On Safe Kernel Stable Coalition Forming among Agents

Bastian Blankenburg, Matthias Klusch
German Research Center for Artifical Intelligence,
Deduction and Multiagent Systems,
Stuhlsatzenhausweg 3, D-66123 Saarbrücken, Germany
e-mail: {blankenb, klusch}@dfki.de

Abstract

We investigate and discuss safety and privacy preserving properties of a game-theortic based coalition algorithm KCA for forming kernel stable coalitions among information agents in face of imperfect information on actual coalition values, and changing agent society. In addition, we analyze the chances of deceiving information agents to succeed in coalition negotiations using the KCA protocol. We show that a certain type of fraud which leads to an increase of individual profit can neither be prevented nor detected, but this comes at the expense of exponentially high computation costs for the deceiving agent.

1. Introduction

Game-theoretic coalition algorithms can be used by intelligent agents as coordination means in a variety of applications in different environments. Applications in the health care and m-commerce domain are required to preserve the privacy of user information to a large extent. In this respect, one interesting question concerns the relation between the safety poperties of a given coalition negotiation protocol, and the privacy of information required to specify the underlying coalition game to be solved by the agents according to the protocol. In particular, what is the minimum amount of information required for a given coalition algorithm to output stable solutions? Will it be individually beneficial to deceive negotiation partners on local information that is used to determine the value of joint coalitions? What kinds of information can be hidden by an agent from selected agents at what costs in terms of its bargaining position in the coalition negotiation?

As to our knowledge, there is no work on this topic available yet. Hence, in this paper, we provide some first thoughts on, and preliminary answers to these questions, taking a special coalition algorithm KCA [4] for kernel stable coalition forming as an example.

In section 2, we introduce the reader to the basics of co-operative game theory, with focus on Kernel stable coalition forming, to an extent that is necessary to understand the results presented. Readers who are familiar with the field can skip this section. In section 3, we analyze and discuss some safety properties of the KCA protocol for negotiation settings with imperfect information on coalition values, and changes in the set of negotiating agents. Finally, the chances of deceiving agents to succeed in coalition negotiations according to the KCA protocol are discussed in section 4.

2. Kernel stable coalition forming

In this section, we briefly introduce the reader to the basic concepts of coalition games, kernel stable coalitions, and a specific negotiation protocol KCA [4] that can be used by agents to form such coalitions. For a more comprehensive introduction to co-operative game theory we refer the reader to, for example, [2, 5].

2.1. Basics

A co-operative or *coalition* game (A, v) is defined by a set A of agents wherein each subset of A is called a *coalition*, and a real-valued characteristic or *coalition value* function $v : A \rightarrow \mathbb{R}$ that assigns each coalition $C \subseteq A$ its maximum monetary gain. Any set of coalition values for all possible coalitions defines a coalition game. The *self-value* $v(\{a_k\}) \equiv v(a_k)$ of an agent a_k denotes the maximum profit it may gain without any cooperation with other agents. It is assumed that each value $v(C)$ does not depend on the actions of agents outside C, any coalition C forms by a binding agreement on the distribution of $v(C)$ among its members, and no side-payments are allowed from C to any agents outside C within the given game.

The sum of both self-value and marginal contribution to a coalition C is called the *local value* or *worth* $lworth_a(C)$ of agent a for C. An individual *agent production utility* function U_k determines the worth of task-related productions of agent a_k. Agents coalesce to increase their individual profits that may result from jointly accomplishing their tasks. The value $lworth_a(C)$ of agent a for coalition C is the total revenue a may obtain for accomplishing its tasks in C on behalf of its user or other agents in C. Each coalition value is the sum of the local values of its members ($v(C) = \sum_{a \in C} lworth_a(C)$).

Stable Solutions of Coalition Games. The solution of a co-operative game with side payments is a *coalition configuration* (S, u) which consists of a partition S of A, the *coalition structure*, and a n-dimensional, real-valued *payoff distribution* vector which components are computed by a real-valued *payoff or utility function* u. The payoff distribution assigns each agent in A its utility $u(a)$ out of the value $v(C)$ of coalition C it is member of in a given coalition structure S.

In *individually rational* payoff distributions each agent gets at least its self-value $u(a) \geq v(\{a\})$. For *group rational* distributions, it holds that the group of all agents maximises its joint payoff. In *Pareto-optimal* payoff distributions no agent is better off in any other valid payoff distribution for the given game and coalition structure.

A configuration (S, u) is called *stable* if no agent has an incentive to leave its coalition in S due to its assigned payoff $u(a)$. Different characteristics and criterions of stability define different solution spaces for a co-operative game.

In general, non-super-additive games at least one pair of potential coalitions is not better off by merging into one. The meaning of stability of coalitions depends on the considered discipline and application domain. Many if not most of the coalition formation algorithms today rely on chosen game-theoretic concepts for stable pay-off division within coalitions according to, for example, the Shapley-value, the Core, the Bargaining Set, or the Kernel [2]. In this paper, we focus on the latter concept of coalition stability.

Kernel Stable Configurations. The *kernel* of a co-operative game (A, v) with respect to a given coalition structure S is a set of *K-stable configurations* (S, u) wherein each coalition in S is in equilibrium. Each pair of agents a_k, a_l in C is in equilibrium, if they cannot outweigh each other in (S, u) by having the option to get a better payoff in coalition(s) *not* in S excluding the opponent agent. The *surplus* of agent $a_k \in A$ with respect to the opponent a_l in a given configuration (S, u) is $s_{kl} = max_{R \notin S, a_k \in R, a_l \notin R}\{e(R, u)\}$, where $e(R, u) = v(R) - u(R)$ denotes the *excess* of alternative coalitions R. Agent a_k outweighs a_l, if $s_{kl} > s_{lk}$ and $u(a_l) > v(a_l)$. Any pair of agents a_k, a_l is in equilibrium with respect to (S, u), if one of the following constraints is satisfied: $(s_{kl} = s_{lk})$, or $(s_{kl} = s_{lk}$ and $u_l = v(a_l)$, or $s_{kl} < s_{lk}$ and $u_k = v(a_k)$.

To compute a K-stable payoff distribution, agents transfer side payments among each other; the demand of agent a_k from a_l is defined as $d_{kl} = min\{\frac{(s_{kl} - s_{lk}}{2}, u(a_l) - v(a_l)\} \geq \alpha$, and zero else, as an upper limit of any side-payment α to be added (subtracted) from the payoff $u(a_k)$ ($u(a_l)$. The transfer scheme converges against a K-stable (S, u) after $O(nlog(re/\epsilon))$ iterations with $O(n2^n)$ steps each, where $re(u)$ denotes the relative error.

The kernel of a game is exponentially hard to compute unless, for example, the size of the coalition is limited by a constant. The kernel appears to be attractive, since it is unique for any 3-agent game (A, v), assigns symmetric agents of some coalition in a given coalition structure for (A, v) equal payoff, and is locally Pareto-optimal in the set K. Polynomial coalition algorithms for polynomial K-stable coalition configurations have been developed for co-operative information agents with perfect [4] or imperfect knowledge [1].

2.2. KCA coalition algorithm

Any set of rational agents can negotiate kernel stable solutions (S, u) of co-operative games (A, v) by using the so called KCA coalition algorithm [4] which proceeds as follows.

Each agent a performs

1. *Communication*
   (a) Set $(\{a_1\}, \ldots, \{a_n\}, (v(\{a_1\}), \ldots, v(\{a_n\}))$ to be the current configuration.
   (b) Set each agent to be the coalition leader of its coalition.
   (c) Generate a totally ordered list of all agents sorted by their overall computational power. *The sorting of this list is the same for all agents. It is not important here how this is exactly done.*
   (d) Send the set of tasks T_a to all other agents.
   (e) Receive the set of tasks of each other agent.
   (f) Evaluate the set of tasks accomplishable by a and send it to all other agents.
   (g) For each other agent, receive the set of accomplishable tasks.
   (h) For every coalition $C \subseteq A$, evaluate $lworth_a(C)$ and send it to all other agents.
   (i) Receive all local values from each other agent.
2. *Generating Proposals*
   (a) If a is not coalition leader of $C, a \in C$, go to 4e.
   (b) For each other coalition $C^* \in S, a \notin C^*$ compute a Kernel-stable configuration (S^*, u^*) with $C \cup C^* \in S^*$ and all other coalitions unchanged. If u^* strictly dominates u, send (S^*, u^*) as a configuration proposal to the leader of coalition C^*.
3. *Evaluating Proposals*

(a) Receive configuration proposals from the other coalition leaders.

(b) Evaluate the received proposals. Choose one proposal (S^+, u^+) that is most beneficial to accept, i.e. for which u^+ strictly dominates u and is not strictly dominated by u^* of any other received proposal (S^*, u^*).

(c) Inform all other coalition leaders about the accepted proposal.

4. *Deciding Upon Coalition Configuration*

(a) Receive all accepted proposals from the other coalition leaders.

(b) If no proposal was accepted, stop.

(c) Choose one proposal to become the new configuration. To do this, determine an order of preference of the proposals according to the following keys, priority in descending order:

i. Bilaterally accepted proposals are preferred to unilaterally accepted ones. An accepted proposal (S^*, u^*) of coalition C^* for coalition C^+ is bilaterally accepted iff C^+ accepted a proposal of C^*.

ii. If any two proposals (S^*, u^*) and (S^+, u^+), $\sum_{a^* \in A} u^*(a^*) > \sum_{a^* \in A} u^+(a^*)$ holds, (S^*, u^*) is preferred to (S^+, u^+).

iii. If any two proposals are equally preferrable according to the above properties, the one which was made by the agent with the greater computational power is preferred.

(d) Inform all other coalition members in C about the new configuration.

(e) If a is not coalition leader, receive the new configuration.

(f) If a is in the coalition, determine the new coalition leader as the agent in the new coalition with the greatest computational power.

(g) If a is coalition leader, do: if a is in the new coalition, inform all other coalition leaders about a being the new coalition leader. If a is not in the new coalition, receive the new coalition leader of the new coalition.

(h) If the grand coaltion was formed or a previously defined time for the coalition formation process is exceeded, stop.

(i) Go to 2.

It is assumed that the time for message exchange is limited, and inter-agent communication is correct. Please note that according to the KCA protocol, in each round at most one new coalition is formed (as a merger of two coalition of the previous configuration), and the computation of a kernel stable payoff distribution with respect to a proposed coalition may affect also the payoffs of those agents that are not involved in that proposal.

3. Safety properties of the KCA

3.1. Safe KCA with incomplete information

Unknown Tasks. Suppose agent a_1 does not receive the complete set of tasks from agent a_2. The local value $lworth_{a_2}(C)$ of a_2 for any coalition C in which both agents are involved depends on the extent a_1 is able to help a_2 in accomplishing its tasks. Since $v(C) = \sum_{a \in C} lworth_a(C)$ holds, and the task sets of agents are mutually exchanged before the coalition negotiation starts, any partial knowledge on tasks to accomplish only changes the original game, thus does not affect the correctness of the output of the KCA protocol.

Unknown Local and Coalition Values. In cases where local values are not known to some agent, it can estimate, or set the corresponding coalition by default. However, this leads to situations in which different agents try to solve different games at the same time with respectively different outcomes of the coalition negotiations according to the KCA protocol.

Example 3.1: *KCA negotiation with estimated coalition value*

Consider a non-superadditive 3-agent game for a set of agents $(\{a_1, a_2, a_3\}$ and coalition values $v(\{a_i\}) = 0, i \in \{1, 2, 3\}$, $v(\{a_1, a_2\}) = 3$, $v(\{a_1, a_3\}) = 1$, $v(\{a_2, a_3\}) = 3$, and $v(\{a_1, a_2, a_3\}) = 0$. The agents may solve this game using the KCA. For example, in the first round, the mutually exchanged kernel stable proposals are $Cfg_1 = (\{\{a_1, a_2\}, \{a_3\}\}, (0.5, 2.5, 0))$, $Cfg_2 = (\{\{a_1, a_3\}, \{a_2\}\}, (0.5, 0, 0.5))$ and $Cfg_3 = (\{\{a_1\}, \{a_2, a_3\}\}, (0, 2.5, 0.5))$. Suppose that no proposal is bilaterally but unilaterally accepted. As a result, both most beneficial coalitions $\{a_1, a_2\}$ or $\{a_2, a_3\}$ could form. In order to obtain a unique configuration, the proposal of the most powerful agent, here a_2, is selected, that is Cfg_1.

Suppose that a_1 did not receive $lworth_{a_3}(\{a_2, a_3\})$ and *under-estimates* the related coalition value $v^*(\{a_2, a_3\}) = 2$. a_1 proposes $(\{\{a_1, a_2\}, \{a_3\}\}, (1, 2, 0))$ to a_2 instead of Cfg_1. Unfortunately, this proposal is not as good as the one in the original game for a_2 such that it rather accepts the alternative proposal of a_3. Thus, the kernel stable configuration Cfg_3 is formed. Thus, in contrast to the above solution of the original game, a_1 stays alone with no profit from joint collaboration.

In case a_1 *over-estimates* the coalition value $v^*(\{a_2, a_3\}) = 3.5$, it proposes $Cfg_{1c} = (\{\{a_1, a_2\}, \{a_3\}\}, (0.25, 2.75, 0))$ instead of Cfg_1 to a_2, and $Cfg_{2c} = (\{\{a_1, a_3\}, \{a_2\}\}, (0.25, 0, 0.75))$ instead of Cfg_2 to a_3. As a result, a_2 and a_3 accept Cfg_{1c}, respectively, Cfg_{2c}, and a_1 accepts either Cfg_1 from a_2 or Cfg_2 from a_3. Given that a_1 and a_2 accepts Cfg_1, respectively, Cfg_{1c}, all agents uniquely decide on the configuration Cfg_{1c} proposed by a_1 based on given criteria for drawing a tie. However, this solution is not kernel stable with respect to the original game but the different game considered by a_1. Even worse, a_1 obtains less profit than it would be possible in most kernel stable solution of the original game.
$\circ$

If an agent decides to compute kernel stable configurations in case of complete information on coalition values only, the interesting question is how its proposals and behavior in negotiations are affected. In Ex. 3.1, a_1 would not make any proposal to any agent, thus non-kernel stable proposals do not appear in the negotiation.

In general, if a does not know the coalition value $v(C^*)$ of a certain coalition c^*, let s^*_{ik} denote the estimated surplus of a_i over a_k, that is the maximum of the set of excesses of coalitions $C \neq C^*$ with $a_i \in C, a_k \notin C$. This surplus estimation is computable by a without knowing $v(C^*)$. Hence, for each kernel stable configuration (S, u) which can exactly be computed by a by using s^*_{ik} instead ofs_{ik}, it holds that (a) $C^* \notin S$, and (b) $\forall C \in S, a_i, a_k \in C, a_i \in C^*, a_k \notin C^* : u(a_k) = v(a_k)$. The second constraint means that a does not need to compute any surplus with $v(C^*)$. Agent a can determine whether a configuration it computes with incomplete information is kernel stable by checking if both constraints are satisfied.

Example 3.2: *Computable configurations without certain local values*

Consider the coalition game in example 3.1. Suppose that a_1 does not know $v(\{a_2, a_3\})$ and decides to neither estimate, nor use any default value for it. As a result, a_1 is not able to compute a kernel stable configuration for the coalition structure $\{\{a_1, a_2\}, \{a_3\}\}$, because the excess of coalition $C^* = \{a_2, a_3\}$ is not computable by a_1. The reason is that if a_1 (wrongly) assumes $s_{21} = v(\{a_2\} - u(a_2)$, then $u = (2, 1, 0)$ is kernel stable with respect to the game it considers. However, the second constraint is not satisfied, since $u(a_1) > v(\{a_1\})$ holds. Thus a cannot be sure that the assumption is right and the resulting configuration is kernel stable. In fact, s_{21} is given by $v(\{a_2, a_3\} - u(a_2) - u(a_3) > v(\{a_2\} - u(a_2)$ since $v(\{a_2, a_3\}) > v(\{a_2\})$ and $u(a_3) = 0$. However, if the game includes $v(\{a_2\}) = 3$, then $u = (0, 3, 0)$ is determinable by a_1 to be Kernel-stable because it is sufficient to know that $s_{21} \geq v(\{a_2\} - u(a_2)$ holds. Because $u(a_1) = v(\{a_1\})$ in this case, it does not matter if $s_{21} > v(\{a_2\}) - u(a_2)$ or $s_{21} = v(\{a_2\} - u(a_2)$ is true and $v(\{a_2\}) - u(a_2) \geq s_{12}$ holds.
$\circ$

To summarize, using the KCA, coalition negotiations are safe with respect to unknown coalition values $v(C)$. Any under- or over-estimation of $v(C)$ by agent a yields a changed game (A, v) for which K-stable solutions can be computed by a following the KCA protocol but with possibly lower payoff $u(a)$ compared to solutions that are computed at the same time by other agents for the original game (A, v).

3.2. Safe KCA with changing agent sets

New agents trying to enter negotiations after these actually started can easily be avoided by verifying the sender of each message to be an expected one. But it may happen that an agent for whatever reason, intended or unintended, becomes unavailable before negotations end and/or the actions as determined by the coalitional contract are carried out. That is, it does not send any more messages required by the protocol. For example, its network connection may break down. The consequences are different for coalitions leaders and members.

Coalition leaders dropping out of negotiation. If the coalitions leader a of a coalition C becomes unavailable, consequently no proposals are made or accepted for C. Other members of C receive no more configuration updates and eventually time out, finishing the negotiation process. After this, the members of C have an outdated configuration if the other coalitions made further merges, which means that their payoffs might no longer be Kernel-stable. This is true even if a becomes available again to accomplish his part of the coalitional contract. If a stays unavailable, the remaining members of C are in fact forming the coalition $C \setminus \{a\}$. If there exist agents in C which rely on payments by a, individual rationality might not be guaranteed any more. Alternatively, a might become available again before negotiations are finished. This might, depending on the state of a then, lead to different configurations considered true by different agents.

Example 3.3: *Coalition leader leaving and re-entering negotiation*

Consider a 3-agent game with $A = \{a_1, a_2, a_3\}$ in

which usually a_1 and a_2 bilaterally accept proposals in the first round because $\{a_1, a_2\}$ is much more beneficial than any other coalition. But a_1 becomes unavailable right after the initial exchange of local values, and thus does not make a proposal for a_2, nor accepts a_2's proposal. Thus, $\{a_2, a_3\}$ is formed. Suppose a_2 becomes coalition leader. In the second round, a_1 becomes available again just when proposals are to be made. a_2 proposes the formation of the grand coalition while a_1, not having received a configuration update, proposes the formation of $\{a_1, a_2\}$. Because this is the most beneficial proposal, it is chosen as the new configuration, thus splitting up the coalition $\{a_2, a_3\}$ again (which is not allowed by the KCA). Suppose a_1 becomes coalition leader of $\{a_1, a_2\}$. a_1 thus sends a configuration update to a_3, but a_3 is waiting for a configuration update by its (now former) leader a_2. a_3 will eventually time out and finish negotiations still 'believing' it would coalesce with a_2.
$\circ$

Other coalition members dropping out of negotiation. The dropping out of coalition members does not influence the coalition formation process unless this happens during task execution which leads to the same problem as for coalition leaders. That is, the configuration may no longer be kernel stable, since the original game no longer models the situation appropriately. As in the case of unavailable coalition leaders, individual rationality of payoffs cannot be guaranteed for agents that rely on payments from unavailable coalition members.

To summarize, using the KCA, coalition negotiations are not safe in case of changing agent society: If agents are leaving the actual coalition game (A, v) a K-stable solution of the changed game (A, v) cannot be guaranteed without total restart of the KCA.

4. Secure and safe KCA

In this section, we show that, using the KCA, agents can safely negotiate K-stable coalitions and preserve individual data privacy (security) at the same time. In particular, any agent that is involved in the negotiations can completely hide its local data and information used to compute its self-value from other agents. Surprisingly, it can do so without even risking any loss of profit in the final coalition configuration.

4.1. Privacy preserving K-stable coalition negotiations

This property of local data privacy preservation in coalition negotiations using the KCA is an inherent property of the definition of kernel stability, which is stated in the following lemma.

Lemma 1. *Let* (A, v) *and* (A, v^*) *with*

$$\exists a^* \in A, r \in \mathbb{R} : v^*(C) := \begin{cases} v(C) + r & \textit{for } a^* \in C \\ v(C) & \textit{otherwise} \end{cases}$$

Then it holds that the configuration (S, u^*) *with* $u^*(a^*) = u(a^*) + r$ *and* $\forall a \in A, a \neq a^* : u^*(a) = u(a)$ *is K-stable with respect to the game* (A, v^*) *iff* (S, u) *is K-stable with respect the game* (A, v).

Proof. Let $s^*_{a^*,a^\circ}(C)$ the surplus of agent a^* over agent a°, $a^*, a^\circ \in C \in S$ in configuration (S, u^*). Then it holds

$$\begin{aligned} s^*_{a^*,a^\circ}(C) &= \max_{C^+ : a^* \in C^+, a^\circ \notin C^+} \{v^*(C^+) - \sum_{a \in C^+} u^*(a)\} \\ &= \max_{C^+ : a^* \in C^+, a^\circ \notin C^+} \{v(C^+) + r(- \sum_{a \in C^+} u(a) + r)\} \\ &= \max_{C^+ : a^* \in C^+, a^\circ \notin C^+} \{v(C^+) + r - \sum_{a \in C^+} u(a) - r\} \\ &= s_{a^*,a^\circ}(C) (\text{in configuration} (S, u).) \end{aligned}$$

$\square$

According to lemma 1, it holds for any K-stable solution (S, u) of coalition game (A, v) that the configuration (S, u^*) for the changed game (A, v^*) with $v^*(C) = v(C) - lworth(a, \{a\}), v^*(\{a\}) = 0, a \in C \in S$, and $u^*(a) = u(a) - lworth(a, \{a\})$ is K-stable. Since $u(a) - u^*(a)$ is constant for all pairs of K-stable proposals (S, u) and (S, u^*), the KCA negotiation protocol is safe against non-disclosure of self-values.

Since $v(C) = \sum_{a \in C} lworth_a(C)$ holds, any agent a may add $r \in \mathbb{R}$ to each of its local values, thereby constantly changing its actual contribution to each C by r, such that a's net result remains the same as in the original game with $v(C)$. In particular, if for each agent a_i the factor $r_i = -lworth_{a_i}(\{a_i\})$ is added to every coalition value $v(C)$ with $a_i \in C$, then the resulting game contains only zero-valued self-values and is equivalent to the original game (A, v).

To understand why that is the case, consider an agent which communicates its worth $lworth^*(a, C) = lworth(a, C) - lworth(a, \{a\})$ to every coalition C in the coalition structure S of configuration (S, u) for given game (A, v). That action changes the original coalition game to a new game (A, v^*) with $v^*(C) = v(C) - lworth(a, \{a\})$ and $v^*(\{a\}) = 0$. However, this change does not affect the value of the excess of any coalition C, since it holds that $e^*(C) := v^*(C) - u^*(C) = v(C) - u(C) = e(C)$. This, in turn, implies that the surplus values of agents in a solution (S, u) of (A, v), and (S, u^*) of (A, v^*) remain the same. By induction over all agents $a \in A$ and S, it can easily be shown that this finally yields equivalent games for any set of agents modulo their self-values.

As a result, self-values are not required to be communicated among the agents at all in order to solve any given coalition game with a K-stable configuration. That means that, using the KCA, any agent a_i can hide any set of local information from other agents in K-stable coalition negotiations, without loss of benefit for anyone, iff this local information is exclusively used to compute its self-value $v(\{a_i\})$. However, the extent to which local information can be hidden depends on the structure of the coalition game, as we will show by means of a simple application example in the following section.

4.2. Application to retailer coalition games

In general, rational agents in e-markets are envisioned to be capable of forming different kinds of coalitions for different purposes. For example, retailer agent coalitions are commonly formed to maximize individual benefits of joint sales to customers. On the other side, customer agent coalitions can be built to maximize individual benefits of joint purchases at retailer site, or to maximize individual brokerage/commission from the customer agents' users.

In the following, we consider a given set A of retailer agents that form coalitions to improve and share their joint benefits of selling requested items to customer agents. In terms of coalition game theory, the coalition value $v(C)$ of a retailer agent coalition $C \subseteq A$ is the maximum joint benefit of retailer agents in C for selling relevant items to their customer agents. Individual item utility $U(a, p)$ for retailer agent a of selling item p to its customers. The self-value $v(\{a\})$ of retailer agent a is the maximum gain of sales without any cooperation. Finally, the retailer agent coalition game (A, v) is the set of all coalition values.

Example 4.1: *Car Retailer Coalition Game*

Consider the following (superadditive) coalition game (A, v) of three car retailer agents $\{a_1, a_2, a_3\}$ and following coalition values defined by maximum car sales in each coalition: $v(\{a_1\}) = 2, v(\{a_2\}) = 1.5, v(\{a_3\}) = 1, v(\{a_1, a_2\}) = 6$, $v(\{a_1, a_3\}) = 8, v(\{a_2, a_3\}) = 7, v(\{a_1, a_2, a_3\}) = 15$.

Using the KCA, after first negotiation round, the agents reach the following K-stable solution of the game: $(S, u) = (\{\{a_1, a_2\}, \{a_3\}\}, (3.5, 2.5, 1))$ which balances the agents' surpluses $s(a_1, a_2) = v(\{a_1, a_3\}) - (u(a_1) + u(a_3)) = 8 - 3.5 - 1 = 3.5$, and $s(a_2, a_1) = 7 - 2.5 - 1 = 3.5$. After a second (final) negotiation round, the grand coalition is formed with the following K-stable payoff distribution among the retailer agents: $(S, u) = (\{\{a_1, a_2, a_3\}\}, (5, 4.25, 5.75))$ which balances the agents' surpluses $s(a_1, a_2) = e(\{a_1, a_3\}) = -2.75$, $s(a_2, a_1) = e(\{a_2\}) = -2.75$, $s(a_1, a_3) = e(\{a_1, a_2\})$ $= -3$, $s(a_3, a_1) = e(\{a_2, a_3\}) = -3, s(a_2, a_3) = e(\{a_2\})$ $= -2.75$, $s(a_3, a_2) = e(\{a_1, a_3\}) = -2.75$.
○

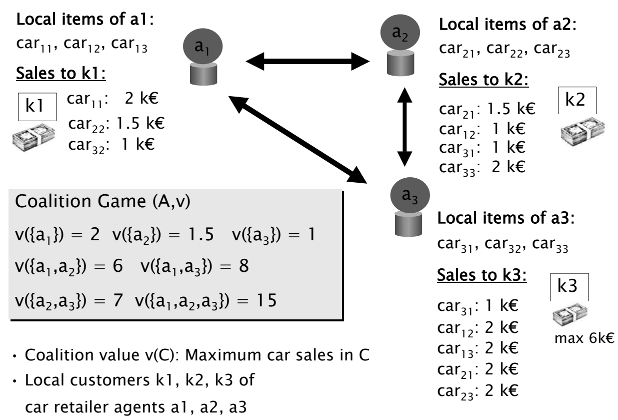

Figure 1. Example of a car retailer agent coalition game

Example 4.2: *Privacy Preservation in K-Stable Coalition Negotiations*

The K-stable solution $(\{\{a_1, a_2\}, \{a_3\}\}, (3.5, 2.5, 1))$ of game (A, v) after first negotiation round bases on the agents' surpluses $s(a_1, a_2) = v(\{a_1, a_3\}) - (u(a_1) + u(a_3)) = 8 - 3.5 - 1 = 3.5$, and $s(a_2, a_1) = 7 - 2.5 - 1 = 3.5$. Hiding of the self-value by each agent induces a new 3-agent game (A, v^*) with a new, unique and K-stable solution $(\{\{a_1, a_2\}, \{a_3\}\}, (1.5, 1, 0))$ which balances the newly formed but equally valued surpluses $s^*(a_1, a_2) = (v(\{a_1, a_3\}) - v(\{a_1\}) - v(\{a_3\})) - ((u(a_1) - v(\{a_1\})) + (u(a_3) - v(\{a_3\}))) = 5 - 1.5 = 3.5$ and $s^*(a_2, a_1) = 4.5 - 1 = 3.5$. This game is equivalent to the original one modulo the agents' self values.

Hiding of self-values implies that, for example, car retailer agent a_1 can prevent the other car retailer agents a_2, a_3 from knowing how many and what kind of own cars it can sell to its local customer for what price. In this example, a_1 can only sell one car of type car11 to its local customer for a price of 2k euros. It can protect this local information without loss of benefit in implied coalition negotiation by simply communicating a zero-valued self-value $v^*(\{a_1\}) = 0$. In this example, agents a_2 and a_3 by no means are able to infer the true self-value of a_1 from their local knowledge, since its car11 cannot be alternatively sold to them to maximize any of their joint coalition values. Both agents do not even know about the existence of car11.

That is not true in cases where local items can be both locally and remotely sold. In such cases the local sales value can be partially inferred by other agents. For example, consider the situation in which car retailer agent a_2 communicates a zero-valued self-value $v^*(\{a_2\})$ to protect its local information that it can sell car21 to its customer for 1.5k euros. Further, it does not tell a_3 that car21 exists. However, in order to determine the maximum joint coalition value with a_3 it has to communicate to a_3 that a_3's car33 can be sold to its local customer $k2$ for 2k euros, that is more than a_3 could obtain locally. As a consequence, a_2 communicates its worth $lworth^*(a2, \{a_2, a_3\}) = 0.5 (= 2 - 1.5)$ to a_3 such that both agents are now able to compute the maximum sales value of a joint coalition. But now a_3 can easily infer that the self-value $v(\{a_2\})$ of a_2 must at least be of value 1.5k $(= 2 - lworth^*(a_2, \{a_2, a_3\}))$ euros, which does not match with the zero value that has been communicated by a_2 to a_3 before.
$\circ$

5. Fraud in Kernel stable coalition forming

Serious threats to safe kernel stable coalition forming based on the KCA protocol are caused by, for example, fraudulent agents that intend to (a) unreasonably strengthen their bargaining position in the coalition negotiation, or (b) influence the building of coalitions for any other strategic reason. We show that this is in principle possible, but at high computational costs in practice.

Suppose that agent a wants to demand more payoff in a given coalition game (A, v) than originally possible. For this purpose, in the first negotiation round, it delays the communication of its worth $lworth_a(C)$ for any coalition $C \subseteq A$ to all other agents. Since $v(C) = \sum_{a \in C} lworth_a(C)$ holds, at this moment, it is the only agent that can compute all coalition values, hence the complete game (A, v). Since a can influence the coalition formation process according to the KCA protocol only at the beginning of the first negotiation round, it must decide on whether to perform a fraud to unfoundedly increase its profit, or not, right before it communicates its local values to the other agents. For this purpose, it locally simulates the KCA protocol to predict the final, kernel stable solution (S, u) for (A, v). In case of 3-agent coalition games, this predicted configuration is even unique.

How can agent a increase its predicted utility $u(a)$ without getting harmed by unmasking claims of other agents within a joint coalition in the predicted final structure S? One option is to choose one coalition $C' \notin S$, and deceive all agents on its individual worth $lworth_a(C')$ for C' such that the unfoundedly increase of its utility $u'(a)$ in the corresponding final configuration (S', u') is maximum. Such a fraud is in principle impossible to detect, since the worth of each agent depends on its individual production utility functions U_a which can be different for all agents.

With increasing $v(C')$, by definition of the KCA the probability that C' is actually formed also increases. How to determine the highest possible value $r \in \mathbb{R}, r \neq 0$ by which agent a can increase its original $lworth_a(C')$? Each change of worth in some round $t \in \mathbb{N}$, that is $lworth_a^{(t)}(C') = lworth_a^{(t-1)}(C') + r^{(t)}$ where $lworth_a^{(0)}(C') = lworth_a(C')$, implies a change of the respective coalition value $v^{(t)}(C')$. That, in turn, changes the current game $(A, v^{(t)})$ to $(A, v^{(t+1)})$ by replacing $v^{(t)}(C')$ with $v^{(t+1)}(C') = v^{(t)}(C') + r^{(t)}$. For each new game $(A, v^{(t)})$, $t = 1...m$, agent a simulates a full run of the KCA protocol until it finds, in round m, a configuration (S_m, u_m) such that $C \in S_m$. In this case, it holds that $u^{(m)}(a) \leq u^{(0)}(a) + r^{(m)}$ (cf. lemma 2). But that is adversive to a, since it would be enforced to bring the additional amount $r^{(m)}$ into the coalition once it is formed.

Lemma 2. *Let (A, v), (A, v^*) games, $r \in \mathbb{R}$, and $C^* \subseteq A$ with*

$$\forall C \subseteq A : v^*(C) := \begin{cases} v(C) + r & \text{for } C = C^* \\ v(C) & \text{otherwise} \end{cases}$$

Further, let $a^* \in C^*$, (S, u) a kernel stable configuration for (A, v), u^* a payoff distribution for which holds that $u^*(a) = u(a) + r$, $a \in A$, if $a = a^*$, and $u^*(a) = u(a)$ otherwise.

Then (S, u^*) is not kernel stable for (A, v^*), if there exists an agent $a^+ \in C^*, a^+ \neq a^*$ such that $s_{a^*,a^+} - s_{a^+,a^*} < r$ holds in (A, v).

Proof. Let $s^*_{a_1,a_2}$ denote the surplus of agent a_1 over agent a_2 in the game (A, v^*) and $Z := \{C | C \subseteq 2^A, a^* \in C, a^+ \notin C\}$. Further, let $e^*(C)$ the excess of coalition C in the game (A, v^*). Then

$$\begin{aligned} s^*_{a^*,a^+} &= \max_{C \in Z} \{e^*(C)\} \\ &= \max_{C \in Z} \{v(C) - \sum_{a' \in C, a' \neq a^*} u(a') - u^*(a^*)\} \\ &= \max_{C \in Z} \{v(C) - \sum_{a' \in C, a' \neq a^*} u(a') - (u(a^*) + r)\} \\ &= \max_{C \in Z} \{v(C) - \sum_{a' \in C} u(a')\} - r \\ &= s_{a^*,a^+} - r < s_{a^+,a^*} \end{aligned}$$

But since $u^*(a^*) > v^*(\{a^*\}) = v(\{a^*\})$, configuration (S, u^*) is not kernel stable for (A, v^*). $\square$

As a consequence, agent a selects $r^{(m-1)}$ as the maximum raise of its original worth for C' to the fraudulent value $lworth^*_a(C') = lworth_a(C') + r^{(m)}$. The value $r^{(m)}$ is the amount of additional profit a expects to gain by communicating its worth $lworth_a(C)$ for all possible coalitions C, including the incorrect value $lworth^*_a(C')$ for coalition C', to all other agents in the first round of the KCA based negotiations.

Example 5.1: *Fraudulent manipulation of coalition value*

Consider a game $(A = \{a_1, a_2, a_3\}, v)$ with $v(\{a_i\}) = 0, i \in \{1, 2, 3\}$, $v(\{a_1, a_2\}) = 5$, $v(\{a_1, a_3\}) = 1$, $v(\{a_2, a_3\}) = 2$, $v(\{a_1, a_2, a_3\}) = 0$. Applying the KCA to this game clearly results in the coalition structure $\{\{a_1, a_2\}, \{a_3\}\}$ and the kernel stable payoff distribution $u = (2, 3, 0)$. Now, the question is what happens if agent a_1 by intention deceives the other agents on the true coalition value $v(\{a_1, a_3\})$ by communicating an artificially increased value of its worth $lworth^*_{a_1}(\{a_1, a_3\}) = lworth_{a_1}(\{a_1, a_3\}) + r$, $r \in \mathbb{R}^+$?

Consider the respectively changed game (A, v^*) with $r = 3$. Then it holds that $v^*(\{a_1, a_3\}) = 4$, and all other coalition values v^* remain the same as in the original game v. Again, using the KCA, the coalition structure $\{\{a_1, a_2\}, \{a_3\}\}$ is formed, but now with different payoff distribution $u^* = (3.5, 1.5, 0)$. Thus, the fraud of a_1 has been succesfull, since it resulted in an increase ($3.5 - 2 = 1.5$) of its profit compared with payoff distribution for the original game.

Now consider the case in which agent a_1 decides to increase its worth even more ($r > 4$), say $r = 5$. For the resulting game (A, v^+) it holds that $v^+(\{a_1, a_3\}) = 6$ and all other coalition values remain the same. This time, using the KCA, a different coalition structure $\{\{a_1, a_3\}, \{a_2\}\}$ forms with payoff distribution $u^+ = (4.5, 0, 1.5)$. Again, it seems that agent a_1 increased its payoff compared with that in the original game by $4.5 - 2 = 2.5$. However, agent a_1 now has the severe problem to actually bring in the amount of $r = 5$ into the formed coalition $\{a_1, a_3\}$, since its real increase in profit is given by $2.5 - 5 = -2.5$, hence actually a loss! How shall a_1 explain that to its committed coalition partner a_3?
$\circ$

The computational efforts that are required to decide whether there exists an individually beneficial option to deceive other agents in KCA based coalition negotiation for a given game are exponentially expensive. Once the predicted configuration (S, u) is locally computed, agent a still has to check each of the $O(2^n)$ alternative coalitions $C' \notin S$ with polynomial computational complexity of simulated KCA based negotiations for each respective game. Besides, the possibly large delay in communicating its values to the other agents after having received theirs may already draw some initial suspicion of fraud on agent a.

6. Conclusion

We showed that the KCA coalition protocol exhibits both desirable and non-desirable properties with respect to safety and privacy preservation. The possibility for any agent to hide its self-values without risking any decrease of its payoff is clearly an advantageous result for kernel based coalition formation procedures such as the KCA. On the other hand, tasks and information may be hidden from other agents only at the cost of giving up some cooperation opportunities. However, if agents hide certain local values from other agents the course of negotiations according to the KCA can seriously be affected such that non-kernel stable solutions of the original game will form. Another threat to KCA based negotiations is caused by agents that drop out of running negotiations. As a consequence, the remaining agents and those that reenter the negotiation process after a while, may end up with different ideas about the final configuration. This makes it impossible for most agents to abide by their coalition contracts. Finally, it has been shown that individual fraud in kernel stable coalition forming using the KCA is in principle possible but appears impractical in terms of computational complexity.

References

[1] B. Blankenburg, M. Klusch, O. Shehory: Fuzzy Kernel-Stable Coalitions Between Rational Agents. Proc. 2nd Intl. Conference on Autonomous Agents and Multiagent Systems (AAMAS 2003), Melbourne, Australia, ACM Press, 2003.

[2] J. P. Kahan, A. Rapoport: Theories of coaltion formation. Lawrence Erlbaum Associates, Hillsdale, NJ, 1984.

[3] M. Klusch, A. Gerber: Dynamic Coalition Formation among Rational Agents. IEEE Intelligent Systems, 17(3), May/June 2002.

[4] M. Klusch, O. Shehory: A Polynomial Kernel-Oriented Coalition Algorithm for Rational Information Agents. Proc. 2nd Intl. Conference on Multi-Agent Systems (ICMAS '96), Kyoto (Japan), AAAI Press, 1996.

[5] G. Owen. Game Theory. Academic Press, NY, 1995.