doi:10.3969/j.issn.1000-436x.2013.07.014

效用驱动的多 agent 合作机制

邓寒冰[1]，张霞[1,2]，刘积仁[1]
(1. 东北大学 信息学院，辽宁 沈阳 110004；2. 东北大学 软件架构国家重点实验室，辽宁 沈阳 110004)

摘 要：提出了一种多 agent 合作机制，以效用为基础，利用效用函数计算 agent 在不同任务下与其他 agent 合作而产生的效用值，以此为依据得到 agent 对不同合作方式的偏好序列，并选择最大偏好参与合作。通过这种方法，agent 可以主动选择最适合自己的合作方式来完成任务，与此同时，以效用为驱动可以使群体的任务收益达到最优，提高了 agent 群体的合作效率。最后给出了用于方法验证的多 agent 原型系统，同时在该系统下证明了合作机制的有效性。

关键词：效用函数；多智能体系统；合作机制；依赖关系
中图分类号：TP311 **文献标识码**：B **文章编号**：1000-436X(2013)07-0124-10

Cooperation mechanism of multi-agent system driven by utility

DENG Han-bing[1], ZHANG Xia[1,2], LIU Ji-ren[1]
(1. College of Information Science and Engineering, Northeastern University, Shenyang 110004, China;
2. State Key Laboratory of Software Architecture, Northeastern University, Shenyang 110004, China)

Abstract: A cooperative mechanism amlong multiple agents was proposed. In the cooperative mechanism, a utility value was used to generate a preferred order for agents under different cooperative patterns. This utility value was achieved by computing the cooperative utility between one agent and the others under different tasks using a utility funct According to this mechanism, agents can automatically choose the cooperative pattern, wh is the best to fit themselves to accomplish their missions. Meanwhile, driven by utility, group task income can achieve the optimum, which enhances the cooperation efficiency of agent groups. A prototype system was designed at last to val te the proposed cooperative mechanism. The effectiveness of this cooperative mechanism has been proved under the prototype system.

Key words: utility function; multi-agent system; cooperation mechanism; dependence relationship

1 引言

普适化、网络化、智能化、代理化、人性化是迄今为止计算的历史发展趋势，而这些趋势导致了计算机科学的一个新领域的出现，这就是多 agent 系统[1]。不同于分布式计算、P2P 计算等计算模式，多 agent 系统求解问题类似于人类的思维过程。受利益的趋势，每个 agent 作为一个独立的计算机系统，代表用户完成独立动作，不需要对问题有全面的分析，而只需要设定目标任务，多个 agent 之间能够进行交互、协商、合作，逐步实现个体目标和总体目标[2,3]。

目前，人们对 agent 的研究已经逐渐由理论向实际应用过度，如何能够准确找到多 agent 系统的应用场景，如何高效地利用多 agent 合作机制以实现其智能化、代理化和人性化是学术界所关心的问题[4]。

从理论研究上来看，广为熟知的包括 agent 个体或群体 BDI 模型[5]、agent 组织与联盟[6]、多 agent 协商机制[7]、多 agent 合作算法[8]以及多 agent 学习[9]等。从应用上看，以云计算为契机，在智能互联的大背景下，多节点合作一直是智能服务领域的研究重点[10]。传统方法在解决这些问题时，总会缺少智能化和人性化，如何让软件具有人的合作方式、如何让软件具有代替人的行为、如何通过软件让人对

收稿日期：2012-11-01；修回日期：2013-05-26
基金项目：国家重点基础研究发展计划（“973”计划）基金资助项目（2012CB724107）
Foundation Item: The National Basic Research Program of China (973 Program)(2012CB724107)

资源的利用达到高效平衡等都是人们亟待解决的问题[11]。

多 agent 技术在解决这些智能难题上具有天然的优越性。首先，agent 作为一个独立的计算系统，具有独立的思维、目标和行为规划。在既定目标的驱动下，agent 会将完成自身预定任务设为最高优先级，这就保证了个体任务的完成；同时，多 agent 之间有标准的通信方式，可以在实现各自目标的过程中保持彼此互联，在协商机制的配合下，完成利益最大化的合作；通过 agent 的代理机制可以将人们从繁琐的人机交互中解放出来，所有的交互均由多 agent 系统自动完成，同时承诺目标任务达成[12~16]。

本文提出一种多 agent 合作机制，利用效用函数计算 agent 在不同任务下与其他 agent 合作而产生的效用值，以此为依据得到 agent 对不同合作方式的偏好序列，agent 根据偏好序列主动选择最适合自己的合作方式来完成任务，以效用为驱动可以使群体的任务收益达到最优，提高了 agent 群体的合作效率。

2　依赖关系与合作模式

在 agent 群体合作过程中，多 agent 之间是存在某种依赖关系的，这种依赖关系与合作模式存在一定的对应关系。Sichman 和 Demazeau 等人给出的基本观点是[17,18]，如果一个 agent 需要其他 agent 帮助来实现自己的目标，则 2 个 agent 之间存在依赖关系。具体的依赖关系类型包括：

1) 独立：2 个 agent 之间没有依赖关系，即 agent 之间不会产生合作；

2) 单向依赖：其中一个 agent 依赖另一个 agent，反之不成立；

3) 相互依赖：2 个 agent 彼此相互依赖，建立合作来完成一个共同的目标；

4) 交互依赖：为了某一目标，第 1 个 agent 依赖第 2 个 agent，而第 2 个 agent 为了实现某一目标，也同样依赖于第 1 个 agent（2 个目标可以相同或不同），其中相互依赖蕴含着交互依赖。

在本文中，agent 的思维方式和行动都是独立的，其运行不依赖于任何其他 agent。在利益驱使下，agent 总在试图找到一种方法来简化自己的工作任务，并且追求得到更多的收益，即让自身利益最大化，这就需要根据任务的不同与其他 agent 进行合作，而建立依赖关系是实现合作的基础。

合作模式包括合作与不合作。而合作又可以包括：单向合作和双向合作。为了便于讨论，假设存在 agent i 和 agent j，t 为 2 个 agent 需要进行合作完成的任务。如图 1 所示，不合作表示 i 与 j 都不想同对方一起参与完成任务 t；单向合作表示 i 在参与任务 t 的过程中希望得到 j 的帮助；双向合作表示 i 和 j 想以合作的方式参与任务 t。

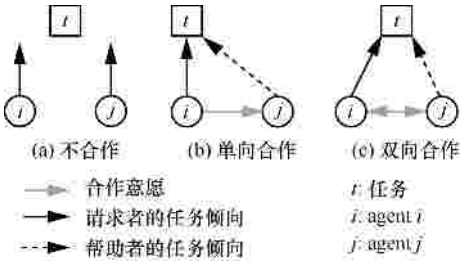

图 1　2 个 agent 之间的合作模式

从依赖关系的角度来看，独立表示 agent 之间不存在合作关系，与不合作对应；单向依赖表示一个 agent 完成任务的过程依赖于另一个 agent，与单向合作对应；相互依赖表示 agent 双方有共同的目标，双方互相依赖，与双向合作对应；交互依赖有些特殊，当目标不相同时，双方互为单向依赖（单向合作），而当目标相同时，双方相互依赖（双向合作）。具体对应关系如表 1 所示。

表 1　合作与依赖的对应关系

依赖关系	不合作(N,N)	单向合作(C,N)/(N,C)	双向合作(C,C)
独立	是	否	否
单向依赖	否	是	否
相互依赖	否	否	是
交互依赖	否	是	是

3　多 agent 的效用与偏好

3.1　效用函数定义

在评价一个人的工作效率时，最常用的方法是计算这个人在单位时间内创造的价值，即如式(1)所示。

$$h = \frac{value}{time} \tag{1}$$

其中，$value$ 表示创造的价值，$time$ 表示创造价值所用的时间。h 的值越高，证明这个人的工作效率越高。可以看出，对 h 值的追求是促使人们更快、更好地完成任务的一个重要驱动因素。

效用是表示状态的数值，效用越高，状态越好。agent 执行任务是实现效用的最大化，与人类工作效率计算过程类似，工作效率的计算公式可以运用到对 agent 的效用定义中，其中，价值可以由 agent 完成任务后得到的收益表示，而创造价值所用的时间可以由 agent 在完成任务所用的运行时间来表示。因此，设 u 为 agent ag 的效用函数，$cash(ag, t)$为 ag 完成任务所得到的任务收益，$time(ag,t)$为 ag 完成任务所需的时间，则效用函数 u 可表示为

$$u_{ag} = \frac{cash(ag,t)}{time(ag,t)} \tag{2}$$

对于多 agent 系统来说，每一个 agent 都有自己的动作集合，其直接决定该 agent 在完成任务过程中所能使用到的动作，而运行时间的长短与执行任务所用的动作数量成正比。因此，本文将时间函数做了一些等价修改，其中，$action(ag,t)$表示 ag 完成任务 t 时所用的动作数，$?$ 表示时间校正系数，则可以得到

$$u_{ag} = \frac{cash(ag,t)}{action(ag,t)\cdot l} \tag{3}$$

其中，$?$ 值等于 agent 执行一个动作的平均时间，设 ag 的动作数为 n，$time(ag, ac)$表示 ag 执行动作 ac 所花费的时间，则 $?$ 可表示为

$$l = \frac{\sum_{i=1}^{n} time(ag, ac_i)}{n} \tag{4}$$

3.2 agent 偏好的性质

agent 在执行任务的过程中会有自己的意愿和偏好，agent 会选择自己最喜欢的方式（偏好）来完成任务[6]。而效用函数的设计正是基于 agent 拥有偏好这一特点，利用效用函数来计算 agent 对于不同合作方式的偏好，给出具体的偏好值，如果偏好值越大，则证明 agent 更愿意选择这样的合作方式参与任务。本文利用任务集合到实数集合的映射关系来设计效用函数 u，具体形式如式(5)所示。

$$u: T? \ R \tag{5}$$

其中，T 表示任务集合，R 表示实数集，这样效用函数可以给出一个关于 agent 对任务的偏好序列。假设 t 和 t'是集合 T 中的任务，如果 $u_{ag}(t) \geq u_{ag}(t')$，则证明 ag 执行任务 t 的结果至少与执行任务 t'的结果一样好。同样，如果 $u_{ag}(t) > u_{ag}(t')$，则证明 ag 执行任务 t 的结果严格好于执行任务 t'的结果。可以看出，偏好具有以下性质：

1) 自反性：对于 $\forall t \in T$，都有 $u_{ag}(t) \geq u_{ag}(t)$；

2) 传递性：对于 $\forall t,t',t'' \in T$，如存在 $u_{ag}(t) \geq u_{ag}(t')$并且 $u_{ag}(t') \geq u_{ag}(t'')$，则 $u_{ag}(t) \geq u_{ag}(t'')$；

3) 比较性：对于 $\forall t,t' \in T$，都有 $u_{ag}(t) \geq u_{ag}(t')$，或者 $u_{ag}(t') \geq u_{ag}(t)$。

合作模式的不同直接决定合作中 agent 的任务收益，而通过自反性、传递性、比较性可以比较 agent 对于合作模式的偏好值，偏好值大小可以确定 agent 是否愿意与其他 agent 以某种形式的合作完成任务。

3.3 效用设计与偏好分析

对于多 agent 系统，如果系统中存在 n 个 agent，则存在 2^n 种不同的合作形式。在 $n = 2$ 的情况下，假设存在 agent i 和 agent j，它们的合作模式分别为：(N_i, N_j)、(N_i, C_j)、(C_i, N_j)、(C_i, C_j)。i 和 j 对于不同合作模式的偏好排序是不同的，而效用函数是最终决定这 4 种偏好排列顺序的关键。通过配置效用函数中的参数值来控制 agent 对合作模式的偏好排序，使多 agent 在合作过程中自动建立依赖关系（独立依赖、单向依赖、相互依赖、交互依赖），即建立了多 agent 的不同合作模式。

3.3.1 效用函数的设计

设存在 agent i 和 agent j，本节要分别给出在(N_i, N_j)、(N_i, C_j)、(C_i, N_j)、(C_i, C_j)4 种组合模式下，效用函数 u_i 和 u_j 的函数形式。首先定义在效用函数中出现的变量和变量之间的关系。其中，

1) t 为 i 和 j 可以合作完成的任务；

2) t_i 为 i 参与合作时，i 所承担的任务；

3) t_j 为 j 参与合作时，j 所承担的任务；

4) t_i'为 i 不参与合作时，i 可以独立完成的任务，其中，$t_i' \geq t_i$；

5) t_j'为 j 不参与合作时，j 可以独立完成的任务，其中，$t_j' \geq t_j$；

6) $cash(ag,t)$，表示 agent ag 完成任务 t 之后所获得的任务收益；

7) $action(ag,t)$，表示 agent ag 完成任务 t 所执行的动作数量；

8) $cost(ag,t)$，表示 agent ag 在建立合作时所产生的合作成本；

9) $punish(ag,t)$，表示当对方选择合作而 ag 选择不合作时，系统给予 ag 的收益惩罚。

为了准确计算效用值，对于同一个 agent 来说，完成相近任务所用的动作数也相近，即动作计数函

数 $action$ 的值也是相近，例如假设 $t_1 + t_2 \sim t$，则可以得到 $action(ag,t_1) + action(ag,t_2) \sim action(ag,t)$；然而，对于同一个 agent 来说，如果任务量相同，则获得的任务收益可能不相同，即如果 $t_1=t_2$，则 $cash(ag,t_1)$? $cash(ag,t_2)$。利用上述参数定义可以给出效用函数 u_i 和 u_j 的具体形态。

以 agent i 为例，效用函数 u_i 可以由式(6)表示，其中，函数中的参数在不同的合作模式下有不同的值。$cash(i,t_i)$表示 i 在参与任务 t 中获得的任务收益；$cash(i,t_i'- t_i)$表示 i 在完成除 t_i 外剩余任务所获得的任务收益；$cost(i,t)$表示 i 建立任务合作时所花费的任务成本；$punish(i,t)$表示当 i 不参与合作时所负担的收益惩罚；$action(i,t_i) + action(i,t_i'- t_i)$表示 i 在完成所有任务时所花费的动作数；$?$ 为时间校正系数。

$$u_i = \frac{cash(i,t_i)+cash(i,t_i'-t_i)-cost(i,t)-punish(i,t)}{(action(i,t_i)+action(i,t_i'-t_i))l} \quad (6)$$

1) 在(N_i, N_j)合作模式下：i 选择不合作，j 选择不合作。因此 $t_i=0$，$cash(i,t_i)=0$，$cash(i,t_i'- t_i)=cash(i, t_i')$，$cost(i,t)=0$，$punish(i,t)=0$，$action(i,t_i)=0$，$action(i, t_i'- t_i) = i.action(i,t_i')$。所以 $u_i(N_i,N_j)$的形式如式(7)所示。

$$u_i(N_i,N_j) = \frac{cash(i,t_i')}{action(i,t_i')\cdot l} \quad (7)$$

2) 在(N_i, C_j)合作模式下：i 选择不合作，j 选择合作。因此 $t_i=0$，$cash(i,t_i)=0$，$cash(i,t_i'- t_i)=cash(i,t_i')$，$cost(i,t)=0$，$punish(i,t) > 0$，$action(i,t_i)=0$，$action(i,t_i'- t_i) = action(i,t_i')$。所以 $u_i(N_i,C_j)$的形式如式(8)所示。

$$u_i(N_i,C_j) = \frac{cash(i,t_i')-punish(i,t)}{action(i,t_i')\cdot l} \quad (8)$$

3) 在(C_i, N_j)合作模式下：i 选择合作，j 选择不合作。因此 $t_c^i=0$，$cash(i,t_i)=0$，$cash(i,t_i'- t_i)= cash(i, t_i')$，$cost(i,t)>0$，$punish(i,t)=0$，$action(i,t_i)=0$，$action(i, t_i'- t_i)=action(i,t_i')$。所以 $u_i(C_i, N_j)$的形式如式(9)所示。

$$u_i(C_i,N_j) = \frac{cash(i,t_i')-cost(i,t)}{action(i,t_i')\cdot l} \quad (9)$$

4) 在(C_i, C_j)合作模式下：i 选择合作，j 选择合作。因此 t_c^i ? 0，$t_i'=t_i- t_c^i \geq 0$，因此 $cash(i,t_i)>0$，$cash(i,t_i'- t_i)\geq 0$，$cost(i,t)>0$，$punish(i,t)=0$，$action(i,t_i)>0$，$action(i,t_i'- t_i)\geq 0$。$u_i(C_i,C_j)$的形式如式(10)所示。

$$u_i(C_i,C_j) = \frac{cash(i,t_i)+cash(i,t_i'-t_i)-cost(i,t)}{(action(i,t_i)+action(i,t_i'-t_i))\cdot l} \quad (10)$$

3.3.2 偏好分析

偏好分析主要是对效用函数值的排序过程，对于(N_i,N_j)、(N_i,C_j)、(C_i,N_j)、(C_i,C_j)这 4 种不同的合作模式，首先利用效用函数分别计算在不同模式下的效用值，然后对效用值进行排序。这里首先给出对于 4 种不同的情况，效用函数的比较结果。同样以 agent i 为例。

1) 对(N_i,N_j)和(N_i,C_j)合作模式下的效用函数进行比较，可以得到

$$u_i(N_i,N_j) - u_i(N_i,C_j) = \frac{punish(i,t)}{action(i,t_i')\cdot l} \quad (11)$$

2) 对(N_i,N_j)和(C_i,N_j)合作模式下的效用函数进行比较，可以得到

$$u_i(N_i,N_j) - u_i(C_i,N_j) = \frac{cost(i,t)}{action(i,t_i')\cdot ?} \quad (12)$$

3) 对(N_i, N_j)和(C_i,C_j)合作模式下的效用函数进行比较，其中，$action(i,t_i') \sim action(i,t_i) + action(i,t_i'- t_i)$，因此可以得到

$$u_i(N_i,N_j) - u_i(C_i,C_j) = \frac{cash(i,t_i')-cash(i,t_i)-cash(i,t_i'-t_i)+cost(i,t)}{action(i,t_i')\cdot l} \quad (13)$$

4) 对(N_i, C_j)和(C_i, N_j)合作模式下的效用函数进行比较，可以得到

$$u_i(N_i,C_j) - u_i(C_i,N_j) = \frac{cost(i,t)-punish(i,t)}{action(i,t_i')\cdot l} \quad (14)$$

5) 对(N_i, C_j)和(C_i, C_j)合作模式下的效用函数进行比较，其中，$action(i,t_i') \sim action(i,t_i) + action(i,t_i'- t_i)$，因此可以得到

$$u_i(N_i,C_j) - u_i(C_i,C_j) = \frac{cash(i,t_i')-cash(i,t_i)-cash(i,t_i'-t_i)}{action(i,t_i'-t_i)\cdot ?} + \frac{cost(i,t)-punish(i,t)}{action(i,t_i'-t_i)\cdot l} \quad (15)$$

6) 对(C_i, N_j)和(C_i,C_j)合作模式下的效用函数进行比较，其中，$action(i,t_i') \sim action(i,t_i)+action(i, t_i'- t_i)$，因此可以得到

$$u_i\left(C_i,N_j\right)-u_i\left(C_i,C_j\right)=\frac{cash\left(i,t_i'\right)-cash\left(i,t_i\right)-cash\left(i,t_i'-t_i\right)}{action\left(i,t_i'\right)\cdot l} \tag{16}$$

以式(11)~式(16)的偏好分析结论为基础，当给公式中的参数附上确定值时，可以很容易建立偏好之间的序列关系。

4 基于偏好的合作建立

根据第 2 节可知，agent 依赖关系分为 4 种：独立、单向依赖、相互依赖和交互依赖，每种依赖关系对应着不同的 agent 合作模式，agent 可以自动选择任意一种依赖关系与其他 agent 建立合作。因此，如何让 agent 选择依赖关系是 agent 自动合作建立的前提。通过第 3 节的证明可知，通过控制参数值来改变 agent 的偏好值，进而可以改变合作模式之间的偏好序列，而合作模式与依赖关系是相关的，因此可以证明，利用参数可以影响 agent 之间的依赖关系。

4.1 独立依赖实现

独立指 agent i 和 agent j 之间彼此不存在任务上的合作关系。无论 i 或者 j，对于不合作的偏好要大于合作的偏好。因此，以 agent i 为例，无论 j 是否合作，i 都首先倾向于不合作。因此，可以给出 agent i 选择独立关系时，4 种同时存在的偏好情况为

P_1：$u_i(N_i, N_j) > u_i(C_i, N_j)$

P_2：$u_i(N_i, N_j) > u_i(C_i, C_j)$

P_3：$u_i(N_i, C_j) > u_i(C_i, N_j)$

P_4：$u_i(N_i, C_j) > u_i(C_i, C_j)$

将式(11)~式(16)代入上述 4 种情况中，可以对应得到偏好 P_1~P_4 的形成条件：

C_1：$cost(i,t) > 0$；

C_2：$cash(i,t_i')–cash(i,t_i)–cash(i,t_i'- t_i)+cost(i,t)>0$；

C_3：$cost(i,t) – punish(i,t) > 0$；

C_4：$cash(i,t_i')–punish(i,t)–cash(i,t_i)– cash(i,t_i'- t_i)+cost(i,t) > 0$。

同理，对于另一方 agent j 来说，无论 i 是否合作，j 也首先倾向于不合作。利用上面对 i 的偏好分析结果，可以相应得到 agent j 形成独立依赖的 4 种偏好条件：

C_5：$cost(j,t) > 0$；

C_6：$cash(j,t_j')- cash(j,t_j)- cash(j,t_j'- t_j)+cost(j,t)>0$；

C_7：$cost(j,t) – punish(j,t) > 0$；

C_8：$cash(j,t_j') –punish(j,t) – cash(j,t_j)–cash(j,t_j'- t_j) + cost(j,t) > 0$。

因此，若想任意 2 个 agent 在交互的过程中保持彼此独立，需要满足条件 C_1~C_8。

4.2 单向依赖实现

单向依赖关系是指，作为合作协议中的双方，其中一方的完成任务过程依赖于另一方，而反方向则没有这种强制的依赖关系。以 agent i 单向依赖 agent j 为例，i 对于合作的偏好要大于不合作的偏好，无论任何情况，i 倾向于与 j 建立合作关系。因此，可以给出 agent i 实现单向依赖的偏好情况：

P_1：$u_i(C_i, C_j) > u_i(N_i, N_j)$

P_2：$u_i(C_i, C_j) > u_i(N_i, C_j)$

P_3：$u_i(C_i, N_j) > u_i(N_i, N_j)$

P_4：$u_i(C_i, N_j) > u_i(N_i, C_j)$

将式(11)~式(16)代入上述 4 种情况，可以对应得到偏好 P_1~P_4 的形成条件：

C_1：$cash(i,t_i')–cash(i,t_i)–cash(i,t_i'- t_i)+cost(i,t)<0$；

C_2：$cash(i,t_i')–punish(i,t)–cash(i,t_i)–cash(i,t_i'- t_i)+cost(i,t)<0$；

C_3：$cost(i,t) < 0$；

C_4：$cost(i,t) – punish(i,t) < 0$。

而对于被依赖方 j 来说，如果想成功建立单向依赖关系，系统希望 j 也可以选择合作，但这种依赖关系下，j 并不偏好于合作。因此，可以得到 agent j 形成独立依赖的偏好情况条件为

P_5：$u_j(C_i, C_j) > u_j(N_i, N_j)$

P_6：$u_j(C_i, C_j) > u_j(C_i, N_j)$

P_7：$u_j(C_i, C_j) > u_j(N_i, C_j)$

对应 agent i 的偏好条件求解过程，可以对应得到偏好 P_5~P_7 的形成条件：

C_5：$cash(j,t_j')–cash(j,t_j)–cash(j,t_j'- t_j)+cost(j,t)<0$；

C_6：$cash(j,t_j')–punish(j,t)–cash(j,t_j)–cash(j,t_j'- t_j)+cost(j,t)<0$；

C_7：$cash(j,t_j')–cash(j,t_j)–cash(j,t_j'- t_j)<0$。

因此，若想在任意 2 个 agent i 和 agent j 之间建立单向依赖关系（i 依赖 j），则需同时满足条件 C_1~C_7。

4.3 相互依赖实现

相互依赖是指 agent i 与 agent j 对于合作的偏好都大于不合作的偏好，然而这种偏好是以同时选择合作为共识的，如果 i 和 j 其中的一方选择不合作，

那么另一方对不合作的偏好将大于合作的偏好。因此以 agent i 为例，可以给出 i 形成相互依赖的偏好情况为

P_1：$u_i(C_i, C_j) > u_i(N_i, N_j)$

P_2：$u_i(C_i, C_j) > u_i(N_i, C_j)$

P_3：$u_i(C_i, C_j) > u_i(C_i, N_j)$

P_4：$u_i(N_i, N_j) > u_i(N_i, C_j)$

P_5：$u_i(N_i, N_j) > u_i(C_i, N_j)$

将式(11)~式(16)代入 P_1~P_5 中，可以对应得到 5 个偏好形成的条件：

C_1：$cash(i,t_i')–cash(i,t_i)–cash(i,t_i'- t_i)+cost(i,t)<0$；

C_2：$cash(i,t_i')–punish(i,t)–cash(i,t_i)–cash(i,t_i'- t_i)+cost(i,t)<0$；

C_3：$cash(i,t_i')–cash(i,t_i)–cash(i,t_i'- t_i)<0$；

C_4：$punish(i,t)>0$；

C_5：$cost(i,t)–punish(i,t)>0$。

由于这种相互依赖是对等的，对于 agent j 来说，其形成相互依赖的偏好条件为

C_6：$cash(j,t_j')–cash(j,t_j)–cash(j,t_j' - t_j)+cost(j,t)<0$；

C_7：$cash(j,t_j')–punish(j,t)–cash(j,t_j)–cash(j,t_j'- t_j)+cost(j,t)<0$；

C_8：$cash(j,t_j') – cash(j,t_j)–cash(j,t_j'- t_j)<0$；

C_9：$punish(j,t_c)>0$；

C_{10}：$cost(j,t_c)–punish(j,t_c)>0$。

因此，若想在任意 2 个 agent i 和 j 之间建立相互依赖关系，需同时满足 C_1~C_{10}。

4.4　交互依赖实现

交互依赖属于相互依赖的一种（当任务一致时）。这里只分析另一种情况，而当任务不一致时，agent i 完成任务的过程需要 agent j 的帮助，而 agent j 完成任务的过程也需要 agent i 的帮助，可以看成是 i 与 j 互为单向依赖。因此，无论 agent i 还是 agent j，对于合作的偏好大于不合作的偏好。以 agent i 为例，则 i 形成交互依赖的偏好情况为

P_1：$u_i(C_i, C_j) > u_i(N_i, N_j)$

P_2：$u_i(C_i, C_j) > u_i(N_i, C_j)$

P_3：$u_i(C_i, N_j) > u_i(N_i, N_j)$

P_4：$u_i(C_i, N_j) > u_i(N_i, C_j)$

将式(11)~式(16)代入 P_1~P_4 中，可以相应得到产生 4 个偏好形成的条件：

C_1：$cash(i,t_i')–cash(i,t_i)–cash(i,t_i'- t_i)+cost(i,t)<0$；

C_2：$cash(i,t_i')–punish(i,t)–cash(i,t_i)–cash(i,t_i' - t_i)+cost(i,t)<0$；

C_3：$cost(i,t)<0$；

C_4：$cost(i,t)–punish(i,t)<0$。

同理，对于 agent j 来说，形成交互依赖的条件为

C_5：$cash(j,t_j')–cash(j,t_j)–cash(j,t_j'- t_j)+cost(j,t)<0$；

C_6：$cash(j,t_j')–punish(j,t)–cash(j,t_j)–cash(j,t_j' - t_j)+cost(j,t)<0$；

C_7：$cost(j,t)<0$；

C_8：$cost(j,t) – punish(j,t)<0$。

这里将 4.1 节~4.4 节中得到的结论进行了总结，可以得到依赖关系与形成条件的对应关系如表 2 所示，只有当 agent i 和 agent j 同时满足这些条件时，才能够实现对应的依赖关系。

表 2　简化后的依赖关系对应的条件

合作方式	agent i	agent j
不合作	$cost(i,t) > punish(i,t) \geq 0$ $cashI > punish(i,t) – cost(i,t)$	$cost(j,t) > punish(j,t) \geq 0$ $cashJ > punish(j,t) – cost(j,t)$
单向合作	$cashI < – cost(i,t)$ $cost(i,t) < 0 \leq punish(i,t)$	$cashJ < 0$
双向合作	$cashI < – cost(i,t)$ $cost(i,t) > punish(i,t) \geq 0$ 或 $cost(i,t) < 0 \leq punish(i,t)$	$cashJ < – cost(j,t)$ $cost(j,t) > punish(j,t) \geq 0$ 或 $cost(j,t) < 0 \leq punish(j,t)$

5　效用驱动的多 agent 合作实现

5.1　支撑平台原型

本文提出了效用驱动的多 agent 合作机制，在对该方法进行验证时，选取了一个主流的多 agent 实现平台。这里选用业界广泛采用的 JADE 作为多 agent 系统的测试平台。JADE(Java agent development framework)[19,20]是基于 Java 语言的 agent 开发框架，是由 TILAB 开发的开放源代码的自由软件。具体的平台设计框架如图 2 所示。

5.2　多 agent 合作应用示例

为了实现通过效用函数中的参数值进而影响 agent 的偏好排序，最终建立多 agent 之间的依赖关系并形成相应的合作机制，作者在 JADE 平台上开发了针对多 agent 合作的测试应用 MSYS（mining system）。该测试应用模拟矿石探测车在矿区自主采集矿石的应用场景，将采矿过程中的各个环境实体进行形式化说明，其主要作用是通过简单的合作任务来检测多 agent 系统的合作效果。

对于该测试应用，系统默认给每个 agent 设定的任务目标是，在规定时间内，尽可采集到最多的矿石数量。这就要求多 agent 群体要对矿区中的矿

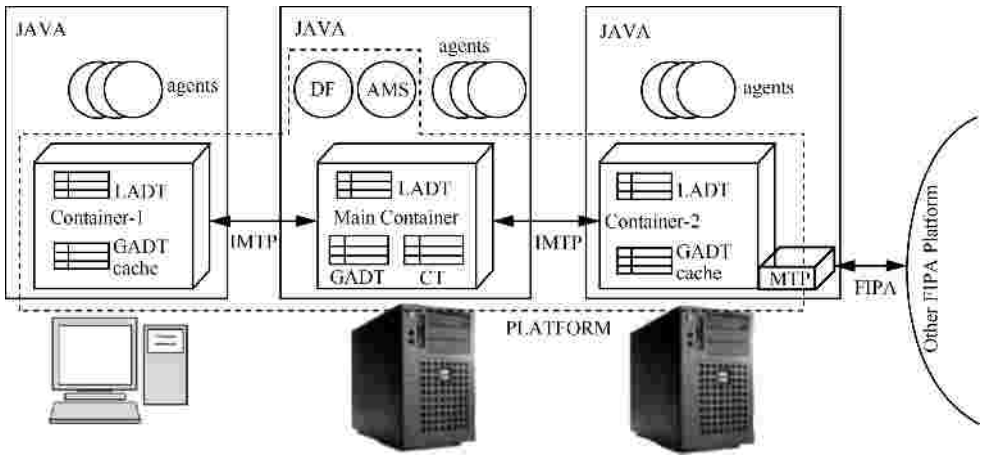

图 2 JADE 平台框架

石进行合理、高效的合作采集。系统为每个 agent 设置了奖励机制，采的矿石越多，得到的奖励就越多，同时在需要建立合作的情况下鼓励合作，惩罚不合作行为，这样在这种机制的驱动下，每个 agent 都以“尽可能多地获得奖励”为目标计算对当前任务的偏好，通过比较不同任务的偏好值指导 agent 本身的行为。

首先对测试应用场景进行简单的描述。

1) 整个场景包括 3 个主要实体：矿车（对应 agent）、矿石和障碍物。

2) 每个 agent 都有一定的采矿上限，不同的 agent 可以采矿的上限也不同，即采矿能力不同。

3) 为了增加环境的动态效果，每个矿石的位置是随机产生的，而且矿石的大小是不相同的。

4) agent 独立采集矿石的重量不能超过自身的上限值，如果超过上限值，这个 agent 就需要与其他 agent 进行合作来采集矿石。

5) 在运行的过程中，agent 相互之间不能碰撞，同时 agent 在行驶过程中需要绕开障碍物。

本文在前面章节中主要分析了 2 个 agent 合作时的各种依赖情况，而在该测试应用中，合作的参与者可能大于 2 个（当矿石的重量很大时，可能会需要多个 agent 同时合作），这是就需要对多个 agent 的合作建立方式进行调整。这里以 3 个 agent 共同参与任务为例来讨论多个 agent 如何建立合作。

假设存在 agent i、j、k，其中，t_i'=1，t_j'=3，t_k'=5。当 i 发现矿石任务 t（t=7）后，决定与 j 进行合作开采，而 i 和 j 的能力不能满足开采需要（$t_i' + t_j' < 7$），这时需要联合 k 来一起合作，此时对于 k 来说，i 和 j 可以看作是一个联合的共同体，即采矿能力为 4 的 agent，而 k 加入合作时，仍以 2 个 agent 建立合作的情况来考虑，如图 3 所示。

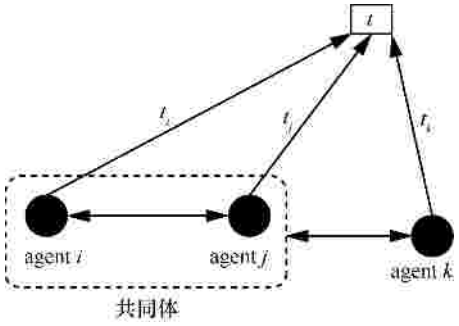

图 3 多 agent 合作实现方法（|*agent*|≥3）

这里对环境中涉及到的参数进行说明。

1) 为了增加 agent 合作的组合方式，这里设定 agent 的采矿能力有 3 种，即 t_i'的值可以为 1、3 或 5。

2) 同时对于矿石的重量来说，为了测试在多种情况下的合作，给定矿石重量的取值范围为 1~10 之间的自然数，每个矿石的重量是 1~10 之间的随机数。

3) *cash*(*ag*,*t*)，表示 agent *ag* 完成采矿任务 *t* 之后所获得的任务收益，收益与采矿的数量成正比。

4) *action*(*ag*,*t*)，表示 agent *ag* 完成采矿任务 *t* 所执行的动作数量。

5) *cost*(*ag*,*t*)，表示合作双方在建立合作时所产生的合作成本，这个值和参与合作的 agent 数量成正比。

6) *punish*(*ag*,*t*)，表示当对方选择合作，而本方选择不合作时，系统给予本方的收益惩罚。

该应用示例的运行界面如图 4 所示，其中带有圆

环的圆点代表负责采矿的 agent，其圆环外延表示感知范围，用来搜索特定范围内的矿石，同时防止 agent 之间的相互碰撞；小面积圆点代表随机产生的矿石，每个矿石的重量不相同；大面积圆点代表障碍物。

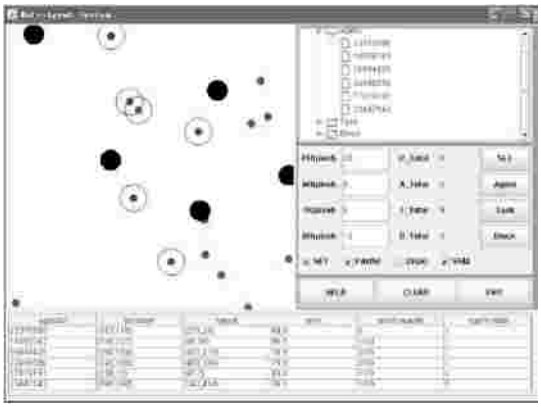

图 4　MSYS 运行界面

5.3　数据分析

根据第 4 节依赖关系的建立过程，$cash(ag,t)$、$cost(ag,t)$、$punish(ag,t)$这 3 个参数值直接决定了 agent ag 如何选择合作方式。为了便于数据分析，这里给出 agent 形成依赖关系的条件。这里将第 4 节中的参数进行等价简化，设定：

$$cashI = cash(i,t_i') - (cash(i,t_i) + cash(i, t_i' - t_i)) \quad (17)$$

$$cashJ = cash(j,t_j') - (cash(j,t_j) + cash(j, t_j' - t_j)) \quad (18)$$

其中，$cash(i,t_i')$表示 agent i 在不参与合作时能得到的任务收益；而 $cash(i,t_i) + cash(i, t_i' - t_i)$表示 agent i 参加合作时能得到的任务收益。

1) 对于相互独立的依赖关系，由 4.1 节中独立依赖形成条件 C_1 和 C_2 可知，$cost(i,t) > 0$，$cost(i,t) - punish(i,t) > 0$，由于 $punish(i,t) \geq 0$，因此可以得到 $cost(i,t) > punish(i,t) \geq 0$；同时由条件 C_3 和 C_4 可以得到：$cashI > punish(i,t) - cost(i,t)$。因此，当满足下列条件时，agent i 和 j 之间不能建立合作：

a) $cost(i,t) > punish(i,t) \geq 0$；

b) $cashI > punish(i,t) - cost(i,t)$；

c) $cost(j,t) > punish(j,t) \geq 0$；

d) $cashJ > punish(j,t) - cost(j,t)$。

2) 对于单向依赖关系（i 依赖 j），因为 $punish(i,t) \geq 0$，由 4.2 节中单向依赖条件 C_3 和 C_4 可以得到：$cost(i,t) < 0$；由条件 C_1 和 C_2 可以得到：$cashI < -cost(i,t)$；由条件 C_5、C_6 和 C_7 可以得到：$cashJ < 0$。因此，当满足下列条件时，agent i 和 j 之间可以建立单向依赖的合作关系。

a) $cashI < -cost(i,t)$；

b) $cost(i,t) < 0 \leq punish(i,t)$；

c) $cashJ < 0$。

3) 对于相互依赖关系，由 4.3 节中相互依赖的形成条件 C_4 和 C_5 可以得到：$cost(i,t) > punish(j,t) > 0$，由条件 C_1、C_2 和 C_3 可以得到：$cashI < -cost(i,t)$。因此，当满足下列条件时，agent i 和 j 之间可以建立相互依赖的双向合作关系。

a) $cashI < -cost(i,t)$；

b) $cost(i,t) > punish(i,t) \geq 0$；

c) $cashJ < -cost(j,t)$；

d) $cost(j,t) > punish(j,t) \geq 0$。

4) 对于交互依赖关系，由 4.4 节中交互依赖的形成条件 C_3 和 C_4 可以得到：$cost(i,t) < punish(j,t)$，进而由条件 C_1 和 C_2 可以得到：$cashI < -cost(i,t)$；同理 agent j 也如此。因此，当满足下列条件时，agent i 和 j 之间可以建立交互依赖的双向合作关系。

a) $cashI < -cost(i,t)$；

b) $cost(i,t) < 0 \leq punish(i,t)$；

c) $cashJ < -cost(j,t)$；

d) $cost(j,t) < 0 \leq punish(j,t)$。

具体简化后的形成依赖关系条件如表 2 所示。

为了验证效用驱动的合作机制在多 agent 系统中的有效性，这里给出了几组动态测试数据来检验在随机环境下，不同参数值对合作建立以及合作结果的影响。

这里首先设定，agent 的任务能力分为 2、3、5，即 agent 可以独立完成 t_i=2，3，5 的任务，为了保证测试的有效性，在每次测试中，不同能力的 agent 比重是一致的；系统中存在的任务量是从 1~5 的自然数中随机产生。由于 agent 每次执行任务时所获得的收益（cash）、产生的成本（cost）和获得的惩罚（punish）都是根据任务量的不同决定的，因此这里给出几组动态测试数据来约束 agent 合作方式，如表 3 所示。

表 3　动态测试数据

组数	$cash(i,t_i)$	$cash(i,t_i')$	$cash(i, t_i' - t_i)$	$cost(i,t)$	$punish(i,t)$
1 组	$1.1t_i$	t_i'	$t_i' - t_i$	$0.2t$	$0.1t$
2 组	$1.1t_i$	t_i'	$t_i' - t_i$	$-0.2t$	$0.1t$
3 组	$1.5t_i$	t_i'	$t_i' - t_i$	$0.2t$	$0.1t$

对于 1 组动态数据来说，根据表 2 的依赖分析，该组数据会使得多 agent 系统的所有 agent 都处于独立状态，同理 2 组会产生一定比例的单向合作，而 3 组会根据任务的不同，形成双向合作。在每次测试验证过程中，不断加大任务群体中合作任务的比重，即提高 $t = 4$ 或 5 在任务量中的比重，这样就可以验证在合作任务量大的情况下，各种不同合作模式的效率情况。这里设定任务能力为 2 的 agent 个数为 3，任务能力为 3 的 agent 个数为 3，任务能力为 5 的 agent 个数为 4；总的任务个数为 100；分 10 次测试，每次测试都提高 $t = 4$ 或 5 的任务 10%的比重。测试结果如图 5 所示。

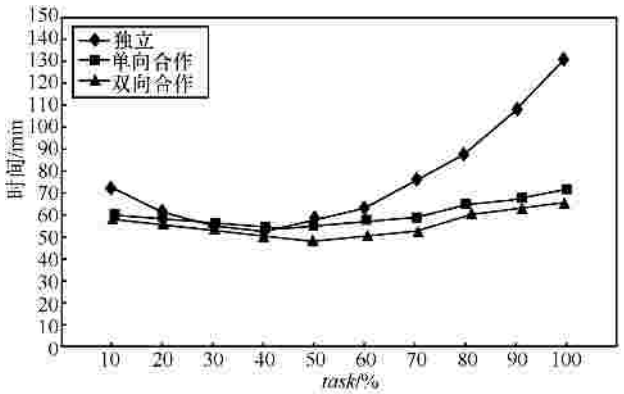

图 5 不同参数下多 agent 的合作情况

由图 5 中的合作情况可见，随着合作任务比重的增加，可独立完成任务的 agent 数量开始下降，对于 1 组来说，系统中的 agent 都处于独立的合作状态，因此随着合作任务的增加，可以完成任务的 agent 相对减少，因此总体完成时间有所增加；而对于 2 组来说，根据任务量的不同，会产生一定比例的单向合作 agent，因此在总体完成时间上要优于 1 组；而 3 组将 agent 系统设置为倾向双向合作方式，不会出现合作拒绝情况，其合作效率要高于单向合作，总体执行时间要优于 2 组。

除了验证效用驱动合作机制的有效性之外，还需要验证该机制是否能在任务合作方面使执行效率提升。这里预先设定 10 个 agent 的合作情况，其中，agent 的任务能力分为：2、3 和 5；任务量为 1~10 之间的随机整数；每分钟产生 10 个任务。分为 3 种情况来比较效用驱动的合作效果，如表 4 所示。其中，表 4 的随机合作是预先设定的合作方式，而在系统运行过程中，由 agent 随机组合来完成。其合作过程是根据任务量的大小决定的，该方法追求多个 agent 以饱满任务量的方式完成合作，例如当任务 $t = 10$ 时，合作倾向“2+3+5”或“5+5”的方式完成合作。

表 4 3 种情况的数据对比

测评参数	无合作	随机合作	效用合作
agent 个数	10	10	10
运行时间/min	10	10	10
任务总量	447	438	445
完成总量	192	326	364
完成比例	42.95%	74.42%	81.79%
单 agent 效率	1.9/min	3.3/min	3.6/min

表 4 中的任务总量和完成量是 5 次测试的平均值。分析表中给出的测试结果，在无合作的情况下，agent 的能力上限为 5，因此对于任务量在 5~10 的任务不能处理，运行效果如图 6(a)所示；在随机合作的情况下，虽然合作后的多 agent 任务能力可以覆盖所有的任务，但预定的合作方式限制了 agent 合作建立的成功率，运行效果如图 6(b)所示；而基于效用的合作机制，不限制 agent 的合作方式，每 agent 以单位时间内尽可能多地完成任务为目标，运行效果如 6(c)所示。

从图 6 可以看到，基于合作机制的多 agent 系统，其完成任务的能力要高于非合作的 agent 系统，而基于效用驱动的合作效果要优于随机合作方式，其原因主要是由于随机合作方式在多 agent 建立合作的过程中，其交互成本要高于效用驱动方式，所以在相同时间内，任务完成量要少于后者。这就证明了效用驱动合作机制在执行效率方面比常规的随机合作有一定的提升。

6 结束语

本文提出的基于效用函数的多 agent 群体合作机制，将效用函数与 agent 偏好关联起来，通过控制效用函数中的参数值来控制每个 agent 的具体偏好，进而影响 agent 对于合作方式选择的自主性。这样就可以使 agent 的行为与抽象的数学模型相关联，将环境中的任务进行形式化抽象，同时附上值属性，这样就可以将环境中的实体以参数的形式影响多 agent 的合作。

下一步将继续深入完善该合作机制，主要包括以下 4 个方面。

1) 在基于效用函数的合作机制中，加入拍卖策

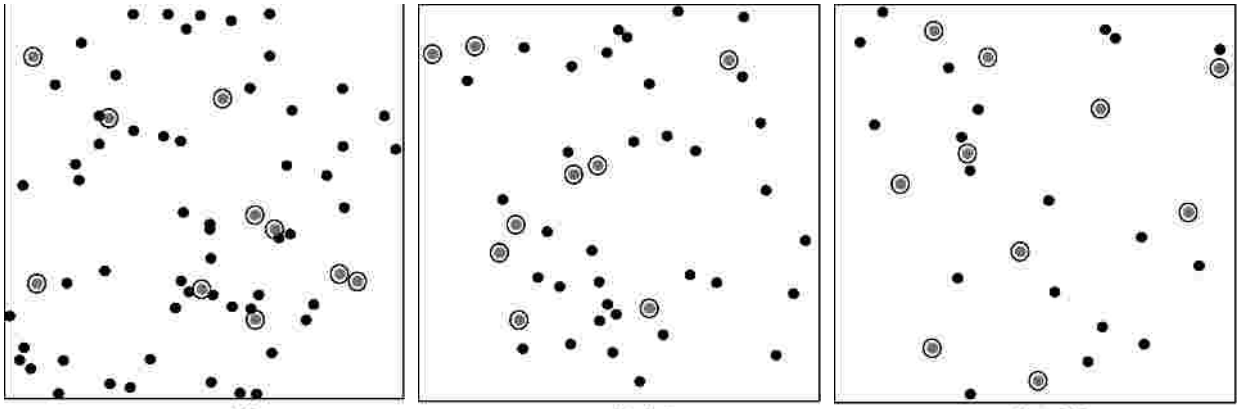

图 6　3 种合作情况下的多 agent 运行结果

略，这主要用于解决资源（任务）与 agent 个数相差较大的情况下，如何实现任务分配的问题。

2) 目前，本文系统地通过控制效用函数的方式来影响agent的合作，并没有考虑agent之间的交互。下一步将增加 agent 之间的交互过程，以此来小范围地辅助合作、优化合作。

3) 目前，本文的测试应用还不具有通用性，当遇到复杂环境时，任务和环境中的实体很难被抽象成 agent 可用的信息。下一步考虑利用本体，将环境抽象成通用的概念模型，agent 可以利用本体工具将环境概念转换成所需要的信息。

4) 目前的效用计算方法在效率上还有优化的空间，下一步主要将算法进行优化，以提高多 agent 系统的合作效率。

参考文献：

[1] WOOLDRIDGE M. An Introduction to Multi agent System[M]. West Sussex: John Wiley & Sons Ltd, 2002.

[2] BELLIFEMINE F, CALRE G, GREENWOOD D. Developing Multi-agent Systems with JADE[M]. West Sussex: John Wiley & Sons Ltd, 2007.

[3] SICHMAN J, DEMAZEAU Y. Exploiting social reasoning to with agency level inconsistency[A]. Proceedings of the 1st International Conference on Multi-agent Systems (ICMAS-95)[C]. San Francisco, USA, 1995. 352-359.

[4] SICHMAN J. A social reasoning mechanism based on dependence networks[A]. Proceedings of the 11th European Conference Artificial Intelligence (ECAI-94)[C]. Amsterdam, Netherlands, 1994.188-192.

[5] MILCH B, KOLLER D. Probabilistic models for agents' beliefs and decisions[A]. Proceedings of the 16th on Uncertainty i Artificial Intelligence (UAI-00)[C]. Stanford, California,2000. 389-396.

[6] PANZARASA P, JENNINGS N R. Social mental shaping: mode ling the impact of sociality on the mental states of autonomous agents[J]. Computation Intelligence, 2001,17(4):738-782.

[7] PLOTKIN T K. Algorithms of distributed task allocation for cooperative agents[J]. Theoretical Computer Science, 2000, 242(1-2):1-27.

[8] POLLACK M E. In Intentions in Communication[M]. Cambridge: MIT Press, 1990.

[9] SURYADI D, GMYTRASIEWICZ P J. Learning models of other agents using influence diagrams[A]. Proceedings of 7th International Conference on User Modeling (UM-99)[C]. Banff, Canada, 1999. 223-232.

[10] WOOLDRIDGE M, JENNINGS N R. Formalizing the cooperative problem solving process[A]. Proceedings of the 13th In national Workshop on Distributed Artificial Intelligence (IWDAI-94)[C]. Lake Quinalt, WA, 1998.403-417.

[11] HADDADI A S. Communication and cooperation in agent system: a pragmatic theory[A]. Lecture Notes in Artificial Intelligence and Lecture Notes in Computer Science[C]. New York: Springer-Verlag Inc, USA,1996.1-134.

[12] DUNIN-KEPLICZ B, VERBRUGGE R. Collective intentions[A]. Proceedings of the 2th International Conference on Multi-agent System[C]. Menlo Park, California, 1996.56-63.

[13] CASTELFANCHI C. From individual intentions to groups and organizations[A]. Proceedings of the 1th International Conference on Multi agent System[C]. San Francisco, USA, 1995.34-40.

[14] ZLOTKIN G, ROSENSCHEIN J S. A domain theory for task oriental negotiation[A]. Proceedings of the 13th International Joint Conference on Artificial Intelligence[C]. California, USA, 1993.416-423.

[15] MATSUBAYASHI K, TOKORO M. A collaboration mechanism on positive interactions in multi-agent environment[A]. Proceedings of the 13th International Joint Conference on Artificial ntelligence[C]. California, USA, 1993.345-351.

[16] BARBER T, LIU H, HAN D C. agent-Oriented Design[M]. Austin: The University of Texas,1999.

[17] WOOLDRIDGE M, JENNINGS N R, KINNY D. A methodology for agent-oriented analysis and design[A]. Proceedings of the 6th National Conference on Artificial Intelligence (AAAI-99)[C]. Orlando, FL,1999. 153-162.

[18] PANZARASA P, JENNINGS N R. Formalizing collaborative decision-making and practical reasoning in multi-agent systems[J]. Journal of Computation, 2002, 12(1):55-117.

（下转第 142 页）

[4] BATINA L, GIERLICHS B, PROUFF E. Mutual information analysis: a comprehensive study[J]. Journal of Cryptology, 2011, (24):269-291.
[5] WHITNALL C, OSWALD E, MATHER L. An exploration of the Kolmogorov-Smirnov test as competitor to mutual information analysis[EB/OL]. http://eprint.iacr.org/2011/380.pdf.
[6] BOGDANOV A. Improved side-channel collision attacks on AES[A]. SAC 2007, LNCS 4876[C]. Berlin:Springer, 2007.84-95.
[7] DINUR I, SHAMIR A. Side channel cube attacks on block ciphers[EB/OL]. http://eprint.iacr.org/2009/127.
[8] RENAULD M, STANDAERT F -X. Algebraic side-channel attacks[A]. INSCRYPT 2009[C]. Berlin:Springer,2009.393-410.
[9] RENAULD M, STANDAERT F X, VEYRAT C N. Algebraic side-channel attacks on the AES:Why time also matters in DPA[A]. CHES 2009[C]. Berlin:Springer, 2009.97-111.
[10] OREN Y, KIRSCHBAUM M, PPOPP T. Algebraic side-channel analysis in the presence of errors[A]. CHES 2010[C]. Berlin:Springer, 2010.428-442.
[11] ZHAO X J, ZHANG F, GUO S Z. MDASCA:an enhanced algebraic side-channel attack for error tolerance and new leakage model exploitation[A]. Proceedings of COSADE 2012[C]. 2012.
[12] GUO J, PEYRIN T, POSCHMANN A. The LED bock cipher[A]. CHES 2011[C]. Berlin:Springer, 2011.326-341.
[13] NICOLAS T. C, GREGORY V. B. Algebraic cryptanalysis of the data encryption standard[A]. 11-th IMA Conference[C]. Cirencester, UK, 2007. 152-169
[14] FU Z, MARHAJAN Y, MALIK S. zChaff SAT solver[EB/OL]. http://www.princeton.edu/ chaff/.
[15] E´EN N, SÖRENSSON N. An open-source SAT solver package[EB/OL]. http://www.cs.chalmers.se/Cs/Research/FormalMethods/Mi iSat/.
[16] SOOS M, NOHL K, CASTELLUCCIA C. Extrending SAT solvers to cryptographic problems[A]. SAT 2009[C]. Berlin:Springer. 2009.244-257.
[17] TOBIAS A. Constraint Integer Programming[D]. TU Berlin, 2007.
[18] BERTHOLD T, HEINZ S, PFETSCH M E. SCIP- solving constraint integer programs[A]. SAT 2009[C]. Berlin:Springer, 2009.
[19] BERTHOLD T, HEINZ S, PFETSCH M E. Nonlinear pseudo-boolean optimization:Relaxation or propagation? [A]. SAT 2009[C]. Berlin:Springer, 2009.441–446.
[20] CARLA P. G, ASHISH S, Handbook of Satisfiability[M]. IOS Press, 2009.633-654.
[21] KNUDSEN L R, MIOLANE C V. Counting equations in algebraic attacks on block ciphers[J]. International Journal of Information Security, 2010, 9(2):127-135.
[22] CHARI S, RAO J R, ROHATAJ P. Template attacks[A]. CHES 2002[C]. Berlin:Springer, 2002.13-28.
[23] BRIER E, CLAVIER C, OLIVIER F. Correlation power analysis with a leakage model[A]. CHES 2004[C]. Berlin:Springer, 2004. 16-29.

作者简介：

冀可可（1988-），女，河南上蔡人，军械工程学院硕士生，主要研究方向为分组密码代数旁路分析。

王韬（1964-），男，河北石家庄人，博士，军械工程学院教授、博士生导师，主要研究方向为信息安全、密码学。

郭世泽（1969-），男，河北石家庄人，博士，主要研究方向为信息安全、密码学。

赵新杰（1986-），男，河南开封人，军械工程学院博士生，主要研究方向为分组密码旁路分析和故障分析。

刘会英（1984-），男，湖北黄石人，军械工程学院博士生，主要研究方向为图像加密和密码旁路分析。

（上接第 133 页）

[19] SHEHORY O, KRAUS S. A kernel-oriented model for coalition formation in general environments: implementation and res lts[A]. Proceedings of the 3th National Conference on Artificial ntelligence (AAAI-96)[C]. Anaheim, CA, 1996. 715-723.
[20] Java agent development framework (JADE)[EB/OL]. http://jade.tilab.com.

张霞（1965-），女，辽宁沈阳人，博士，东北大学教授，主要研究方向为软件工程、数据库、面向服务智能应用等。

作者简介：

邓寒冰（1984-），男，辽宁沈阳人，东北大学博士生，主要研究方向为软件 agent **与本体技术。**

刘积仁（1955-），男，辽宁沈阳人，博士，东北大学教授、博士生导师，主要研究方向为软件工程、数据库、计算机图形学、医学影像计算、网络安全与管理和嵌入式技术与系统等。