[文章编号] 1004- 0609(2001)06- 1104- 05

基于 Multi-agent 的分布式专家系统协作机制①

景广军[1], 李松仁[2], 陈松乔[3]
(1. 南京大学 计算机科学与技术系, 南京 210093; 2. 中南大学 矿物工程系, 长沙 410083;
3. 中南大学 信息科学与工程学院, 长沙 410083)

[摘 要] 探索了基于 Internet/ Intranet 松耦合环境设计分布式选矿专家系统 DMPES, 采用面向 Agent 的系统开发方法和 Multi-agent 分布协作求解模式。基于 DCOM 技术, 以 C+ + 构建框架类加速了 Agent 的实现。文中提及的智能主体的通信机制、协同求解中的领域问题描述、协商及分布式协作机制的建立思想对复杂问题求解提供了支持。DMPES 系统在网络环境下基于 Windows NT-WorkStation 采用 C+ + 开发实现, 这里描述的协作机制已在该系统中获得应用, 运行结果表明其具有较好的性能。

[关键词] 多智能体; 通信机制; 协商式协同; 分工式协同; 组件对象模型; 框架类
[中图分类号] TD 913; TP 39 [文献标识码] A

分布式选矿专家系统 DMPES[1] 中多 Agent 协作求解过程如下: 针对某一问题输入求解信息, 首先交互 Agent 开始与用户交互, 获得等求解任务的性质和内容, 它负责分解子任务, 即产生若干个假设结论, 然后根据这些假设结论分属的求解 Agent, 由管理 Agent 分别指派给不同的求解 Agent 去验证。一旦某个假设结论被证实, 验证该假设的 Agent 就启动下一层次的交互 Agent, 它又产生若干个假设结论, 又进行指派。这种"假设产生—假设验证"过程一直循环, 直到提不出新的假设为止, 由集成 Agent 形成最终问题求解结论。由此可见, 合作求解的关键是通信。通过两个或多个 Agent 之间的信息传递, 各 Agent 共同分担求解任务和共享中间结果[2]。然而通信速度要比计算速度慢, 尤其是在松散耦合的结点之间。因此, 针对具体的应用领域, 研究有效的通信及协作机制就显得特别重要。

1 多 Agent 通信机制

1.1 通信方式

本系统中, 各 Agent 之间采用消息传递, 而在 Agent 内部各模块之间采用共享存贮器(即黑板结构)来交换信息[3]。这种混合通信方式综合了二者的优点, 既便于语义理解, 又易于实现。消息传递提供了一种抽象的通信方式, 它是系统中各 Agent 之间进行交互的唯一手段。本系统中, 消息是以信件的形式由通信处理器传递, 各 Agent 设有接收信箱, 存放其它 Agent 发来的信件。黑板被划分成一定的抽象层次, 每个信息层次又包含一些数据项。黑板上记录了一些原始数据、问题求解过程中的部分解及完整解。

1.2 通信内容

本系统中, 通信内容含有 2 类信息: 数据信息和控制信息。数据信息包括结点之间互相交换的故障假设、征兆信息、中间结果等, 通过这些信息的传递, 各结点分担整个选矿问题求解任务。控制信息是指有关 Agent 的当前求解活动的状态信息, 用来提高系统的全局观察, 促进 Agent 之间的相互协作和协调。

1.3 通信协议

通信需要有通信语言, 由语言组成语句, 为了各 Agent 正确地理解语句, 保证有效通信, 需要有语法和语义。通信协议(高层协议)确定了语法, 通信处理器解释了语义。DMPES 中通信协议的 BNF 定义为:

〈通信语言〉: = 〈发送者〉〈接收者〉〈正文体〉
〈发送者〉: = 〈结点名〉
〈接收者〉: = * | 〈结点名〉

① [基金项目] 国家自然科学基金资助项目(59574034); 湖南省自然科学基金资助项目
[收稿日期] 2001- 03- 02; [修订日期] 2001- 06- 11 [作者简介] 景广军(1972-), 男, 博士.

〈结点名〉: = 〈标识符〉

〈正文体〉: = 〈指派〉| 〈查询〉| 〈回答〉

〈指派〉: = ALLOCATE〈指派内容〉

〈指派内容〉: = 〈求解假设〉〈假设状态〉

〈求解结论假设〉: = 〈标识符〉

〈结论状态〉: = NOT EVALUATED | NOT CONFIRMED| CONFIRMED

〈查询〉: = INQUIRY〈查询内容〉

〈查询内容〉: = ES〈结点名〉STATE | HYPOTHESIS〈结论假设〉STATE| SYMPTOM〈事实〉STATE

〈事实〉: = 〈标识符〉

〈回答〉: = ANSWER〈回答内容〉

〈回答内容〉: = ES〈Agent 名〉STATE〈Agent 状态〉|

〈Agent 状态〉: = IDLE| RUN| INTERRUPT| STOP

〈事实状态〉: = EXIST | NOT EXIST | UNKNOWN

〈标识符〉: = 〈字母〉{〈字母数字〉}

〈字母数字〉: = 〈字母〉| 〈数字〉

上面定义中，所有英文单词为 DMPES 定义的关键字，其语义取其字面意思。另外，字符‘ * ’表示广播消息。

1.4　通信策略

本系统采用选择通信和广播通信 2 种方式。首先根据元级知识(合作知识)选择通信伙伴，如果失败，再进行广播通信。信息需求方式有 2 种：需求响应式和主动提供式。前者是指需求结点向一组有关结点或某个结点发出需求信息，可以提供信息的结点，按需求提供的信息，向需求结点发送需求结果。后者是指结点依据系统规划，检查它所拥有的信息，如果可断定某个信息为某些或某个结点所需求，则结点主动把它发送给可能需求的结点[4]。为了系统通信简单、有效，本系统采用需求响应式。

1.5　通信处理器

DMPES 中设置了一个通信处理器，它能根据通信协议组织和解释通信语句，使通信双方相互正确理解通信处理器对通信语句的解释。我们采用了基于知识的方法，即为每类通信语句建立了一个模板存于知识库中，有关这类通信语句的解释以及解释所需要的信息都存储在该模板下，通信语言的解释程序就像一个独立于领域的推理机一样，只依赖于知识库中的通信模板。当接收到的一个通信语句与一个模板匹配时，变量进行一致性替换并触发一个相应的进程，然后按通信语句中的信息来执行这个进程。比如，下面的通信语句:

(Node1 Node2 ALLOCATE(Main_ Axis_ Unbalance)(NOT EVALUATED))

同模板

(〉Node2 ALLOCATE(〉)(〉))

相匹配，相对应的变量替换后，与 ALLOCATE 相对应的进程被触发，它把这个还未验证的故障假设按照其优先系数插入到本结点的假设对列，等待验证。由此可见，使用这种基于模板的通信，简单、有效和易于扩充，满足了系统对通信的要求。

2　多 Agent 协作机制

在分布式专家系统中，多 Agent 系统就是由多个能自治运行的子问题求解 Agent 组成的，能共同协作完成某一特定任务的分布式智能系统。在多 Agent 系统中，应能相互对话、相互适应、协同工作，以有效地完成它们共同的任务[5]。

多 Agent 协同求解的方式有两种：协商式协同求解，分工式协同求解[6]。协商式协同求解指多个 Agent 就某一主题，根据各自的知识能力和工作经验求解问题，并进行协商讨论，最后由协同 Agent 总结协商结果，确定任务的解决方案。分工式协同求解就是将一个大的任务分解成若干粒度较小的子任务，并根据各 Agent 的能力调度子任务给 Agent，Agent 完成子任务之后，由协同 Agent 进行子任务合成，得出最终求解结果。为了叙述方便，这里约定 (A_1, A_2, A_3, …, A_n) 为 n 个 Agent 组成的系统，A_i 表示第 i 个 Agent。$G_i[A_i, T]$ 为 A_i 对任务 T 的求解结果。

2.1　协商式协同求解

协商式协同求解的过程一般为：(1) 协同 Agent在网络上发布任务；(2) 各 Agent 对任务分别求解，构成求解空间$\{G_i\}$；(3) 多 Agent 在线讨论；(4) 协同 Agent 协调各 Agent 的求解并得出最终求解结果[7]。由于各 Agent 的知识背景、求解问题的出发点不同，它们的求解结果存在差异，因此如何协同解空间$\{G_i\}$，是多 Agent 系统求解的关键所在。

对本文主要采用以下两种方法。

1) 服从多数法。如果$\{G_i \mid i = 1 \sim n\}$中有多个

相同或相近的意见，则取意见集中的方案为最后的方案。

2) 可信度优先法。这里将 $DF[A_j, G_i]$ 定义为相对可信度，含义为 A_j 认为结果 G_i 正确的程度大小。$DF[G_i]$ 定义为综合可信度，即综合考虑各 Agent 评定情况后确定结果 G_i 的可信度。$DF[A_j, G_i]$ 和 $DF[G_i]$ 在[0, 1]区间。确定 DF 的步骤为：首先各 Agent 向协同 Agent 提交结果；协同 Agent 在网上发布某一结果 G_i；其他 Agent 向提交这一结果的 Agent 即 A_i 提问，A_i 回答各 Agent 的问题；同时可以修正自己的意见。各 Agent 根据 A_i 回答问题的情况给修正后的 G_i 评定可信度，构成 G_i 的相对可信度集合$\{DF[A_1, G_i]\ DF[A_2, G_i]\ \cdots DF[A_n, G_i]\}$；协同 Agent 对相对可信度集合进行处理得出综合可信度 $DF[G_i]$，处理方式可以分多种情况进行。

等同对待：对各 Agent 的评定意见平等对待，即简单求解可信度集合$\{DF[A_j, G_i] \mid j = 1 \sim n\}$的平均数

$$DF[G_i] = \frac{1}{n} \times \sum_{j=1}^{n} DF[A_j, G_i]$$

尊重个人意见：一般来说，各 Agent 对自己的意见最有发言权，因此我们应该加大 A_i 对$DF[G_i]$的贡献。

$$DF[G_i] = W_i \times DF[A_j, G_i] + W_t \times \sum_{j=1(\neq i)}^{n} DF[A_j, G_i]$$

式中 W_i，W_t 是由协同Agent分配给各 Agent 的权重，它们应满足 $W_i + W_t = 1$ 且 $W_i > W_t$。

尊重权威意见：由于各 Agent 对任务 T 的知识背景和求解经验上的差异，决定它们对任务求解结果评定的意见具有不同的意义，可以加大对任务 T 具有权威的 Agent 在协同 $DF[G_i]$ 中作用。因此

$$DF[G_i] = \sum_{j=1}^{n} (W_j \times DF[A_j, G_i]$$

式中 $W_1 + W_2 + \cdots + W_n = 1$，其大小由协同 Agent 根据经验确定。

究竟选用上面哪一种方式确定 $DF[G_i]$ 应视具体情况而定。重复上述的步骤对所有 Agent 提交的结果进行综合可信度的评定，就构成综合可信度集$\{DF[G_i] \mid i = 1 \sim n\}$。协同 Agent 根据 $DF[G_i]$ 的大小选取相应的求解结果 $G_i[A_i, t]$。如果存在几个综合可信度相等且为最大值，而它们对应的求解结果又存在差异，则让它们重新进行辩论，并重新评定其综合可信度，最后得出一个最优的求解结果。

2.2 分工式协同求解

分工式协同求解的一般过程[8]：

1) 任务的分解：协同 Agent 将一个大的任务 T 分解成粒度较小的子任务空间$\{T_i \mid i = 1 \sim r\}$。同时给出相应求解子任务的资源需求表 T：$R_i = \{t_j \mid j = 1 \sim k\}$。

2) 子任务调度：协同 Agent 根据各 Agent 的能力 A：CAP_i 及资源表 A：R_i 的状况和求解子任务的资源需求表进行子任务的分配。

3) 子任务的求解：各 Agent 对接受到的子任务进行求解。求解完毕，向协同 Agent 提交结果。

4) 子任务的合成：当所有的 Agent 都完成任务后，协同 Agent 进行任务合成。

设任务 T 的条件约束域 domain(X) 为：$F(x) = PBX$，其中 $P = (P_{ij})_{m*n}$ 为约束系数，$B = (b_1, b_2, \cdots, b_m)^{\mathrm{T}}$ 为约束边界，$X = (x_1, x_2, \cdots, x_3)^{\mathrm{T}}$ 为约束参数。而子任务的求解过程实际上是在域 domain(X) 内寻求一个理想的点 X，使子任务的解达到最好的效果，即让 $T_i(X) = c_{i1}x_1 + c_{i2}x_2 + \cdots + c_{in}x_n$ 达到最优。由此构成子任务的求解空间为

$$\text{Best}\begin{cases} T_1(X) = C_{11}X_1 + C_{12}X_2 + \wedge + C_{1n}X_n \\ T_2(X) = C_{21}X_1 + C_{22}X_2 + \wedge + C_{2n}X_n \\ \wedge \\ T_r(X) = C_{r1}X_1 + C_{r2}X_2 + \wedge + C_{rn}X_n \end{cases}$$

显然，要使 r 各子任务在同一个点达到最优是不可能的，为了解决这一矛盾，可以采用模糊数学的方法对需求解的子任务进行模糊处理，处理方法如下[9]。

首先，各 Agent 在约束条件下求出各子任务的最优解 $T_i = \text{Best}(T_i)$，它们也是局部的最优解，一般情况下，满足它们要求的约束点落在约束空间域 domain(X) 的不同地方，为了使这些约束点尽可能的一致或靠近，可以放宽子任务的要求。由协同 Agent 给每个子任务一个伸缩因子 d_i（不小于 0）。d_i 的大小根据子任务的重要性而定，越重要的任务 d_i 越小。这样我们就把子任务的最优解模糊化了，设 T_i 的模糊化解为 $G_i(X)$，则其隶属度函数表示为

$$G_i(X) = \begin{cases} 0 & T_i(X) < T_i - d_i \\ 1 - [T_i - T_i(X)/d_i] & T_i - d_i \leqslant T_i(X) < T_i \\ 1 & T_i(X) \geqslant T_i \end{cases}$$

经过上述处理后，使用模糊数学的方法在条件域

domain(X) 内对模糊化后的解集{ $G_i(X) \mid i = 1 \sim r$} 进行合成，就可以在域 domain(X) 内找到一个点或一组充分靠近的点集满足各子任务的要求。

3　基于分布式技术的 Agent 实现

分布式选矿专家系统 DMPES 的设计及知识分布在文献[1]中已予以介绍，将选矿知识首先分解成粗粒 Agent，即矿石可选性预测 Agent、选矿厂设计 Agent、生产过程控制 Agent、设备故障诊断 Agent 及生产经营管理 Agent。然后，再进一步分为细粒 Agent，如流程及设备选择 Agent、选别方法 Agent、选别原则流程 Agent、选别工艺流程 Agent、选矿指标预测 Agent、浮选生产控制 Agent、碎磨设备故障诊断 Agent 等[10~12]。将上述细粒 Agent 分布在松耦合的 Internet/Intranet 网络环境中，根据所求解选矿子领域问题的需要，设计对应的符号系统 Agent、神经网络 Agent、模糊 Agent、多媒体 Agent 及遗传 Agent，它们紧耦合在单处理器上，完成问题求过程[13, 14]。

3.1　框架类的设计

基于 C++ 构造类框架，它是面向特定的应用范畴的，一旦与应用特征相符，就可以整体地被重用，被重用构件的粒度越大，所要关心的细节就越少，因而抽象程度就越高。这种基于类框架的 Agent具有面向特定应用领域和大粒度的特征。相关领域的 Agent 通过消息相互作用，构成了一个开放的信息系统。一个 Agent 因自己不具有所需的能力而不能解决的问题，可以请求相关的 Agent 予以代理。每个 Agent 都具有自己的能力可以独立地解决问题，因而可以独立地运行。

首先定义 Agent 的形式化构造部分。由于一个多 Agent 系统是一个开放的信息系统，因此每个 Agent 都是独立构造、独立编译、并能独立运行的。形式部分语法的 BNF 描述如下：

〈AgentProgram〉::= Agent 〈Agentname〉
　〈ClassFrameWork〉
　〈Initialization〉
　〈MessageLoop〉
EndAgent

每一个 Agent 系统程序有唯一的标识〈Agentname〉，并由 3 部分组成，即类框架(〈ClassFrameWork〉)，初始化部分(〈Initialization〉)和消息循环(〈MessageLoop〉)。所有 Agent 的消息监听部分因为都相同，初始化部分声明、定义与初始化全局变量和对象。就对象来说，初始化部分调用类框架中各个类的构造函数以生成暂态或持久对象。

下面给出类框架说明：

〈ClassFrameWork〉::= 〈ApplicationClasses〉
　〈UserInterfaceClasses〉
　〈AgentInterfaceClasses〉
　〈MentalStateClass〉
　〈BehaviorRuleClass〉

其中，应用类部(〈ApplicatonClasses〉)为面向领域的类集合，体现了 Agent 的实际能力。这些类既有继承关系，也有横向关联而形成应用领域的类框架，是重点开发环节。用户接口类部(〈UserInterfaceClasses〉)可以采用 Visual C++ 提供的图形用户接口类框架。用户接口类与应用类通过横向关联而形成联结，用户可以使用用户接口类提供的接口调用应用类部提供的若干方法。而 Agent 接口类部(〈AgentInterfaceClasses〉)同样通过横向关联而与应用类形成联结，提供了其它 Agent 对应用类中若干方法的可能的调用界面，实质上是对 Agent 之间交互作用的范围及深度作了一种限制。当然，其它 Agent 对应用类中方法的最终调用要得到心态类和行为规则类的允许。心态类(〈MentalStateClass〉)和行为规则类(〈BehaviorRuleClass〉)的 BNF 说明在文中省略。

3.2　基于 DCOM 设计 Agent

分布式组件对象模型(DCOM，Distributed Component Object Model)作为分布式计算策略，是在开放性软件 DEC 远程过程调用协议的基础上开发的。DCOM 是组件对象模型 COM 的一个扩增版，而 COM 是 ActiveX 的基础技术。COM 和 DCOM 最大的不同在于 COM 组件是运行在单机上，而 DCOM 组件则是分布在网络上。

DCOM 支持动态和静态两种对象调用。DCOM 提供了两种界面，不同的调用方法需要不同的界面来实现。使用静态调用，要定义一个界面和它的组件，让 MIDL 编辑程序自动产生连接这些组件的代理存根模块(proxy-stub)代码。DCOM 是通过类库来支持动态调用的，类库中包含了描述组件的文件，客户应用在运行期间通过一个被称为 IDispatch 的 COM 界面来获取这些界面，优点是无需手工定义界面、让 MIDL 产生 proxy-stub 代码了，可以使用缺省的 IDispatch 代理存根模块代码。

在本系统中，基于 DCOM 技术的优势，以 Vi-

sual C+ + 语言为开发工具，设计实现了分布式专家系统 DMPES 中的多 Agent，每个领域子问题求解 Agent 采用统一的基于关系模式的知识表示模式及数据库引导推理，各求解 Agent 之间的知识和数据的传输不需要转化，大大提高了通信速度[15]。

1) 设计用户与 Agent 及 Agent 之间的交互界面，是信息及数据传输的格式化具。利用 Visual C+ + 的 ATL COM 应用向导生成一个 IDL 基本框架文件，根据需要在 IDL 文件中增加界面定义语言 MIDL，形成通用界面生成 C+ + 框架类。

2) 基于关系模式表征子领域知识，用 C+ + 实现对数据库及知识库的搜索推理，建造 Agent 内部求解的 C+ + 框架类。

3) 通过一个远程对象代理和对象远程过程调用 ORPC，实现调用远程服务器对象的能力。而本地跨进程调用则是通过独立于操作系统的进程间通信完成的，包括静态和动态两种调用方法。

①客户应用通过将对象的类标识和界面标识传送到 COM 库发出对类实例的请求。

②COM 库检查服务器系统登记，找到服务工具，然后启动它。

③该服务产生一个对象并将界面指针返回给 COM 库，由它将指针传送给客户。

④此时客户机即可自由地调用该对象的方法了。对远程服务，客户机要把请求传递给远程对象代理，它通过对象远程过程调用 ORPC 和服务器存根模块与远程服务器连接。

4) 使用 MIDL 编译程序编译 IDL 文件，生成必要的 proxy 和 stub 代码；使用 ATL 对象向导添加你要的 COM 对象；编译并建立最后的 COM 对象；编写应用，访问该 COM 对象，将该对象移到远程机器上并在那里登记。

4 结语

DMPES 系统在网络环境下基于 Windows NT-WorkStation 用 C+ + 编程实现，这里描述的协作机制已在该系统中获得应用，运行结果表明其具有较好的性能。在复杂问题求解中，采用多智能主体的分布式并行协同求解方法，在 TCP/IP 的控制方式下有效地组织系统中多个智能主体的并行协同求解规划，是提高系统求解效率达到全局最优解的关键。文中提及的智能主体的通信机制、协同求解中的领域问题描述、协作机制的建立思想对复杂问题求解提供了支持。分布式人工智能技术在工程领域有着广泛的应用前景，尤其是随着计算机应用水平的提高，它会越来越多地应用于复杂问题的求解。

[REFERENCES]

[1] LI Song-ren (李松仁), JING Guang-jun (景广军), CHEN Song-qiao (陈松乔), et al. 基于 multi-agent 的分布式专家系统原理与应用 [J]. J Cent South Univ Technol (中南工业大学学报), 2001, 32(1): 20- 24.

[2] JING Guang-jun (景广军), LI Song-ren (李松仁), LIANG Xue-mei (梁雪梅). 基于 Internet/Intranet 技术建造的选程专家系统 [J]. The Chinese Journal of Nonferrous Metals (中国有色金属学报), 2001, 11(3): 489 - 494.

[3] SHI Dian-xi (史殿习), WANG Huai-min (王怀民), ZHOU Peng (邹 鹏), et al. 一种基于 Agent 的分布式集成框架的设计与实现 [J]. Computer Science (计算机科学), 1998, 25(4): 42- 45.

[4] ZHOU Xiao-bo (周笑波), XIE Li (谢 立), ZHOU Xiao-fang (周晓方). 一个基于多服务员系统的 Internet 市场服务管理模型 [J]. Chinese J Computers (计算机学报), 1999, 22(4): 424- 430.

[5] HU Zhi-gang (胡志刚), ZHONG Jue (钟 掘). 解耦设计中分布式并行协同求解方法 [J]. J Cent South Univ Technol (中南工业大学学报), 1998, 28(1): 69- 71.

[6] ZHONG Jue (钟 掘), HU Zhi-gang (胡志刚). 基于耦合问题的多智能主体协作模型 [J]. J Cent South Univ Technol (中南工业大学学报), 1998, 29(2): 165- 168.

[7] Kraus S. Negotiation and cooperation in multi Agent environments [J]. Artificial Intelligence, 1997, 94(3): 79 - 97.

[8] Lekkas G P, Avouris N M. Development of distributed problem solving systems for dynamic environments [J]. IEEE Transactions on Systems, Man and Cybernetics, 1995, 25(3): 400- 414.

[9] Jennings N R. Controlling cooperative problem solving industrial multi-agent systems using joint intention [J]. Artificial Intelligence, 1995, 75(2): 195- 240.

[10] LI Song-ren, JING Guang-jun, ZHOU Xian-wei, et al. Study of mineral processing expert system skeletal tool [J]. Trans Nonferrous Met Soc China, 2000, 10(1): 94- 97.

[11] LI Song-ren (李松仁), JING Guang-jun (景广军), ZHOU Xian-wei (周贤渭), et al. MPES 通用骨架系统的研究 [J]. Nonferrous Metals (有色金属), 2000, 52(1): 46- 48.

[12] LUO Xin-xing (罗新星), LI Song-ren (李松仁), GAO Yang (高 阳). 浮选厂自学习咨询控制专家系统知

识库设计 [J]. J Cent South Univ Technol(中南工业大学学报), 1996, 27(5): 534- 537.

[13] YANG Jia-hong(杨家红), LI Hong-gui(李洪桂), ZHAO Xian-fu(赵显富), et al. 矿物机械活化人工神经网络模糊专家系统的开发 [J]. J Cent South Univ Technol(中南工业大学学报), 1999, 30(1): 21- 25.

[14] LUO Zhou-quan(罗周全), SU Jia-hong(苏家宏). 基于 Intranet 的矿山企业网络管理信息系统 [J]. J Cent South Univ Technol(中南工业大学学报), 1999, 30(1): 99- 101.

[15] Liedekerke M H, Avouris N M. Debugging multi-agent system [J]. Information and Software Technology, 1995, 37(2): 103- 112.

Cooperative mechanism of distributed expert system based on Multi-agent

JING Guang-jun[1], LI Song-ren[2], CHEN Song-qiao[3]

(1. Department of Computer Science and Technology, Nanjing University, Nanjing 210093, P. R. China;
2. Department of Mineral Engineering, Central South University, Changsha 410083, P. R. China;
3. College of Information Science and Engineering, Central South University, Changsha 410083, P. R. China)

[Abstract] The design of Distributed Mineral Processing Expert System (DMPES) based on Internet/Intranet was explored under slack coupling with Agent-oriented Programming (AOP) method and Multi-agent distributed cooperating solution pattern. Inhomogeneous agents were proposed, that is, agents may be built on WAN, LAN and single CPU by tense coupling mode, and solution agents need interoperation of heterogeneous software agents based on symbol system, neural network, genetic algorithm for background-analytical applications. All agent communication is performed through message passing and blackboard mechanism. Message passing includes synchronous communication method and asynchronous communication method. Cooperative mechanism includes negotiatory mechanism and dividual mechanism. Based on Distributed Component Object Model (DCOM) technology, building frame class by C++, it accelerates the execution of Agent. It makes knowledge transferred among Agents easy, and brings up communicating efficiency greatly.

[Key words] multi agent; communication mechanism; negotiatory cooperation; dividual cooperation; distributed component object model; framework class

(编辑　袁赛前)