Tl 去检查状态区中各个 Agent 的心跳计数值。当 Agent 的计数值为 0 时,则认为该 Agent 出现故障;否则,将心跳值设置为 0。其中 Ta <Tl.

对于控制 Agent 节点失效,由检测节点通知其上层管理节点,用同一区域的另一个控制 Agent 节点承接失效控制 Agent 节点的任务。

如果是一般的任务 Agent 出错,依据一定的替换策略,由具备同等功能的单个或多个任务 Agent 去替代完成。

5.4 通信技术

多 Agent 系统中,Agent 间的通信策略一般分为两类:信息传递方式和黑板方式。在本论文讨论的系统中,普通任务节点是通过黑板方式通信的,把信息放到黑板中,等待控制节点收集,进而调度相应的任务 Agent 执行。这种方式不要求 Agent 之间有较深的了解。而控制节点之间信息交互重要而频繁,因此直接采用消息进行通信,传递速度快,满足实时性要求,并能应付紧急情况的出现。

6 结论

提出了一种基于层次黑板模型的多 Agent 系统框架。将树形框架依据任务分解粒度大小确定层次,每个层次划分多个区域。然后考虑在区域内建立双层控制 agent 节点确定系统稳定性和效率,使用系统在现有资源和代价下,达到更大可用性和可靠性。文中只给出了部分思路和想法,还没有付诸于实现,这是今后的工作之一,相信该思想在提升大规模实时系统的健壮性方面将会有广阔的应用前景。

参考文献

1 李海刚、吴启迪. 多 Agent 系统研究综述. 同济大学学报, 2003,31(6): 728 – 731.

2 王斌、张尧学、陈松乔. 一种基于黑板模型的多 Agent 系统通信方法. 小型微型计算机系统,2002,23(11): 1355 – 1358.

3 高志军、颜国正、丁国清. 多 Agent 协作环境下的任务分配. 系统工程与电子技术,2005,27(1): 134 – 136.