A Polynomial Kernel-Oriented Coalition Algorithm for Rational Information Agents

Matthias Klusch
Institut für Informatik,
Christian-Albrechts-University Kiel,
24118 Kiel, Germany

Onn Shehory
Department of Mathematics & Computer Science,
Bar Ilan University,
Ramat Gan, 52900 Israel

Abstract

Information agents behave like active intelligent front-ends of stand-alone information systems. The main purpose of such an agent is to gather intensionally relevant information in non-local domains. They may either work as individuals or efficiently cooperate in order to satisfy their own set of information search tasks given by their local customers. However, in particular the need to respect the database autonomy and to cope with semantic heterogeneity hinders such a cooperation. In addition, as the Internet becomes increasingly commercialized, the agents will receive monetary rewards for their services and may negotiate with each other to maximize their expected utility.
In this paper we present a new solution for such an utilitarian information-gathering by rational cooperation. For this purpose, a game-theoretic based algorithm for coalition formation among the information agents is used. The local calculation of agent-utility bases on each agent's productions, resulting from the execution of information search tasks. These monetary utilities and respective coalition values are utilized by the proposed coalition algorithm. It enables decentralized cooperation via the formation of Kernel-oriented stable coalitions among rationally cooperating information agents in polynomial time.

1 Introduction

The increase in the amount of information in the last decades has caused the formation of multiple heterogeneous, autonomous databases in a decentralized environment as well as their global interconnections in the Internet. As a result, the search for semantically relevant, non-local data has become an exhaustive procedure, either for a human user or for a computerized data-search server.

Solutions to this information-gathering problem were provided by different types of agent-based systems, like the central information agent approach in (Barbuceanu & Fox 1994), work in the InfoSleuth project (Jacobs & Shea 1995), the Amalthaea system in (Moukas 1996) or the MACRON system in (Decker & Lesser 1995). In addition, there is also a recent trend to create mobile software agents which are capable for an user-dependent information search in the WWW, like e.g. in (Jacobs & Shea 1996).

So far it is assumed that there is sufficient common interest among the information agents that they will frequently volunteer to help each other and receive no reward for their labor. But as the Internet becomes more and more commercialized, it is highly probable that these agents will act on behalf of their creators to make a monetary profit. In such a scenario information agents demand payments from their local customers and other agents for the services they provide and may negotiate with each other to maximize their expected individual utility.

The first approach to provide such an utilitarian information-gathering by rationally cooperating information agents was introduced in (Klusch 1994; 1996b). A system FCSI[1] of so-called *cooperative information agents* was designed to solve the problem of recognizing intensional data dependencies between autonomous, stand-alone databases. For this purpose, methods from terminological knowledge representation and inference, as well as game-theoretic based formation of utilitarian coalitions were applied. In this paper we will focus on the latter aspect of how to build rational coalitions among autonomous information agents.

Several solutions to the problem of utilitarian coalition formation have been suggested by researchers in the field of Distributed Artificial Intelligence (DAI). Some solutions concentrate on the special case of autonomous agents in a super-additive environment (Shehory & Kraus 1993; Ketchpel 1993; Zlotkin & Rosenschein 1994). provided for coalition formation in Multi-Agent Systems (MAS), where each agent tries to increase its own personal utility via cooperation. Other MAS solutions for general environments are e.g. given in (Shehory & Kraus 1996) and in (Sandholm & Lesser 1995). These solutions strongly base on the game theory concepts. However. some coalition formation algorithms in DAI are based on operations research, combinatorial algorithms and graph theory (Shehory & Kraus 1995). These are most appropriate for coali-

[1]FCSI is a shortcut for '*F*ederative *C*ell *S*ystem for an utilitarian discovery of *I*nterdatabase dependencies'.

tion formation in Distributed Problem Solving (DPS) cases, where the agents cooperate in order to increase the outcome of the whole system.
The coalition formation model proposed for the FCSI in (Klusch & Shehory 1996) leads to individually rational solutions but still does not guarantee stability in a coalition-theoretical sense (cf. Sect.3 in (Kahan & Rapoport 1984)). Therefore, we present an algorithm which enables utilitarian coalition formation for information agents in general types of environments with a guaranteed stability in polynomial time. We base our solution on the Kernel stability concept from game theory. It is to some extent similar to the work in (Shehory & Kraus 1996), but the latter is inappropriate for the specific case of information agents. Hence, we provide a new Kernel-oriented coalition algorithm in terms of specific functions and procedures to be performed by each information agent locally, state its complexity and discuss its essential properties.

2 Information Agents

In the literature there exist many different views on what an agent is: the answer mostly depends on who poses the question. This is also the case for the term *information agent*. In (Wooldridge & Jennings 1995) it is defined as an agent which has access to at least one, and potentially many information sources, and is able to collate and manipulate information obtained from these sources to answer queries posed by users and other information agents. Others restrict the term to a more linguistic-based negotiation between knowledge-based entities which relies on a presumed common ontology (see e.g. (Jacobs & Shea 1995; Weigand 1995)). We define a *cooperative information agent* as an active, intelligent database front-end which tries to satisfy its own information search tasks alone or by an utilitarian cooperation with others.
The federative agent system FCSI is a set of cooperative information agents (cf. Fig. 1) which try to rationally coalesce with each other to gather some relevant information in non-local domains.

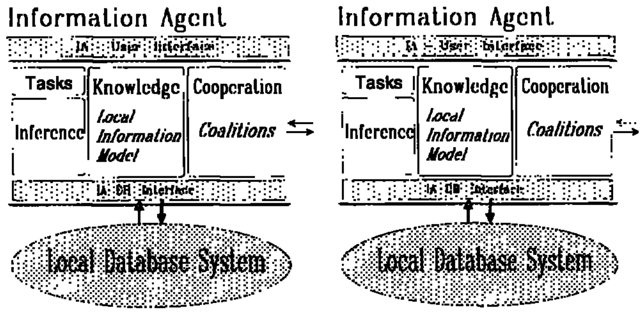

Figure 1: FCSI Information Agents

For this purpose each FCSI agent builds its own local terminological information model LIM using an appropriate description logic. Such an information model contains a linguistic description of semantic aspects of selectively exportable schema data[2]. Each agent is able to detect several kinds of interrelation knowledge concerning these schema data locally[3]. Fig. 2 summarizes the cooperation phases of the FCSI.

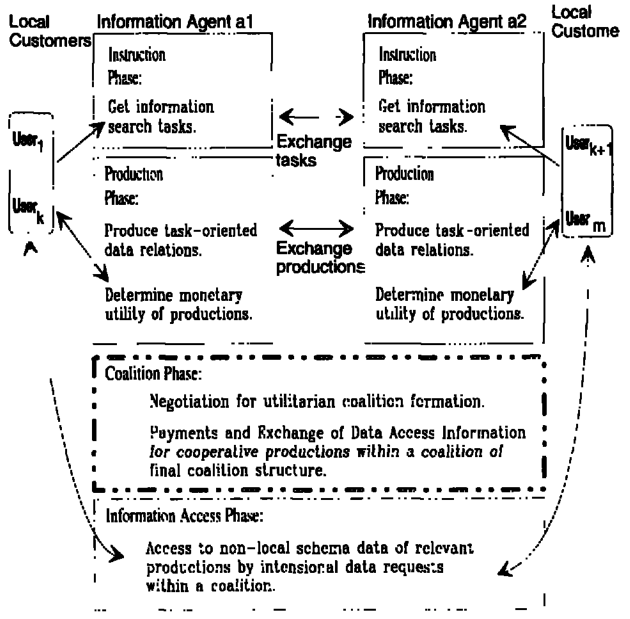
Local Customers
Information Agent a1
Information Agent a2
Local Customers
Instruction Phase:
Get information search tasks.
Exchange tasks
Instruction Phase:
Get information search tasks.
User1
Userk
Userk+1
Userm
Production Phase:
Produce task-oriented data relations.
Exchange productions
Production Phase:
Produce task-oriented data relations.
Determine monetary utility of productions.
Determine monetary utility of productions.
Coalition Phase:
Negotiation for utilitarian coalition formation.
Payments and Exchange of Data Access Information for cooperative productions within a coalition of final coalition structure.
Information Access Phase:
Access to non-local schema data of relevant productions by intensional data requests within a coalition.

Fig. 2: Cooperation among FCSI-Agents

The agents receive their information search tasks from their local customers as well as from other agents. During a knowledge-based production phase each agent determines its task-oriented productions. By processing the set of search tasks an agent produces a set of *terminological relations* which relate its local schema data to non-local one. This is done by comparing the linguistic data descriptions which are part of the received tasks. For this purpose, methods for terminological classification are applied. Evaluation of monetary utility of a task-oriented production for an agent bases on the price that each customer is willing and committed to pay for relevant information, i.e., the parts of the agent's productions which satisfy the given search tasks[4]. It is possible to use also local production costs for utility calculation.
A complementary view of monetary utility can be obtained by the so-called informative utility of a production. It measures the relevance of the recognized data

[2]Related works are (Blanco 1994; Jacobs & Shea 1995) and the LICS/LIKE project, e.g. (Burg 1993).

[3]This includes the recognition of so-called interdatabase dependencies which are comparable e.g. to the boolean-valued data dependency predicate in (Sheth, Karabatis, & Rusinkiewicz 1991).

[4]A task t is satisfied by another one t' (in short $t' \xrightarrow{tr} t$) iff both data descriptions are related by a kind of recognized terminological relation tr as it is specified in task t.

relations in terms of the quantity of task satisfactions. In addition, numeric textual similarity factors from the area of information retrieval may be considered for a qualitative production utility. For reasons of space limitation, we omit the discussion of this issue (cf. (Klusch 1996a)).
Upon the completion of the coalition formation phase, each customer is authorized to access the relevant data for his search tasks. Access is restricted to the information domains of the agents with which the local information agent has formed a coalition. No coalition breaking is allowed in the FCSI. Electronic payments and information transfers among the agents are performed only if the coalition configuration is guaranteed not to alter[5].
We will focus now on the coalition formation process where each information agent tries to get a maximal monetary profit for its productions.

3 Game-Theoretic Definitions and Concepts

Below we provide definitions of some game-theoretic concepts which are necessary to understand our utilitarian coalition formation process in Sect. 5.

Definition 3.1: *Cooperative Game* $(\mathcal{A}, v)$

Let $\mathcal{A}$ be a set of agents a_i, $i \in \{1,..n\}$ and $C \subseteq \mathcal{A}$ be a coalition.

1. *Characteristic function* (or coalition value function) of the game $(\mathcal{A}, v)$ assigns to each coalition C a value which measures its utility achievable as a whole by cooperation among its members: $v : \mathcal{P}(\mathcal{A}) \mapsto \mathbb{R}^+, v(\emptyset) := 0$.
2. $v(\{a_i\})$ denotes the *self value* of a single-agent coalition.
3. Each partition of $\mathcal{A}$ (a set of non-empty, mutually disjunctive subsets $C \subseteq \mathcal{A}$) is called a *coalition structure* $\mathcal{C}$, $|\mathcal{C}| = m \leq n$.
4. A *payment configuration* $PC = (u, \mathcal{C})$ for the coalition structure $\mathcal{C}$ denotes a *payoff distribution* $u : P(\mathcal{A}) \mapsto \mathbb{R}$ of coalition values in $(\mathcal{A}, v)$ to each agent: $(u_1, ..., u_n: C_1, ..., C_m)$, $u_i \equiv u(\{a_i\})$, $u(R) := \sum_{a_i \in R} u_i$ with efficient (utility) distribution $u(C_k) = v(C_k), \forall C_k \in \mathcal{C}$.

Main assumptions for a game $(\mathcal{A}, v)$:

- each coalition $C \subseteq \mathcal{A}$ is given a value $v(C)$.
- a payment configuration represents a solution of the game.
- (monetary) utility units are comparable and transferable among the agents.

□

Definition 3.2: *(Non-)Super-additive Games*

Let $\mathcal{A}$ be a set of agents a_i, $i \in \{1,..n\}$, $(\mathcal{A}, v)$ a cooperative game.

$(\mathcal{A}, v)$ is *super-additive* $:\Leftrightarrow$
$\forall C_k, C_l \subseteq \mathcal{A} : v(C_k \cup C_l) \geq v(C_k) + v(C_l)$

$(\mathcal{A}, v)$ is *non-super-additive* $:\Leftrightarrow$
$\exists C_k, C_l \subseteq \mathcal{A} : v(C_k \cup C_l) < v(C_k) + v(C_l)$

a_i, a_j are *symmetric agents* $:\Leftrightarrow$
$\forall C_k \subset \mathcal{A}, a_i, a_j \notin C_k : v(C_k \cup \{a_i\}) = v(C_k \cup \{a_j\})$.
□

Definition 3.3: *Rational Configurations*

Let $PC = (u, \mathcal{C})$ be a payment configuration for a coalition structure $\mathcal{C}$.

1. Rationality of the PC:

Individually rational	$\forall a_i \in \mathcal{A} : u_i \geq v(\{a_i\})$
Group rational	$u(\mathcal{A}) = v(\mathcal{A})$
Coalition rational	$\forall T \subseteq \mathcal{A} : u(T) \geq v(T)$

2. Pareto-Optimality of the PC:
pareto-optimal: $\neg \exists u' : \forall i \in \{1,..n\}\ u'_i \geq u_i$
□

A solution of a cooperative game $(\mathcal{A}, v)$ is denoted by a set of payment configurations PC which are at least individually rational.
The meaning of the notion of stability depends on the particular coalition theory it is associated with. Some well-known notions of stable utility division are e.g. those of the Core **C**, Stable Set **S**, Bargaining Set **M** and the Kernel **K**. A calculated solution of a given game is called CT-stable ($CT \in \{\mathbf{C}, \mathbf{S}, \mathbf{M}, \mathbf{K}\}$), if all its PCs are stable in a sense given by the actually considered coalition theory CT. Thereby, each coalition theory defines the possible solution space for a cooperative game. For an introductory survey of the theories we refer to e.g. (Kahan & Rapoport 1984; Shehory & Kraus 1996).
Since the present work provides a Kernel-oriented coalition algorithm we give the definition of the Kernel **K** as well as the respective basic calculation scheme, the modified transfer scheme for a **K**-stable PC, and omit the other coalition theories.
The Kernel (Davis & Maschler 1965) is a set of payment configurations PC in which the coalition structure $\mathcal{C}$ is stable in the sense that there is an equilibrium between all pairs of individual agents which are in the same coalition, i.e. they do not dominate each other in such a PC. It leads symmetric agents to receive equal payoffs.

Definition 3.4: *The Kernel* **K**

The set of payment configurations of a cooperative game, where each pair of agent is in an equilibrium is called *Kernel*
$\mathbf{K} := \{(u, \mathcal{C}) \mid \forall a_k, a_l \in C \in \mathcal{C} : (a_k, a_l) \text{ in equilibrium}\}$, with

1. *Excess of a coalition* $C \notin \mathcal{C}$ with respect to an utility distribution u in $(u, \mathcal{C})$: $e(C, u) := v(C) - u(C)$.

[5]This assumption can be weakened iff for each agent there is no decrease in monetary benefit or amount of available information.

2. *Surplus*, or strength, s_{kl} of agent a_k over agent a_l in the same coalition ($a_k, a_l \in C \in \mathcal{C}$) wrt. a payment configuration $(u, \mathcal{C})$: $s_{kl} := \max_{R \notin \mathcal{C}, a_k \in R, a_l \notin R} e(R, u)$.
3. Agent a_k *dominates* agent a_l in the same coalition ($a_k, a_l \in C \in \mathcal{C}$) with respect to a payment configuration $(u, \mathcal{C})$ iff $s_{kl} > s_{lk}$ and $u_l > v(\{a_l\})$.
4. A pair of agents a_l, $a_k \in C \in \mathcal{C}$ is in *equilibrium* iff $(s_{kl} = s_{lk}) \vee (s_{kl} > s_{lk} \wedge u_l = v(\{a_l\})) \vee$ $(s_{kl} < s_{lk} \wedge u_k = v(\{a_k\}))$.

□

Definition 3.5: *Modified Transfer Scheme for* **K**-*stable Payment Configurations*

1. Each sequence $(u^{(1)}, \mathcal{C}), ..., (u^{(i)}, \mathcal{C}), ...$ of payment configurations transferring in each step i positive sidepayments α among the agents a_k, a_l by the following assignments
$$(u^{(i+1)}, \mathcal{C}) : \begin{cases} u_k^{(i+1)} & := u_k^{(i)} + \alpha \\ u_l^{(i+1)} & := u_l^{(i)} - \alpha \\ u_j^{(i+1)}, (j \neq k, l) & := u_j^{(i)} \end{cases}$$
is called a *Transfer Scheme*. The calculation of α is determined by the considered coalition theory, means the notion of stability. A *convergent transfer scheme for* **K**-*stable PCs* iteratively computes for a given coalition structure $\mathcal{C}$ and an initial payment configuration $PC^0 = (u, \mathcal{C})$ another payment configuration $(u', \mathcal{C})$ which is in the Kernel **K**.
2. The convergent Transfer Scheme for **K**-stable PC (Stearns 1968) is determined by the mutual *demand* d_{kl} for each pair of agents a_k, a_l as an upper bound for sidepayments α.

   Let $PC = (u, \mathcal{C})$, s_{kl} surplus of agent a_k over agent a_l in PC, $a_k, a_l \in C \in \mathcal{C}$, $\alpha \leq d_{kl}$ with
$$d_{kl} := \begin{cases} \min\{(s_{kl} - s_{lk})/2, u_l\} & \text{if } s_{kl} > s_{lk} \\ 0 & \text{else} \end{cases}$$
3. *Modified Transfer Scheme* for the Kernel **K**:

   Calculation of an ε-approximately **K**-stable payment configuration $PC = (u, \mathcal{C})$ by terminating Stearns transfer scheme[6] for **K**-stable PC iff the *relative PC-kernel error* $re(u)$ is sufficiently small
   $re(u) := \frac{\max\{s_{ij} - s_{ji} \mid a_i, a_j \in C \in \mathcal{C}\}}{u(\mathcal{A})} \leq \varepsilon.$

□

There exist also well-known transfer schemes for the Core and the Bargaining Set in game-theory literature, but in general no algorithm for calculating a stable payment configuration *and* concurrently forming the respective coalition structure[7]. Thus, most of the game theory work only predicts, given a coalition configuration, how the players (or agents) will distribute the payoffs. They do not provide them with an algorithm for the formation of coalitions.

Definition 3.6: *Coalition Models*

A *Coalition Environment CE* defines a kind of cooperative games $(\mathcal{A}, v)$, i.p. all assumptions which are necessary to calculate the characteristic function and payoffs[8].
If an environment CE induces superadditive or allows also non-superadditive games, the environment is called *superadditive* or *general*, respectively.

A *Coalition Algorithm CA* defines the sequence of actions which has to be executed by each agent locally in order to reach a stable payment configuration as a solution of a game in an environment CE. The algorithm may have the following properties

- be computation- or negotiation-oriented.
- be centralized or decentralized.
- terminate with a polynomial computational complexity.
- provide a stable payment configuration, optionally pareto-optimal.
- determine behaviour and decision-making process of each coalition unit.

A *Coalition Model* $CM = (CE, CA)$ is defined by both, the environment CE and the algorithm CA. □

Development of coalition models by designing regularities and strategies in an abstract manner for various coalition environments appears in the area of DAI, e.g. in (Zlotkin & Rosenschein 1994; Shehory & Kraus 1993; 1996). Recently, interdisciplinary research started to investigate how to adapt such proposals, if possible, to ones which are also appropriate for application-specific environments, like in (Fischer & Müller 1996; Klusch & Shehory 1996).
The coalition algorithm we will present in Sect.4.1 is negotiation-oriented, decentralized, terminates within a polynomial time and yields a stable payment configuration. It is appropriate for general coalition environments, thus not restricted to super-additive games.

4 Coalitions of Information Agents

For coalition formation among the rational information agents of the FCSI we adapted the production-oriented calculation of utility values in (Shehory & Kraus 1993). Each information agent produces interrelation knowledge based on their information search tasks (cf. Sect.2). The utility an agent obtains for productions on its own search tasks by itself is denoted as the agent's self-value. The value of a coalition is the sum of all production utilities of all its members. We assume that the agents know of the commonly used utility function and agree on the utility division method. The used coalition algorithm (cf. Sect.4.1)

[6]The truncated transfer scheme still guarantees convergence since the α-reduction process which leads to convergence is not modified.

[7]Note that each transfer scheme in game-theory computes a payoff vector for a given, *fixed* coalition structure $\mathcal{C} \subseteq \mathcal{P}(\mathcal{A})$.

[8]This includes the monetary system as well as requirements for the coalition formation process.

converges to individually rational and Kernel-oriented stable coalition structures (cf. Def.3.1)[9].

One important issue in the information agents domain is the aspect of information availability. Nearly all practical information systems provide methods for permitting access rights to local data for particular user groups. These authorizations must be respected by the agents. In this sense, the agents productions are called available and production utilities are contributable to a particular coalition value. All agents have to satisfy the following negotiation constraint (IAC): only members of the same and fixed coalition are mutually committed to provide the access to the local production data they used for evaluating the value of their coalition. Consequently, each search task or task-relevant production which is sent by one information agent to another, implies an access-right for the latter to the respective production data of the first, if they will be in the same fixed coalition[10].

<u>**Definition**</u> **4.1:** *Productions and Utilities*

Let $\mathcal{A}_{FCSI}$ be a set of n FCSI information agents, $a_i, a_k, a_j \in \mathcal{A}_{FCSI}$, $C \subseteq \mathcal{A}_{FCSI}$. Let further be

1. *Agents Productions*:
   - $p(t^{a_k}_{tid_x,a_j})$ the knowledge-based production of agent a_k for own or received search task $t_{tid_x} \in OTS^i_j \cup RTS^i_j$ from agent a_j, i.e., all discovered interdatabase dependencies between schema data of a_k and a_j with respect to the search term of the task t_{tid_x} and the desired type of terminological relation as the task goal of t_{tid_x},
   - $Prod^s_{a_k}$ the set of self-productions for own search tasks restricted on local information model LIM,
   - $Prod_{a_k,C}$ the set of a_k's productions for search tasks received from agents of coalition C,
   - $Prod_{a_k,C}[S]$ the set of parts of productions $p \in Prod_{a_k,C}$ which are available for all members of coalition S,
   - $Prod^{(C)}_{a_k,C}[S]$ the set of parts of productions $p \in Prod_{a_k,C}[S]$ whose elements relate data exclusively within the local information domains of agents in C[11],
   - $Prod_{a_i} := Prod^s_{a_i} \cup Prod_{a_i,\mathcal{A}\setminus\{a_i\}}$, and $Prod^{ca}_{a_k,C} := Prod^s_{a_k}[C] \cup Prod^{(C)}_{a_k,C\setminus\{a_k\}}[C]$.
2. *Agent Utility Function:* $U^{type}_{agent_{id}} : Prod_{a_i} \rightarrow \mathbb{R}^+$.
3. *Coalition Value* $v(C)$: $\mathcal{P}(\mathcal{A}) \mapsto \mathbb{R}^+$, $v(C) := \sum_{a_k \in C,\ p \in Prod^{ca}_{a_k,C}} U_k(p)$.
4. **The *contributable amount* of the coalition value for n-agent coalition entity C to another coalition entity S is defined as $v(C)[S] := \sum_{a_k \in C,\ p \in Prod^{ca}_{a_k,C}[S]} U_k(p)$.**

□

Every calculated polynomial Kernel-stable solution is defined to be *pK-stable*.

Different types of agent utility functions lead to respectively different coalition types. In particular, we consider formation of coalitions relying on quantitative coalition types. One coalition type bases on the amount of satisfaction of all own tasks (C_{ots}). Its utility function is defined as the sum of all payments the agent a_i will get from its local customers for satisfying their search tasks $t' \in OTS^i$ by a particular production $p(t) \in Prod_{a_i}$:

$$U^{ots}_i(p(t)) := \begin{cases} \sum\limits_{\{t' \in OTS^i : t \xrightarrow{tr} t',\, tr \in p(t)\}} w(t'), t \in RTS^i \\ \sum\limits_{\{t' \in OTS^i : t' \xrightarrow{tr} t,\, tr \in p(t)\}} w(t)\ , t \in OTS^i \end{cases} \tag{1}$$

In general, non-super-additive environments are expected in cases where e.g. the increase in the size of the coalition is costly. There, it may be less beneficial to form a joint coalition than to let coalitions remain disjoint. Such costs may arise from internal communication and coordination requirements. Therefore, large coalitions are costly, and the expected utility from satisfying information search tasks may be less than the overhead of internal coalition costs.

For the considered coalition value function v (cf. Def.4.1) it can be proven that there may exist cooperative games $(\mathcal{A}_{FCSI}, v)$ which are non-super-additive. This is mainly caused by the need to restrict the coalition value calculation to the productions which are available for all members. Therefore, the coalition environment for the FCSI, including Def.4.1 and the (IAC) assumption, is of general type.

4.1 A Kernel-Oriented Coalition Algorithm (KCA)

The following coalition algorithm KCA advances the information agents from one coalition structure to another in order to increase the agents' payoff, thus increasing rationally-motivated cooperation.

In each step, at least one coalition will make an attempt to improve the payoffs of its members. This is done by calculating and sending respective, pK-stable payment configurations as proposals to other coalitions which are actually most promising. After having received such proposals for bilateral coalition formation, each coalition rationally decides upon the preferred one. This accepted payment configuration is then broadcasted to all other coalitions. Among all such proposals which are accepted by a coalition, the one with the fastest creation time is selected as the committed coalition configuration for this round. Thereby, some agents are temporarily enforced to accept the corresponding payoff distribution which is valid for all

[9]Some coalition algorithms which use the so-called bilateral Shapley-Value(Ketchpel 1993) as a fair and individually rational division method were already provided in (Klusch 1994; Klusch & Shehory 1996).

[10]Note that an exportable local schema data is not necessarily available for all other agents. Thus, every information-search task with attached local data must be selectively broadcasted in the FCSI.

[11]Following the (IAC) assumption, coalition external data is not available for C, even not paid for. Otherwise it would have been rational for C to be respectively extended.

agents. Since this affects in particular agents which were not involved in the bilateral coalition agreement of the selected proposal, they may be dissatisfied with their payoff[12] and will react with a proposal to the newly formed agent community in the next round. Negotiation continues until all proposals of all coalition entities of the currently valid coalition structure were rejected or a grand coalition has been already formed or a predefined time-period was exceeded.

Algorithm 4.1: *KCA*

Let be $\mathcal{A}_{FCSI}$ a set of FCSI information agents, and

- $csize_{min}$ and $csize_{max}$ given lower and upper bound for coalition size,
- $RPropSet_i$ the set of configuration proposals PC_{ti} the coalition C_i received from C_t in the actual negotiation round;
- $PC^{(rnum)}$ the committed coalition configuration for negotiation round $rnum$;
- $PC_{ij}^{(rnum)}$ a configuration proposal of coalition C_i for coalition C_j in round $rnum$;
- $rtime$ given time-period unit for receiving messages. $RPTout$, $RTout$, $CNTout$ boolean flags to indicate if time for receiving proposals, a round or the negotiation at all is exceeded, respectively;
- RecMess($\{m_1,..,m_s\}$,$C^{(rnum)}$) checks, if a message of type in $\{m_1,...m_s\}$ was received from other coalitions in actual coalition structure $C^{(rnum)}$;
- RecMessQ(m,$MQueue_i$) checks if some message m is in FIFO-Queue $MQueue_i$;
- $PList_i$ is list of coalition preferences in monotone decreasing order obtained by Rank;
- DecideRepresentative(C_i,$CRList$) selects the agent in coalition C_i with most computational ressources wrt. commonly known agents ressources list $CRList$;
- $CRList$ ordered list of all FCSI agents in $\mathcal{A}_{FCSI}$ concerning their computational strength which will be available during negotiation;
- SelectFirstCalc(M) selects the first created message in set M, i.e. the message with the fastest creation time stamp.

```
each $a_m \in \mathcal{A}_{FCSI}$ performs:
begin $rnum$:=0;
$PC^{(0)}$:=$(\{\{a_1\},...,\{a_n\}\}, (v(\{a_1\}),...,v(\{a_n\})))$;
repeat
   $rnum$:=$rnum$+1; $PC^{(rnum)}$:= $PC^{(rnum-1)}$;
   $RPropSet_i$:=$\emptyset$; $SPropSet_i$:=$\emptyset$; $PAccept$:=FALSE;
   [$a_m \in C_i$: DecideRepresentative($C_i$,$CRList$);
   $a_m \in C_i$ is representative of $C_i$: begin perform]

   $PList_i$:= Rank($C_i$, $C^{(rnum-1)} \setminus C_i$, $csize_{min}$, $csize_{max}$);
   for each $C_j \in PList_i$ do
   begin
      GetKVal($C_i$,$C_j$);
      $PC_{ij}^{(rnum)}$:= CalculatePC($C_i \cup C_j$,$PC^{(rnum-1)}$);
      if EvalPC($C_i$,$C_j$,$PC_{ij}^{(rnum)}$) then
      begin Send($PC_{ij}^{(rnum)}$,$C_j$);
       $SPropSet_i$:=$SPropSet_i \cup \{PC_{ij}^{(rnum)}\}$;
      end else Send($NoCProp_i$,$C_j$);
   end;
   repeat Wait($rtime$);
   $RPropSet_i$:= $RPropSet_i \cup$ Mess($PC_{ki}^{(rnum)}$,$MQueue_i$);
   until RecMess($\{PC, NoCProp\}$,$C^{rnum}$) or $RPTout$;
   $PListEmpty$:=FALSE;

   repeat
      $C$:= Top($PList_i$); delete($C$,$PList_i$);
      if $C$= nil then begin
         Broadcast($NoPAccept$);
         $PrefListEmpty$:=TRUE; end;
      if $C = C_k$ and $PC_{ki}^{(rnum)} \in RPropSet_i$
      then if Decide($PC_{ki}^{(rnum)}$,$SPropSet_i$,$RPropSet_i$)
         then begin
           $PAccept$:=TRUE; $PAcc.Coal$:=$C_k$;
           $PAcc.Config$:=$PC_{ki}^{(rnum)}$; $PAcc.Sender$:=$C_i$;
         end;
      $RecPAccept$:=RecMessQ($PAcc$,$MQueue_i$);
      if not $RecPAccept$ and $PAccept$ then
       Broadcast($PAcc$);
   until $PrefListEmpty$ or $PAccept$ or $RecPAccept$;
   repeat Wait($rtime$);
     $PAccSet$:= $PAccSet \cup$ Mess($PAcc$,$MQueue_i$);
   until RecMess($\{PAcc, NoPAccept\}$,$C^{(rnum)}$) or $RTout$;


   $Proposal$:= SelectFirstCalc($RecPAccSet$);
   if $Proposal.Coal = C_i$ and $Proposal.Sender$=$C_k$
      then $PC^{(rnum)}$:= $PC_{ik}^{(rnum)}$;
   else if $Proposal.Sender = C_i$ and $Proposal.Coal$=$C_k$
      then $PC^{(rnum)}$:= $PC_{ki}^{(rnum)}$;
   else if $Proposal.Sender \neq C_i$ and $Proposal.Coal \neq C_i$
      then $PC^{(rnum)}$:= $Proposal.Config$;
   [end perform]
until RecMess($\{NoCProp\}$,$C^{(rnum)}$) or $CNTout$;
FormCoal($PC^{(rnum)}$);
end;
□
```

Due to space limitations we give a brief, informal description of the used functions.

- **function** Rank(C:Coal;P:Coal-set; $cmin$, $cmax$: integer): PreferenceList; ranks the other coalitions $C_k \in P$, $cmin \leq |C_k| \leq cmax - |C|$, wrt. their contributable amounts $v(C_k \cup C)[C]$[13]. Local calculation of these values bases on productions for

[12]Note that Kernel-stability is restricted to agents within the same coalition. Thus, in a pK-stable payoff distribution in a bilateral coalition proposal the payoffs for all other agents remain unchanged.

[13]Note that for the characteristic function in Def.4.1 $v(C \cup S)[S] = v(C \cup S)$ holds. In an environment where all prices $w(t)$ for task satisfactions are the same for all tasks t and the agent utility function in eq.(1) is chosen, there is an ordinal relation between monetary and informative utility.

tasks and information that were received from the C_k agents.

- **function** GetKVal(C_i, C_j: Coal): void; gets non-local coalition values which are relevant for the calculation of surplus values s_{kl} for agents a_k in C_i wrt. agents a_l in coalition C_j (cf. Def.3.4) from the representative of C_j.
- **function** CalculatePC(C: Coal; PC: PayConfig): PayConfig; calculates a new, pK-stable payment configuration proposal using the modified transfer scheme (cf. Def. 3.5) for coalition C using the payoffs of the (old) configuration PC.
- **function** EvalPC(C_i, C_j: Coal; PC_{ij}: PayConfig): boolean; checks if the bilateral proposal is better for both than the currently valid payment configuration, i.e., **EvalPC**:=TRUE iff

$$\sum_{a_x \in C_i \cup C_j} u_x^{(PC_{ij})} \geq \sum_{a_x \in C_i \cup C_j} u_x^{(PC^{(rnum-1)})}$$

- **function** Decide($PC_{ki}^{(rnum)}$: PayConfig; $SPropSet$, $RPropSet$: PropSet): boolean;
  checks if the received proposal $PC_{ki}^{(rnum)}$ allows every agent $a_l \in C_i \cup C_k$ to gain more benefit u_l than the own one $PC_{ik}^{(rnum)}$ calculated for C_k, and if it is better for all agents in C_i than all proposals C_i received or has sent ($PC_{si}^{(rnum)} \in RPropSet$, $PC_{is}^{(rnum)} \in SPropSet$) to other coalitions $C_s, s \neq k$.
- **function** FormCoal(PC: PayConfig); forms the coalitions wrt the committed payment configuration PC and handles the necessary financial payments among the agents as well as the information exchange within the coalitions.
- Functions like Send or Broadcast are communication functions.

4.2 Complexity of the KCA

The number of coalitions to be considered during the KCA is

$$n_{coalitions} = \sum_{i=csize_{min}}^{csize_{max}} \binom{n}{i} = \sum_{i=csize_{min}}^{csize_{max}} \frac{n!}{i!(n-i)!}$$

which is, in the worst case, of order $O(n^{csize_{max}})$. This may be multiplied by the number of coalition types, which is a constant and therfore does not affect the order of the complexity. When the agents compute the set of coalitions, the coalitional values and the coalition structures, the order of complexity is equal to the order of the number of the coalitions.

During the KCA process, an agent is required to compute $O(n_{coalitions})$ coalition values and design $O(n)$ coalition structures. In each iteration round, whenever a coalition structure is considered, a pK-stable PC shall be calculated. This is done by CalculatePC, where the KCA employs a truncated transfer scheme for calculating such PCs. Each transfer scheme iteration step consists of the following parts:

- $n_{coalitions}$ excesses are calculated, each requires $O(n)$ calculations. Among the excesses, the surpluses are searched for. The composite complexity of the excess and surplus calculations is $O(n \times n_{coalitions})$.
- The maximum surplus is located by $O(n_{coalitions})$ operations. Other calculations are of a lower complexity.

Summing up, the complexity of one transfer scheme iteration step is $O(n \times n_{coalitions})$. The resulting payoff vector of the transfer scheme will converge to an element of the polynomial-Kernel (with a relative error not greater than ε) within $n \log_2(e_0/\varepsilon)$ iteration steps (Stearns 1968), where e_0 is the initial PC error and ε is the predefined allowed error[14].

An agent will perform the transfer scheme $O(n)$ times per KCA iteration round from which there are in turn $O(n)$. Other procedures performed during the KCA are of lower complexity concerning the functions studied above. Therefore, the complexity of the KCA per agent is $O(n^3 n_{coalitions})$ computations. In addition to the computational operations, some communication operations are required. In general, each agent is required to perform $O(n)$ communication operation per KCA iteration round, and this is done $O(n)$ times. Therefore, the communication complexity per agent is $O(n^2)$. The most calculation-consuming step concerns the knowledge-based production phase which will be performed prior to starting the KCA.

4.3 Properties of the KCA

The KCA strongly relies on local computations of the information agents. Other coalition algorithms for rational information agents which can also be used for general environments are in (Klusch 1994; Klusch & Shehory 1996). There exist algorithms which could possibly be adapted as well, however most of them assume (explicitly or implicitly) complete information for each agent. This contradicts the availability assumption (IAC) for information agents in the more realistic coalition environment of the FCSI. Other algorithms also provide perfect stability wrt. the considered coalition theory, but with exponential computation efforts (e.g. the Bargaining Set). For practical reasons, information search by many agents requires a reduction to a polynomial complexity. The KCA satisfies this requirement as proved in the previous section. The KCA is an anytime algorithm, i.e., it provides the agents with an individually rational, pK-stable payment configuration and associated coalition structure, at any time it terminates. But it does not necessarily lead to a Pareto-optimal payment configuration.

In each negotiation round a pK-stable configuration will be accepted by the agent community. Moreover, the algorithm provides each agent with a regulation

[14]Note that the logarithm does not depend on n, and therefore does not affect the order of complexity, although it may contribute a large factor.

method to choose among proposed configurations due to its personal rationality.
Changes of the production situation in the FCSI causes a new cooperative game (A_{FCSI}, v). In such cases there must be a renewal of the coalition negotiation by restarting the KCA using the last calculated payment configuration as an initial one.

5 Conclusion

We introduced a new coalition algorithm which enables utilitarian coalition formation for rationally interacting information agents in general environments. The utility distribution is guaranteed to be stable concerning the polynomial Kernel equilibrium. Such information agents solve the problem of utilitarian information-gathering among several autonomous databases by rational cooperation in the Internet. The algorithm is given in terms of specific functions and procedures to be performed by each information agent locally. It is a polynomial anytime algorithm and respects the specific requirements the FCSI agents must deal with. Toward an implementation of FCSI information agents a development toolkit IDEAS (Klusch 1996b) for Multi-Agent Systems as well as the knowledge-based component (Gao 1996) has been implemented in C and Tcl/Tk on HP-UX and SUN-Solaris workstations. For first simulation results of the basic negotiation schema of the KCA we refer to (Shehory & Kraus 1996).

References

Barbuceanu, M., and Fox, M. 1994. The information agent: an infrastructure for collaboration in the integrated enterprise. In *Proc. SIG Meeting on Cooperating Knowledge-Based Systems CKBS-94.*

Blanco, J. 1994. Building a federated relational database systems: an approach using a knowledge-based system. *Intern. Journal on Intelligent Cooperative Information Systems* 3(4).

Burg, J. 1993. A data-dictionary as a lexicon: An application of linguistics in information systems. In *Proc. 2nd Intern. Conf. on Information and Knowledge Management.*

Davis, M., and Maschler, M. 1965. The kernel of a cooperative game. *Naval research Logistics Quarterly* 12:223–259.

Decker, K., and Lesser, V. 1995. Macron: an architecture for multi-agent cooperative information gathering. In *Proc. CIKM-95 Workshop Intelligent Information Agents.*

Fischer, K., and Müller, J. P. 1996. A decision-theoretic model for cooperative transportation scheduling. In de Velde, W. V., and Perram, J. W., eds., *Lecture Notes in Artificial Intelligence, Vol. 1038.* Berlin:Springer-Verlag.

Gao, H. 1996. *Implementierung eines terminologischen Wissensrepräsentations- und Inferenzsystems TEWIS.* CSD University of Kiel: MSc Thesis (in german).

Jacobs, N., and Shea, R. 1995. Carnot and infosleuth - database technology and the www. In *ACM SIGMOD Intern. Conf. on Management of Data.*

Jacobs, N., and Shea, R. 1996. The role of java in infosleuth: Agent-based exploitation of heterogeneous information ressources. In *Proc. of Intranet-96 Java Developers Conference.*

Kahan, J. P., and Rapoport, A. 1984. *Theories of coalition formation.* Hillsdale NJ,: Lawrence Erlbaum Associates.

Ketchpel, S. P. 1993. Coalition formation among autonomous agents. In *Proc. of MAAMAW-93.*

Klusch, M., and Shehory, O. 1996. Coalition formation among rational information agents. In de Velde, W. V., and Perram, J. W., eds., *Lecture Notes in Artificial Intelligence, Vol. 1038.* Berlin:Springer-Verlag. 204–217.

Klusch, M. 1994. Using a cooperative agent system for a context-based recognition of interdatabase dependencies. In *Proc. CIKM-94 Workshop on Intelligent Information Agents.*

Klusch, M. 1996a. *Rational kooperative Erkennung von Interdatenbankabhängigkeiten.* CSD University of Kiel, Germany: Dissertation (in german, in preparation).

Klusch, M. 1996b. Utilitarian coalition formation between information agents for a cooperative discovery of interdatabase dependencies. In Kirn, S., and O'Hare, G., eds., *Cooperative Knowledge Processing.* London:Springer-Verlag. Chapter 13.

Moukas, A. 1996. Amalthaea: Information discovery and filtering using a multiagent evolving ecosystem. In *Proc. Intern. Conf. on Practical Applications of Intelligent Agents and Multi-Agent Technology PAAM-96.*

Sandholm, T. W., and Lesser, V. R. 1995. Coalition formation among bounded rational agents. In *Proc. of IJCAI-95*, 662–669.

Shehory, O., and Kraus, S. 1993. Coalition formation among autonomous agents: Strategies and complexity. In Castelfranchi, C., and Müller, J. P., eds., *Lecture Notes in Artificial Intelligence, Vol. 957.* Berlin:Springer-Verlag. 57–72.

Shehory, O., and Kraus, S. 1995. Task allocation via coalition formation among autonomous agents. In *Proc. of IJCAI-95*, 655–661.

Shehory, O., and Kraus, S. 1996. A kernel-oriented model for coalition-formation in general environments: Implementation and results. In *Proc. of AAAI-96.*

Sheth, A.; Karabatis, G.; and Rusinkiewicz, M. 1991. Specifying interdatabase dependencies in a multidatabase environment. *IEEE Computer.*

Stearns, R. E. 1968. Convergent transfer schemes for n-person games. *Transactions of the American Mathematical Society* 134:449–459.

Weigand, H. 1995. Integrated semantics for information and communication systems. In *Proc. IFIP Conference on Data Semantics.*

Wooldridge, M., and Jennings, N. 1995. Intelligent agents: theory and practice. *Knowledge Engineering Review*, 10(2):115–152.

Zlotkin, G., and Rosenschein, J. S. 1994. Coalition, cryptography, and stability: Mechanisms for coalition formation in task oriented domains. In *Proc. of AAAI94*, 432–437.