嵌入基于案例记忆机制的个人辅助 Agent

陈可佳[1] BARTHES A. Jean-Paul[2]
(南京邮电大学计算技术研究所 南京 210003)[1]
(UMR CNRS 6599, Heudiasyc, Université de Technologie de Compiègne, France)[2]

摘 要 个人辅助 agent 是一种能够帮助用户解决工作环境中特定任务的认知 agent。为个人辅助 agent 设计了一种记忆机制,用于表示和组织个人辅助 agent 的记忆。受基于案例推理思想的启发,构建了一个记忆处理器,以对记忆进行更新和管理。在多 agent 平台——OMAS 中实现了个人辅助 agent 原型系统。实验表明,具有记忆机制的个人辅助 agent 能更有效地执行用户任务。

关键词 辅助 agent,基于案例的推理,记忆组织包
中图法分类号 TP39 **文献标识码** A

Personal Assistant Agents with a Case-based Memory

CHEN Ke-jia[1] BARTHES A. Jean-Paul[2]
(Institute of Computer Technology, Nanjing University of Post and Communication, Nanjing 210003, China)[1]
(UMR CNRS 6599, Heudiasyc, Université de Technologie de Compiègne, France)[2]

Abstract Personal Assistant(PA) agents are cognitive agents capable of helping users handle tasks at their workplace. The paper concerned the design of a memory mechanism for PA agents, which leads to fairly complex cognitive agents. Inspired by the idea of Case-based Reasoning(CBR) method, a cognitive memory processor was built to monitor and update the memory of PA. A prototype of the PA agents was developed and implemented in our multi-agent platform named OMAS. Several experiments show the memory makes the PA agent more efficient in handling the tasks requested by the user.

Keywords Assistent agent, Case-based reasoning, Memory organization packets

1 简介

人机交互需要通过有效的用户界面。自 20 世纪 80 年代以来,研究者已设计一系列复杂的工具(如多个窗口、菜单、控件等)来实现用户界面。使用这些工具,用户可以亲自执行所有任务并能显式控制所有事件。然而,采用这种"直接操纵式界面",计算机只能被动等待用户指令。此外,随着信息处理技术的发展,计算机能够处理越来越多的复杂任务,人机界面的复杂度也日益增长(如可供使用的工具数目日益增加)。直接操纵式界面对于未经训练的用户来说不再是一种理想的交互方式,因此,一些研究者[1,2]提出了一种新型的人机交互方式,以解决这一问题。他们将智能 agent 引入人机交互领域,从而得到了个人辅助 agent(Personal Assistant Agent, PA),如图 1 所示。在某些文献[3]中,PA 也被称为界面 agent(Interface Agent)。

PA 是智能 agent 研究中最引人注目的子领域之一[4]。在早期的一些工作[5,6]中,PA 主要用于减少用户的工作负担,处理用户任务、隐藏任务的复杂性,学习并调整用户的偏好,以及与其他 agent 进行合作[7]等。PA 可以执行诸如"安排会议"、"处理电子邮件"等日常任务[4],或者"网页浏览"等较特殊的任务[8]。近几年,PA 领域得到了越来越多的关注,如 DARPA 于 2003 年开始启动 PAL(Personalized Assistants that Learns)计划,AAAI 于 2007 年组织了"智能助手的交互挑战"(Interaction Challenges for Intelligent Assistants)专题研讨会等等。

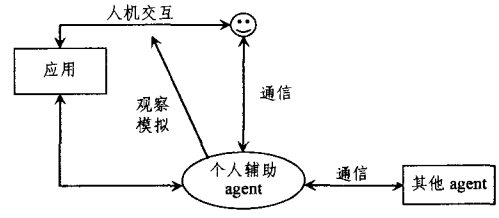

图 1 个人辅助 agent 原型[1]

PA 的设计和构建是一项艰巨的任务,需要整合推理、规划、调度、自然语言处理、多模态接口等不同的研究领域[9]。在早期,研究者主要研究 PA 对用户的偏好、目标的学习与适应。研究基于的假设是:一旦 PA 知道用户的需求,即可为用户提供有效的帮助。现实应用中,一个有效的 PA 还应具备多源、多类型的知识,能够通过自动推理来执行任务,并能从

到稿日期:2009-11-30 返修日期:2010-02-20
陈可佳(1980—),女,讲师,CCF 会员,主要研究方向为智能 agent、信息检索、数据挖掘等,E-mail:kejia.chen@gmail.com;**BARTHES A. Jean-Paul**(1950—),男,教授,主要研究方向为多 agent 系统、知识表示等。

失败经验中学习以适应新的任务[10]。没有记忆机制的 PA 将无视过去的经验，当处理与已完成任务相似的新任务时，它只能从零开始重新处理这一任务。因此，为 PA 构建记忆机制非常重要。

为 PA 构建一个记忆机制，增加了 PA 结构的复杂性，但也得到一个更加智能且有效的 PA。本文提出了一种基于认知模型的记忆机制，并将此整合到 PA 的对话机制中，从自然语言对话中以及任务完成的结果中自动获取和形式化当前 PA 的状态。为了解释这一模型的原理，本文首先描述实验中使用的 PA 框架结构和行为机制。

2 研究背景

法国 UTC(Université de Technologie de Compiègne)的 Heudiasyc 实验室研制了一种能使用自然语言与用户进行交互的 PA 模型[11-13]。这一认知 agent 使用了一种支持 SCA (Structured Cognitive Agent)的特殊平台。平台上所有的 agent 都是完全独立的，可以在任何时候进入或退出平台。agent 的技能模型、认知本体、任务模型和对话机制均嵌入在 agent 的结构中。

本文根据 SCA 构建的 PA 模型如图 2 所示。

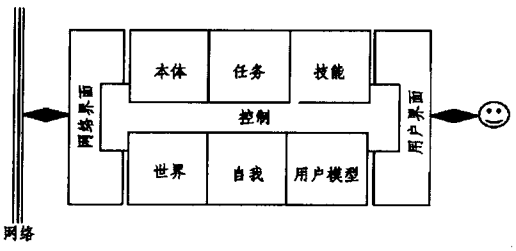

图 2 SCA 中的 PA 模型

PA 的各组成部分如下：

网络界面，将 PA 与网络中的其他 agent(服务 agent 或者 PA)连接；

技能模块，定义并实现 PA 的行为；

本体模块(ontology)，包含 PA 的各种认知本体(agent 本体、领域知识本体、任务知识本体等)，所有本体及知识库均使用一种框架(frame)语言 MOSS[1)]进行表示；

世界模块，表示网络中其他 agent 的信息，可自动更新；

任务模块，表示 PA 正在执行的任务；

自我模块，是 PA 的核心部分，描述了 PA 的技能、目标和记忆机制；

用户界面，采用文本、语音等多模式通信与用户对话；

用户模型，让 PA 知道如何将任务库中的任务转给其他 agent 处理、每种任务对应何种会话模块，这一层目前不处理用户的偏好问题；

控制模块，包含若干调整 PA 行为的函数，由多线程实现。

3 记忆机制

记忆表示方式的选择应符合以下原则：表示语言(representation language)应足够充分，记忆的存储开销和计算开销均应足够小，此外对记忆的检索应较为容易。人工智能领域提供了多种知识表示机制，例如框架(frames)、剧本(scripts)、语义网络、谓词、对象、规则等[14]。之前本文初步开发的 PA 使用了一个存放特征的堆栈作为记忆模块，为会话提供一个局部的上下文。这种短期记忆(short-term memory)可以确保会话的连续性。然而，对于构造一个长期记忆(long-term memory)，堆栈形式则显得过于有限。

本文注意到 PA 的每一段子会话都对应了一个任务的完成情况，受基于案例推理[15,16](Case-Based Reasoning，CBR)思想的启发，本文很自然地选择了一种基于案例的记忆表示方法。本文的主要目的是探讨构建记忆机制的可能性，而非如何利用案例解决问题。为了构建记忆机制，本文选择了一种动态记忆模型，即 Schank 等人[17]提出的记忆组织包(Memory Organization Packets，MOPs)模型。MOPs 定义了一个语义网格，本文在此基础上添加了索引以便于检索。下面详细介绍这一方法。

3.1 MOPs 结构

在动态记忆模型中，领域知识和事件(episodes)都是以 MOPs 的格式来表示和组织的。一个 MOP 可以由一组范数(norms)表示，作为此 MOP 的基本元素，包括正发生的事件、待实现的目标、事件的执行者等。

MOPs 主要分为两类：实例 MOPs 和抽象 MOPs。前者涉及一个具体的事件，后者描述若干类似的实例 MOPs。本文中的“案例”(case)通常指实例 MOPs，用来描述一个事件或者一个问题得到解决的情况。所有的 MOPs 之间由两种关系连接：抽象关系(abstraction relations)和打包关系(packaging relations)。其中，打包关系由一组角色(role)和填充值(filler)组成。前者一般是属性名；后者可以是一个属性值，也可以是另一个 MOP。图 3 展示了一个办公 PA 的记忆片段。办公 PA 能够执行诸如电子邮件管理、会议管理等日常办公任务。图 3 中，除了底部的一个实例 MOP“Send-Email-Case.1”之外，所有的方框均表示抽象 MOPs。箭头表示抽象关系，直线表示打包关系。一个实例 MOP 也可以通过抽象关系连接到某个抽象 MOP(例如 MOP “Send-Email-Case”是 MOP “Send-Email-Case.1”的抽象)。值得指出的是，在记忆活动过程中，实例 MOPs 一旦产生，就必须通过模式匹配的方法逐步精化到 MOPs 结构的最低层。

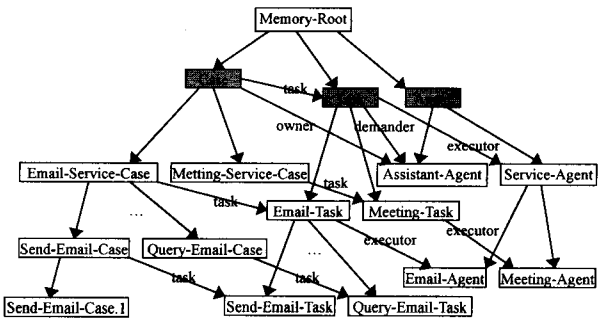

图 3 PA 的一个基于 MOPs 的记忆片段

3.2 插入一个新案例

在 PA 与用户的交互过程中，新案例能自动载入到记忆中，并导致记忆结构的调整。

图 4 中，PA 当前处理的任务是“给 Mike 发送一封电子邮件”。经过分析，PA 将任务表示为 MOP“Send-Email.16”。在 MOP“Case”下，立即产生一个对应这一任务的新 MOP

1) 详细文档可查 http://www.utc.fr/barthes/MOSS/

"Case. 42"。随后,这一 MOP 精化为 MOP"Email-Service-Case"下的 MOP"Email-Service-Case. 38",然后又进一步精化为 MOP "Send-Email-Case"下的 MOP"Send-Email-Case. 16"。新案例最终由 MOP"Send-Email-Case. 16"表示并存放到记忆中。

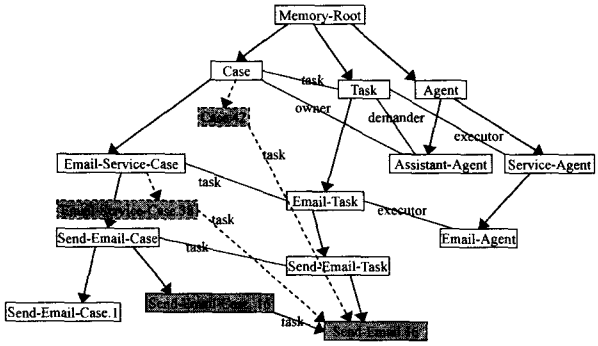

图 4 一个新案例被动态载入记忆结构

随着记忆结构的动态更新,PA 还同时执行了案例的检索(或称为"回忆")。根据抽象关系,PA 搜索出与新案例类似的旧案例,它是与新案例的打包关系描述相兼容的所有 MOPs 中最特殊的一个。

3.3 MOPs 结构的优势

作为一种记忆建模方法,MOPs 既是记忆中知识的组织者,也是知识的处理者。一组 MOPs 围绕一个特殊主题用多种链结组织在一起,能够把记忆网络结构中的知识和经验进行有效分类。因此,MOPs 的组织结构决定了在记忆过程中可以获取的信息。

此外,MOPs 不是一个静态结构,而是可动态调整的结构。记忆一旦使用,便会发生改变。通过模式匹配,我们可以找到与一个输入兼容且最特殊的子 MOP。如果输入的描述不存在于实例 MOPs 中,则将创建一个新的实例 MOP。如果一个 MOP 下的若干子 MOPs 拥有相同的描述,那么可创建一个包含这些描述的抽象 MOP。如果一个抽象 MOP 的大部分实例 MOP 移到这个抽象 MOP 的某一个子抽象 MOP,那么可删除或合并结构中的一些 MOPs。

4 实现

这一部分详细介绍 PA 记忆机制的实现,以及如何在 PA 模型中整合此记忆机制。

4.1 MOPs 表示

本文中基于 MOPs 的记忆结构与本体、知识库使用了同样的表示形式。表 1 定义了 MOP"Case"(定义为一个"概念"),这是整个记忆的核心。

表 1 以 MOSS 形式表示的"Case"

```
(defconcept "Case"
  (:doc "A CASE represents an episode of memory.")
  (:att "time")
  (:rel "owner" (:to "Assistant Agent"))
  (:att "term" (:one-of :short :long))
  (:att "rtag")
  (:att "rcount")
  (:rel "task" (:to "Service Task")))
```

在"Case"下,可用":is-a"域定义新的 MOPs。这些 MOPs 形成了一个关于电子邮件管理任务的 MOPs 分层结构和一个关于会议管理任务的 MOPs 分层结构。例如表 2 中,"Email Service Case"是一种"Case","Send Email Case"是一种"Email Service Case"。MOPs 在结构中的位置越低,所涉及的任务也越精确。因而,在 PA 的记忆中,还需要对"任务"进行表示(见表 2),用于填充案例的内容。

表 2 "Case"下若干子 MOPs 的表示

```
...
(defconcept "Email Case"
  (:is-a "Case")
  (:rel "task" (:to "Email Task")))
(defconcept "Send Email Case"
  (:is-a "Email Case")
  (:rel "task" (:to "Send Email")))
...
(defconcept "Meeting Case"
  (:is-a "Case")
  (:rel "task" (:to "Meeting Task")))
(defconcept "Arrange Meeting Case"
  (:is-a "Meeting Case")
  (:rel "task" (:to "Arrange Meeting")))
...
```

表 3 中的"Service Task"与"Case"一样使用了分层结构进行定义。其中,"Task Arg"定义了完成某类任务所需要的参数。"weight"属性表示某个任务参数的重要性,用于案例相似性的计算。

表 3 "Service Task"和"Task Arg"的表示

```
(defconcept "Service Task"
  (:rel "executor" (:to "Service Agent"))
  (:rel "applicant" (:to "Assistant Agent"))
  (:att "result" (:one-of :success :failure))
  (:att "return value")
  (:att "fail reason")
  (:rel "target" (:to "Universal Class"))
  (:rel "arg" (:to "Task Arg")))
(defconcept "Task Arg"
  (:doc "A TASK-ARG is an argument required or optional to the task.")
  (:att "name")
  (:att "val")
  (:att "weight"))
```

此外,agent 本身的信息对于任务和案例的表示也很重要(见表 4):

每个案例中的任务通常由一个服务 agent 执行;

为某个用户服务的 PA 可与若干服务 agent 连接;

某一案例涉及的 agents 可以是远程的或者本机的。

表 4 "Agent"的表示

```
(defconcept "Agent"
  (:att "name")
  (:att "position" (:one-of :local :remote)))
(defconcept "Assistant Agent"
  (:doc "An ASSISTANT-AGENT is a personal assistant agent to help user.")
  (:is-a "Agent")
  (:rel "master" (:to "Person"))
  (:rel "staff" (:to "Service Agent")))
(defconcept "Service Agent"
  (:doc "A SERVICE-AGENT owns a domain-related knowledge and processes domain-related tasks.")
  (:is-a "Agent")
  (:rel "skill" (:to "Skill")))
```

4.2 记忆过程管理

本文为 PA 构建的记忆机制主要由两个基本元素构成：①基于案例的记忆体，②记忆处理器，如图 5 所示。

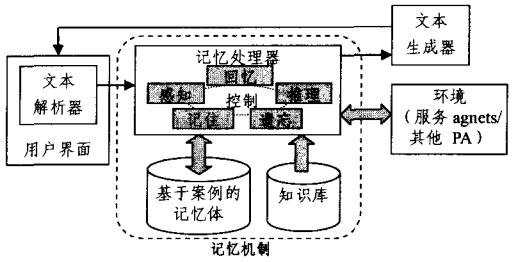

图 5 PA 记忆机制结构图[18]

4.2.1 基于案例的记忆体

短期记忆和长期记忆在记忆过程中起到中心作用，体现了 PA 的认知能力。短期记忆存在于工作记忆（working memory）中，为 PA 提供当前任务的上下文。与用户交互的过程中，PA 一旦接收到信息，便会激活短期记忆。长期记忆存储了 PA 过去的经验，有助于进行回忆、学习、推理等认知活动。

因此，基于案例的记忆体中的事件案例（episodic cases）也分为两种类型：①短期案例（short-term cases，STC）；②长期案例（long-term cases，LTC）。案例的类型由 MOP“Case”中的属性“term”表示。一个新案例首先产生于短期记忆中，若案例最终成功保留在记忆里，它将转化为长期案例。

4.2.2 记忆处理器

记忆机制的第二个组成部分为记忆处理器，如图 6 所示，用于检索、处理案例和更新记忆结构。记忆处理器由 5 个认知子函数和 1 个引擎组成。参考人类记忆过程，本文实现了 5 个认知子函数：感知、回忆、推理、记住和遗忘。它们的构建类似于 CBR 的经典 4R 循环过程（Retrieval，Reuse，Revision 和 Retention）[19]。根据 PA 当前状态，引擎将激发某认知子函数。

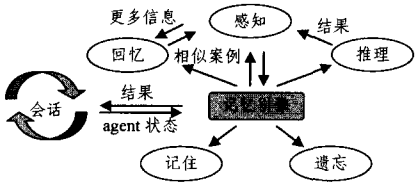

图 6 记忆处理器中的认知活动

一旦 PA 获得用户或其他 agent 发送的消息，它的感知函数就得到激发。如果消息涉及一个新的任务，则产生一个短期案例。否则，PA 检索出刚才的一个短期案例并使用当前环境信息扩充这一案例。若 PA 已收集了这一任务的足够必要信息，它将启动记忆函数和推理函数。

记忆函数在记忆处理器中起关键作用，用于发现 PA 记忆体中与当前案例相似的案例。本文提出了一种分段搜索方法：1）先使用动态记忆过程发现一个抽象 MOP。此 MOP 包含与当前情况最为匹配的特征（例如任务类型、任务请求者等）；2）选择此 MOP 下最相似的实例 MOP（或事件案例）。本文采用加权最近邻（Nearest Neighbor）的方法来计算事件案例间的距离。如果未发现相似案例，则从零开始处理此任务。

若发现了一个相似的旧案例，PA 启动推理函数。对于不同的任务，如何使用这些旧案例也是不同的。举例来说，如果回忆到的案例涉及了一个如“获取 email 地址”的查询任务，则 PA 检查旧案例中任务的结果，看其是否是当前案例中任务预期的答案。如果旧案例涉及的任务为“安排会议”，PA 将检查旧案例和当前案例中的“会议时间”信息，以避免时间冲突。

一旦当前任务结束，PA 就启动记住函数。任务的短期案例将转换成长期案例并保存在 PA 的记忆体中。

遗忘函数有规律地删除一些无用的（重复性的）或者过时的经验，以维持基于案例记忆体的正确性和检索效率。目前，本文将那些重复性的案例作为无用的案例，将包含某些不再正确的信息的案例作为过时的案例。

记忆引擎是控制组件。一旦启动，记忆引擎就会分析 PA 当前状态里的数据，包括一个会话的当前状态（即任务的起始、中间或结束状态）和会话中任务的描述数据。分析结果决定数据应被传往何处：一个任务刚刚开始处理时传给感知函数，如图 7 所示，或者当任务完成时传给记住函数和遗忘函数。

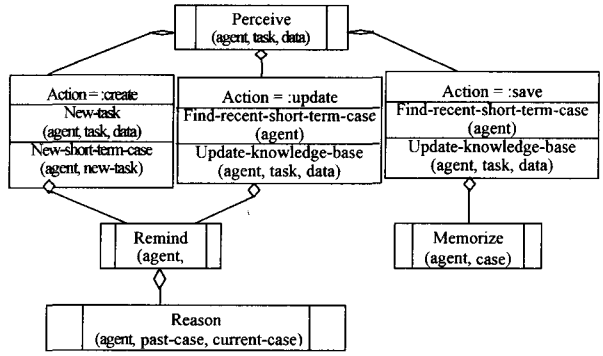

图 7 感知函数实现

4.3 记忆机制的整合

记忆机制一旦构建，就整合到 PA 的会话模块中。在本文实现中，当开始一个新的会话时，PA 启动记忆机制。记忆机制在整个会话过程中都处于激活状态。

本文的 PA 有一个文本分析器[12]，可以通过分析用户语言来识别任务类型。一旦任务类型被识别，PA 就进入此任务对应的会话入口，激发一个或多个子会话来处理这一任务。例如图 8 为一个“获取 email 地址”会话的状态图。会话由一系列的状态组成。一旦进入一个状态，PA 就调用“=execute”方法。如果在这个状态中，PA 向用户或其他 agent 提问，被问题的回答将由“=resume”方法来处理。如果 PA 没有提问，在“=execute”方法中将实现一次状态转变。PA 的记忆引擎就整合在“=execute”方法和“=resume”方法中。

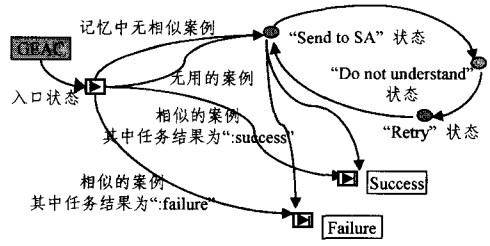

图 8 GEAC(_get-email-address-conversation)会话的状态图

在会话中，PA 从用户语言中抽取任务数据，进行简单的

文本处理。将其中的代词性引用(如"her","it"等)由前面案例中的相应实体来代替。对于人物性代词,实体是案例中相关"person"概念的任务参数值。对于物性代词,实体是案例中的任务参数"target"的值。

4.4 记忆机制与 PA 其他模块的关系

记忆机制在自我模块中建立,与其他模块也有密切的关系,特别是本体模块、世界模块和用户界面。

4.4.1 记忆机制与本体模块

本文中,PA 的本体(如图 9 所示)有 3 个作用:①帮助抽取消息中的信息;②帮助选择一个服务 agent 处理用户任务;③构成记忆结构的基础信息。

记忆的构建基于本体,记忆中对事件案例的定义使用了本体中的概念,并且这些事件案例可能涉及到基于本体知识库(由概念的实例组成)中的某些事实或个体。

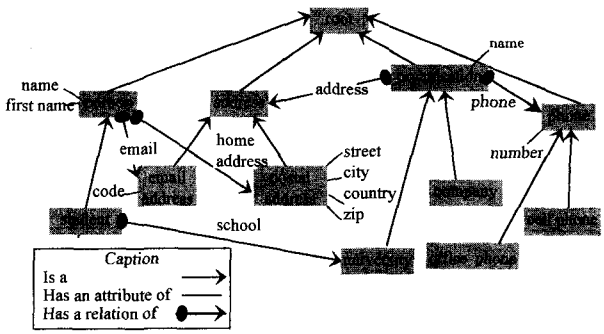

图 9 一个 PA 中的本体结构

此外,记忆中的所有信息都必须能被 PA 获知和理解。因而,在记下一个新案例时,PA 应该更新其知识库,增加从环境感知的新知识。例如,假设用户查询一个叫"Kelly"的人的个人信息。开始,PA 不认识此人,将任务发送给一个服务 agent 执行。后来,PA 将这一案例保存在记忆中,记录下这一新知识,并更新知识库,即构建一个"person"概念下的实例,人名为"Kelly"。

4.4.2 记忆机制与世界模块

PA 内部需要登记系统中其他 agent 的信息,将世界模块中关于"谁能做什么"的知识逐步更新至 PA 的知识库中,并保存在 PA 的记忆机制里。例如,委托给 PA"Assistant-Agent. 1"的任务"Send-Email-Task. 1"可由"Assistant-Agent. 1"的助理 agent——服务 agent"Email-Agent. 1"来执行,如图 10 所示。

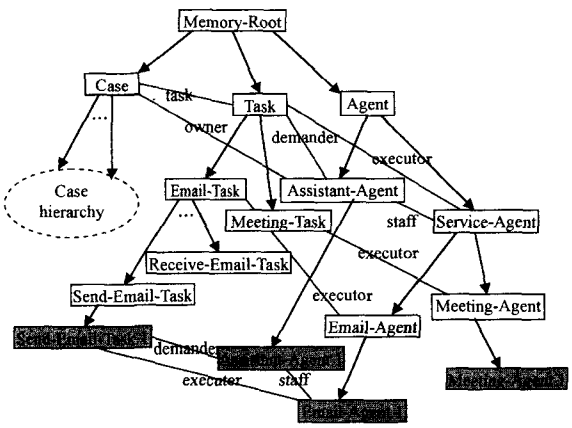

图 10 "谁能做什么"知识结构

一旦识别出任务,PA 就首先检查是否存在能处理这一任务的 agent(远程的某个 PA 或者本地的服务 agent)。否则,PA 将在记忆中查询近期是否存在一个成功完成这类任务的案例。对于"谁能执行这类任务",这样的案例可提供准确的答案。

4.4.3 记忆机制与用户界面

记忆机制仅在 PA 从环境中接受信息时才启动。因此,4.3 节介绍的记忆机制需要整合到用户界面模块,尤其是会话机制中。此处不再赘述。

4.5 Agent 平台

PA 在多 agent 平台 OMAS(Open Multi-Agent System)[11] 中实现,平台中所有的 agents 在 Allegro Common Lisp® Version 8.1 环境下实现。OMAS 平台是一个研究平台,目标在于使用认知 agent 设计和实现一些应用。OMAS 中存在 3 种 agent 模型:①服务 agents(service agents,SA);②个人辅助 agents(personal assistant agents,PA);③用于平台间通讯的传递 agents(transfer agents,XA)。服务于同一个 PA 的服务 agents 构成以这个 PA 为中心的团队[20],通常在用户主机上运行。其他的 SAs 和 XAs 通常运行于不同的机器。PA 的基于案例的记忆体使用了知识表示语言 MOSS 来表示。MOSS 语言是一种基于 PDM(Property Driven Model)模型的面向对象语言[21]。

5 测试

在研究中,人们最常构建的是具备电子邮件管理和会议管理功能的 PA,通过实验测试了它的有效性。本文的实验初步使用了 3 个 agent:一个具有记忆机制的 PA 和与之相连的两个 SAs。PA 设计为一个传统的办公助理,以帮助用户处理日常工作:收发电子邮件、安排会议、查询会议信息等。SAs 包含一个邮件 SA 和一个会议 SA。邮件 SA 具有查询电子邮件地址、发送和接收电子邮件的技能。会议 PA 具有组织会议、取消会议和查询会议的技能。

本文在 OMAS-MOSS V7.0 系统中对记忆 PA 和无记忆 PA 进行比较实验。结果表明,在任务处理方面、在可能场景中与用户交互方面,记忆 PA 比无记忆 PA 更为显著有效。这些改善主要在于帮助解决新问题,对可能的失败给以警告,以及处理语言中的代词等。

本文的实验结果为用户与记忆 PA 交互的几组精简后的会话。表 5 显示记忆 PA 执行对电子邮件地址的查询,并成功识别人物代词。无记忆 PA 不能用过去案例帮助解决问题,且无法识别代词所指。表 6 显示了记忆 PA 成功地识别一个对象代词。表 7 显示了记忆 PA 能够对"会议时间冲突"问题给出警告。

表 5 查询"Kejia"的电子邮件地址的会话

…
User: Please arrange a meeting for tomorrow 10am.
Agent: Who will attend the meeting?
User: Kejia and kelly.
…
User: What is Kejia's email address?
Agent: As far as I recall, the email address you want should be: ("kejia.chen@utc.fr").
Agent: Would you like to do something else?
User: Send a mail to her.
Agent: What is the object?
…

表 6　识别物代词"it"

…
User: Do I have a meeting tomorrow?
Agent: Yes. You have meetings:
("time: 10; Attendees: Kejia; Room: C230; Topic: About thesis modification.")
User: I want to cancel it.
Agent: Well, this meeting has been cancelled.
…

表 7　"安排会议"任务中的一个警告信息

…
User: Please arrange a meeting for tomorrow 10 am.
Agent: Who will attend the meeting?
User: Kejia and Kelly.
Agent: Where will be hold the meeting?
User: Room C230.
Agent: What is the meeting topic?
User: About thesis modification.
Agent: This meeting has been arranged. (A new case has been added in the memory)
Agent: What else can I do for you?
…
User: Please arrange a meeting for tomorrow 10am.
Agent: Sorry, you have arranged a meeting at that time. What about anthor time?
User: At 3 pm.
Agent: Who will attend the meeting?
…

6　相关工作

近期，一些研究者研究了对 PA 的经验的存储记忆问题：

Vere 和 Bickmore 创建了 HOMER agent[22]。HOMER 有一个有限的记忆体(一组之前的会话)，仅仅用于回答涉及 HOMER 过去经验的问题。

Lashkari 等人[7]选择了基于记忆的推理(Memory-Based Reasoning)的方法来捕捉用户模型。每一组用户行为和相应的情形都记录在 agent 的记忆中。收集记忆中最接近的匹配情况之后，agent 在新的情况中对一个行为进行预测。

Ferret 和 Grau[23]提出一个称为 MLK(Memorization for Learning Knowledge)的系统，将特殊的情况以概念图(conceptual graphs)的形式组织到记忆中。MLK 用于解决理解和学习的问题，尤其在缺少领域知识的时候。

Lerman 和 Galstyan[24]将 agent 的历史行为合并到一个多 agent 系统的数学模型中。agent 使用记忆，从 agent 状态和过去遭遇的情况环境来评估系统的全局状态。

有 3 个项目更接近本文的工作：

Nguyen[24]研发的个人辅助 agent，提出了和本文 PA 相类似的服务，但并未建立一个结构化的记忆模型。他们的 agent 建于 BDI 模型，可以进行一个非直接的商务会话。

Koide 和 Yamauchi[26]为一个界面 agent 实现了一个基于案例的记忆。此界面 agent 能够区分有经验和无经验的状态。然而，这一 agent 只能工作于一个非常特殊的领域，并且不能与多 agent 环境交互。

Guzzoni 等人[27]在 DARPA PAL 计划的 CALO 项目中提出的方法也是基于 BDI 模型。然而，这一项目意在通过使用 wizzards 和图形编程研发一种认知系统。与这一项目相同，本文得到的 agent 系统也是完全整合的。

结束语　本文研究工作表明研究一个一般性的记忆机制是可行的。本文工作包含了案例的表示、组织、产生和使用，并建造了一个短期记忆和一个长期记忆。

记忆机制增加了 PA 的复杂度，目前这种代价是可以忍受的。然而，我们需要在更多复杂的情况下进行实验，而不仅仅局限在这一领域中使用一般性的情况来测试方法。在本文的 agent 模型(SCA)框架中，对于新的领域应用需要扩展 PA 的本体和额外的子对话。这需要在今后工作中得到改进。

将来工作还应包含 PA 的学习问题，主要包括两种学习：①通过交互，增加和更新 PA 的知识。这一学习在文中已经实现；②规则的泛化问题，例如发现任务之间的规律。这样 PA 可以预测未来任务，甚至任务的解决方案。我们可以用机器学习方法来实现这一类学习。

参 考 文 献

[1] Maes P. Agents that Reduce Work and Information Overload[J]. Communications of the ACM, 1994, 37(7): 30-40

[2] Lieberman H. Autonomous Interface Agents[C]// Proc. of the ACM Conference on Computers and Human Interface. Atlanta, GA, March 1997: 67-73

[3] Middleton S E. Interface Agent: A Review of the Field[R]. EC-STR-IAM01-001. School of Electronics and Computer Science, University of Southampton, 2002

[4] Modi P J, Veloso M, Smith S F, et al. CMRadar: A Personal Assistant Agent for Calendar Management[C]// Proc. of the 6th International Workshop on Agent-Oriented Information Systems. Riga, Latvia, June 2004: 134-148

[5] Maes P, Kozierok R. Learning Interface Agents[C]// Proc. of the 11th National Conference on Artificial Intelligence. Washington, DC, July 1993: 459-465

[6] Mitchell T, Caruana R, Freitag D, et al. Experience with a Learning Personal Assistant[J]. Communications of the ACM, 1994, 37(7): 81-91

[7] Lashkari Y, Metral M, Maes P. Collaborative Interface Agents [C]// Proc. of the 12th National Conference on Artificial Intelligence. Seattle, WA, Aug. 1994: 1-6

[8] Lieberman H. Letizia: An Agent that Assists Web Browsing[C]// Proc. of the 14th International Joint Conference on Artificial Intelligence. Montreal, QB, Aug. 1995: 924-929

[9] Guzzoni D, Baur C, Cheyer A. Modeling Human-Agent Interaction with Active Ontologies[C]// AAAI Spring Symposium of Interaction Challenges for Intelligent Assistants. Stanford University, CA, USA, AAAI Press, 2007: 52-59

[10] Wobcke W R, Ho V, Nguyen A, et al. A BDI Agent Architecture for Dialogue Modeling and Coordination in a Smart Personal Assistant[C]// Proc. of the IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology. Compiègne, France, Sept. 2005: 323-329

[11] Barthès J-P A, Ramos M. Agents Assistants Personnels dans les Systèmes Multi-Agents Mixtes-Réalisation sur la Plate-forme OMAS[J]. Technique et Science Informatiques, 2002, 21(4): 473-498

[12] Enembreck F, Barthès J-P A. Architecture d'un Système de Dialogue avec un Agent Assistant[C]// 15ème Conférence Francophone sur l'Interaction Homme-Machine. ACM international, New York, 2003: 95-105

(下转第 192 页)

从图3中可以看出，在随机噪声干扰方差为0.5的条件下，改进自适应粒子群算法(VAPSO)的定位均方误差RMSE在1.5m以内。为了进一步分析该算法与标准粒子群算法(PSO)、变异自适应粒子群算法(MPSO)和速度自适应粒子群算法(VPSO)的差别，对其作平均适应度分析，结果如图4所示。

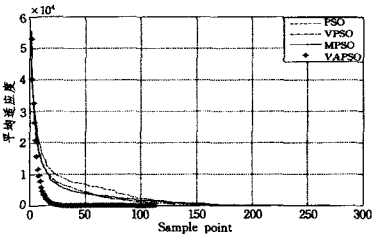

图4 平均适应度随迭代次数变化关系曲线

从图4可以看出，VAPSO的收敛速度最快，PSO算法的收敛速度最慢，而MPSO和VPSO的收敛速度基本一致。进一步从误差均值、误差方差、计算量方面进行比较分析，如表1所列。

表1 目标定位求解算法指标比较

定位算法	均方误差均值	均方误差方差	计算量(s)
PSO	0.8709	0.1643	15.635974
VPSO	0.8266	0.0993	10.717399
MPSO	0.8402	0.1115	16.609592
VAPSO	0.7711	0.0936	7.305790

从表1可以看出，改进后的自适应粒子群算法均方误差均值最小，方差最小，计算量比PSO减少一半，整体性能更好，更容易在短时间内达到较高的定位精度。

结束语 本文提出了速度自适应和变异自适应融合的改进自适应粒子群算法，并将其应用到目标定位研究中，同时分析了该方法与其它改进方法的区别。从分析结果可以得出，改进后的自适应粒子群算法收敛速度快，定位精度高，计算量小。因此，利用改进后的自适应粒子群算法对空间目标进行定位，能有效地获取目标的空间位置。同时，该方法对移动通信等其它定位问题也具有一定的应用价值。

参考文献

[1] 魏秀业，潘宏侠. 速度自适应粒子群优化算法在故障诊断中的应用[J]. 太原理工大学学报，2009，40(1)：47-50

[2] 蔡昭权，黄翰. 自适应变异综合学习粒子群优化算法[J]. 计算机工程，2009，35(7)：170-171

[3] Yun S，Lee J，Chung W，et al. A soft computing approach to localization in wireless sensor networks[J]. Expert Systems with Applications，2009，36：7552-7561

[4] 唐勇，王兴春. 基于粒子群优化算法的空中目标定位[J]. 指挥控制与仿真，2007，29(4)：31-32

[5] Cura T. Particle swarm optimization approach to portfolio optimization[J]. Nonlinear Analysis：Real World Applications，2009(10)：2396-2406

[6] 徐刚，瞿金平，杨智韬. 一种改进的自适应粒子群优化算法[J]. 华南理工大学：自然科学版，2008，36(9)：6-10

[7] 龚纯，王正林，等. 精通MATLAB最优化计算[M]. 北京：电子工业出版社，2009，4：303-306

[8] 李如琪，周媛媛. 自适应变异粒子群算法在输电网规划中的应用[J]. 广东电力，2008，21(12)：18-22

(上接第189页)

[13] Paraiso E C，Barthès J-P A. Une Interface Conversationnelle pour les Agents Assistants Appliqués à des Activités Professionnelles[C]// Actes IHM 04. Namur，Belgique：ACM Press，2004：243-246

[14] Davis R，Shrobe H，Szolovits P. What is a Knowledge Representation? [J]. AI Magazine，1993，14(1)：17-33

[15] Lopez de Mantara R，McSherry D，Bridge D，et al. Retrieval，Reuse，Revision and Retention in Case-Based Reasoning[J]. The Knowledge Engineering Review，2006，20(3)：215-240

[16] Kolodner J L. Case-based Reasoning[M]. San Mateo，CA：Morgan Kaufmann，1993

[17] Riesbeck C K，Schank R C. Inside Case-based Reasoning[M]. Hillsdale，NJ：Lawrence Erlbaum，1989

[18] Chen K-J，Barthès J-P. Giving an Office Assistant Agent a Memory Mechanism[C]// Proc. of the 7th IEEE International Conference on Cognitive Informatics. Stanford University，CA，IEEE CS Press，2008：402-410

[19] Watson I，Marir F. Case-based Reasoning：A Review[J]. Knowledge Engineering Review，1994，9(4)：355-381

[20] Tacla C A. De l'Utilité des Systèmes Multi-Agents pour l'Acquisition des Connaissances au Fil de l'Eau[Z]. Thèse de Doctorat，Université de Technologie de Compiègne，France，2003

[21] Shen W，Barthès J-P A. Description and Applications of an Object-oriented Model PDM[Z]. Modeling Complex Data for Creating Information：Real and Virtual Objects，1995：15-24

[22] Vere S A，Bickmore T W. A Basic Agent[J]. Computational Intelligence，1990，6：41-60

[23] Ferret O，Grau B. An Aggregation Procedure for Building Episodic Memory[C]// Proc. of the 15th International Joint Conference on Artificial Intelligence. Nagoya，Japan，1997：280-285

[24] Lerman K，Galstyan A. Agent Memory and Adaptation in Multiagent Systems[C]// Proc. of the 2nd International Joint Conference on Autonomous Agents and Multi-Agent Systems. Melbourne，Australia，July 2003：797-803

[25] Nguyen A. An Agent-based Approach to Dialogue Management in Personal Assistants[D]. Sydney，Australia：School of Computer Science and Engineering，U. of New South Wales，2007

[26] Koide S，Yamauchi S. Interface Agent for Process Plant Operation Towards the Next Generation Interface[C]// Proc. of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics. Atlanta，GA，Sept. 1999：197-202

[27] Guzzoni D，Baur C，Cheyer A. Modeling Human-Agent Interaction with Active Ontologies[C]// AAAI Spring Symposium of Interaction Challenges for Intelligent Assistants. Stanford University，CA，USA：AAAI Press，2007：52-59