ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.21, No.8, August 2010, pp.1795−1809
doi: 10.3724/SP.J.1001.2010.03669

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于 Agent 和蚁群算法的分布式服务发现*

郑 啸[1,2+], 罗军舟[1], 宋爱波[1]

[1](东南大学 计算机科学与工程学院,江苏 南京 210096)

[2](安徽工业大学 计算机学院,安徽 马鞍山 243002)

Distributed Service Discovery Based on Agent and Ant Colony Algorithm

ZHENG Xiao[1,2+], LUO Jun-Zhou[1], SONG Ai-Bo[1]

[1](School of Computer Science and Engineering, Southeast University, Nanjing 210096, China)

[2](School of Computer, Anhui University of Technology, Maanshan 243002, China)

+ Corresponding author: E-mail: xzheng@seu.edu.cn

Zheng X, Luo JZ, Song AB. Distributed service discovery based on agent and ant colony algorithm. *Journal of Software*, 2010,21(8):1795−1809. http://www.jos.org.cn/1000-9825/3669.htm

Abstract: This paper suggests an ant-like agent service discovery mechanism. There are two types of agents cooperating to search target services: Search Agent and Guide Agent. Search Agent simulates the behavior of an ant that searches for services on the network. Guide Agent is responsible to manage a service route table that consists of pheromone and hop count, instructing Search Agent's routing. Volatile pheromones make Search Agent sense the change of topology and service resource, and hop count makes them know the distance. Semantic similarity is also introduced in routing selection as a heuristic factor, which improves the recall. The life-span control policy makes query traffic controllable. With system size increasing, the query traffic would increase slightly and has an upper bound. The result of simulation shows that the suggested mechanism is scalable and adaptable enough to be suitable for large-scale dynamic computing environments.

Key words: service discovery; peer to peer network; agent; ant colony algorithm; routing mechanism

摘 要: 提出一种类似蚂蚁觅食活动的 Agent 服务发现机制.有两类 Agent 合作寻找目标服务:Search Agent 和 Guide Agent.前者模拟蚂蚁的行为在网络上发现目标服务,后者管理一个由信息素和跳数组成的服务路由表,用以指导 Search Agent 的行进路线.动态变化的信息素可以让 Search Agent 感知到网络拓扑和服务资源的变化,而跳数可以让它们了解距离.路由选择中还使用语义相似度作为启发因子,用于提高召回率.Search Agent 生命周期控制机制

* Supported by the National Natural Science Foundation of China under Grant Nos.60773103, 60903161, 60903162, 90912002 (国家自然科学基金); the National Basic Research Program of China under Grant No.2010CB328104 (国家重点基础研究发展计划(973)); the Specialized Research Fund for the Doctoral Program of Higher Education of China under Grant No.200802860031 (高等学校博士学科点专项科研基金); the Jiangsu Provincial Natural Science Foundation of China under Grant Nos.BK2007708, BK2008030 (江苏省自然科学基金); the Jiangsu Provincial Key Laboratory of Network and Information Security of China under Grant No.BM2003201 (江苏省网络与信息安全重点实验室); the Key Laboratory of Computer Network and Information Integration (Southeast University), Ministry of Education of China under Grant No.93K-9 (东南大学计算机网络和信息集成教育部重点实验室)

Received 2008-09-22; Revised 2008-12-18; Accepted 2009-06-01

使查询流量负载成为可控的,并具有确定的上界.实验结果表明,该方法在大规模的分布式计算环境下具有良好的可扩展性和动态环境下的适应性.

关键词: 服务发现;peer-to-peer 网络;agent;蚁群算法;路由机制

中图法分类号: TP311 文献标识码: A

在服务计算环境下,服务是组成分布式异构系统的基本元素.为了使服务请求者可以迅速而有效地发现并定位服务,服务提供者要将自己提供的服务发布到一个服务注册库中.这个注册库记录了服务的功能、接口、参数和提供者等信息.服务请求者只要到注册库中即可寻找满足功能需要的服务.注册库是服务请求者和服务提供者之间的桥梁,服务查找和发现性能的好坏是影响整个面向服务应用系统性能的关键.

国内外的研究已经相继提出了多种基于 P2P(peer-to-peer)网络的分布式服务注册和发现模式[1−5].这些模式将注册系统划分为若干个相对独立的注册节点.注册节点之间以 P2P 方式互连,并可以共享数据.分布式的服务发现通常包括两个问题[1]:一是如何迅速而有效地找到目标服务所在的注册节点;二是在注册节点上查找目标服务.后面一个问题可以用集中式服务发现的研究成果加以解决,本文只研究前面一个问题.对于此问题,以往的研究只关心召回率、响应时间等基本性能.解决方法一般是根据服务的内容,将服务按照一定的规则或映射关系存储在注册节点中,服务的内容和存储位置有一定的内在关系.在查询服务时,可以按照事先约定的规则或映射关系定位服务所在的注册节点.这种方法的组织结构复杂,发现机制简单,可以达到较短的响应时间和较高的召回率.但是在动态性较强的开放的服务计算环境下,具有一定自主性的注册节点的加入和退出会变得频繁和不确定.过分强调系统组织结构的这类方法,将导致结构的重新组织,计算开销大,可扩展性弱,适应性差.然而,系统的可扩展性和适应性是在大规模的、动态的分布式计算环境中必须解决的挑战性问题.

本文提出一种面向大规模、动态、开放的服务计算环境下的分布式服务发现机制.该机制可以保障用户有效发现服务并具有可扩展性和适应性.首先,提出一个基于非结构化 P2P 网络的服务注册系统模型.注册系统中的每个注册节点都独自按照服务行业划分和管理服务,为了消除歧义,行业按照一个共同的服务行业分类本体命名.其次,提出了基于 Agent 的服务发现技术,并用蚁群算法的思想指导 Agent 的行为.具体来说,就是在 P2P 注册系统上派出多个自主执行查找任务的 Agent.在每一个注册节点,Agent 根据所要查找的目标服务所属行业、相应路径上的信息素(pheromone)浓度以及距离目标服务行业的跳数来选择路径.这是一种非直接的协作方式,不需要 Agent 之间直接通信或传递关于路径的消息,不但可以避免盲目寻找服务,而且可以减少网络负载,节省网络带宽,有利于系统扩展.路径上的信息素由先前经过的 Agent 留下并积累,它可以指导后来的 Agent 去搜索服务.当有新的节点或资源加入到系统中时,也要向邻居节点扩散信息素.信息素具有挥发机制,那些长时间没有得到增加的信息素最终会挥发完.依靠信息素更新机制和动态变化的信息素浓度,Agent 可以自主适应系统拓扑结构和注册中心服务资源的变化.需要说明的是,本文提出的机制和相关算法并不适用于结构化 P2P 网络.

本文第 1 节介绍相关研究工作.第 2 节介绍分布式服务注册系统体系结构,给出服务注册节点的组织方式.第 3 节介绍基于 Agent 的服务发现机制,给出 Agent 的行为以及基于蚁群算法的 Agent 路由与服务发现机制.第 4 节对本文提出的算法进行仿真实验和性能分析.第 5 节总结全文并给出下一阶段的工作.

1 相关工作

在分布式服务注册与发现的研究中,目前广泛采用 P2P 结构作为服务注册系统内部的组织方式,但在具体的服务发现机制和研究侧重点上有所不同.

METEOR-S[1]是一个支持语义 Web 服务发布与发现的基于 P2P 的分布式 UDDI(universal description, discovery and integration).它使用一个共享的注册本体对各个 UDDI 服务器按照服务领域进行分类,每个 UDDI 服务器必须映射到注册本体上的 1 个或多个结点上.UDDI 服务器和注册本体之间的映射关系用于组织 P2P 网络的拓扑结构.在这种机制下,UDDI 服务器的内容和系统拓扑结构紧耦合,存在注册本体和服务注册信息同步的问题,自适应的能力较差.而且,由于注册本体的维护是另外一套语义维护体系,增加了大规模环境下的实现

和使用的难度.文献[3]提出了基于传统单层 P2P 网络的分布式服务发现机制.它使用 DNS 服务来登记和管理对等结点的信息,并设计了一个称为 P2P Registry 的 Agent 作为 P2P 网络和 DNS 之间的中间件.在 Agent 的帮助下,每个对等节点能够通过 DNS 注册和发现目标服务.文中提出了用 Agent 实现服务自动注册和发现,但是没有充分发挥 Agent 自主性和自适应性的特点,还要依靠一个集中式的 DNS 来定位目标.这个 DNS 类似于混合 P2P 结构中的超级节点.文献[4]提出基于本体社区的双层 P2P 结构的服务发现模型.本体社区是注册中心的集合,这些注册中心的内容是有限制的,要根据模块化本体中的每一个模块设立注册中心,多个社区之间再通过相同的本体建立连接.因此,这种模型的结构是与本体模型结构相关的.服务发现时,先在社区内采用洪泛式的消息传递,然后在社区之间进行一步转发的消息转发方式,以实现跨社区的服务发现.该模型的特点是根据本体概念之间的关系建立分层的 P2P 网络,可以适应多本体共存的环境.遗憾的是,实验中注册中心和服务的数量都比较少,不能体现出所提出的模型和相关算法在较大规模分布式环境下的效率如何.与以上的研究不同的是,本文的工作面向大规模的分布式服务发现环境,更加强调系统的可扩展性和适应性.

为了减少服务发现的响应时间,已有一些研究建议对注册库中的服务按类别进行组织和管理.这样可以直接在与服务相关的领域或行业中搜索,缩小了搜索范围.UDDI 提供了服务分类机制[6],包括北美行业分类系统、通用标准产品和服务分类法等,但这些分类标准并不统一而且较为粗糙,大都依靠服务的领域或地域进行划分,没有用本体方法统一分类中的概念,不支持语义发现.文献[7]提出按照 GICS 全球行业分类标准建立 adUDDI.一个 adUDDI 中可以注册多个行业的服务,所有 adUDDI 的信息注册在一个根注册中心.具有相同行业的 adUDDI 相互连接,组织成一个虚拟的区域.与本文的方法相比,这种方法没有考虑行业分类中的概念之间的层次关系,即子行业和父行业之间的关系.没有建立本体库,因此并不支持语义查询.此外,GICS 是投资型分类标准,只考虑了以赢利为目的的经营活动单位,行政事业单位、社会团体、各种组织协会等是不属于其分类对象范围的.因此,为了获得更全面的划分,本文采用管理型的分类标准,如中国的国民经济行业分类标准.另外,本文建立了行业分类本体,可以在语义上比较行业之间的相似度,提高了召回率.

本文运用蚁群算法的思想指导 Agent 的服务发现行为.蚁群算法已经成功地应用在网络路由选择[8]、负载均衡[9]、分布式 QoS 路由[10]、移动 Agent 迁移[11]等问题中.文献[12]通过在动态的、不可靠的和大规模的网络环境中模拟生物系统的一些特性,如免疫机制(immunity)、药物趋化现象(chemotaxis)等,来设计自适应、鲁棒的分布式算法,并总结了在非结构化 P2P 网络中的文件查找、移动自组织网络的路由、负载平衡等方面的应用.Anthill[13]是一个用于构建 P2P 应用的开发环境和工具.它借用多 Agent 和进化计算技术,利用称为蚂蚁的移动 Agent 在网络上漫游并执行任务.由于 Anthill 仅仅是一个开发环境,因此并没有提出具体的蚂蚁的路由算法.

2 服务注册系统体系结构

图 1 显示了本文提出的基于非结构化 P2P 的服务注册系统的体系结构.图中每个注册节点都是服务注册系统中的对等节点.它们以随机方式互连,组成一个松散耦合的覆盖网络(overlay).每个注册节点的逻辑结构由 3 个模块组成,即通信模块、服务路由管理模块和服务注册模块.通信模块实现注册节点之间按照非结构化 P2P 方式互连和通信,注册节点的加入和退出遵循非结构化 P2P 网络的算法.这部分利用已有的研究成果实现,本文不再讨论.服务路由管理模块维护一个服务路由表,表中记录了从本地到目标服务行业的路由信息.这个服务路由表由一个称为 Guide Agent 的 Agent 来维护.服务注册模块的主体是注册库,它包含注册 Web 服务的元数据信息文件.为了在分布式的注册系统上发现目标服务,设计了一个可以在系统中自主漫游的移动 Agent(mobile agent),这个 Agent 称为 Search Agent.Search Agent 具有生命周期,它接受查询任务,负责在服务注册系统中查找目标服务.

在服务计算环境下,服务是面向某个具体服务行业领域的.例如,租车服务属于交通运输业,而酒店预订服务属于旅游业.因此,服务注册节点可以对注册的服务按照所属行业领域进行划分、归类和管理.为了统一并确定共同认可的服务行业分类的概念和术语,有必要建立服务行业分类本体.每个注册节点都根据共同的服务行业分类本体来管理和聚集服务.本文按照国民经济行业分类标准(GB/T 4754-2002)[14]建立服务行业分类本体.

由于本体构建是一个复杂的工程问题,本文不再讨论如何建立一个行业分类本体.图 2 给出了一个服务行业分类本体的片段.根据行业分类标准的特点,该本体是一个由概念组成的树型层次结构,概念之间的层次关系是 subClassOf 关系,表示父类和子类的继承关系.值得注意的是,对于按照其他行业分类标准建立的本体,本文的模型和算法也同样适用.

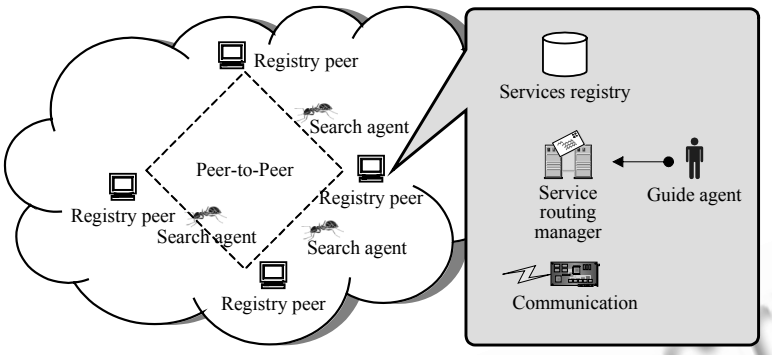


Fig.1 Architecture of a distributed services registry system

图 1 分布式服务注册系统体系结构

一个用服务行业分类本体进行语义标注的 Web 服务可以表示为 $WS(I)$,I 为服务归属的行业集合(这里忽略了 Web 服务的输入、输出等).图 3 是一个服务注册模块的组织结构,它包括一个行业分类本体实例和服务池.行业分类本体实例是一个树状目录结构,表示了当前注册节点中 Web 服务的行业分类情况.服务池中存储了 Web 服务的元信息.一个 Web 服务可以属于 1 个或多个行业.如,$WS_1(si_i,si_j)$表示该服务可以属于 si_i 和 si_j 两个行业.为了加快查询速度,可以利用索引技术.索引技术可以参考信息检索中的成熟算法,本文不再深入研究.

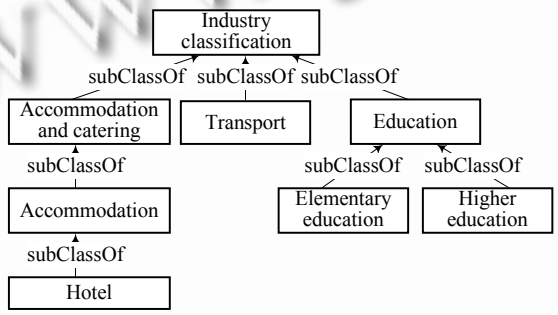


Fig.2 Sample service industry classification ontology

图 2 服务行业分类本体例子

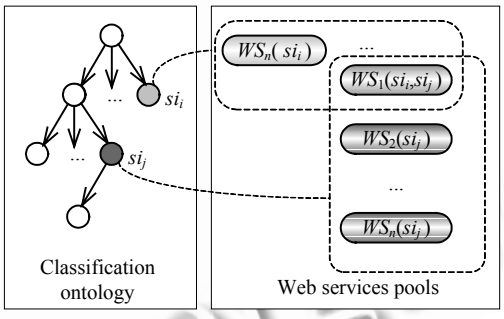


Fig.3 Organization of service registry module

图 3 服务注册模块组织结构

3 基于蚁群算法的 Agent 服务发现

基于真实蚂蚁行为而提出的蚁群算法起初被用于求解 TSP 问题(traveling salesman problem)[15].在该算法中,多个人工蚂蚁被放置在图的多个顶点上.它们模仿真实蚂蚁的行为去寻找最短周游路径.蚁群算法的精髓是利用在路径上保留的信息素来在蚂蚁之间传递关于路径的信息,而这些信息素又是以前蚂蚁走过后散布下的,代表了一定的先验知识.在算法中,信息素全局更新的正反馈机制缩小了搜索范围,其隐含的负反馈机制又保持了搜索范围.受到一些分布式系统中生物群体智能研究[8−12]的启发,本文运用蚁群算法解决分布式服务注册系统中的服务发现问题.对于实际应用中的每一次 Web 服务查询请求,系统产生 n 个 Search Agent 模拟蚂蚁的行为去执行查询任务.分布式的服务注册系统中,一般会有不同组织注册的多个功能相同而 QoS 不同的服务,服务请求者需要尽可能地得到所有这些服务的信息,以便选择 QoS 最适合的服务.Search Agent 的目标就是在一定

的约束条件下尽可能多地找到 QoS 不同的目标服务信息,并提交给请求者.

3.1 Agent的行为

Search Agent 在注册系统 P2P 网络上漫游并查找目标服务.它包含了目标服务(target service)、生命值(time to live,简称 TTL)、跳数(hop count)、禁忌表(tabu)等属性变量.生命值表示 Agent 可以存活的时间,跳数记录了最近一次发现目标服务后,该 Agent 经过的注册节点(以下简称节点)数,禁忌表中记录了它经过的节点,防止节点重复访问,也记录它的访问线路.Search Agent 具有以下行为(涉及到的算法将在下一节加以介绍):

(1) 漫游.为了在网络上查询到请求的目标服务,Search Agent 必须按照一定的路由机制在节点之间移动.网络中的每个节点都维护了一个服务路由表.Search Agent 根据服务路由表的信息选择一个邻居节点作为下一跳节点.如图 4 所示,服务路由表记录了从本地到目标服务行业的信息,它包括目标服务行业 Industry、下一跳节点地址 Next hop、到 Industry 的跳数 Hop count 以及信息素浓度 Pheromone 这 4 个字段.例如,图 4 的路由表中的第 1 条记录表示如果目标服务属于行业 si_i,就可以选择 h_i 作为下一个路由节点,从本地到 si_i 所在节点的距离(跳数)为 hc_i,这条出口上的信息素浓度为 ph_i.同时,它将经过的节点都登记到自己的禁忌表中,然后更新自己的跳数属性.

(2) 查询服务.它在到达的每一个节点上查询目标服务.节点有关于注册服务的语义描述和 QoS 属性,所以可以采用语义匹配技术来查询.

(3) 发送信息素更新消息.消息包括服务行业字段和跳数字段.在两种情况下发送更新消息.如果查询成功,就向所有邻居节点的 Guide Agent 群发更新消息,报告在此节点上可以找到一个目标服务,跳数为 1.无论查询成功与否,如果自己的跳数不为 0,则都要将跳数封装到一个更新消息中,并发送给当前节点上的 Guide Agent.

(4) 生命周期控制.Agent 具有生命周期,当它被创建时,就被赋予一个初始生命值,此后在漫游时,生命值根据一定的规则更新.这样就可以控制在网络上传播的 Agent 的数量,并确保查询请求结束后不会再有其产生的 Agent 还在网络上不停地漫游.此外,通过调节生命值的初始值,可以增加或减少搜索半径.由于 Search Agent 在生命周期结束时会自行销毁,这种机制实现了生命周期的自管理,系统不必管理它的整个生命周期.

Industry	Next hop	Hop count	Pheromone
si_i	h_i	hc_i	ph_i
si_i	h_j	hc_j	ph_j
si_j	h_k	hc_k	ph_k

Fig.4 Example of service routing table

图 4 服务路由表

Guide Agent 主要维护服务路由表,它具有以下行为:

(1) 监听并接收 Search Agent 的消息.Guide Agent 一直监听是否有来自 Search Agent 或邻居节点上 Guide Agent 的信息素更新消息.

(2) 服务路由表管理.分析接收到的更新消息,如果路由表中存在此类服务行业,就增加相应表项的信息素浓度,更新相应表项的跳数;如果不存在此类服务行业,则增加一个新的表项.Guide Agent 根据收到的信息素更新消息中包含的跳数来替代当前表项中的跳数值.由于注册库中的服务有可能发生变化,路由表需要定时更新并减少信息素.那些长时间没有得到增加的信息素最终会挥发完,表示其相应的路径上可能已经不存在相应的服务注册信息,或者没有查找此类服务的需求.如果某个表项的信息素浓度为 0,就删除该表项.

(3) 扩散信息素更新消息.当所在的节点加入注册系统后,或者有新的服务注册成功后,向所有邻居节点扩散信息素更新消息.消息中的跳数设置为 1.

3.2 Agent服务发现机制

下面从 Search Agent 的路由机制、信息素的产生和更新以及 Search Agent 的跳数更新、生命周期控制这

4 个方面进行讨论.不失一般性,假设 Search Agent 要查询的目标服务只属于 1 个服务行业.

3.2.1 Search Agent 的路由机制

Search Agent 查询当前节点的服务路由表.首先,根据服务行业分类本体确定目标服务的行业都和表中的哪些行业语义相似;然后再根据匹配的结果,从多个符合语义相似条件的路由中选择出适当的路由.

(1) 服务行业的语义匹配

本体概念间的语义相似度计算是服务行业语义匹配的基础.目前提出了很多种概念间语义相似度计算方法,但本文中的服务行业分类本体是一个树型结构,所以采用基于树的计算方法.这类方法利用两个概念在概念树中的最短路径、概念在概念树中的层高以及所在层高的节点密度等作为度量指标.文献[16]提出的两个概念 C_1,C_2 间语义相似度的计算公式为

$$Sim(C_1,C_2)=\mathrm{e}^{-al}\cdot\frac{\mathrm{e}^{\beta h}-\mathrm{e}^{-\beta h}}{\mathrm{e}^{\beta h}+\mathrm{e}^{-\beta h}} \tag{1}$$

其中,l 表示两个概念间的最短路径的长度,h 表示两个概念在树中最近的相同父辈概念在树中的高度,参数 α,$\beta\geqslant 0$ 是调节因子.本文采用该计算方法作为两个行业本体概念间语义相似度的计算方法.给定一个语义相似度阈值 $\xi\in(0,1)$,在当前节点 i 上的服务路由表 RT_i 中的集合 I 就是筛选出的与目标服务行业 si_k 语义相似的行业集合.I 可以表示为

$$I=\{si\mid Sim(si_k,si)>\xi\wedge si\in\pi_{Industry}(RT_i)\} \tag{2}$$

其中,$\pi_{Industry}(RT_i)$表示 RT_i 在 $Industry$ 上的投影.

(2) 路由选择

在服务路由表中选择路由时,会出现两种情况:一是要查询的目标服务所属行业在语义上不与当前服务路由表中任何一个目标服务行业匹配,这时 Search Agent 就在所有邻居节点中随机选择一个;二是找到 1 个或多个目标服务行业可以匹配,这时 Search Agent 要根据语义相似度以及表项中相应的信息素值和跳数来确定选择哪一个邻居.下面讨论第 2 种情况下的路由选择机制.

信息素是用蚁群算法思想求解问题的核心因素,它代表了一定的先验知识,其大小代表了在相应的路径上查找成功次数的多少.本文算法还引入两个启发因子,即语义相似度和跳数.语义相似度表示了路由表中的目标服务行业与 Search Agent 的目标服务所属行业的相似程度,跳数代表了与目标服务行业的距离.Search Agent 漫游到语义相似度大的行业上查询的准确率可能会提高,漫游到信息素大的邻居节点后的查找成功率可能要高一些,而选择跳数小的可能响应时间较短.因此,要综合考虑这 3 个参数.

假设当前节点为 i,Search Agent k 在第 2 种情况下以如下概率(公式(3))选择邻居节点 j:

$$p_k(i,j)=\begin{cases}\dfrac{ph(i,j)(1/hops(i,j))^{\lambda}Sim(si_k,I_j)}{\sum_{u\in N}ph(i,u)(1/hops(i,u))^{\lambda}Sim(si_k,I_u)}, & j\in N\\ 0, & j\notin N\end{cases} \tag{3}$$

$$N=\{n\mid(si,n)\in\pi_{Industry,Nexthop}(RT_i)\wedge si\in I\}-Tabu(k).$$

其中,$ph(i,j)$表示节点 i 到节点 j 上的信息素值;$hops(i,j)$表示以节点 j 为下一跳时,到达目标服务行业的跳数;$\lambda>0$ 是调节因子,调节信息素和跳数在路由选择中的比重;RT_i 表示节点 i 上的服务路由表;I_x 表示 RT_i 中下一跳节点 x 所对应的行业;si_k 表示 Search Agent k 要查询的目标服务的行业;$Sim(si_k,I_x)$为语义相似度(可用公式(1)计算).$\pi_{Industry,Nexthop}(RT_i)$表示 RT_i 在$(Industry,Nexthop)$上的投影,I 的定义如公式(2).$Tabu(k)$是 Search Agent k 的禁忌表.

在服务路由表中,可能会出现多个不同行业对应同一个下一跳节点的情况.因此,为了确保在集合 N 中不会删除重复的节点,可以首先对其进行预处理.方法是对于 m 个重复的节点 n,用 m 个不同的记号 $n_1,n_2,\ldots,n_m$ 标记.

3.2.2 信息素的产生和更新

信息素在 Search Agent 发现服务的过程中发挥了重要的作用,它引导 Search Agent 的路由方向.因此,如何产生和更新信息素是影响本文算法性能好坏的关键.在以下几种情况下会产生或更新信息素:

(1) 已经找到目标服务的 Search Agent 在所经过的路径上会留下信息素.它产生的新的信息素会累加到原来该路径已有的信息素上.设 Search Agent 从节点 n 进入本地节点 i,$ph(i,n)$表示从 i 到邻居节点 n 的路径上的信息素值,则有

$$ph(i,n)=\alpha\cdot ph(i,n)+(1-\alpha)\Delta p_1 \tag{4}$$

其中,$\alpha\in(0,1)$,Δp_1 为常数.

(2) 当在节点 n 上找到目标服务时,它还会向节点 n 的所有邻居节点扩散信息素.邻居节点上的 Guide Agent 在收到信息素更新消息后,会在自己服务路由表的相应表项上更新信息素.即

$$ph(m,n)=\beta\cdot ph(m,n)+(1-\beta)\Delta p_2, m\in J(n) \tag{5}$$

其中,$\beta\in(0,1)$,Δp_2 为常数,$J(n)$是节点 n 的邻居集合.

(3) 对于服务路由表上的每一项,都会定期作如下更新:

$$ph(i,n)=\rho\cdot ph(i,n) \tag{6}$$

其中,$\rho\in(0,1)$是挥发系数.这样,如果某个邻居长时间始终没有被访问,则它的信息素将会接近为 0.这样的节点上可能注册的服务较少,或者已经退出了网络.

(4) 当有新的节点加入网络后,它会向其所有邻居发送信息素更新消息.按照公式(5)的规则来更新邻居的服务路由表.

(5) 当某个节点上有新的服务注册时,它会向其所有邻居发送信息素更新消息.按照公式(5)的规则来更新邻居的服务路由表.

3.2.3 Search Agent 跳数的更新

Search Agent 跳数值初始为 0,表示目前还没有发现目标服务.如果第 1 次发现,就将跳数设置为 1.此后,如果在沿途节点上继续发现目标服务,就重置为 1;否则就做加 1 操作.因此,Search Agent 在查找服务的同时,还可以记录最近一次发现到的服务距离当前位置的跳数,这个跳数将被包含在信息素更新消息中,用于更新服务路由表.Search Agent k 在节点 i 上时,其跳数 HC_k 的更新规则为

$$HC_k=\begin{cases}0, & HC_k=0\wedge si_k\notin C(i)\\ 1, & si_k\in C(i)\\ HC_k+1, & HC_k>0\wedge si_k\notin C(i)\end{cases} \tag{7}$$

其中,si_k 表示 Search Agent k 要查找的服务所属的行业,$C(i)$表示节点 i 上的服务集合.

3.2.4 生命周期控制

本算法使用 TTL 控制 Search Agent 的生命周期.在 Search Agent 产生时,它的 TTL 会被设置一个初值.在经过的每个节点,TTL 值都会被更新.如果在当前节点上没有找到目标服务,TTL 会减少;如果找到了,则不对 TTL 值处理.这样可以使这个 Agent 再去查找更多的节点.如果当前节点的所有邻居节点都在 Agent 的禁忌表中,则它的 TTL 值被强制为 0.TTL 值为 0 的 Agent 就不再漫游,并被销毁.Search Agent k 在节点 i 上时,其生命值 TTL_k 的更新规则为

$$TTL_k=\begin{cases}TTL_k-1, & si_k\notin C(i)\\ TTL_k, & si_k\in C(i)\\ 0, & J(i)\subseteq Tabu(k)\end{cases} \tag{8}$$

其中,si_k 和 $C(i)$同公式(7),$J(i)$是节点 i 的邻居集合.

基于以上的讨论,下面分别给出服务路由表更新算法和路由选择算法的伪代码.

算法 1. 服务路由表更新.Guide Agent 在收到信息素更新消息后,在上面提到的情况(2)、情况(4)、情况(5)这 3 种情况下运用本算法.情况(1)、情况(3)这两种情况下的服务路由表更新算法与此类似.

1: **Input**: A service routing table n_1.*SrvRoutingTab* of a local node n_1, a neighbor node n_2,
a pheromone update message *msg*, a pheromone increment *ph*, a const *beta*;

2: **IF** (*msg.Industry*,n_2) in n_1.*SrvRoutingTab* **THEN**

3: n_1.*SrvRoutingTab*[(*msg.Industry*,n_2)].*ph*=*beta*×n_1.*SrvRoutingTab*[(*msg.Industry*,n_2)].*ph*+(1−*beta*)×*ph*;
4: n_1.*SrvRoutingTab*[(*msg.Industry*,n_2)].*hc*=*msg.hc*;
5: **ELSE**
6: Add (*msg.Industry*,n_2,*msg.hc*,*ph*) to n_1.*SrvRoutingTab*;

算法 2. 路由选择算法.Search Agent 在每个节点执行此算法,执行查询任务和选择下一个访问节点的任务.

1: **Input**: A Seach Agent *SA*, a target service *WS*, a local node *node*, pheromone increments ph_1 and ph_2;
2: *SendUpdatePheromoneMessage* to *node* with *SA.HopCount* and ph_1;
3: *SA.UpdateHopCount*;
4: *SA.Tabu.Add*(*node*);
5: **IF** Agents having the same task had arrived **THEN**
6: *SA.TTL*−−;
7: **ELSE**
8: *SA.Query*(*WS*);
9: **IF** (NoFound) **THEN**
10: *SA.TTL*−−;
11: **ELSE**
12: **FOREACH** $node_i$ **IN** *node.neighbour*
13: *SendUpdatePheromoneMessage* to $node_i$ with *msg.hc*=1 and ph_2;
14: **IF** All *node.neighbour* in *SA.Tabu* **THEN**
15: *SA.TTL*=0;
16: **ELSE**
17: *SA.Nexthop*=*SelectingRoute*(*node*);

4 实验分析

下面用仿真实验来评估算法的性能.假设在注册节点上可以准确地找到所有目标服务,实验仅重点考察算法在系统中查找到目标服务所在注册节点的性能.

4.1 实验环境

仿真实验工具为 Repast 3(recursive porous agent simulation toolkit)[17].这是一个可以模拟大规模系统的开源的多 Agent 仿真平台.仿真程序用 C#开发,运行的机器为 Pentium 4 3.0GHz CPU,2G 内存.仿真软件模拟了网络结构符合 Power-law 规则[18]的非结构化 P2P 服务注册系统,每个注册节点可以有最多 20 种类型的服务行业,每个行业最多有 50 种类型的服务,每个注册中心随机产生 200 个注册服务,服务路由表的长度为 256.基于蚁群算法的机制都包含大量的调节参数.然而到目前为止,这些参数的取值还没有科学的严格的理论指导[19,20],需要经过多次实验确定优化的参数.根据文献[16,20]中的建议和反复的实验,公式(1)的参数设置为α=0.2,β=0.6.本文算法的参数设置为α=0.9,β=0.85,ρ=0.95,Δp_1=0.25,Δp_2=0.35,ξ=0.5.λ参数决定了信息素和跳数哪一个对路由选择的影响大一些.在拓扑结构和注册服务信息基本不变的静态环境下,跳数可以比较真实地反映服务信息的注册位置;而在动态环境下,信息素的更新机制发挥了适应环境变化的作用,所以在静态实验环境下设置λ=0.5,放大跳数值的影响,动态实验环境下设置λ=2,缩小其值.仿真实验基于离散时间模型,所有事件在确定的离散时间步上执行.这样的时间步称为一个 Tick.以所有 Search Agent 产生直到它们生命全部终止为一个查找轮次(round).

为了分析算法的性能,我们分别做了 3 组实验.第 1 组实验评估本文算法的性能以及可扩展性和适应性,第 2 组实验将本文算法与另两种经典的 P2P 资源发现算法进行比较,第 3 组实验讨论本文算法的参数对性能的影响.本文定义并使用下列度量指标:

- 召回率(recall,也称作查全率)是指查找到的目标服务数量和总的目标服务数量的比例.它反映了算法的查找能力.这是服务发现算法的基本性能指标.
- 查询流量负载(query traffic load)是指对于一个查询请求,网络所有节点接收到的查询消息的总数量.它反映了算法消耗的网络带宽资源,可以衡量在分布式服务注册网络中服务发现算法花费的代价.
- 查找性价比(search performance-price ratio)是指查找到的目标服务数量和查询消息数量的比例.它反映了平均每个消息可以找到的目标服务的数量,表现出算法的性价比.

4.2 算法性能分析

4.2.1 基本性能评估

首先评估网络中注册中心数、拓扑结构以及服务种类和数量不发生变化的静态环境下的算法基本性能.图 5 是在一个缺少信息素的初始环境下,不断迭代查找同一个目标服务后,算法的召回率变化趋势以及不同轮次下的召回率的比较.系统中有 1 000 个注册中心,随机产生 185 个目标服务,并派遣 15 个 Search Agent 执行发现任务,它们的生命初值都为 20.在第 1 轮的查找过程中,Search Agent 的查找效率是很低的,当所有 Agent 都完成查找任务生命终止时,召回率大约为 25%.但在第 2 轮查找时就可以在很短的时间步内达到较高的召回率.此后,在第 10 轮查找时则可以达到很高的召回率,大约为 82%.在多轮的迭代过程中,从起初缺乏信息素的较盲目搜索到信息素发挥作用后的启发搜索,召回率在不断地提高,Search Agent 的生命周期得以延长(因为查找成功时,生命值不衰减,参考公式(8)).图 5 的实验结果表明,本文算法的收敛速度较快,可以在较短的时间内达到较高的召回率.

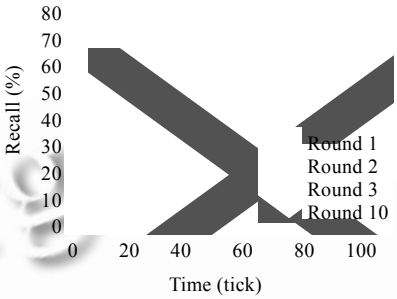


Fig.5 Comparison between recalls of various rounds

图 5 多轮迭代时的召回率比较

4.2.2 可扩展性评估

可扩展性是指算法适应系统不同规模的能力.特别是在大规模的系统环境下,算法的基本性能是否会降低是可扩展性的重要标志.这里考察系统规模对召回率的影响、提高召回率的方法以及为了达到一定的召回率会产生多少查询消息负载.

图 6 显示了不同系统规模对算法召回率的影响.在 Search Agent(SA)的数量(等于 20)和初始生命值(等于 20)不变的情况下,在 15 轮搜索后,随着注册节点数量的剧增(从 500 到 8 000 的规模),召回率下降的速度比较缓慢,呈现平稳适度下降的态势.由于 Search Agent 的数量和初始生命值都没有改变,因而随着网络规模的激增,这种下降的趋势是合理的.Search Agent 的查找能力与自身的数量、初始生命值紧密相关,数量影响查找的范围,而初始生命值与搜索路径长度有关.图 7 显示了在注册节点数量为 5 000 时,不同 Search Agent 数量和初始生命值大小对算法召回率的影响.数量的增加提高了查找的范围,而初始生命值的增加又提高了搜索路径的长度,所以可以通过提高数量和初始生命值来提高召回率.特别是当系统规模增大时,这种提高是必要的.

定理 1. 在本文算法中,查询流量负载 L_q 不大于 n 个 Search Agent 初始生命值 *InitialTTL* 的总数与系统中目标服务总数 Num_{ts} 的和,即

$$L_q \leqslant n \times InitialTTL + Num_{ts}.$$

证明:假设对于一次查询请求,系统产生 n 个具有相同 *InitialTTL* 值的 Search Agent.对于一个 Search Agent i,它产生的查询流量负载等于它在整个生命周期中访问过的节点数,而这是由生命值 TTL_i 决定的.根据公式(8),如果 Search Agent 在某个节点上找到一个目标服务,它的 TTL_i 不变化,这等于生命值增加了.因此,设ΔTTL_i 是第 i 个 Search Agent 的生命增量,则有

$$L_q = \sum_{i=1}^{n}(InitialTTL + \Delta TTL_i) = n \times InitialTTL + \sum_{i=1}^{n}\Delta TTL_i.$$

生命增量ΔTTL_i是在查询过程中动态变化的,与 Search Agent i 的查找过程及生命周期控制策略(公式(8))有

关,只有当 Search Agent i 的生命周期结束时才可以准确获得.根据公式(8),TTL_i 只有在发现目标服务时才会保持不变,相当于生命值增加了 1 个单位,所以生命增量总数最多等于发现的目标服务总数.当召回率最大达到100%时,$\sum_{i=1}^{n}\Delta TTL_i = Num_{ts}$. 在服务发现中,某些 Search Agent 可能由于邻居节点都在禁忌表中而终止生命,而且召回率≤100%,所以可得 $\sum_{i=1}^{n}\Delta TTL_i \leqslant Num_{ts}$. 进而 $L_q \leqslant n\times InitialTTL+Num_{ts}$. □

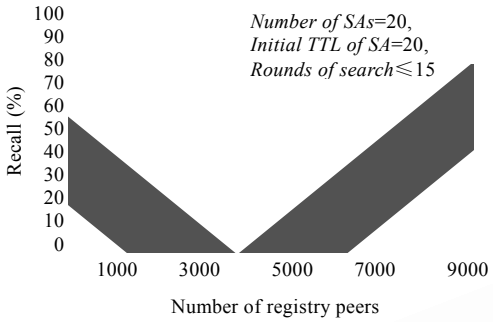


Fig.6 Recall vs. the number of registry peers

图 6 不同数量注册节点时的召回率

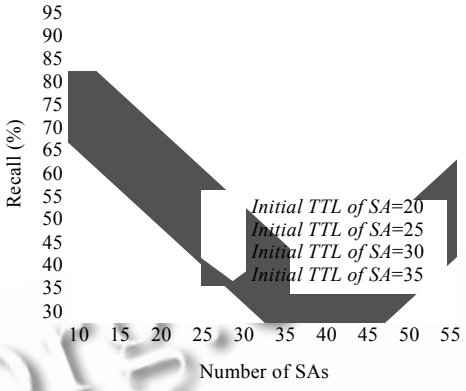


Fig.7 Recall vs. the number of SAs

图 7 不同数量 Search Agent 时的召回率

如上所述,在网络规模增大时,为了达到一定的召回率,必须增加 Search Agent 的数量和初始生命值.根据定理 1,这势必增加查询流量负载.图 8 表明在不同系统规模下,为了达到 80%的召回率,查询流量负载随系统规模的增加呈线性增加的趋势.通过理论分析和仿真实验可知,L_q 的大小是可控的,可以根据定理 1 计算其最大值,而且为了达到一定的召回率,L_q 必须增加,但其大小是随网络的规模线性增加.

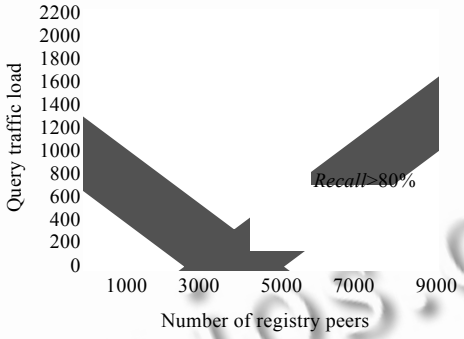


Fig.8 Query traffic load vs. the number of registry peers

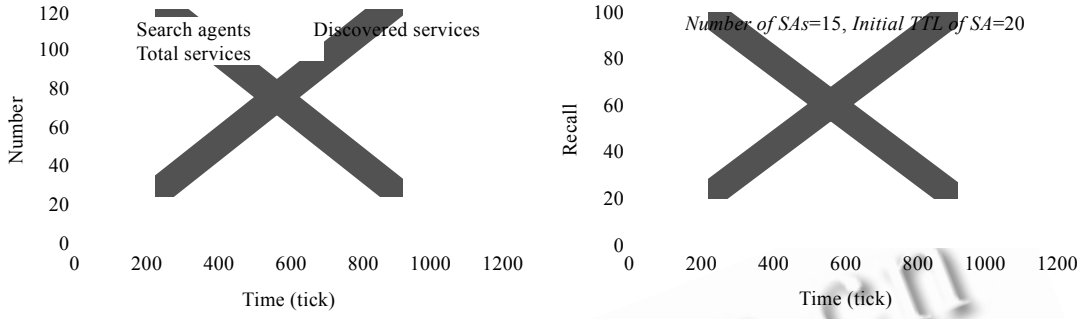
图 8 不同数量注册节点时的查询流量负载

以上理论分析和实验表明,本文算法的可扩展性较好,适应于大规模的服务发现系统.

4.2.3 适应性评估

适应性是指算法对系统结构、外界环境动态变化后的调整能力,调整的结果应该使算法性能可以迅速稳定,并恢复到一定的水平.为了考查算法在节点频繁加入或退出的动态环境下的适应性,实验模拟了一个注册节点数量发生变化的环境.为了获得稳定的统计数据,假设在每一轮查找过程中,节点数量和拓扑结构都不发生变化.实验中,在每一轮查找之间随机增加或减少 10%的注册中心,观察算法召回率的变化.图 9 是仿真软件动态输出的曲线图.此时,网络的节点数为 500.在 6 轮查找后,增加 10%的注册节点.图 9(a)显示了总服务数、已发现的

服务数、Search Agent 数量的变化情况,图 9(b)显示此过程中召回率的变化.可见,注册节点增加后,第 7 轮查找的召回率立刻下降到 85%.这是因为注册节点增加后,不但可能会增加目标服务数量,而且网络拓扑结构也会有变化.这导致 Search Agent 之前积累的信息素、跳数等路由信息可能不再准确,也需要自适应地改变.经过 3 轮查找后,召回率又回到了之前 90%的水平.这是信息素的动态更新机制发挥了作用,算法动态适应了网络拓扑结构的变化.



(a) Process of multi-round search (b) Trend of recall

(a) 多轮查找过程 (b) 召回率变化趋势

Fig.9 Experiments in a dynamic environment

图 9 动态环境下的实验

4.3 与经典算法的比较

Gnutella[21]和 Random Walks[22]是非结构化 P2P 网络中两种经典的资源发现算法,也常被用于 P2P 服务发现机制.我们在提出的服务注册系统中实现了这两种算法指导的 Search Agent 服务查找机制,以便与本文算法作对比.图 10 是在不同系统规模下,3 种算法查询流量负载的比较.图 11 是 3 种算法查找性价比的比较.由于 Gnutella 和 Random Walks 在每个节点都要生成新的 Search Agent,Gnutella 以类似泛洪(flooding)的方式产生,*k*-Random Walks 在每个注册中心都产生 *k*–1 个新的 Search Agent,所以在大规模的环境下,为了达到较高的召回率会产生大量的 Search Agent,因而算法产生的查询流量就很高,而查找性价比就很低.而本文算法在整个查找过程中不再产生新的 Search Agent,所以在减少了查询流量负载的同时,也获得了很高的查找性价比.

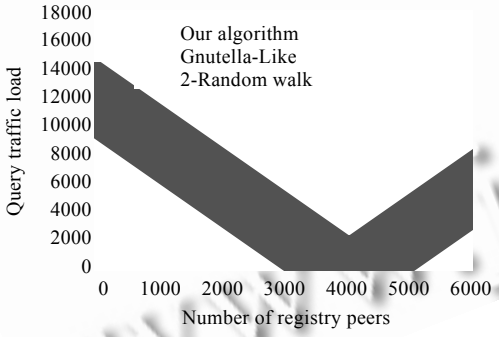


Fig.10 Comparison between three algorithms in query traffic load

图 10 3 种算法查询流量负载的比较

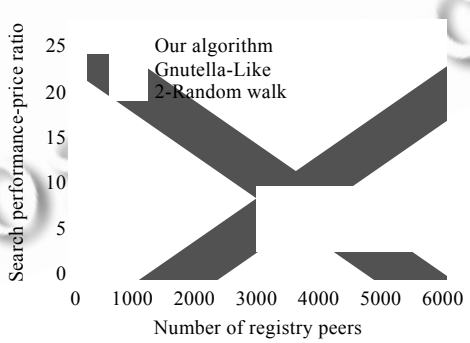


Fig.11 Comparison between three algorithms in search performance-price ratio

图 11 3 种算法查找性价比的比较

再如图 12 所示,在基于信息素的指导下,本文算法在访问了 30%左右的注册节点后就可以获得很高的召回率,而另两种经典算法由于缺少历史信息指导的盲目搜索,其召回率与访问过的节点数有关,基本成等比关系.

对比实验结果表明,本文算法在访问较少的节点后就可以达到较高的召回率,具有更好的查找性能.而且当节点数增加,系统规模扩大时,为了达到较高的召回率,不必增加过多的 Search Agent.这样可以减少网络负载,适合于大规模的网络环境.

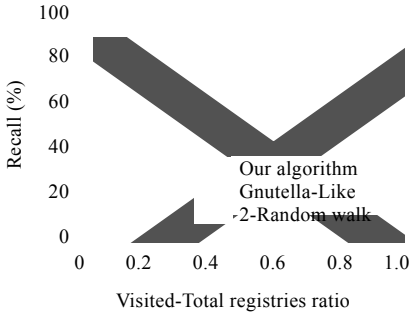


Fig.12 Comparison between three algorithms in recall

图 12 3 种算法召回率的比较

4.4 算法参数对性能的影响

这组实验考察本文算法的参数设置对其性能的影响.在本文算法中,语义相似度阈值ξ、调节因子λ和挥发系数ρ对于算法的召回率、查找时间的影响较大.因此,本节重点讨论这 3 个参数.

图 13 是语义相似度阈值ξ和召回率之间的关系,不同系统规模下的召回率曲线总体趋势相似(图例中的 n 表示注册节点数).当ξ=0.5 时,召回率达到最高;当ξ<0.5 时,召回率有所下降,但还是比较高;但是当ξ>0.5 时,召回率呈下降趋势.这是因为当ξ的值减小时,集合 I(见公式(2))中的元素增多,即服务路由表中可以与目标服务行业语义匹配的服务行业增多.这会扩大 Search Agent 的搜索范围,一些实际上与目标服务行业语义相似度很小的服务行业也有机会被选择到.但同时,由于公式(3)中语义相似度因子的作用,相似度较小的服务行业被选择的概率也较小,召回率的减小变化并不十分明显,还维持在较高水平.当ξ较大时,可供选择的服务行业较少,可能会筛选掉一些与目标服务行业语义相似的服务行业,因此会减少召回率.实验结果表明,当$\xi\approx$0.5 时会达到较好的平衡.

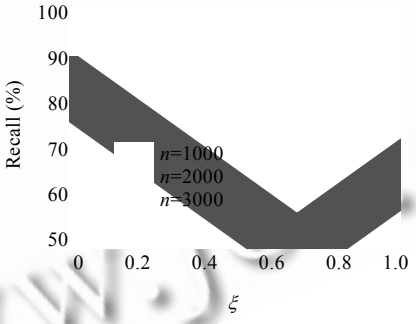


Fig.13 Relation between ξ and recall

图 13 ξ和召回率之间的关系

在公式(3)中,λ越小,越突出跳数在路由选择中的作用,相应地减小信息素的作用.在静态环境下,系统拓扑结构和注册服务基本不变化,跳数可以比较准确地反映出目标服务的距离,选择跳数小的路径可以减少查找时间.因此在这种情况下,λ要取小值.而在动态环境下,跳数并不一定可以准确地反映距离,此时要减小跳数的作用,增加信息素的作用,即增加λ.图 14 反映了不同系统规模下,调节因子λ和查找时间之间的关系(图例中的 n 表示注册节点数).在静态环境下,随着λ的增加,查找时间会不断增加,当λ>0.5 时,显示了一个明显的增长趋势.然而

在动态环境下,随着λ的增大,查找时间减小.当λ>2 后,查找时间趋向较小的稳定值.

图 15 是在不同的挥发系数ρ下,算法的召回率和 Search Agent 的节点访问率之间的关系.在ρ=0.95 时,算法具有最好的查找性能;而在ρ=0.98 时,虽然在节点访问率不大于 10%时有最高的召回率,但在访问更多的节点后,召回率低于ρ=0.95 时的召回率.这是因为ρ决定了信息素的挥发程度,如果挥发慢,则 Search Agent 在搜索路径时容易陷入某些经验路径,难以探索其他路径.虽然在查找初期可以较快找地到目标服务,但减小了找到更多目标服务的可能.如果挥发快,则信息素不能保留积累的经验,信息素的指导作用减小.由图 15 可知,当ρ=0.9 时,召回率低于其他两种信息素挥发较慢时的情况.

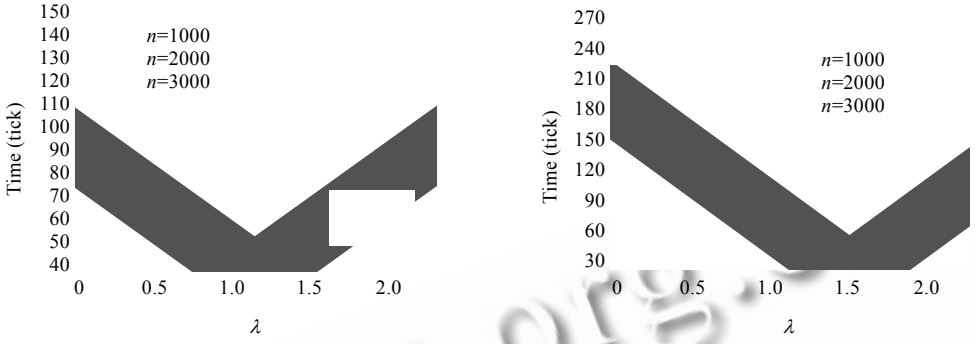


Fig.14 Relation between λ and search time

图 14 λ和查找时间之间的关系

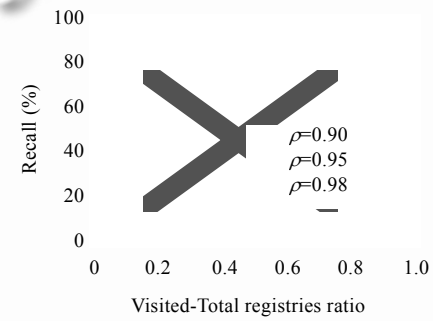


Fig.15 Comparison between different volatilization coefficient ρ in recall

图 15 不同挥发系数ρ下召回率的比较

5 结 论

本文从提高大规模服务计算环境下服务发现机制的可扩展性和适应性角度考虑,提出一个基于 Agent 的服务发现机制.该机制模拟蚁群觅食活动,用两种类型的 Agent 合作完成服务发现任务,Search Agent 模拟蚂蚁的行为,在服务注册系统的网络上寻找服务;Guide Agent 驻留在注册节点,维护一个服务路由表,用来指导 Search Agent 选择路由.仿真平台 Repast 上的实验结果表明,本文提出的机制具有较好的可扩展性和适应性.

本文利用 Agent 和蚁群算法的思想初步解决了大规模服务发现系统的可扩展性和适应性问题.为了说明问题,注册系统自身的结构并不复杂,没有考虑层次化的 P2P 结构.下一步工作也可以研究在分层结构下的问题.此外,P2P 结构的注册系统类似于一个复杂适应系统(complex adaptive system),正在进行的工作是从群体智能的角度研究服务计算环境下的自治和适应性问题.

References:

[1] Verma K, Sivashanmugam K, Sheth A, Patil A, Oundhakar S, Miller J. METEOR-S WSDI: A scalable P2P infrastructure of registries for semantic publication and discovery of Web services. Journal of Information Technology and Management, 2005,6(1): 17−39. [doi: 10.1007/s10799-004-7773-4]

[2] Chen DW, Xu B, Cai YR, Li JZ. A P2P based web service discovery mechanism with bounding deployment and publication. Chinese Journal of Computers, 2005,28(4):615−626 (in Chinese with English abstract).

[3] Chen CW, Gan PS, Yang CH. A service discovery mechanism with load balance issue in decentralized peer-to-peer network. In: Barolli L, ed. Proc. of the 11th Int'l Conf. on Parallel and Distributed Systems (ICPADS 2005). Washington: IEEE Computer Society Press, 2005. 592−598.

[4] Liu ZZ, Wang HM, Zhou B. A two layered P2P model for semantic service discovery. Journal of Software, 2007,18(8):1922−1932 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/18/1922.htm [doi:10.1360/jos181922]

[5] Guo DK, Ren Y, Chen HH, Xue QW, Luo XS. A QoS-guaranteed and distributed model for Web service discovery. Journal of Software, 2006,17(11):2324−2334 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/17/2324.htm [doi:10.1360/jos172324]

[6] Clement L, Hately A, Riegen CV, Rogers T. Universal description discovery & integration (UDDI) 3.0.2. 2004. http://www.uddi.org/pubs/uddi_v3.htm

[7] Du ZX, Huai JP. Research and implementation of an active distributed Web service registry. Journal of Software, 2006,17(3): 454−462 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/17/454.htm [doi:10.1360/jos170454]

[8] Di Caro G, Dorigo M. Antnet: Distributed stigmergetic control for communications networks. Journal of Artificial Intelligence Research, 1998,9:317−365.

[9] Liu JM, Jin XL, Wang YS. Agent-Based load balancing on homogenous minigrids: Macroscopic modeling and characterization. IEEE Trans. on Parallel and Distributed Systems, 2005,16(7):586−598. [doi: 10.1109/TPDS.2005.76]

[10] Xu H, Wu SQ. A distributed QoS routing based on ant algorithm for LEO satellite network. Chinese Journal of Computers, 2007, 30(3):361−367 (in Chinese with English abstract).

[11] Du RH, Yao G, Wu QY. Application of an ant colony algorithm in migration of mobile agent. Journal of Computer Research and Development, 2007,22(2):282−287 (in Chinese with English abstract).

[12] Babaoglu O, Canright G, Deutsch A, Di Caro GA, Ducatelle F, Gambardella LM, Ganguly N, Jelasity M, Montemanni R, Montresor A, Urnes T. Design patterns from biology for distributed computing. ACM Trans. on Autonomous and Adaptive Systems, 2006,1(1):26−66. [doi: 10.1145/1152934.1152937]

[13] Babaoğlu Ö, Meling H, Montresor A. Anthill: A framework for the development of agent-based peer-to-peer systems. In: Rodrigues L, Raynal M, Chen W, eds. Proc. of the 22nd Int'l Conf. on Distributed Computing Systems (ICDCS 2002). Washington: IEEE Computer Society Press, 2002. 15−22.

[14] General Administration of Quality Supervision, Inspection and Quarantine of P.R.C. Industrial Classification for National Economic Activities. GB/T 4754-2002, Beijing: Standards Press of China, 2007 (in Chinese).

[15] Dorigo M, Gambardella LM. Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Trans. on Evolutionary Computation, 1997,1(1):53−66. [doi: 10.1109/4235.585892]

[16] Li YH, Bandar ZA, McLean D. An approach for measuring semantic similarity between words using multiple information sources. IEEE Trans. on Knowledge and Data Engineering, 2003,15(4):871−882. [doi: 10.1109/TKDE.2003.1209005]

[17] North MJ, Collier NT, Vos JR. Experiences creating three implementations of the repast agent modeling toolkit. ACM Trans. on Modeling and Computer Simulation, 2006,16(1):1−25. [doi: 10.1145/1122012.1122013]

[18] Adamic LA, Lukose RM, Puniyani AR, Huberman BA. Search in power law networks. Physical Review E, 2001,64(4): 46135−46143.

[19] Ridge E, Curry E. A roadmap of nature-inspired systems research and development. Multiagent and Grid Systems, 2007,3(1):3−8.

[20] Parunak HVD, Brueckner SA, Matthews R, Sauter J. Pheromone learning for self-organizing agents. IEEE Trans. on Systems, Man, and Cybernetics—Part A: Systems and Humans, 2005,35(3):316−326. [doi: 10.1109/TSMCA.2005.846408]

[21] Clip 2. The Gnutella protocol specification. v0.4. 2000. http://www.clip2.com/GnutellaProtocol04.pdf

[22] Lü Q, Cao P, Cohen E, Li K, Shenker S. Search and replication in unstructured peer-to-peer networks. In: Gupta M, ed. Proc. of the 16th ACM Int'l Conf. on Supercomputing (ICS 2002). New York: ACM Press, 2002. 84−95.

附中文参考文献:

[2] 陈德伟,许斌,蔡月茹,李涓子.服务部署与发布绑定的基于 P2P 网络的 Web 服务发现机制.计算机学报,2005,28(4):615−626.

[4] 刘志忠,王怀民,周斌.一种双层 P2P 结构的语义服务发现模型.软件学报,2007,18(8):1922−1932. http://www.jos.org.cn/1000-9825/18/1922.htm [doi:10.1360/jos181922]

[5] 郭得科,任彦,陈洪辉,薛群威,罗雪山.一种 QoS 有保障的 Web 服务分布式发现模型.软件学报,2006,17(11):2324−2334. http://www.jos.org.cn/1000-9825/17/2324.htm [doi:10.1360/jos172324]

[7] 杜宗霞,怀进鹏.主动分布式 Web 服务注册机制研究与实现.软件学报,2006,17(3):454−462. http://www.jos.org.cn/1000-9825/17/454.htm [doi:10.1360/jos170454]

[10] 许辉,吴诗其.LEO 卫星网络中基于蚂蚁算法的分布式 QoS 路由.计算机学报,2007,30(3):361−367.

[11] 杜荣华,姚刚,吴泉源.蚁群算法在移动 Agent 迁移中的应用研究.计算机研究与发展,2007,44(2):282−287.

[14] 中华人民共和国国家质量监督检验检疫总局.国民经济行业分类.GB/T 4754-2002,北京:中国标准出版社,2007.

郑啸(1975－),男,福建莆田人,博士生,副教授,CCF 会员,主要研究领域为网络与分布式系统,服务计算.

宋爱波(1970－),男,博士,副教授,CCF 会员,主要研究领域为网格计算,海量数据处理,Petri 网理论与应用.

罗军舟(1960－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为下一代网络体系结构,协议工程,网络安全与管理,网格计算.