一种多 agent 系统框架与协商机制研究 *

王 鲁, 王志良, 杨 溢
(北京科技大学 信息工程学院, 北京 100083)

摘 要: 从基于动态、异构网络上快速构建稳健的多 agent 系统出发,设计了多 agent 远程过程调用通信模型,定义了三种基本类型的 agent,对 KQML 消息规范进行扩展,增加了对消息生存周期的控制,设计了双缓存消息推送器以实现 agent 消息的主动推送,并在 WCF 的基础上实现了该通信框架。针对同目标多 agent 协作系统提出了基于开销均衡的 agent 系统交互协商策略,通过实例证明相对于独立运行和基于正交互协商策略的 agent 系统,本协商策略可有效降低系统总开销,并可使运行负载更为均衡。

关键词: 多 agent; 通信模型; 开销均衡; 协商策略
中图分类号: TP302.1 **文献标志码**: A **文章编号**: 1001-3695(2012)03-0852-04
doi:10.3969/j.issn.1001-3695.2012.03.014

Framework of multi-agent system and consultation mechanism

WANG Lu, WANG Zhi-liang, YANG Yi
(*School of Information Engineering*, *University of Science & Technology Beijing*, *Beijing* 100083, *China*)

Abstract: Based on dynamic, heterogeneous network to quickly build robust multi-agent system, this paper designed a remote procedure call model for multi-agent communication, and defined three basic types of agent, extended KQML message specification to control message life cycle. It designed a double-cache message pusher to push agent message actively, and realized the communication framework on the basis of WCF. This paper proposed a multi-agent negotiation strategy based on cost balance for multi-agent collaboration system with same target. The example proves that, compared to the system that agents operate independently or use interactive negotiation strategy, this negotiation strategy can effectively reduce the total system cost, and can run the load more evenly.

Key words: multi-agent; communication model; cost balance; negotiation strategy

软件 agent 是一种具有自主性和协作性的软件程序,可以帮助用户完成一些特定问题的求解,具有自治性、社会性、反应性、能动性等特点,可根据周围环境的变化调整自身的状态。在多 agent 系统中,agent 根据自身具有的问题求解能力通过与其他 agent 的协调、协商、协作共同完成单个 agent 不能完成的任务[1]。良好的通信方式是 agent 高效完成任务的前提。目前已经有多种可以利用分布式对象技术实现多 agent 系统通信的方法,如 OGM 组织提出的公共对象请求代理体系结构 CORBA[2], Microsoft 提出的分布式组件对象模型 DCOM[3]。由于分布式系统的大量出现,对于快速构建 agent 及其系统提出了更高的要求。例如,在家庭环境下,需要对每种设备构建不同的 agent,由这些具有不同能力的 agent 在动态、异构的网络中保证可靠性,并维持网络的健壮性。上述 agent 构建方法在跨平台和访问对象上存在不足或本身实现复杂,难以满足快速构建 agent 系统的要求。现有 agent 的通信方法多采用基于黑板的通信,当仅具有简单功能的 agent 的数量很多时,消息在黑板中呈现的数量会急剧增长,消息的频繁存取使得多 agent 系统的整体性能下降,难以满足灵活可靠的要求。另外,在异构网络中,由于系统中 agent 所代理设备的功能单一,系统的负载将是不平衡的,影响整体运行的鲁棒性。如何平衡个体与整体的协同性是需要解决的重要问题[4]。文献[5]提出了一种基于正交互的协商策略,在两个 agent 之间进行协商,不能达到全局最优。基于上述分析,本文针对分布式环境中异构网络 agent 系统中通信的特点,提出了基于 WCF 的多 agent 系统框架,并使用扩展的正交互 agent 协商策略用于减低异构网络系统运行的开销和平衡负载。

1 多 agent 协作通信框架

1.1 多 agent RPC 通信模型

远程过程调用(remote procedure call,RPC)可以使计算机通过网络访问远程计算机,而无须了解网络底层技术,这使得开发分布式网络程序变得简单易用[6]。本文提出的多对多 agent RPC 通信模型是对传统 RPC 模型的改进。在传统的 RPC 模型中,客户端在远程调用服务器方法前需要获知标准的方法接口,否则难以远程调用方法;本文的 agent 在生成客户端前,动态或静态添加服务引用,通过下载服务接口获取服务器提供的所有方法的集合,agent 搜索集合中有价值的方法,自主决定是否调用服务中的方法。本文设计的多对多 agent RPC 通信模型遵循服务器/客户端模式,如图 1 所示,agent 设

收稿日期: 2011-08-29; **修回日期**: 2011-10-08 **基金项目**: 国家自然科学基金资助项目(60903067);国家高科技研究计划资助项目(2007AA04Z218)

作者简介:王鲁(1986-),男,内蒙古赤峰人,博士研究生,主要研究方向为人工智能、智能决策、信息集成(wangluking163@163.com);王志良(1956-),男,教授,博导,主要研究方向为人工心理及情感计算、智能机器人学、和谐人机交互、3C 融合;杨溢(1984-),男,博士研究生,主要研究方向为人工智能、智能决策、信息集成.

备向外界提供服务作为服务器端,同时自身也可作为客户端调用环境中其他 agent 设备提供的服务。

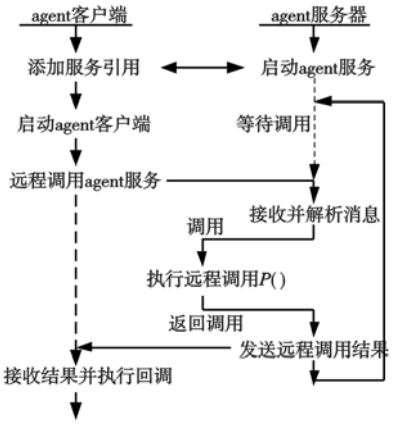

图1　Agent服务器/客户端通信模式

Agent 客户端与 agent 服务器通信的过程如下:

a) Agent 客户端

(a) 动态或静态添加 agent 服务器端的服务引用,下载相应的服务接口,获取服务的方法集;

(b) 定义服务执行的回调,生成 agent 客户端;

(c) 远程调用 agent 服务器所提供的某项方法 $P()$;

(d) 等待 agent 服务器返回应答;

(e) 执行回调,继续 agent 客户端的其他功能。

b) Agent 服务器

(a) 启动 agent 服务器的服务;

(b) 等待 agent 客户端的调用;

(c) 收到 agent 客户端调用方法 $P()$ 的消息后,执行相应的方法,并将返回值传送回 agent 客户端;

(d) 回到(b),继续监听等待请求。

传统的多 agent 系统中的 agent 以独立的个体存在,当有新的 agent 设备加入到系统中时,需要向已有的每个 agent 设备请求服务列表,信息的重用率较低;当 agent 设备的数量很多时,网络的性能与可靠性将降低。在本文设计的多 agent 系统中增加了 agent 协调器(agent coordinator),对系统中 agent 设备的工作状态进行监督,如图 2 所示。

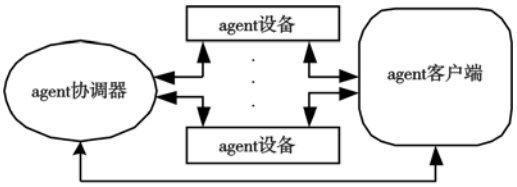

图2　多对多的agent RPC通信模型

本文提出的多对多 agent RPC 通信模型中包括三种基本类型的 agent,功能如下:

a) Agent 协调器(agent coordinator)。负责监测多 agent 系统中 agent 的上下线情况,收集所有 agent 设备的服务、状态、消息,将这些信息以其自身服务的方式供 agent 设备和 agent 客户端调用。

b) Agent 设备(agent device)。它是底层设备面向上层应用的代理程序,既可向外提供服务,又可从 agent 协调器处获取其他 agent 的服务,为其内部的推理结构提供依据。

c) Agent 客户端(agent client)。它是多 agent 系统中的非必要设备,可从 agent 协调器和 agent 设备处获取服务,但不对外提供任何服务。

为了使不同的 agent 设备在上述模型中提供统一的服务,规定 agent 设备必须实现以下四种服务原语:

a) Agent. Beacon(state)。Agent 在线宣告信标原语,该原语在 UDP 端口每隔 15 s 广播一次,其唯一参数 state 为 XML 数据格式,其中包括该 agent 设备上线时间、服务发布地址。若上线后 30 s 内其他设备没有再次收到在线宣告信标,则可认为该 agent 已经下线。

b) Agent. State(operating-id)。Agent 运行状态原语,为其他 agent 提供当前的运行状态,其唯一参数为要获取的 agent 运行状态的 ID,返回值的类型由运行状态的类型决定。

c) Agent. DeviceInfo(deviceinfo)。Agent 信息原语,为其他 agent 提供该 agent 的信息,在其唯一的参数 deviceinfo 中提供了 agent 名称、序列号、设计日期、位置和设备类型。

d) Agent. SubscribeState(state)。Agent 状态改变订阅原语,其唯一参数 state 的值可以为 Subscribe_True 或 Subscribe_False,以确定是否订阅 agent 状态改变时所产生的消息。

1.2　消息通知机制

黑板可以用在任务共享和结果共享的系统中,为基于事件的问题求解提供了一种方法。多 agent 系统中的 agent 需要不定时地访问黑板并从大量信息中搜索感兴趣的消息,先进的黑板系统为不同的 agent 提供了不同的区域以便于搜索。即便如此,当 agent 的数量增加时,黑板中的数据和访问黑板的次数将呈指数增长,且黑板中的消息随着等待时间的增长而面临失效的状态[7]。本文在消息结构中对消息的时效性作了规定,双缓存消息推送器(double-cache message pusher, DMP)收到消息后在其时效内会立即对消息进行处理,并推送给可执行的 agent,从而减少 agent 客户端在消息搜索上的开销以及避免由轮询检索而产生的等待。

本文提出的多 agent 协作通信框架中所使用的消息结构是对 KQML(knowledge query and manipulation language)消息规范的扩展,如表 1 所示。其中,startTime 和 deadline 规定了消息的生存时间,单位精确到 ms。本文采用基于时间优先级排序的消息冲突消解策略,消息首先按照 startTime 排序,优先级相同的消息按照 deadline 排序,最后优先级相同的消息将随机决定优先级的高低。

表 1　分割实验数据

关键词	意义	关键词	意义
performative	消息类型	sender	消息发送方
content	消息内容	startTime	消息形成时间
ontology	本体名称	deadline	消息有效截止时间
receiver	消息期望接收方		

双缓存消息推送器的结构如图 3 所示。

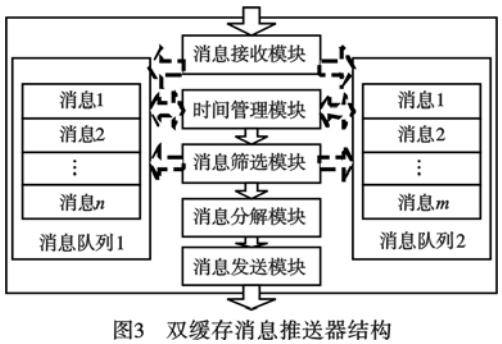

图3　双缓存消息推送器结构

DMP 位于上述多 agent 系统中的 agent 协调器上,其结构

中包括两个用于存储消息请求的双缓存和一系列的推送模块。

由消息接收模块接收来自 agent 设备发送来的请求消息，根据缓存中的运行情况决定消息推送的位置。当消息队列 1 处于工作状态时，将消息推送到消息队列 2 上，反之亦然。两个消息队列共同拥有一个维护程序，不断轮流扫描两个消息队列，当消息队列中有未完成的消息时，将该消息推送给时间管理模块。时间管理模块可保证分布式系统中 agent 协调器与 agent 设备之间的时空一致性，完成消息推送的同步，使得消息按照 agent 的意愿执行。时间管理器对比当前时间和推送过来的消息截止时间 deadline，以决定是否丢弃该消息。被保留的消息被推送到消息筛选模块，该模块搜索多 agent 系统中所有 agent 设备的功能，判断是否有可以完成该消息的设备。若没有，则将该消息推送回消息队列中，等待可用 agent 上线；若有，则将该消息推送给消息分解模块。消息分解模块将推送来的消息进一步分解为 agent 设备服务中的具体方法，通过消息发送模块调用 agent 设备提供的服务，完成整个消息的实现。

2　多 agent 通信框架实现

2.1　WCF

WCF(windows communication foundation)是微软构建面向服务的应用提供的分布式通信框架，是. NET Framework 的重要组成部分。WCF 整合了如 enterprise service、net remoting、Web service 和 MSMQ 等分布式系统相关的技术，可实现跨进程、跨网络，支持多种形式的宿主、协议和安全模式，简化了开发基于 SOA 的分布式系统的过程[8]。

WCF 服务由服务类、宿主和终结点三个基本要素组成。服务类包括服务契约(service contact)、操作契约(operation contact)和数据契约(data contact)，通过公开契约向外提供服务。宿主可为服务提供运行环境。终结点由地址(address)、绑定(binding)和契约(contact)组成，指明了服务的具体调用方式。本文在 WCF 的基础上设计实现了多 agent 通信系统。

2.2　Agent 协调器

Agent 协调器负责监管整个系统中 agent 的上下线情况、收集 agent 的功能，并推送 agent 请求的消息。Agent 协调器的结构如图 4 所示。

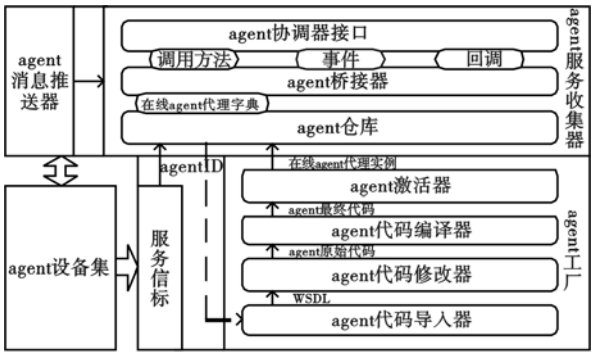

图4　Agent协调器的结构

Agent 协调器的服务信标负责监听并接收来自 agent 设备广播的在线宣告信标和下线宣告信标，将结果通报给 agent 服务收集器中的 agent 仓库。Agent 仓库维护一个哈希表类型的在线 agent 列表，若收到的是下线宣告信标，将列表中对应的 agent 删除；若收到的是上线宣告消息，判断 agent 设备是否已经在列表中，若没有，则提取 agent 设备的 Web service 描述语言(Web services description language，WSDL)中的地址发送给 agent 工厂，否则标明该 agent 处于在线状态。当列表中的 agent 在 30 s 没有更新在线状态，agent 仓库自动清除该 agent。

Agent 工厂负责将刚上线的设备包装成一个可访问的 agent 实例。Agent 代码导入器通过访问 WSDL 地址下载 agent 服务信息，导入 WSDL，agent 代码修改器生成可用的原始代码，agent 代码编译器动态编译并加载程序集获得 agent 最终代码，最后，agent 激活器实例化 agent 设备代理、注册回调，并将其推送给 agent 服务收集器中的 agent 仓库。

Agent 服务收集器中的 agent 仓库为上层提供了在线代理字典引用。Agent 代理桥接器处理 agent 协调器接口的请求，为其提供调用方法、事件和回调等服务。Agent 设备集中的服务请求消息由 agent 协调器中的双缓存消息推送器按照本文第 2 章中的机制调用服务收集器中的服务，完成服务请求。

在 agent 协调器中完成了上述框架中的通信原语，这包括：获得 agent 协调器的基本信息；当前在线 agent 的数量；获得指定/所有 agent 的设备信息；获得指定在线 agent 的服务列表；获得指定服务的详细信息；提供指定服务的调用方法；订阅/取消订阅 agent 在线状态；订阅/取消订阅指定 agent 运行状态。为了提供订阅消息，agent 协调器提供了 agent 设备上线/下线回调 AgentAliveChanged，以及 agent 设备运行状态回调 AgentStatusChanged。考虑到 agent 设备不同的访问能力和功能请求，agent 协调器提供了四种 WCF 服务的绑定方式：webHttpBinding、basicHttpBinding、wsHttpBinding、wsDualHttpBinding。

2.3　Agent 设备

本文设计的基于 WCF 的 agent 设备结构如图 5 所示。

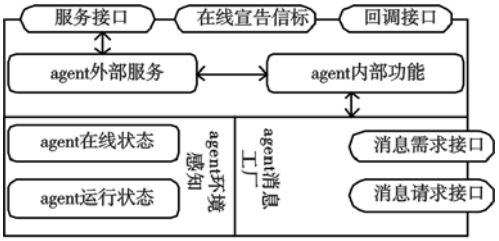

图5　Agent的一般组成

Agent 设备将自身具有的特定功能进行封装，提供给外部服务和消息工厂调用。在线宣告信标为 XML 格式，将 agent 设备的在线状态、上线时间和 WSDL 地址发布给 agent 协调器和其他潜在客户端。WCF 的发布机制包括 HttpGet 和 MexMetadata 两种。当收到在线信标时，服务需求者会由服务发布地址自动获取服务信息。

Agent 设备的服务接口提供了所有可用的设备功能，除此以外还包括：获取该 agent 的设备信息；订阅/取消订阅该 agent 运行状态；获取该 agent 的所有服务状态；获取该 agent 指定服务状态。Agent 设备的服务发布采用 wsDualHttpBinding 的绑定方式以支持双向通信。

Agent 设备的回调接口为服务使用者提供了更多信息共享的能力。当该 agent 设备的状态发生改变时，会调用 StatusChanged 回调方法，通知所有已注册的 agent。通过这种方法减少了 agent 之间不必要的通信，有效地降低了网络通信开销。agent 设备利用网络中其他设备的回调消息和在线宣告信标维护自身的环境感知列表，包括其他 agent 的在线状态和运行状态。Agent 消息工厂提供了用于接收其他设备服务需求的消息需求接口和用于向 agent 协调器及客户端发布请求的消息请求接口。消息的格式遵循本文第 2 章的规定。

3　基于开销均衡的交互协商策略

多 agent 系统工作的环境多具有动态性、开放性的特点,因此,拥有相同目标的 agent 需对系统中的资源进行交互和再分配,以实现资源的最高利用率并提升系统的整体性能。本文提出基于开销均衡[5]的 agent 系统交互协商策略,通过多 agent 对相同目标自主竞争协商,以合作的方式实现任务的委托,最终达到均衡 agent 的运行开销和降低系统运行负载的目的。

3.1　基本假设

在多 agent 系统中,有如下假设:agent 可完成相同的任务;每个 agent 可以与其他 agent 进行通信和协商;对于特定任务 agent 的运行开销和通信开销是可确定的。在此假设的基础上,有如下定义:

执行开销——Agent A 执行某一特定任务所需要的开销,记做 $C_{\mathrm{exe}}(A)$;

委托开销——Agent B 将其与 agent A 重叠的部分委托 agent A 执行,agent A 执行所需要的开销,记做 $C_{\mathrm{del}B}(A)$;

附加开销——Agent A 执行 agent B 的委托,执行任务所产生的附加开销,记做 $C_{\mathrm{ext}B}(A)$;

运行开销——Agent A 执行某一特定任务并完成 agent B 的委托所产生的开销,记做 $C_{\mathrm{opr}B}(A)$,且满足

$$C_{\mathrm{opr}B}(A)=C_{\mathrm{exe}}(A)+C_{\mathrm{del}B}(A)+C_{\mathrm{ext}B}(A)$$

总开销——Agent A 完成多次运行所产生的开销,记做 $C(A)$,满足 $C(A)=\sum_{i=1}^{n}C_{\mathrm{opr}}^{i}(A)$。其中 n 为运行的次数。

3.2　协商策略算法

在基本假设的前提下,本文提出的协商策略要实现 agent 的最终开销相近,系统运行的总开销最小。假设需要完成的任务为 T,系统中有 $N(N>1)$ 个具有完成任务 T 能力的 agent,定义 agent 的集合为 M。具体算法描述如下:

a)确定每个 agent 的执行开销

$$\{C_{\mathrm{exe}}(I)\mid I\in M\}$$

确定每个 agent 相对于其他 agent 的委托开销

$$\{C_{\mathrm{del}I}(J)\mid I\in M,J\in M\backslash I\}$$

确定每个 agent 在执行委托任务时产生的附加开销

$$\{C_{\mathrm{ext}I}(J)\mid I\in M,J\in M\backslash I\}$$

初始化每个 agent 的总开销

$$\{C(I)\mid I\in M\}$$

b)计算每个 agent 执行一次任务产生的运行开销

$$\{C_{\mathrm{opr}\bar{I}}(I)\mid I\in M\}$$

其中,$C_{\mathrm{opr}\bar{I}}(I)$ 按如下公式计算:

$$C_{\mathrm{opr}\bar{I}}(I)=C_{\mathrm{exe}}(I)+\sum_{J=1,J\neq I}^{N}C_{\mathrm{del}J}(I)+\sum_{J=1,J\neq I}^{N}C_{\mathrm{ext}J}(I)$$

c)计算每个 agent 的开销评估函数

$$E_I=C(I)+C_{\mathrm{opr}\bar{I}}(I)$$

d)获取 E_I 中最小值对应的 I 值,定义为 K

$$\{K\mid E_K=\min\bigcup_{I=1}^{N}E_I\}$$

(a)当 K 存在唯一值时,E_K 即为 N 个 agent 协商后的最优值,调整对应 K 值的 agent 的总开销 $C(K)$,$C(K)=E_K$,保持其他 agent 的总开销不变。

(b)当 K 存在多个值时,获取 K 值对应的运行开销中的最小值,最小值对应的 K 值定义为 K':

$$\{K'\mid C_{\mathrm{opr}\bar{K'}}(K')=\min\bigcup_{I=K}C_{\mathrm{opr}\bar{I}}(I)\}$$

①当 K' 存在唯一值时,$E_{K'}$ 即为 N 个 agent 协商后的最优值,调整对应 K' 值的 agent 的总开销 $C(K')$,即 $C(K')=E_{K'}$,保持其他 agent 的总开销不变。

②当 K' 存在多个值时,从 K' 中随机确定接受任务的 agent,更新相应的总开销,方法同①。

e)由上一步确定的 agent 执行委托任务,回到 b)继续确定执行任务的 agent 并完成后续的任务。

3.3　算法实例

假设多 agent 系统中存在三个 agent,即 agent A、agent B、agent C,同时完成任务 T,任务 T 包括三个子任务。设 agent A 完成任务 T 的执行开销为 7、2、9;agent B 完成任务 T 的执行开销为 7、4、11;agent C 完成任务 T 的执行开销为 8、3、6;agent 间通信的附加开销均为 1,三个 agent 的委托开销分别为 2、4、3,则对于该系统运行 100 次任务 T,运用本文的协商策略可得到如下结论:

a)运行 100 次任务 T 后,agent 协商结果如表 2 所示。若三个 agent 独立运行,则各自的开销分别为 2 700、2 900 和 2 500,系统总开销为 8 100;运用文献[5]中的正交互协商策略,则 agent 各自的开销分别为 1 541、1 550、1 562,系统总开销为 4 653,是独立运行的 57.4%;运用本文的协商策略,系统的总开销为 2 690,是独立运行的 33.2%,这表明采用本文的协商策略使得 agent 各自的开销和系统运行的总开销都明显降低。

表 2　任务 T 的 agent 协商结果

协商次数	C(A)	C(B)	C(C)	协商结果/次数
10	81	87	100	A(3) B(3) C(4)
20	189	174	175	A(7) B(6) C(7)
30	270	261	275	A(10) B(9) C(11)
40	351	348	375	A(13) B(12) C(15)
50	459	435	450	A(17) B(15) C(18)
60	540	522	550	A(20) B(18) C(22)
70	621	638	625	A(23) B(22) C(25)
80	702	725	725	A(26) B(25) C(29)
90	810	812	800	A(30) B(28) C(32)
100	891	899	900	A(33) B(31) C(36)

b)Agent 运行负载标准差与运行次数关系如图 6 所示。当 agent 系统独立执行任务 T 时,agent 的运行负载标准差随着运行次数的增加而增加,并且类似于线性增长;当采用文献[5]中的策略和本文的协商策略时,agent 的运行负载标准差在较小的范围内波动,且随着运行次数的增加,运行负载标准差远小于独立运行时的标准差。采用文献[5]策略的标准差最大值为 18.4,最小值为 1.7,而采用本文策略的标准差最大值为 12.5,最小值为 1.2,这表明采用本文的协商策略可使得 agent 的运行负载更为均衡。

4　结束语

本文设计的多 agent 通信模型在实验室环境下的智能家居网络 LookeyHome 中得到测试,满足动态、异构网络上快速构建稳健的多 agent 的需求。基于开销均衡的 agent 系统交互协商策略为同目标多 agent 系统高效运行提供了理论依据,在上述实际的测试中可有效降低系统运行总开销　(下转第 858 页)

2 实验结果分析对比

为了验证本文软聚类算法的有效性,从中国知网上下载了数学类、物理类、政治类、生物类和计算机类各 100 篇文章作为测试数据集,对文献的引用关系和摘要进行了聚类分析,采用常用的性能评价方法:召回率、查准率[9],并且与 S2FCM[10] 和改进的 S2FCM 算法[11] 进行了实验结果对比。表 2 为算法召回率对比,表 3 为算法查准率对比。

表 1 算法时间复杂度对比

算法	类型	时间复杂度	备注
划分算法	K-means	$O(Nkt)$	迭代,非增长
	PAM	$O(kt(N-k)^2)$	非增长
层次聚类	单连接	$O(kN^2)$	非增长
	平均连接	$O(kN^2)$	非增长
	全连接	$O(kN^2)$	非增长
本文算法		$O(N(n+\log N))$	增长类型

对比表 2 和 3 可以发现,本文提出的算法在召回率和准确率上对比已存在的算法 S2FCM 和改进的 S2FCM 都有不同程度上的提高,值得注意的是,由于计算机属于服务类交叉学科,即计算机类的文章与很多数学、物理、生物上的文章相关,因此,在查询过程中,准确率受到了一定的限制;而政治类的文献与数学、物理等方面均没有交叉性,因此,形成了高内聚、低耦合的聚类,查询准确率就比较高。

表 2 算法召回率对比 %

算法	数学类	物理类	政治类	生物类	计算机类
S2FCM	79.4	81.3	83.5	76.2	72.5
改进 S2FCM	84.6	84.2	88.7	82.4	79.6
本文算法	91.2	92.1	95.3	91.3	87.3

表 3 算法查准率对比 %

算法	数学类	物理类	政治类	生物类	计算机类
S2FCM	82.5	83.4	88.6	81.4	76.8
改进 S2FCM	86.4	87.6	90.7	84.8	86.8
本文算法	93.7	94.5	96.8	91.5	92.6

3 结束语

本文提出了一种新颖的文献聚类算法,通过分析文献之间的引用关系进行角色划分,可以有效地查询到同时属于多种聚类的交叉文献;通过对文献关键词进行分析和研究,构造聚类主题,有效地抑制了“主题漂移”现象。实验证明,本文算法提高了对文献的搜索精度和搜索效率,具有较高的实用价值。该算法已在 SNS 科技论文管理平台[12] 中使用,并取得了不错的实验效果。

参考文献:

[1] 孟海涛, 陈笑蓉. 基于模糊相似度的科技文献软聚类算法[J]. 贵州大学学报:自然科学版,2007,24(2):175-178.

[2] BIANCONI G, BARABSI A. Competition and multiscaling in evolving networks[J]. Europhysics Letters,2001,5(4):436-442.

[3] 王娴,谢弛,荣雪,等. SNS 走向何方——SNS 网站运营的现状和未来趋势研究[EB/OL]. 2008-12-03. 人民网.

[4] 艾伯特. 巴拉巴西. 链接网络新科学[M]. 徐彬,译. 长沙:湖南科学技术出版社,2007.

[5] LI Xiao-dong. Adaptively choosing neighborhood bests using species in a particle swarm optimizer for multimodal function optimization[C]// Proc of Genetic and Evolutionary Computation Conference. 2004:105-116.

[6] FAN Cong-xian, XU Ting-rong. Research and improved algorithm of HITS based on Web structure mining[C]//Proc of Computer Information. 2010:160-162.

[7] 高琪,张永平. PageRank 算法中主题漂移的研究[J]. 网络与通信,2010,3(3):117-119.

[8] 胡健,董跃华, 杨炳儒. 大型网络中社区结构发现算法[J]. 计算机工程,2008,34(19): 92-93.

[9] 范聪贤,徐汀荣,范强贤. Web 结构挖掘中 HITS 算法改进的研究[J]. 微计算机信息,2010,26(1-3):160-162.

[10] 裴继红, 范九伦, 谢维信. 一种新的高效软聚类方法:截集模糊 C-均值(S2FCM)聚类算法[J]. 电子学报,1998,26(2):83-86.

[11] 白似雪,陆萍. 一种基于文本分类的特征选择方法[J]. 南昌大学学报:工科版,2008,30(1): 87-90.

[12] http://www.linkscholar.com/[EB/OL].

(上接第 855 页)和均衡运行负载。但本文的交互协商策略只考虑了多 agent 系统中单一任务的委托,在将来的研究中,笔者希望进一步研究多任务的委托执行,也试图将这种策略应用在其他工程领域中。

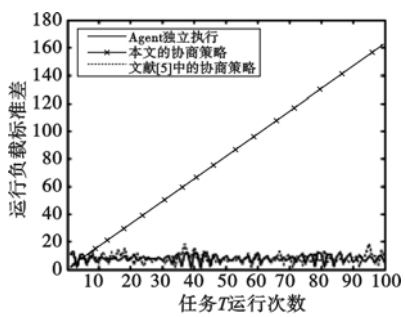

图6 运行负载标准差与运行次数关系

参考文献:

[1] RAJI F, LADANI B T. Anonymity and security for autonomous mobile agents[J]. IET Information Security,2010,4(4):397-410.

[2] FELBER P, NARASIMHAN P. Experiences, strategies, and challenges in building fault-tolerant CORBA systems[J]. IEEE Trans on Computers,2004,53(5):497-511.

[3] COLOMBO A W, SCHOOP R, NEUBERT R. An agent-based intelligent control platform for industrial holonic manufacturing systems [J]. IEEE Trans on Industrial Electronics, 2005, 53 (1): 322-337.

[4] 吴菊华, 吴丽花, 甘仞初. 基于规范的多 agent 协同机制研究[J]. 计算机应用研究,2009, 26(5):1778-1781.

[5] 何炎祥,陈苹萌. Agent 和多 agent 系统的设计与应用[M]. 武汉:武汉大学出版社,2001.

[6] SANG-HOON K, JIN-SOO K, SEUNGRYOUL M. Modeling and evaluation of serial multicast remote procedure calls (RPCs) [J]. IEEE Communications Letters,2009,13(4):283-285.

[7] SUNGJIN C, HONGSOO K, EUNJOUNG B, *et al*. Reliable asynchronous message delivery for mobile agents [J]. IEEE Internet Computing Magazine,2006,10(6):16-25.

[8] ZHANG Wei, CHENG Gui-xue. A service-oriented distributed framework-WCF[C]//Proc of International Conference on Web Information Systems and Mining. 2009:302-305.