面向智能电子商务的移动 Agent 迁移策略研究

甘早斌 孔祥印 肖国强
(华中科技大学计算机科学与技术学院 武汉 430074)

摘 要 以智能电子商务系统为应用研究背景,综合考虑商家服务质量、网络负载等影响查询效率的因素,引入时间片概念,提出了一种基于商品智能搜索的 Agent 迁移策略模型。在模型中给出了一种客观准确的商家服务质量评价方法,采用申请与应答的方式有效避免了因网络负载和商家服务器负载等原因造成的迁移失败的问题,同时减少了网络带宽和服务器资源的浪费。

关键词 电子商务,移动 agent,迁移策略,时间片
中图法分类号 TP391 **文献标识码** A

Migration Strategy of Mobile Agents for the Intelligent E-commerce Applications

GAN Zao-bin KONG Xiang-yin XIAO Guo-qiang
(School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China)

Abstract Considering some factors (the quality of service and network load) on searching efficiency for the intelligent e-commerce applications, this paper defined the concept of time slice, and presented a migration strategy model of mobile agents searching goods. In this context, the evaluation approach of QoS (the quality of service) was given for businesses objectively and accurately. The request-response mode is utilized to effectively avoid the migration failure because of network overload and server overload, and reduce the waste of the network bandwidth and server resources.

Keywords E-commerce, Mobile agent, Migration strategy, Time slice

1 引言

近年来,随着 Internet 相关技术的发展,电子商务应用领域不断扩大,网上购物人数和网上交易量不断攀升[1]。传统的电子商务已不能满足人们的需求,尤其是 B2B 模式下的电子商务,如果一个企业生产的商品需要多个厂家的零部件,就要一个一个地访问每个厂家的门户网站去查询相关的零件产品信息,以找到满意商品进行交易。同时,对于供应商来说,如果靠客户访问站点才能提供服务,是非常被动的。随着企业之间竞争越来越激烈,企业与其合作伙伴的紧密联系与互助合作和对客户的需求与偏爱做出快速反应变得十分重要。在这种情况下,建立快捷、高效、智能化的电子商务已经成为电子商务的发展趋势。

由于移动 Agent 具有移动的特点,并支持离线计算、实时远程交互、动态均衡负载、并行任务求解、动态适应环境、感知环境变化、快速自主地做出反应等诸多优点[2],因此将移动 Agent 技术引入商品搜索中,既可以充分发挥移动 Agent 的技术优势,又可以适应智能电子商务发展的需要,并能为智能电子商务活动的参与者提供一系列的个性化服务,具有广阔的应用前景。

移动 Agent 的迁移策略是移动 Agent 的核心和基础支撑技术之一,也是提高商品搜索性能的核心技术。迁移策略是 Agent 迁移的决策机制,是根据移动 Agent 的任务、当前网络资源和远程主机资源等外界环境,动态地为 Agent 规划出迁移路径,使移动 Agent 在开销最小的情况下最快、最好地完成其任务。迁移策略的优劣直接影响移动 Agent 系统的性能乃至其任务的完成。

2 相关研究工作

移动 Agent 的迁移策略是移动 Agent 技术的核心领域,国内外很多学者对移动 Agent 的迁移策略问题做了比较深入的研究。Bewington 等人基于 D'Agent 开发了分布式信息查询系统"Technical-Report Searcher"[3]。在该系统中,他们提出了"旅行 Agent 问题(Traveling Agent Problem, TAP)",即移动 Agent 在出发前,TAP 求解模块根据当前网络负载和信息在主机上的分布概率,为移动 Agent 规划出迁移路径,使移动 Agent 以最快的速度找到待查信息。该迁移策略需要一个集中的信息检测模块,负责收集网络负载信息,而且没有考虑在移动 Agent 迁移过程中网络负载的动态变化情况。选择路径时只考虑了 Agent 的查询速度,并没有考虑服务质量。文献[4]对理想迁移策略、异步迁移策略和学习迁移策略 3 种迁移策略做了对比,通过实验得出第 3 种迁移策略的平均迁移

到稿日期:2010-09-10 返修日期:2010-11-15 本文受国家自然科学基金项目(70672041)以及湖北省自然科学基金项目(2007ABA307)资助。

甘早斌(1968—),博士,副教授,主要研究方向为电子商务、信任计算、软件经济学,E-mail:zgan@mail.hust.edu.cn;孔祥印(1985—),硕士,主要研究方向为智能搜索、电子商务;肖国强(1953—),副教授,主要研究方向为计算机图像处理、电子商务、可信计算。

时间最短、具有较高的迁移效率的结论。文献[5]针对网格环境提出了一种动态市场旅行商问题模型，其充分描述了移动Agent在网格环境中的行为方式，在规划移动Agent的迁移路径时充分考虑了网络负载及服务器的服务质量动态变化等因素。

移动Agent迁移计划问题(Mobile Agent Planning, MAP)为移动Agent迁移提供了新的解决方法。假设有N个能够独立完成相应的子任务的节点，移动Agent可以在规定的时间内访问相应的节点，完成相应的子任务，节点与节点之间的网络延时是不确定的，且每个节点只能被访问一次。MAP问题就是找到N个节点的访问顺序，使其在最短的时间内完成任务。文献[6]对移动Agent迁移计划问题做了深入的研究，提出了其相应的迁移策略，较为有效地提高了Agent系统的效率。文献[7]研究了多Agent和单Agent等情况下的基于MAP的移动Agent迁移策略问题。文献[8]提出了一种基于信息传输多移动Agent的迁移策略，该策略把移动Agent迁移分为两部分：一部分为迁移计划设定，另一部分为迁移路径的最优化。文献[9]应用了移动Agent静态迁移、动态迁移、可预测的动态迁移3种不同的方法，解决了无线传感器网络中的目标跟踪问题，并对这3种策略的性能进行了分析，实验证明了可预测的动态迁移比起前两种方法效率更高一些。

文献[10]中提出的信息检索模型中，需要计算访问每个节点所需要的时间，进而规划出一条迁移路径，使得移动Agent能够在设定的访问时间限制内完成信息检索。但该模型没有提出一个明确的访问时间限制的设定方法。文献[11]提出了一种Tmap(Timed Mobile Agent Planning)的迁移策略，有效地解决了在某些节点有访问时间限制的情况下移动Agent数量和执行总时间的最优化问题。文献[12]提出了一种基于预测Agent生命时间的迁移算法，保证了在Agent服务区域内有足够长的生命时间。

3 搜索移动Agent迁移策略

3.1 问题描述

在基于移动Agent电子商务环境下，如何让用户能够最快、最准确地找到所需要的商品信息，即如何让移动Agent在给定的商家中遍历所有商家，完成商品搜索任务的时间最短、效率最高，是本文要解决的问题。这种如何确定在多个目标点之间的移动顺序以获得最高效率的问题，类似于旅行商问题(TSP, Traveling Salesman Problem)。TSP的历史很久，最早的描述是1759年欧拉研究的骑士周游问题，即对于国际象棋棋盘中的64个方格，走访64个方格一次且仅一次，并且最终返回到起始点。对于经典旅行商问题，可定义为给定的n个城市，寻求单一旅行者由某一城市出发，通过另外$n-1$个城市之后，最后再回到原点的最小路径成本。而智能电子商务中移动Agent的迁移路径问题与TSP问题有一些差异：

(1) 提供商品搜索的服务器的服务能力不同；

(2) 网络环境的动态性很强，网络负载经常出现波动，影响网络传输速度；

(3) 用户等待结果返回时间是有限制的，不可能长期处于等待状态；

(4) 商家服务器可能需要处理多个用户的查询请求，并且服务状态和能力是不断变化的；

(5) 非对称性；

(6) 任务的模糊性，没有严格的执行条件和确定的执行结果。

根据模型的特殊性，提出了一种基于时间片的移动Agent动态规划迁移策略：给定n个服务器，代表销售商品种类不同的商家节点，m个移动Agent从客服服务器出发不重复地访问其路径上的商家节点，直到路径上商家节点被全部访问完，或其时间片用完为止。在保证搜索时间最短的情况下，其搜索到的商品数量最多。其中移动Agent在每个商家节点搜索的商品数量是不同的，商家节点之间的迁移所需要的时间是动态变化的，所以移动Agent需要在迁移之前与商家节点进行协商，确定是否迁移到该商家节点。另外，商家也可能动态增加或删除。

为了便于对模型进行数学描述，这里引入几个重要的定义。

定义1(执行时间) 从移动Agent到达某个节点开始，直到在该节点上的查询任务结束的这段时间。

定义2(迁移时间) 移动Agent从一个节点迁移到另一个节点所花费的时间，与网络带宽及网络负载相关。

定义3(时间片) 移动Agent被创建时就付给的，它表示移动Agent的寿命，只要时间片剩余值大于零，移动Agent就可以继续执行搜索任务，否则就会自行销毁。

定义4(协商时间) 从发送搜索请求到收到商家服务器回复的这段时间。其中包括信件在移动Agent与商家节点之间的往返时间和评价商家节点负载情况所用的时间。

定义5(旅行时间) 移动Agent被客户节点派出执行搜索任务开始到当前节点的这段时间。

该模型的数学描述如下：

(1) 设有n个商家节点，用集合H表示，其中某个商家用h_i表示，$h_i \in H, 1 \leqslant i \leqslant n$。

(2) 由客户服务器节点创建完成某个搜索任务的搜索移动Agent。然后，搜索移动Agent根据任务情况克隆m个子搜索移动Agent，用集合A表示，其中某个子搜索移动Agent用a_i表示，$a_i \in A, 1 \leqslant i \leqslant m$。

(2) 移动Agent a_i在任意两个商家节点间的距离用在两个商家之间迁移需要的时间表示为$d(h_j^i, h_k^i)$，$0 \leqslant j, k \leqslant n$；$h_0^i$表示移动Agent a_i在客户服务器节点HOME。

(3) 移动Agent a_i在每个商家节点进行商品检索所用的执行时间为$e(h_k^i)$，$1 \leqslant k \leqslant n$；在每个商家节点的商品检索量$r(h_k^i)$，$1 \leqslant k \leqslant n$；与每个商家节点协商所需要的协商时间为$c(h_k^i)$，$1 \leqslant k \leqslant n$；移动Agent a_i的时间片为$l(a_i)$，$1 \leqslant i \leqslant m$。

(4) 假定每个搜索移动Agent a_i需要访问的商家节点集合为$P(a_i)=\{h_1^i, h_2^i, \cdots, h_j^i\}$，$1 \leqslant i \leqslant m$，要求确定一个最佳访问顺序$h_{k_1}^i \rightarrow h_{k_2}^i \rightarrow, \cdots, \rightarrow h_{k_j}^i$，可用向量$Q(a_i)$表示，即

$$Q(a_i)=(h_{k_1}^i, h_{k_2}^i, \cdots, h_{k_j}^i), 1 \leqslant i \leqslant m, 1 \leqslant j \leqslant n, 1 \leqslant k_1, k_2, k_j \leqslant j$$

每个搜索移动Agent访问迁移路径上的所有商家节点后的旅行时间$T(a_i)$可定义为

$$T(a_i)=d(h_0^i, h_1^i)+c(h_q^i)+e(h_q^i)+\sum_{j=1}^{q-1}[d(h_j^i, h_{j+1}^i)+c(h_j^i)+e(h_j^i)]$$

式中，$1\leqslant i\leqslant m, q\leqslant length(P(a_i))$，$length()$为求解集合大小的函数，使得搜索商品数量 $S=\sum_{i=1}^{n} r(h_i)$ 最大，且 $MAX\{T(a_i)\}$ 最小，$T(a_i)\leqslant l(a_i)$，$1\leqslant i\leqslant m$。

3.2 商家服务质量评估

该策略充分考虑了移动 Agent 在选择迁移路径过程中可能遇到的多种制约因素。从软件资源角度分析来看，影响路径选择的主要因素是商家销售的商品质量、客户对商家服务信誉评价等。从硬件的角度分析，认为影响路径选择的主要因素是网络的传输时间、商家服务器的 CUP 运算速度、商家服务器的负载。这些因素都是移动 Agent 在商品搜索过程中不得不考虑的。这些影响因素是动态变化的，服务器之间不具有直接可比性，而且难以预知。下面用服务质量作为商家服务器的服务能力的评价标准。

商家服务器服务质量是不断变化的，而且每个用户提交的搜索任务可能是不同的，对商家服务质量的评估自然也不会相同。综合上述考虑，商家服务质量评估算法为

$$p_j=p_j'+\rho\Delta p_j \tag{1}$$

客户节点用此公式来更新商家服务器 h_j 的总体服务质量，最初每个商家节点的总体服务质量被赋予同样的初始值 1，p_j'为商家节点 h_j 总体服务质量被更新以前的值，p_j 为商家节点 h_j 总体服务质量被更新以后的值，Δp_j 为商家 h_j 在某次搜索中提供的服务质量，由下面的式(5)计算得到，参数 ρ 决定商家 h_j 在此次查询中的影响值，这里可以根据商家节点搜索得到的商品数量的大小来取值。

对商品检索来说，其商家参与搜索并且成功搜索到的商品数量越多，表明商家服务器的性能越好，其在下一次搜索中提供的服务质量可能越好。因此将搜索商家得到的商品数量，作为后续移动 Agent 提供路径选择的参考。搜索得到的商品数量影响函数如下，

$$\mu_j=\begin{cases}\dfrac{\alpha r(h_j)/e(h_j)}{\sum_{i=1}^{n} r(h_i)/\sum_{i=1}^{m}\min(l(a_i),T(a_i))}, & h_j\in P(a_i)\\ 0, & h_j\notin P(a_i)\end{cases} \tag{2}$$

式中，$\sum_{k=1}^{m}\min(l(a_k),T(a_k))$为 m 个搜索 Agent 完成同一个搜索任务总的花费时间，α 是访问 h_j 的移动 Agent a_j 在 m 个移动 Agent 中的影响因子，可以设为

$$\alpha=l(a_j)/\sum_{i}^{m} l(a_i) \tag{3}$$

商家访问量过大、网络阻塞等原因可能导致搜索响应时间过长，因此搜索的响应时间也是一个非常重要的影响因素。搜索的响应时间函数为

$$\upsilon_j=\begin{cases}\dfrac{d(h_i,h_j)+c(h_j)}{\sum_{k=1}^{m}\min(l(a_k),T(a_k))/m}, & h_j\in P(a_i)\\ 0, & h_j\notin P(a_i)\end{cases} \tag{4}$$

式中，$\sum_{k=1}^{m}\min(l(a_k),T(a_k))$为 m 个搜索 Agent 完成同一个搜索任务总的花费时间。$d(h_i,h_j)$是负责搜索商家节点 h_j 的移动 Agent a_k 从 h_j 的上一个商家节点 h_i 迁移到 h_j 所花费的时间，$c_j=c(h_j)$是 a_k 与节点 h_j 的通信时间。

综合上述因素考虑，商家节点 h_j 在一次查询任务中的服务质量综合评价公式为

$$\Delta p_j=\alpha\mu_j+\frac{(1-\alpha)}{\upsilon_j} \tag{5}$$

式中，α 为影响因子，其取值在 0～1 之间。

利用商家服务质量的评定值能保证服务质量好的商家被较早地访问，也就是在 Agent 迁移路径过程中，应该采取服务质量好的节点被优先访问的原则。

3.3 移动 Agent 与商家之间的协商

由于网络连接的不稳定性，商家与商家之间的网络连接可能会中断。网络负载过大，或者由于商家服务器正在执行搜索任务而导致服务器负载过大时，不能执行其他的搜索任务。以上多种原因可能导致移动 Agent 不能及时迁移到目标商家节点。这就需要一种协商机制，在移动 Agent 迁移之前判断下一个访问节点能否到达或执行搜索任务。这里采用请求和应答的方式，移动 Agent 通过发送信件的方式与商家服务器进行协商。信件格式如下，

(*LetterType*, *SourceHost*, *TargetHost*, *Port*, *BeginTime*, *Requst*)

信件分为“请求信件”和“应答信件”，由 LetterType 标识，BeginTime 标识信件的发送时间，Requst 标识商家服务器是接受搜索请求还是拒绝搜索请求，SourceHost 为信件的发送方地址，TargetHost 为信件的目的地址，Port 为商家服务器通信的端口号。

移动 Agent 向商家节点发送搜索请求信件，然后等待商家节点的回复信件。商家服务器接收到某一移动 Agent 发送的搜索请求信件时，如果商家节点正在执行搜索任务，服务器负载过大，不能接受其他的搜索任务，则向移动 Agent 发送拒绝搜索信件；如果商家服务器是空闲的，也就是没有执行搜索任务，则向移动 Agent 发送接受搜索信件。如果移动 Agent 接收到了商家服务器的接受搜索信件，则迁移至节点执行查询任务；如果接收到了拒绝搜索信件，或超过一定时间仍然没任何回复(这时认为节点之间的链接中断，Agent 无法迁移至搜索的节点)，则 Agent 从路由表中查找下一个商家节点，向其发送搜索请求信件。通信步骤描述如下。

(1) 移动 Agent(a_i)向商家节点 h_j 发送搜索请求信件 RequestLetter，并且 a_i 状态为等待状态，初始化等待时钟 $Time(a_i)$为 0：

$$a_i \xrightarrow{\text{RequestLetter}} h_j;\ Time(a_i)=0$$

(2) h_j 接收到 a_i 的信件，打开 RequestLetter，取出信中的 SourceHost 和 $BTime$。计算信件在网络中的传输时间 $TranTime=NTime-BTime$。若 $TranTime>waitt/2$(这里 $waitt$ 为 Agent 等待时间回复最长时间)，则丢弃 RequestLetter，不对该信做回复。否则执行(4)。

(3) 对服务器现在的负载情况进行评估：

$\omega_t(h_j)=f(CPU_t(h_j), LastTask_t(h_j))$

式中，$CPU_t(h_j)$表示 t 时刻商家服务器 CPU 负载情况，$LastTask_t(h_j)$表示 t 时刻商家服务器正在执行的搜索任务的剩余情况。商家节点 h_j 根据 $\omega_t(h_j)$确定回复类型，如果 $\omega_t(h_j)\geqslant\alpha$($\alpha$ 为服务器负载界定值)，服务器向移动 Agent 发送拒绝搜索信件，否则发送接受搜索信件。

$$h_j \xrightarrow{AnswerLetter} a_i$$

(4)a_i 打开 AwserLetter，取出 Requst。如果为拒绝搜索信件，则 a_i 从其路由表中找出下一个访问节点发送搜索请求；如果为接受搜索信件，则 a_i 向 h_j 迁移。

(5)若 $Time(a_i)>waitt$ 还没有收到回信，则 a_i 从其路由表中找出下一个访问节点发送搜索请求。

3.4 时间片设定

根据统计学规律，用户等待搜索结果返回有一个响应时间。如果超过了这个时间，用户就不愿继续等待下去，就可能终止查询任务。这时派出去的移动 Agent 可能还长时间地在商家节点之间迁移，执行搜索任务，这样就会造成网络带宽的浪费和商家服务器资源的浪费。采用为每个移动 Agent 设定一个时间片的方法来解决资源浪费问题，也就是每个移动 Agent 都有生命期限。如果移动 Agent 旅行的时间超过这个时间片，标志它生命已经结束，这时移动 Agent 就不需要再继续执行搜索任务了，自行销毁。

如果时间片设定得太短，可能导致用户得到的数据较少，不能满足用户的需求；如果过长，用户搜索结束时，移动 Agent 还长时间在网络中迁移，会造成网络带宽的浪费。综合上述考虑，时间片设定为

$$l(a_i)=(1+\beta)\sum_{j=1}^{q}AV(e(h_i^j))$$

式中，$1\leqslant i\leqslant m$，$q=length(Q(a_i))$，$AV()$是求解某个商家在以往的搜索任务中所需要的平均查询时间函数，$\beta\in[0,0.5]$，时间片大于商家所需查询时间的总和，因为考虑到了移动 Agent 迁移过程中所需的迁移延迟。迁移延迟主要包括商家节点与商家节点之间的协商时间、移动 Agent 在网络中的迁移时间、当迁移失败时移动 Agent 调整迁移路径所需的时间。迁移延迟的大小影响整个搜索任务完成所需时间的长短，也就影响到移动 Agent 时间片的大小。由于迁移延迟是由移动 Agent 的迁移路径选择、当前网络的软硬件环境和其他的网络负载信息等多方面因素决定的，并且网络负载信息是极不稳定的，因此无法预测移动延迟的准确值。因为如果迁移延迟超过查询时间的一半，就会严重影响查询效率，搜索时间会超出查询结果返回用户所能容忍的等待时间，所以这里设定迁移延迟时间小于商家所需查询时间总和的 0.5 倍，也就是间片长度设定为小于平均查询时间的 1.5 倍。

4 迁移策略的算法实现

为了便于理解算法，列出模型中移动 Agent 的主要状态有：创建(*create*)、等待(*wait*)、迁移(*migrate*)、执行(*execute*)、克隆(clone)、通信(communicate)、销毁(self-destruct)、返回(return)。状态的相互转换如图 1 所示。

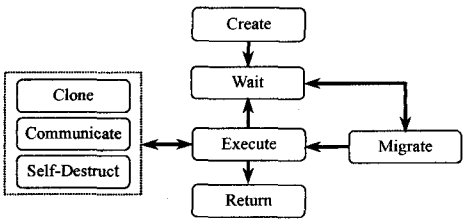

图 1 迁移状态转换图

其中，

(1) 创建(Create)：客服服务器接到客户提交的查询申请后，创建移动 Agent，并给其分配查询任务。

(2) 等待(Wait)：移动 Agent 从其携带的路由表中查找下一个要搜索的商家节点，向其发送搜索请求信件，等待商家节点的回复信件。

(3) 迁移(Migrate)：移动 Agent 接收到商家节点的接受搜索请求信件后，移动 Agent 向商家节点迁移。

(4) 执行(Execute)：移动 Agent 迁移到商家服务器后，执行搜索任务。

(5) 克隆(Clone)：移动 Agent 克隆一个相同的移动 Agent 来访问后面的商家节点。

(6) 销毁(Self-Destruct)：当移动 Agent 的时间片用完后，自行销毁。

(7) 通信(Communicate)：移动 Agent 迁移到商家节点与之进行通信。

(8) 返回(Return)：商家节点的搜索结束后克隆一个返回 Agent，返回到客服服务器。

这里主要介绍模型中最重要的两个算法：移动 Agent 任务分配算法及其迁移算法。移动 Agent 任务分配算法(算法 1)中，客户服务器首先计算所有的商家节点服务质量评分值，对商家节点进行排序，然后为每个 Agent 分配搜索任务。算法保证了每个移动 Agent 迁移路径上的商家节点顺序是按商家节点服务质量评分值降序排列的，并总体上保证了服务质量好的商家节点最先得到访问。移动 Agent 迁移算法(算法 2)中，移动 Agent 首先检查自己的时间片，确认自己的寿命是否已到期。如果到期，则 Agent 自行销毁；如果未到期，Agent 继续执行搜索任务。Step3 中，移动 Agent 向商家节点发送搜索请求。如果商家节点接受了搜索请求，则移动 Agent 迁移至该商家节点执行查询任务；如果商家节点拒绝了搜索请求，则移动 Agent 从路由表中查找下一个商家节点。最后(Step7)路径上的所有商家节点访问完毕，Agent 自行销毁。

算法描述如下。

算法 1 移动 Agent 任务分配

Step 1 计算各商家节点的服务质量评分值 P_i，然后根据评分值对商家节点进行降序排序 $h_1,h_2,\cdots,h_n$。

Step 2 计算此次搜索所需要的迁移 Agent 的数目 $AN=BN/k$，其中 BN 为参与此次搜索的商家节点个数，k 为每个移动 Agent 要负责搜索商家节点的个数。生成 AN 个移动 Agent，$a_1,a_2,\cdots,a_{AN}$。

Step 3 AssignNode(a_i) /* 移动 Agent 任务分配算法 */

```
r=1;
for j=1 to BN
for k=0 to AN-1 /* 把剩下的前 AN 个商家节点分别分配给 AN 个 task_r(a_{k+1})=B_j；移动 Agent */
k=k+1;
end for
j=j+AN;
end for
```

Step4 按分配算法依次把商家节点加入到移动 Agent 路由表中。

算法 2 移动 Agent 迁移算法

Step 1 计算移动 Agent a_i 旅行时间 $T_i=T(a_i)$。如果 $T_i>l(a_i)$，则 a_i 自行销毁，不再继续执行搜索任务。否则按以下步骤执行。

Step 2 a_i 从路由表中查找下一个节点 h_j。

Step 3 a_i 向 h_j 发送搜索请求信件，a_i 处于等待状态，并启动等待计时器 $Time(a_i)$。商家节点操作如下：

(1) h_i 接收到 a_i 发送的搜索请求信件后，首先评价服务器现在时刻的负载情况 $\omega_t(h_j)$，可以通过检测 CPU 的使用量或现在时刻服务器执行的搜索任务的剩余情况来计算。

(2) 如果 $\omega_t(h_j)<\alpha$(α 为服务器负载界定值)，就给 a_i 发

送一个接受搜索信件；否则，发送一个拒绝搜索信件。

Step 4　如果 $Time(a_i)<\delta$ 内没有收到回复信件或收到的是拒绝搜索信件，则执行 Step1－Step4；如果 a_i 收到了接受搜索信件，则执行下面的步骤。/＊没有收到答复可能是由于节点之间的网络中断，拒绝答复可能是由于节点正在执行查询任务＊/

Step 5　a_i 迁移到 h_j，将搜索条件提交给商家节点的静态查询 Agent(SQA)，商家服务器执行如下操作：

(1) SQA 与 a_i 通信，获得搜索条件、客户服务器 IP 等信息。

(2) 根据 a_i 的时间片 $l(a_i)$ 和 a_i 的旅行时间 $T(a_i)$，设置 SQA 查询时间限制 $LimT=l(a_i)-T(a_i)$；为 SQA 设置时钟 $Time(SQA)=0$。

(3) SQA 对本地商品信息执行搜索任务，$Time(SQA)=0$，SQA 生成一个返回 Agent(BA)，把搜索到的结果返回给中心服务器。

(4) 循环执行(1)－(3)。

Step 6　把此节点访问标记为已搜索。

Step 7　如果所有的节点搜索完毕，则 a_i 自行销毁。否则循环执行以上操作。

5　实验及性能分析

利用课题组开发的 Smart Market 电子商务系统平台作为实验环境。在实验室所在的校园网内，采用 9 台 PC 机模拟 8 个商家服务器和 1 台客户服务器，如图 2 所示。客户的搜索任务需求通过浏览器提交给客户服务器，客户服务器创建移动 Agent，从客户服务器出发搜索每台商家服务器 SQL SERVER2000 数据库中的商品信息。实验室每台主机的配置是不同的，所以服务能力也不相同。每台主机数据库的商品信息记录数目也是不同的。根据实验室每台主机配置的高低和商品信息记录数目的大小，给 8 个商家服务器赋予不同的服务质量评定值。

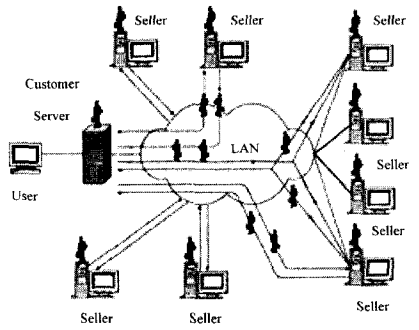

图 2　实验环境拓扑结构图

实验分别采用了 3 种不同的迁移策略：静态迁移策略(ST)、服务器迁移策略(SS)、移动 Agent 动态规划迁移策略(DY)。其中，(1)静态迁移策略(ST)主要特点为，在移动 Agent 出发前，随机规划出每个移动 Agent 的迁移路径，在迁移过程中移动 Agent 严格按照初始的节点顺序进行访问。此策略的主要缺点为移动 Agent 迁移路径的规划是随机的，没能反映出节点的服务能力对迁移路径规划的影响，也没有考虑 Agent 迁移过程中可能遇到网络中断、访问的主机出现故障及服务实效等情况。(2)在等服务器迁移策略(SS)中，移动 Agent 能根据初始迁移路径和当前网络的软硬件环境及其它负载信息，自主地调整迁移的路径。但是，移动 Agent 迁移路径的规划仍然是随机的。(3)移动 Agent 动态规划迁移策略(DY)，除了兼顾等服务器迁移策略(SS)优点以外，可对每个节点的服务质量进行评价，并根据评价值为每个移动 Agent 规划迁移路径，反映了节点的服务能力对迁移路径规划的影响。

实验对于 3 种不同的迁移策略各自进行了 10 次实验。每次实验，客户服务器创建 3 个移动 Agent，每个移动 Agent 负责搜索商家服务器中的 3 个服务器。在前两种策略中，按服务器 1－8 顺序分配给每个移动 Agent，第 3 种策略中按上面一节的分配算法给移动 Agent 分配搜索任务。在实验过程中，通过关闭某些主机来模拟网络连接出现的中断现象。实验结果如表 1 所列。

表 1　不同迁移策略实验数据

strategy	time	number
ST	220s	1005
SS	195s	1155
DY	140s	1460

分析实验结果可以得出，移动 Agent 动态规划迁移策略用较少的搜索时间，搜索回来的商品信息要比其他两个迁移策略多。在实验中还发现起初 3 种迁移策略的搜索效率相差不是很大，但随着搜索次数的增加，第 3 种策略的搜索效率要明显地比其他两种策略高。这说明第 3 种策略有一定的自我学习能力。

对移动 Agent 有无时间片设定做了不同的实验。当查询结束后，有时间片的迁移中的移动 Agent 继续迁移的平均时间要比无时间片的少，也就说明有时间片的迁移策略更能节省网络资源和商家服务器资源。所以可以得出基于时间片的移动 Agent 动态规划迁移策略可以提高商品搜索的搜索质量和效率的结论。

基于时间片的移动 Agent 动态规划迁移策略主要有以下贡献。

首先，本文提出的迁移策略，客观地评估了商家服务器的服务质量，能较为公正、准确地反映商家服务器的服务能力，为迁移计划提供了路径选择的基础。

其次，为了防止网络阻塞可能造成的移动 Agent 迁移失败，或商家服务器负载过大造成查询不能及时进行，采用了申请、应答机制，由移动 Agent 提出搜索申请，商家服务器根据自己的情况应答 Agent 的申请，从而有效地避免了上述两种情况。

最后，如果用户查询请求结束，但移动 Agent 仍然可能在网络中迁移到还没有访问的商家服务器节点，这样造成了网络带宽和商家服务器资源的浪费。本文采用了时间片的机制，给每一个移动 Agent 赋一个时间限制，也就是该 Agent 的生命值。当 Agent 生命值结束时，就会自行销毁，不再进行迁移。这样就能够较好地解决上面的问题。

总之，这种基于时间片的移动 Agent 迁移机制能够根据商家服务器的服务能力选择迁移路径，能够更好、更快地把搜索结果返回给用户，提高了商品搜索的性能。

结束语　本文提出了基于时间片的移动 Agent 动态规划迁移模型，让用户能够更快、更准确地得到想要的商品信息。该策略能够客观地评价商家服务器的服务能力，并且动态感应网络负载及商家服务器负载信息，动态改变迁移路径。A-

gent 查询会返回大量的查询结果，将来的工作中，我们将研究这些结果的排序和归类。

参 考 文 献

[1] Meeker M, Devitt S, Wu Liang. Internet Trends [OL]. http://www.morganstanley.com/institutional/techresearch/pdfs/MS_Internet_Trends_060710.pdf, 2010-06-07

[2] Fuggetta A, Picco G, Vigna G. Understanding code mobility [J]. IEEE Transactions on Software Engineering, 1998, 24(5): 352-361

[3] Brewington B, Gray R, Moizumi K, et al. Mobile agents in distributed information retrieval [C]// Klush M, ed. Intelligent Information Agents. Heidelberg, Germany: Spring-Verlag. 1999, 15: 355-395

[4] 刘大有，杨博，杨鲲，等. 基于旅行图的移动 Agent 迁移策略[J]. 计算机研究与发展，2003，40(6)：838-845

[5] 张至柔，罗四维，陈欲，等. 移动 Agent 在网格中的路径优化算法研究[J]. 计算机研究与发展，2006，43(5)：791-796

[6] Moizumi K. Mobile Agent planning problems [D]. Dartmouth College, 1998

[7] Moizumi K, Cybenko G. The traveling Agent problem, Mathematics of Control [J]. Signals, and Systems, 2001, 14(3): 213-232

[8] Zgaya H, Hammadi S, Gh'edira K. A migration strategy of mobile Agents for the transport network applications [J]. Mathematics and Computers in Simulation, 2008, 76: 345-362

[9] Xu Ying-yue, Qi Hai-rong. Mobile Agent migration modeling and design for target tracking in wireless sensor networks [J]. Ad Hoc Networks, 2008, 6: 1-16

[10] Tzitzikas Y, Constantopoulos P. A decision theoretic model for information retrieval supporting time constraints [C]// Proceedings of the BCS-IRSG European Colloquium on IR Research. Darmastadt, 2001: 1-12

[11] Baek J-W, Yeom H Y. A timed mobile Agent planning approach for distributed information retrieval in dynamic network environments [J]. Information Sciences, 2006, 176: 3347-3378

[12] Shiokawa S. A Migration Algorithm for Moblile Agent Based on Predicted Lifetime in MANET [C]// Proceedings of the 2nd IFIP Wireless Days Conference. December 2009, Paris, France, 2009: 1-6

(上接第 135 页)

[4] Huller J. Reducing Time to Market with Combinatorial Design Method Testing[C]// Proceedings of 10th Annual International Council on Systems (INCOSE2000). Minneapolis, MN, USA, July 2000: 16-20

[5] Lei Y, Tai K C. In parameter order: a test generation strategy for pairwise testing[R]. TR-2001-03. Raleigh, North Carolina: Department of Computer Science, North Carolina State University, 2001

[6] Seroussi G, Bshouty N H. Vector Sets for Exhaustive Testing of Logic Circuits [J]. IEEE Transaction on Information Theory, 1988, 34(3): 513-522

[7] Williams A W, Probert R L. Formulation of the Interaction Test Coverage Problem as an Integer Program [C]// Proceedings of 14th International Conference on the Testing of Communicating Systems. March 2002: 283-298

[8] Wallace D R, Kuhn D R. Failure Modes in Medical Device Software: An Analysis of 15 Years of Recall Data[J]. Int'l Jour. of Reliability, Quality and Safety Engineering, 2001, 8(4): 351-371

[9] Smith B, Feather M S, Muscettola N. Challenges and Methods in Testing the Remote Agent Planner[C]// Proc. 5th Int'l Conf. on Artificial Intelligence Planning and Scheduling (AIPS 2000). 2000: 254-263

[10] Nie Chang-hai, Xu Bao-wen. A Survey of Combinatorial Testing [J]. Journal of the ACM, 2008, 2(3)

[11] Bryce R C, Colbourn C J. Constructing Interaction Test Suites with Greedy Algorithms [C]// Proceedings of 20th IEEE/ACM International Conference on Automated Software Engineering (ASE2005). November 2005: 440-443

[12] Lei Y, Kacher R, Kuhn D R, et al. IPOG: A General Strategy for T-Way Software Testing [C]// Proceedings of 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS2007). March 2007: 549-556

[13] Cohen D M, et al. The AETG system: An approach to testing based on combinatorial design [J]. IEEE Transaction on Software Engineering, 1997, 23(7): 437-444

[14] Tung T W, Aldiwan W S. Automating Test Case Generation for the New Generation Mission Software System [C]// Proceedings of IEEE Arospace Conference. 2000: 431-437

[15] Colbourn C J, Cohen M B, Turban R C. A Deterministic Density Algorithm for Pairwise Interaction Coverage [C]// Proceedings of IASTED International Conference on Software Engineering (SE2004). February 2004: 345-352

[16] Bryce R C, Colbourn C J. The Density Algorithm for Pairwise Interaction Testing [J]. Software Testing, Verification and Reliability, 2007, 17: 159-182

[17] Ghazi S A, Ahmed M A. Pair-wise Test Coverage Using Genetic Algorithms[C]// Proceedings of the 2003 Congress on Evolutionary Computation (CEC2003). Volume 2. Canberra, Australia, December 2003: 1420-1424

[18] Shiba T, Tsuchiya T, Kikuno T. Using Artificial Life Techniques to Generate Test Cases for Combinatorial Testing [C]// Proceedings of 28th Annual International Computer Software and Applications Conference (COMPSAC2004). Volume 1. Hong Kong, China, September 2004: 72-78

[19] Bryce R C, Colbourn C J. One-test-at-a-time Heuristic Search for Interaction Test Suites [C]// Proceedings of 9th Annual Conference on Genetic and Evolutionary Computation (GECCO2007). July 2007: 1082-1089

[20] Cohen M B, Colbourn C J, Ling A C H. Augmenting Simulated Annealing to Build Interaction Test Suites [C]// Proceedings of 14th International Symposium on Software Reliability Engineering (ISSRE2003). Denver Colorado, USA, November 2003: 394-405

[21] Cohen M B, Colbourn C J, Ling A C H. Constructing Strength Three Covering Arrays with Augmented Annealing [J]. Discrete Mathematics, Available online June 2007

[22] Cohen M B, Gibbons P B, Mugridge W B, et al. Variable Strength Interaction Testing of Components [C]// Proceedings of 27th Annual International Computer Software and Applications Conference (COMPSAC2003). Dallas, TX, USA, 2003: 413-418