基于面向自治计算的 Agent 系统动态重构模型*)

陶 丽[1] 张自力[1,2]

(西南大学智能软件与软件工程实验室 重庆 400715)[1] (迪肯大学工程与信息技术系 澳大利亚 VIC3217)[2]

摘 要 动态重构问题是支持 Agent 系统对环境自适应的关键挑战之一,亦是 Agent 领域亟待解决的关键问题。面向自治的计算是一种全新的自底向上解决问题的计算范型,擅于提取并刻画复杂、自组织系统的行为规则,特别适合于对复杂问题和复杂 Agent 系统进行建模。因此,本文采用面向自治的计算 AOC(Autonomy Oriented Computing),提取出了基于多 Agent 的动态重构模型 ADRM,重点讨论了模型中自治实体、环境、自治实体行为等关键要素的定义,分析了该模型如何能够实现动态重构,并提供了 ADRM 有效工作的动态重构算法 ASDR(Agent System Dynamic Reconfiguration)。经实验验证,在任务和环境动态变化的情况下,模型 ADRM 能够较好地解决 Agent 系统的动态重构问题。

关键词 多 Agent 系统,动态重构,面向自治的计算 AOC,模型

Dynamic Reconfiguration Model of Multi-agent Systems Based on Autonomy Oriented Computing

TAO Li[1] ZHANG Zi-Li[1,2]

(Laboratory of Intelligent Software and Software Engineering, Southwest University, Chongqing 400715)[1]

(School of Engineering and Information Technology, Deakin, Geelong, Australia, VIC 3217)[2]

Abstract Dynamic reconfiguration has been listed as one of the key challenges in support of Agent adaptation to environments, which has attracted much attention of researchers world wide. To tackle this tough problem, an Agent-Based Dynamic Reconfiguration Model (ADRM) is proposed from the autonomy-oriented computing (AOC) point of view. This paper presents an autonomy-oriented-computing model ADRM, for solving Agent system dynamic reconfiguration. And an algorithm ASDR (Agent System Dynamic Reconfiguration) has been developed for implementing reconfiguring. In order to test the efficient of this model, the performance of this model was investigated.

Keywords Multi-agent system, Dnamic reconfiguration, Atonomy oriented computing, Model

1 引言

多 Agent 系统和混合多 Agent 系统在探索大规模分布式开放系统和理解、刻画现实世界中的许多复杂问题都有非常广泛的应用[1,2]。但在基于多 Agent 的应用中,有一个亟待解决的困难问题,即在用户需求、Agent 所处环境、甚至 Agent 本身均可动态变化的情况下,如何根据任务或者环境的变化来动态重构基于多 Agent 的系统,以避免因静态的组织结构造成不恰当的甚至是错误的配置,造成错误的解决方案或者产生不可接受的严重质量问题。

事实上,这一问题称为"运行时重构与重设计",被列为支持 Agent 对环境自适应能力的关键挑战之一[3]。针对这一问题,国内外学者探索了多种解决途径。Sycara 等提出了利用不同种类的中间 Agent,以实现系统中 Agent 的动态增加或删除[4];Hannebauer 提出了侧重于个体 Agent(微观)而不是社会(宏观)级别的两种重配置操作——"Agent 合并"与"Agent 分解"[5];Palma 等研究者探索了修改系统中 Agent 地理分布的重构技术[6];van der Hoek 和 Wooldridge 提出了授权和协作的动态逻辑(Dynamic Logic of Delegation and Cooperation, DCL-PC),变量可以在 Agent 联盟中动态分配,使得联盟的能量结构(Power Structure)发生变化[7];一些研究者借鉴博弈论和社会学的方法,对在动态网络中构建健壮、敏捷的 Agent 动态联盟进行了探索[8]。目前,中科院的诸葛海研究员领导的小组也在与该问题类似的方面展开研究,称为"网络资源的智能聚融"(http://www.semgrid.net/)。

Agent 系统的动态重构与 Agent 联合关系密切,它实际上是对某种 Agent 联盟联合成功后的动态管理过程。联合是多 Agent 系统和电子商务领域的研究热点。Shehory 等提出了一个在大规模电子市场中的多 Agent 联盟形成模型,描述了 Agent 加入、离开直至达到联盟稳定状态的宏观行为[9];Nathan Grffiths 和 Michael Luck 给出了基于动机和信任的联盟形成机制,阐述了自利的 Agent 如何建立、维持、解体一个部落(Clan)的过程[10];Wooldridge 和 Jennings 提出了完整的包含联合意图、全局规划等协商各阶段的多 Agent 协作问题求解方案[11];Kraus 等人提出了一种协议,某个 Agent 可以用边际启发和专家启发的方法来选择联盟的伙伴,而不需要确切地知道其他 Agent 完成子任务的耗费,从而使 Agent 面对不完全信息可以更好地形成联盟[12];Lerman 等人提出了一种基于动态物理微分方程的 Agent 进出联盟管理方法,通过增大模型的普遍性和复杂性允许 Agent 在一定条件下离开联盟,并能计算联盟的总体效用[13]。

但是在以上的研究中,都没能提出一个在任务个数、要

*)本课题得到国家教育部重点项目基金(104160)资助。陶 丽 硕士研究生,研究方向为人工智能,面向自治的计算。张自力 教授,研究生导师,目前从事人工智能、基于代理的计算、混合智能系统等方面的研究。

求，Agent 的个数、能力均动态变化情形下，使完成任务的多 Agent 系统本身自适应变化的有效解决途径。

为此，本文采用一种新的计算范型——面向自治的计算（Autonomic Oriented Computing，简称 AOC）[14,15]，提取出了 Agent 系统动态重构的抽象模型 ADRM，设计了完成动态重构的算法 ASDR（Agent System Dynamic Reconfiguration）。由于系统的动态重构可能涉及到 Agent 系统的重新形成，因此，我们改进了面向多 Agent 解决 CSP 问题的算法 ERA（Environment, Reactive rules, and Agents）[16]用于支持重构问题，该算法另文讨论[17]。

本文第 2 节将首先描述动态重构问题在现实世界中的一个真实场景；第 3 节将重点阐述为什么面向自治的计算 AOC 适用于动态重构问题的解决；第 4 节将介绍基于 AOC 的动态重构系统 ADRM 模型和 ASDR 算法；第 5 节给出实验结果以及讨论；最后对全文进行总结，并对未来工作进行展望。

2 Agent 系统动态重构例示

动态重构问题普遍存在于基于 Agent 的系统中。为了更好地阐述本文将解决的动态重构问题，本文选择国际贸易 Agent 竞赛中的供应链管理单元作为研究背景（http://www.sics.se/tac）（TAC SCM）。在 TAC SCM 中，Agent 将模拟一家小型计算机装备工厂的管理人员，它需要对从原料获取、产品生产到完成客户订单的整条供应链进行良好的管理，以获取最大的利润。为更好地阐明何种情况需要动态重构，在原有的应用背景基础上，依据对现实世界需求的分析，本文引入了 3 种动态变化的情况。

- 每一种原料的供应商是不稳定的。除了 TAC SCM 原本规定的两家外，还有无数家供应商愿意提供他们的原料。原料供应商的总数目和各自的报价可能每天都在改变。
- 每个原料供应商的状态是变化的。比如某个供应商可能会一夜破产或者因为一些其他不可抗力导致无法按时到货。
- 在订货后的某个时间范围内，客户的需求是可以改变的。比如，某个客货的订单表明他只需要由 CPU、主板、内存和硬盘组装的电脑，但在订货的第 2 天，他又希望每台电脑都能拥有 DVD 光驱。

由于现有的经典供应链管理模型和 TAC SCM 参赛选手的解决方案都忽略了这样的动态情形，因此现有的策略不能满足这样的动态变化要求。

为了实现如场景中描述的动态重构情况，我们构建的动态重构模型 ADRM 需要能够满足：(1)一种信息服务的机制，以便系统能够检测、收集、定位需要的供应商 Agent；(2)一种表述供应商 Agent 特征的方法，以便能够较为方便地比较不同供应商间的优劣；(3)一种能够在众多供应商 Agent 中找到最优解的算法，该算法即使在有限时间内不能找到最优，至少也能够提供一个可以接受的次优解；(4)一种用来维持和动态重构 Agent 系统的机制。

对于问题 1，本文采用一个中间 Agent——黄页 Agent，来实现供应商 Agent 的发现和注册；对于问题 2，采用约束求解和 Redline[18] 相结合的方法进行解决，把用户的每一项要求都作为一个约束，然后采用 Redline 的方法进行匹配，从而将每一个供应商 Agent 的能力都通过一个约束值和来表示；改进面向多 Agent 解决 CSP 问题的算法 ERA 用以解决问题 3；通过一个 ADRM 系统的动态重构控制器和动态重构算法 ASDR 来实现问题 4 的解决。

在本文中，我们将重点对模型的系统结构、动态重构实现机制进行详细的阐述。

3 为什么采用面向自治的计算 AOC

面向自治的计算 AOC，是一种非常适用于提取并刻画复杂、自组织系统的行为规则的计算范型[14,15]，已经被广泛应用到约束满足问题求解[16]、图像特征提取[17]、优化[18]等众多领域当中。

一个 AOC 的形式框架是一个三元组$\langle\{e_1,e_2,\cdots,e_N\},E,\Phi\rangle$。其中$\{e_1,e_2,\cdots,e_N\}$是自治实体，$E$ 是环境，Φ 是整个系统的目标函数。一个自治实体包括感受器（一个或多个）、效应器（一个或多个）和一个本地行为规则库。自治实体被定义为五元组$\langle S,F,G,B,R\rangle$。其中 S 是实体 e 当前的状态，F 是评价函数，G 是系统的目标函数，B 和 R 分别是实体的原始行为和原始行为规则。

采用 AOC 的方法提取 Agent 系统动态重构的抽象模型，就是要把系统中的元素映射到 AOC 的形式框架，即抽取并定义环境、自治实体、自治实体的本地行为规则库。

动态重构问题还可归纳为约束满足问题 CSP（Constraint Satisfaction Problem）。CSP 问题的基本组成元素为变量 V、变量的域 D 以及约束 C[21]。变量的域 D 是变量可能取值的集合，变量 V_i 只能在它的域 D_i 中进行取值；约束 C 描述了变量 V 之间必须满足的关系。

据此，我们可以对基于多 Agent 的动态重构问题进行建模：

- 网络中愿意提供服务的 Agent 可以对应于 CSP 中的变量 V；
- Agent 所处的网络环境就是变量的域 D；
- 任务对 Agent 的各种要求自然的表示为约束条件 C。

本文将采用 AOC 的方法来解决动态重构这一 CSP 问题。

4 基于 AOC 的动态重构系统模型 ADRM

基于 AOC 的动态重构模型 ADRM 实际上包含了多种不同的 Agent：提供资源的成员 Agent（Magent），承担资源搜寻工作的建组 Agent（Cagent），在联盟中执行用户任务的雇员 Agent（Eagent），提供资源发现、注册、归类的黄页 Agent（YP），以及动态重构行为的控制 Agent（Reconfiguration Controller，RC）。

ADRM 模型的体系结构如图 1 所示。

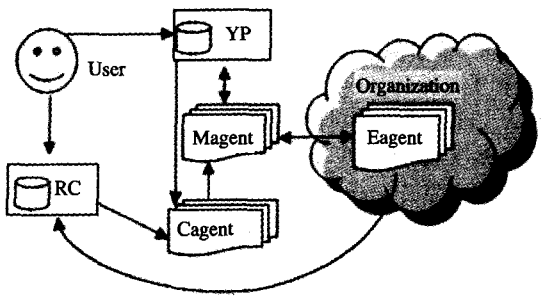


图 1 ADRM 模型体系结构

4.1 Agent 系统动态重构问题形式化描述

动态重构模型 ADRM 在某个时刻可形式化描述如下：

·网络中存在 n 个位于不同地理位置、具备不同行为能力的成员 Agent：

$Magent=\{Magent_1, Magent_2, \cdots, Magent_n\}$

·需要 i 个建组 $Cagent$ 承担资源搜索角色：

$Cagent=\{Cagent_1, Cagent_2, \cdots, Cagent_i\}$

·约束条件 m，即任务需要成员 Agent 具备的 m 种能力：

$C=\{C_1, C_2, \cdots, C_m\}$

4.2 ADRM 模型中的动态重构流程

Agent 系统的动态重构行为发生在联盟形成后、任务执行过程中，有如下几个原因导致重构行为的发生：

·网络中成员 Agent 执行任务的愿望是动态变化的；

·成员 Agent 执行任务的能力，或者叫做可提供的资源是动态变化的；

·执行任务的组员 Agent 的愿望是动态变化的；

·执行任务的组员 Agent 的个体能力是动态变化的；

·用户或者任务对于组员 Agent 的要求是动态变化的，这个要求包括组员 Agent 的个数和个体能力等。

在 ADRM 模型中，Cagent、任务、Eagent 的状态改变信息主要是通过黄页 Agent 和控制 Agent 来搜集的。黄页 Agent YP 将根据任务的不同要求和网络真实环境的变化，动态地组织和管理成员 Agent，接受 Magent 加入和退出的申请。而控制 Agent 将根据任务的不同分配相应的建组 Agent，同时接受组员 Agent 的退出申请，并指派相应的 Cagent 重新开始建组行为。控制 Agent 将会尽可能地重用以前的 Cagent 种群，以缩短建组时间。但是，YP 和 RC 仅仅是一个信息搜集者、咨询者和指令发出者的角色，ADRM 模型中最核心的部分——组织形成和动态维护则主要是通过 Cagent 在逻辑环境中的行为来实现的。

ADRM 模型的动态重构流程图如图 2 所示。

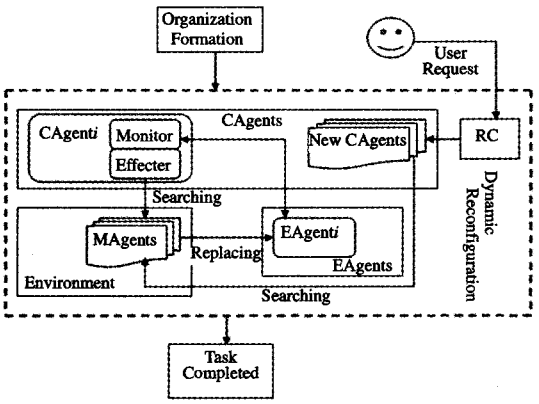


图 2 ADRM 模型的动态重构流程图

4.3 ADRM 模型中的 Agent 类别及原始行为

ADRM 模型中的 Agent 类别及原始行为分别定义如下：

4.3.1 成员 Agent(Magent)

Magent 种群是 ADRM 模型中最基本的组成部分，是真实网络中的一些具有任务要求的能力，并愿意提供这些能力的自治实体。在系统的运行过程中，Magent 将把自己的能力与任务的要求相匹配，并将匹配后的约束值和自己在网络中的位置等相关信息注册到黄页 Agent 当中。在系统的运行过程中，Magent 整个种群的大小、个体能力、个体位置等都是动态变化的。

一个 Magent 可表示为一个 4 元组：

$Magent=(MID, RT, CV, ML)$

其中，MID：Magent 的唯一标识号；RT：Magent 提供的某种资源类型；CV：Magent 所提供的某种资源约束值，采用约束求解和匹配算法计算得出，本文不进行详细阐述；ML：Magent 在真实网络中的所在位置。

4.3.2 建组 Agent(Cagent)

Cagent 是形成和维持 ADRM 系统最重要的自治实体。它由控制 Agent RC 根据任务的要求或者环境的变化指派产生相应的 Cagent 种群。Cagent 有两个主要的功能：一是在环境中通过本地搜索行为，选择出最适合任务需求的 Magent，组成完成任务的组织。二是组织形成，对自己挑选出来的 Magent 进行侦听，一旦发现该 Magent 出现不能继续完成任务执行的状况，将向 RC 发起重构申请。

4.3.3 雇员 Agent(Eagent)

当 Cagent 搜索行为结束之后，由 Cagent 寻找到的那些 Magent 就构成了当前执行任务的一个组织，从而执行任务的这些 Magent 就转变为了 Eagent 的角色。因此，Eagent 种群实际上是任务的最后执行者。在任务的执行过程中，每个 Eagent 可以根据自己的能力、愿望等变化状况，可以向控制 Agent 提出停止自己承担的任务，退出当前的组织。

4.3.4 黄页 Agent(YP)

黄页 Agent YP 是 Magent 种群的管理者。它的主要功能是提供网络中的 Magent 将自己的能力和相关信息注册并存储，然后通过随机选择划分，形成逻辑环境表，以提供给 Magent 搜索和查询。如果某 Magent 不愿意提供自己的能力，那么它需要向 YP 申请，YP 将注销这个 Magent 的相关信息。

黄页 Agent YP 要求的注册信息是一个 3 元组：

$Register=(RT, CV, ML)$，

其中，RT：Magent 提供的某种资源类型；CV：Magent 所提供的某种资源约束值；ML：Magent 在网络中的所在位置。

4.3.5 控制 Agent(Reconfiguration Controller，RC)

控制 Agent ——Reconfiguration Controller(RC)，是建组 Agent 的分配者和协调者。根据任务的要求，RC 将分配满足用户要求个数的 Cagent，并指派各自的搜索类型。同时，它还能够根据 Cagent 提出的重构申请，判断到底是对 Agent 系统进行重构还是解散整个系统重新组织。本文采用要求重构的 Eagent 占整个 Eagent 种群的比例作为判断的标志，一旦该比例超过给定的阈值(比如 0.4)，控制 Agent 就决定不再实施重构，而是解散整个 Agent 系统。值得注意的是，RC 不会对每个 Cagent 的本地搜索行为进行控制，只是在宏观上对整个 Cagent 种群进行协调。

4.4 环境描述

环境是成员 Agent 的所在地，也是建组 Agent 行为的发生地。这里的环境包括两个层面，一是物理环境：由 n 个成员 Agent 构成的实际网络环境；二是逻辑环境：即 n 个成员 Agent 注册后，经过 YP 的映射、划分，由各自的限制值和一些相关信息构成的一张二维表。在后面的描述中，环境就特指逻辑环境，它是我们动态重构行为的发生地，用于传递 Cagent 感兴趣的特征，以及 Cagent 间可共享的本地知识，比如当前的 Magent 是否已经被其他 Cagent 挑选。

物理环境与逻辑环境的关系如图 3 所示。

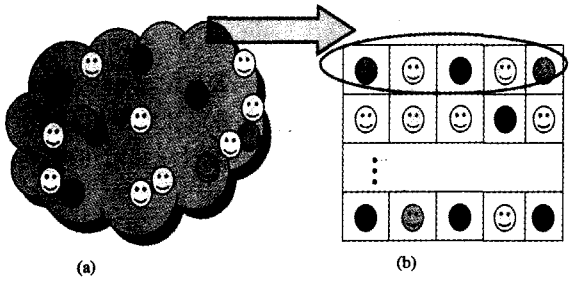


图 3 (a)真实网络中的 Agent 构成的物理环境
(b)经映射和划分后形成了逻辑环境

定义 1 环境的数据结构：

(1) 表大小

• n 行⇔n 个建组 Agent. E=〈 row_1, row_2, …, row_n〉;

• $\forall i \in [1,n]$, row i⇔建组 Agent Cagent i 的行为域⇔D_i, 因此第 i 行便有 $|D_i|$ 列;

row_i=〈 $lattice_{i1}$, $lattice_{i2}$, …, $lattice_{i|D_i|}$〉;

• E 是一个大小为 $\sum|D_k|$ 的数组。$e(i, j)$就等于 $lattice_{ij}$ 的位置。

(2) 表值,这里的表值由两个数据项组成:

• $e(i, j)$. value 记录了位于第 i 行、第 j 列的成员 Agent $Magent_{ij}$ 的值,即该成员 Agent 满足约束值的个数,如果 $e(i, j)$. value=m,那么说明这个 Agent 完全满足任务要求的 m 个约束条件,是一个非常适合完成任务的 Agent;

• $e(i, j)$. violate 记录了位于第 i 行、第 j 列的成员 Agent $Magent_{ij}$ 是否已被挑选成为雇员 Agent 的候选者。$e(i, j)$. violate=0,表示未被挑选,而 $e(i, j)$. violate=1,则表示已被挑选,其他的 Cagent 将不能再把这个 Magent 作为候选者。

4.5 建组 Agent 本地行为规则库

建组 Agent 在环境中的行为需要遵循本地行为规则库的规定。行为规则是自治实体 Cagent 的核心,它用来指导每个 Cagent 如何动作,如何对从环境、黄页 Agent 和控制 Agent 收集到的信息进行反应。建组 Agent 的本地行为规则库定义如下:

4.5.1 较好移动(better-move)

建组 Agent 以概率 better-p 移动到一个比目前所在位置的成员 Agent 更好的某个成员 Agent 所在位置。

定义 2 找到一个更好的成员 Agent 所在位置 $e(x, j)$ 的函数 Ψ better 为:

Ψ better$(x,y)=j \mid j \in [\text{low}, |Dx|], e(x, j).\text{value} \geqslant e(x,y).\text{value}$

$$low=\begin{cases}1 & \text{当 } y-s\leqslant 1\\ y-s & \text{当 } y-s>1\end{cases}$$

其中,$e(x,y)$为当前建组 Agent 所在位置,x 表行号,y 表列号;s 代表搜索的步长;low 是当前搜索范围的下限。

4.5.2 最好移动(best-move)

建组 Agent 以概率 best-p 移动到一个在本次搜索范围内约束值最高的位置。

定义 3 找到本次搜索范围内最好的成员 Agent 所在位置 $e(x, j)$的函数 Ψ best 为:

Ψ best$(x,y)=j \mid j \in [low,top], (\forall t \in [low,top])\ e(x, j).\text{value} \geqslant e(x,t).\text{value}$

$$low=\begin{cases}1 & \text{当 } y-s\leqslant 1\\ y-s & \text{当 } y-s>1\end{cases}$$

$$top=\begin{cases}y+s & \text{当 } y+s<|D_x|\\ n & \text{当 } y+s\geqslant|D_x|\end{cases}$$

其中,$e(x,y)$为当前建组 Agent 所在位置,x 表行号,y 表列号;s 代表搜索的步长;low 是当前搜索范围的下限,top 为上限。

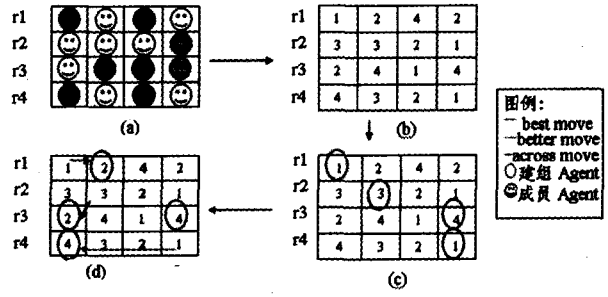
4.5.3 交叉移动(across-move)

建组 Agent 以概率 across-p,或者在已经无法找到更好成员 Agent 情况下,移动到一个新行,重新开始搜索。在选择新行的时候,它可以随机选择或者选择所有成员 Agent 的均值最高的那一行,同时,还应当避免选择已经有多个建组 Agent 存在的行数。

定义 4 找到本次搜索范围内最好的成员 Agent 所在位置 $e(x, j)$的函数 Ψ across为:

$$\Psi \text{ across}(x,y)=\begin{cases}i & \forall t[1,n]\sum_{j=1}^{|D_i|}e(i,j)>\sum_{j=1}^{|D_t|}e(t,j)\\ \text{Random}(n)\end{cases}$$

建组 Agent 的三种本地行为如图 4 所示。



(a)16 个物理环境中的真实 Agent 经映射和划分后构成的逻辑环境。(b)由成员 Agent 的约束值构成的 4×4 的表格——建组 Agent 存在的环境。(c)某时刻,4 个建组 Agent 在环境中的位置示意。(d)下一时刻,原 r1 行的建组 Agent 采取 better_move 移动到位置(1,2);原 r2 行的建组 Agent 采取 across_move 移动到位置(3,1);原 r3 行的建组 Agent 已经找到最优解,不移动;原 r4 行的建组 Agent 采取 best_move 移动到位置(4,1)。

图 4 建组 Agent 的本地行为示意

4.6 动态重构算法 ASDR

动态重构算法 ASDR(Agent System Dynamic Reconfiguration)用于提供 Agent 系统实现动态重构。它包括对模型中的各种 Agent 初始化、侦听系统状态、俘获重构请求、发起重构行为等工作。函数 Refresh 用于在动态情况下,对 Cagent 种群的大小、搜索任务、逻辑环境、雇员 Agent 和成员 Agent 状态进行更新。函数 OrganizationFormation 是在动态重构情形下进行建组的算法,在文[16]中有详细阐述。

Function ASDR

```
1.  While organization formation completed and task has not accomplished
2.    do task monitoring;
3.    if dynamic reconfiguration action put forward
4.      Refresh(Cagents,Eagents,environment);
5.      call OrganizationFormation;
6.    End if;
7.  End while.
```

5 实验结果及讨论

在假设成员 Agent 的个数分别为 50、100、1000、10000,建组 Agent 的个数分别为 4、10、50、100,三种行为选择概率(概率在 0.0～1.0 范围内)共 66 种不同组合的情况下,我们对 ADRM 模型的收敛和运行情况进行了实验。

实验表明，只要概率选取在合理的范围内，模型都能够收敛到比较好的解。这里比较好的解指的是最优解，或者是位于最优解附近一个很小范围内的解集范围内。找到较好解的概率 solution-p≥95%。

图 5 和图 6 是在 Magents=50，Cagents=4，max_time-step=100，better_p=0.6，best_p=0.2，across_p=0.2，foot=2，Max_eagent_value=12 时的一次运行情况。在某个时刻，50 个 Magents 中愿意提供服务的只有 32 个，这 32 个 Magent 及它们的约束值经黄页 Agent 划分后的二维表如表 1 所示。需要重构的成员 Agent 有 4 个，所以分派了 4 个建组 Agent 来寻找合适的解。

表 1　32 个 Magent 形成的逻辑环境，表中的数值为某 Magent 的约束值

1	3	5	7	9	11	12	10
2	3	4	5	12	11	7	9
4	3	5	5	9	12	12	12
11	2	3	5	7	8	1	9

图 5 表示在该次运行中 Cagent 找到的候选 Eagent 的约束值总和随时间步变化的情况。可以看到，在第 8 步的时候就收敛到了最优解——每个 Cagent 都找到了最满足要求的 Magent，即约束值为 12 的 Magent。

图 6 是 4 个 Cagent 随时间步的变化，在逻辑环境中的搜寻位置变化的情况。

同时，结合第 2 节例示的模拟实验表明，在任务和环境动态变化的情况下，采用模型 ADRM 和算法 ASDR 能够较好地解决 Agent 系统的动态重构问题。在模拟试验中，黄页 Agent 不断地接收未加入供货联盟的供应商 Agent 的各种申请，动态维持建组 Agent 的行为环境，并通过重构发起后的刷新保证了环境的一致性；建组 Agent 通过 Monitor 模块的监控，能够及时发现联盟中的供货商供货优劣情况，比如某个供货商的货物迟迟不到，那么它将申请 RC 发起重构行为，替换该不合格的供应商；RC 还能在货物供应过程中接受客户的要求，通过对供货商联盟（Agent 系统）进行重构，使得在很短的时间内，组建起一个新的满足用户需求的联盟。

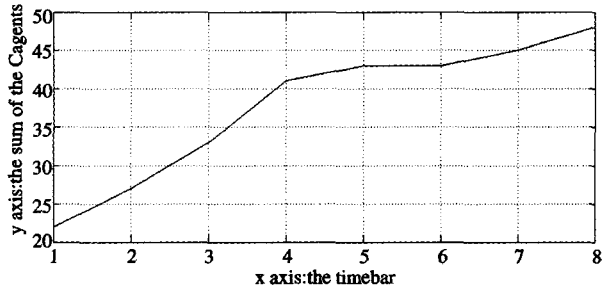


图 5　Cagent 种群找到的候选 Eagent 的约束值总和随时间步变化的情况

结束语　动态重构不仅是 Agent 领域中一个关键问题，类似问题亦是网格领域中一个亟待解决的难问题，被称为“虚拟组织的形成和管理”[22]。

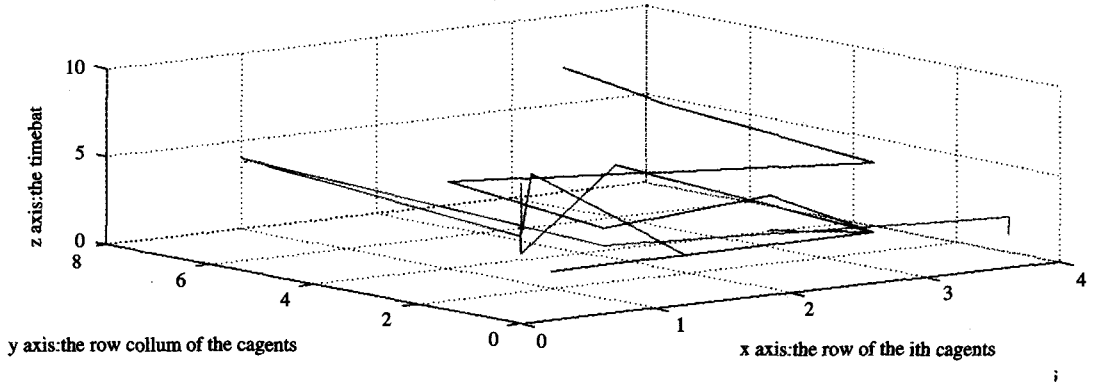


图 6　Cagents 在逻辑环境中的位置变化情况

本文结合供应链管理的背景，利用最新的可计算范型 AOC，提取并设计出了基于多 Agent 系统的动态重构模型 ADRM；提出了动态重构算法 ASDR；对模型和算法的有效性和收敛性进行了实验型研究，证明在参数设定恰当时，能够保证在可接受的时间范围内收敛。

ADRM 模型从宏观上实现了在任务不同和环境变化的情况下动态重构基于 Agent 的系统；配套的 ASDR 为 ADRM 模型的有效工作提供了技术和方法。在 Agent 领域，有了动态重构技术，Agent 系统可以适用于不同的任务和不断改变的环境。在某种意义，Agent 系统就具有了按需而变的学习和适应能力。在网格领域，动态重构技术可以对虚拟组织的形成和管理提供参考和有力支持。

尽管我们提出了基于多 Agent 系统动态重构的模型和算法，并得到了一些初步结果，模型中还有不少问题留待进一步解决。比如在模型中添加启发式机制，或者给 Cagent 增加推理能力，实现在不同情况下更理智、更恰当的行为选择；可以在模型中增加信任和信誉，防止某些成员 Agent 的欺骗和雇员 Agent 不负责任的随意退出行为；可以放松约束值的计算方法，通过约束条件和约束值的模糊，使约束值更恰当地体现不同资源间的差异。

参 考 文 献

1　Wooldridge M. An Introduction to Multi-Agent System. John Wiley and Sons Ltd, 2002

2　Zhang Z, Zhang C. An Agent-Based Hybrid Intelligent System for Financial Investment Planning. In: Proceedings of PRICAI 2002, LNAI-2471, Springer, August 2002. 255～364

3　Luck M, Mcburney P, Preist C. A Manifesto for Agent Technology: Towards Next Generation Computing. Autonomous Agents and Multi-Agent Systems, 2004, 9:203～252

（下转第 158 页）

息以及不完整的信息。因此，我们需要探讨在上述环境下的近似查询。本文考虑到多代理系统中不存在完整匹配的语义术语或者存在一些所谓的缺省信息和不完整信息的情况下如何实现正确的语义查询。Schaerf 和 Cadoli[18] 定义了一个良构的逻辑并提供了一个快速算法，用于近似推理。他们的查询方法特别依赖于具体参数的选择，这些参数是一个字母表的子集，用于确认近似逼近的程度。但是就这些参数来说，它们的哪些值能得到最好的逼近，是很难确定的。我们的方法可以通过术语的替换或者重写，将一个代理中不理解的术语通过协调交给另外一个代理去理解。我们的方法比较近似于文[19]的工作，但是我们的研究与它的区别主要有：1) 我们使用的是基于 OWL 本体语言的描述逻辑，OWL 目前已经成为标准。2) 通过上面的例子可以看出，我们的查询表达式中的概念可以具有概念并(disjunction)描述，而不是只具有交(conjunction)描述(文[19]只具有交描述)。3) 我们执行术语查询得到的所有结果（如果查询结果存在的话），一定是根据本体的语义而得到的用户想要的结果，因为替换术语的概念语义上比查询术语的概念是缩小的。而文[19]刚好相反，使用比查询概念语义范围上更大的概念替换查询术语，因此不能保证语义上总是正确的。

结论 本文提出了一种在多代理通讯情形下如何更好地实现代理之间的协调，为用户提供更好的查询能力。我们主要介绍了如何实现基于分布环境下的本地代理的查询，并提出了一系列相应的算法。值得注意的是，本文只是从代理通讯的本体级别考虑代理的通讯问题。但是实际上，多代理通讯至少包含本体级通讯、功能级通讯以及代理通讯语言 3 种层次的研究[20, 21]。我们只考虑第一种类型的通讯问题，从理论上提供了一种基于本体的代理通讯解决机制，并加以实现。

参 考 文 献

1 Baader F, et al. Rewriting concepts using terminologies. In: Proceedins of KR2000, Morgan Kaufmann, 2000
2 Uschold M. Barriers to effective agent communications. In: Proceedings of Ontologies in Agent Systems (OAS01), 2001
3 Steels L. The origins of ontologies and conmmunication conventions in multi-agent systems. Autonomous Agents and Multi-agent Systems, 1998(1):169～194
4 Hendler J. Agents and the Semantic Web. IEEE Intelligent Systems, 2001,16(2):30～37
5 Neches R, Fikes R E, Gruber T R, etal. Enabling Technology for Knowledge Sharing. AI Magazine, 1991,12(3):36～56
6 Studer R, Benjamins V R, Fense l D. Knowledge Engineering, Principlesand Methods. Data and Knowledge Engineering, 1998, 25(122):161～197
7 William S, Austin T. Ontologies. IEEE Intelligent Systems, 1999 Jan/Feb: 18～19
8 Chandrasekaran B, Josephson J R, Benjamins V R. What Are Ontologies, and Why Do We Need Them? 1999 Jan/Feb: 20～25
9 KMG. Using Protégé-2000 to Edit RDF: [Technical Report]. Stanford University, January 2001. http://protege. stanford. edu/protege-rdf/protege-rdf. html
10 Bechhofer S, et al. OilEd: a Reason-able Ontology Editor for the Semantic Web. In: Proceedings of KI2001, Joint German/Austrian conference on Artificial Intelligence, Vienna. Springer-Verlag LNAI, Vol 2174, 2001. 396～408
11 Sure Y, et al. 2002. OntoEdit: Collaborative Ontology Development for the Semantic Web. International Semantic Web Conference (ISWC02), LNCS, Vol. 2342, Sardinia, Italy, June 2002. 221～235
12 Hewlett-Packard Semantic Web Lab. Jena Semantic Web Framework. 2004. http://www. hpl. hp. com/semweb/
13 Ma Yinglong, Wu Kehe, Jin Beihong, et al. Approximate Semantic Query Based on Multi-Agent Systems. In: 2006 International Conference on Rough Sets and Knowledge Technology (RSKT2006), Lecture Notes on Artificial Intelligence, vol 4062, 2006
14 Selman B, Kautz H. Knowledge compilation and theory approximation. Journal of ACM, 1996,43(2):193～224
15 Stuchenschmidt H. Approximate information filtering with multiple classification hierarchies. International Journal of Computational Intelligence and Applications, 2002, 2(3):295～302
16 Stuchenschmidt H. Approximate terminological queries. In: Proceedings of FQAS02, 2002
17 Akahani J, et al. Approximate query reformulateion for ontology integration. In: Proceedings of ICWS2003
18 Schaerf M, Cadoli M. Tractable reasoning via approximation. Artificial Intelligence, 1995,74(2):249～310
19 Stuckenschimdt H. Exploiting Partially Shared Ontologies for Multi-Agent Communication. In: Proceedings of CIA'02, 2002
20 Payne R T, Paolucci M, Singh R, et al. Communicating Agents in Open Multi Agent Systems. In: First GSFC/JPL Workshop on Radical Agent Concepts (WRAC'02), 2002
21 Sycara K, et al. LARKS: Dynamic Mathmaking Among Heterogeneous Software Agents in Cyberspace. Autonomous Agents and multiagent Systems, 2002,5:173～203

(上接第 151 页)

4 Sycara K, Lu J, Klusch M, et al. Matchmaking among Heterogeneous Agents on the Internet. In: Proceedings of AAAI Spring Symposium on Intelligent Agents in Cyberspace, Stanford, USA, 1999
5 Hannebauer M. Autonomous Dynamic Reconfiguration in Multi-Agent Systems: Improving the Quality and Efficiency of Collaborative Problem Solving. LNAI 2427, Springer, 2002
6 De Palma N, Bellissard L, Riveill M. Dynamic Reconfiguration of Agent-Based Applications. In: Third European Research Seminar on Advances in Distributed Systems, Madeira Island (Portugal), 1999
7 Van der Hoek W, Wooldridge M. On the Dynamics of Delegation, Cooperation and Control: A Logical Account. In: Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems, Utrecht, the Netherlands, July 2005
8 Klusch M, Gerber A. Issues of Dynamic Coalition Formation Among Rational Agents on the Internet. In: Proc. 2nd International Conference on Knowledge Systems for Coalition Operations (KSCO), Toulouse, France, 2002
9 Lermann K, Shehory O. Coalition Formation for Large Scale Electronic Markets. In: Proceedings of the Fourth International Conference on MultiAgent Systems, IEEE Computer Society, Boston, 2000. 167～174
10 Griffiths N, Luck M. Coalition Formation Through Motivation and Trust. In: Proceedings of the Second International Joint Conference on Autonomous Agents and Multi-Agent Systems, Melboume, Australia, 2003
11 Wooldridge M, Jennings N. The Cooperative Problem Solving Process. Journal of Logic and Computation, 1999,9 (4): 563～592
12 Kraus S, Shehory O, Taase G. Coalition Formation with Uncertain Heterogeneous Information. In: Proceedings of the Second International Joint Conference on Autonomous Agents and Multi-Agent Systems, Melbourne, Australia, 2003
13 Lerman K, Shehory O. Coalition Formation for Large Scale Electronic Markets. In: Proceedings of the Fourth International Conference on MultiAgent Systems, Boston, 2000. 216～222
14 Liu J, Jin X, Tsui K. Autonomic Oriented Computing: From Problem Solving to Complex Systems Modeling. Springer, 2005
15 Liu J, Jin X, Tsui K. Autonomy Oriented Computing (AOC): Formulating Computational systems with Autonomous Components. IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans, 2005,35(6):879～902
16 Liu J, Han J, Tang Y Y. Multi-agent oriented constraint satisfaction. Artificial Intelligence, 2002,136(1):101～144
17 陶丽，张自力. 一种面向自治计算的启发式联盟形成算法. 已投计算机学报
18 Liu C, Foster I. A Constraint Language Approach to Matchmaking. In: Proceedings of the 14th International Workshop on Research Issues on Data Engineering (RIDE 2004), Boston, 2004
19 Liu J, Tang Y Y, Cao Y. An Evolutionary Autonomous Agents Approach to Image Feature Extraction. IEEE Transactions on Evolutionary Computation, 1997,1(2):141～158
20 Tsui K, Liu J. Evolutionary Diffusion Optimization. In: Proceedings of the 2002 Congress on Evolutionary Computation, Honolulu, Hawaii, 2002
21 Russell S, Norvig P. Artificial Intelligence: A Modern Approach (2nd edition). Prentice Hall, 2002
22 Foster I, Kesselman C. The Grid: Blueprint for a New Computing Infrastructure (2nd ed.), Morgan Kaufmann, 2004
23 Zhang Z, Zhang C, Zhang S. An Agent-Based Hybrid Framework for Database Mining. Applied Artificial Intelligence, 2003, 17(5-6):383～398
24 Zhang Z, Zhang C. Agent-Based Hybrid Intelligent Systems: An Agent-Based Framework for Complex Problem Solving. LNAI 2938, Springer, 2004