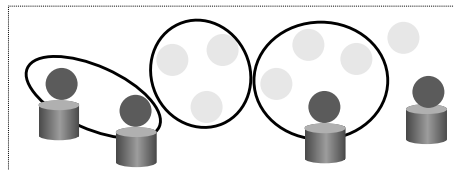


Issues of Coalition Forming: Dynamics, Security, and Trust

FR Informatik
In-Depth Course C6132
Dr. Matthias Klusch



Dynamic Coalition Forming



Dynamic Changes of the Environment?

Dynamic and efficient re-organising of task-oriented cooperation in stable agent teams or coalitions.

Simulation-based DCF-S coalition algorithms

A. Gerber, Ph.D. Thesis, University of the Saarland, 2004.



The DCF-S Scheme

Each agent concurrently simulates, selects, and negotiates one coalition for each of its sets of tasks $G_1 \dots G_n$

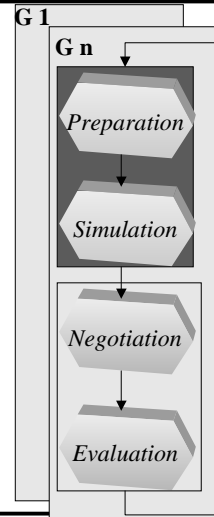
(1) Prepare simulation of potential coalitions

- select top-ranked matching candidates:
Task-related capabilities Vs Risk of cooperation
- approximate needed information

(2) Randomly simulate coalitions

- until coalition w/ maximum utility of accomplishing given tasks is found

If relevant changes of the environment occurred then continue simulation with updated information.



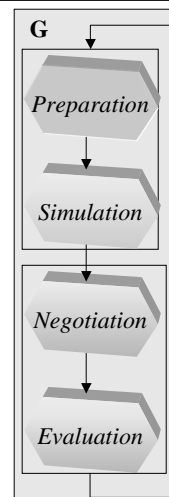
The DCF-S Scheme: Preparation

Update local knowledge base

- Check goal G to be accomplished for modifications
- Check and update information records for each agent
 - Goal-related capabilities of agent in coalition $C[G]$ (resources, costs, quality, performance etc.)
 - Risk of cooperation with agent in coalition $C[G]$ (history-based/subjective estimation)
- Approximate required information using local knowledge and/or via interaction with nearest world-utility agent.

Determine candidates for joint coalition

- capability based matching
- select top-ranked agents



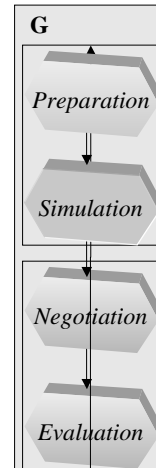
DCF-S Scheme: Simulation

Simulate randomly coalitions $C[G]$ of limited size each of which are able to accomplish goal G with an acceptable ratio between estimated risk of failure and profit.

- Estimate the risk of coalition failure
 - History of agent leaving coalitions (trust penalties)
 - History of deception & fraud; subjective trust
 - Effectiveness of agent (performance, capabilities)
- Add agent with minimum estimated risk of cooperation in, and maximum contribution to $C[G]$
- Remove agent from $C[G]$ with opposite properties.

Select coalition candidate $C[G]^*$ as new $C[G]$ **iff**

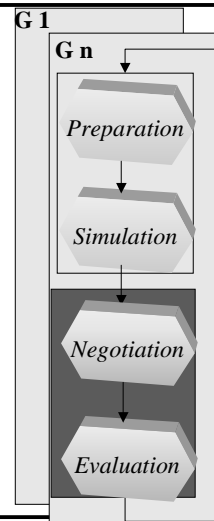
- increase of benefit: $v(C[G]^*) \gg v(\text{current } C[G])$
- acceptable trust penalty payments by current $C[G]$ for forming $C[G]^*$



The DCF-S Scheme (2)

- **Negotiate top-ranked simulated coalition**
 - Multi-attribute utility theory based bilateral negotiation including payoff distribution
- if relevant changes of the environment occurred then {stop negotiation; keep contracted agents in coalition}
- **Evaluate negotiation** (and update local knowledge)
- Goto (1)

A. Gerber/M. Klusch, IEEE Intelligent Systems, 17(3), 2003.



DCF-S Scheme: Negotiation

Negotiate simulated coalition $C[G]$

- bilaterally with each potential member of $C[G]$ based on
- payoff distribution according to given stability criterion

Contract agents whenever negotiation successful
= Iterative creation of $C[G]$.

Payments only after creation of $C[G]$.

If an event causes changes to the game $(A,v)[G]$ **then**

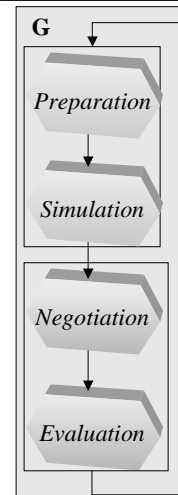
- stop negotiation immediately
- keep current $C[G]$ for partial restart of simulation

Evaluate recent negotiation process;
update local knowledge base; report evaluation to nearest WUA.

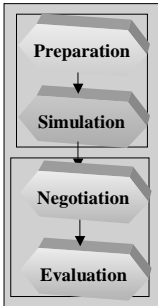


Discussion of the DCF-S Scheme

- The scheme is of high-risk and opportunistic:
There is no guarantee for optimal solutions
(Random simulation; High-frequent occurrence of events)
- Goal-oriented coalitions may overlap
- Effectiveness of coalition may improve
Via continuous simulation with updated
(approximated) information
- Supports trusted coalition forming
Via trust penalties for agents breaking coalitions
- Agents may discover their neighbourhood via WUAs



DCF-S Coalition Algorithms



DCF-S Scheme:

Each agent concurrently **simulates** and **negotiates** appropriate **coalition** to accomplish its **set of tasks**.

DCFS-Random+ : Random selection of relevant agents

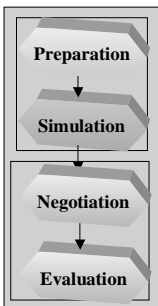
- No agent capability based matching, low computational complexity and selection accuracy p

DCFS-Matching+ : Probabilistic selection of relevant agents

- Probabilistic reasoning on agents' capabilities based on updated local knowledge
- Reduced negotiation costs, better selection accuracy



DCF-S Coalition Algorithms (2)



DCFS-SVM: Adaptive selection of relevant agents

- DCFS-Matching+ with **support vector machine (SVM)** for capability based clustering of agents
- Improved selection accuracy for simulation
- High computational costs $O(m^3)/O(mn)$

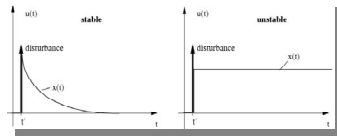
Capability-based matching of agents to tasks relies on appropriate attribute vector space representation of tasks, services, individual and system of agents.



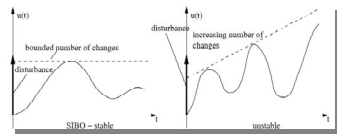
Stability of DCF-S Coalitions

not allocated tasks/time

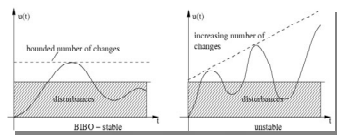
Dynamic capability-based re-organization of agent teams and corresponding re-allocation of tasks to agents within the multi-agent system.



• Asymptotically stable system



• Single-input/Bounded-output (SIBO) stable

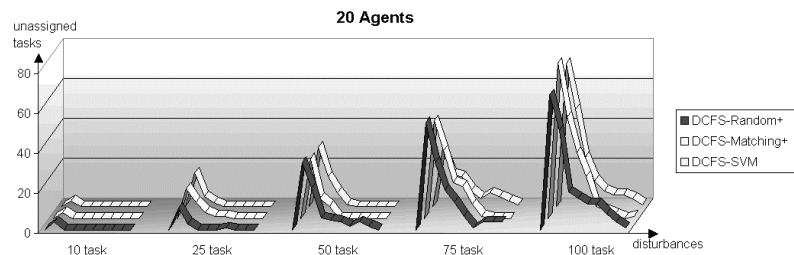


• Bounded-input/Bounded-output (BIBO) stable



Evaluation Results

Stability of dynamically re-organising multi-agent system is measured against the number of required re-allocations of tasks to relevant agents.

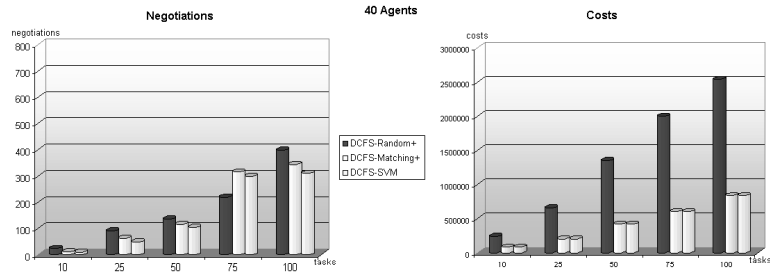


For given set of changed tasks per negotiation round

- each DCF-S algorithm allows the agent system to reach an asymptotically stable state in reasonable time.



Evaluation Results (2)



Knowledge-based adaptation to dynamic changes in the simulation phase

- improves capability-based selection of agents, hence
- reduces costs of both task execution, and negotiations until stable state.

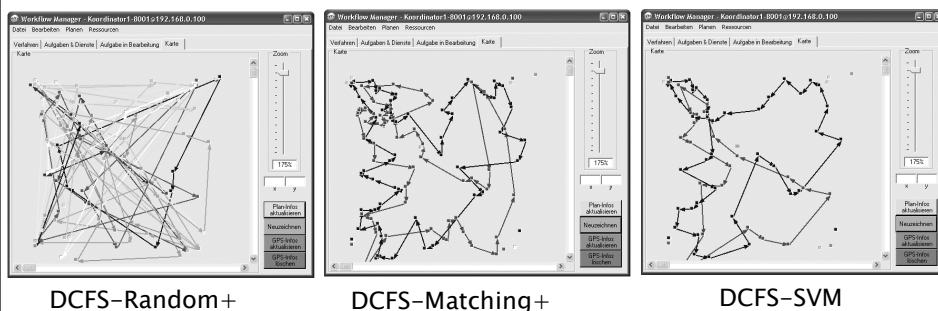
A. Gerber/M. Klusch, German AI Journal, 1 / 2004.



Evaluation Results (3)

Application: Mobile Planning Services for Cereal Harvesting

Planned routes of agricultural machines for given harvesting tasks



A. Gerber, Ph.D. Thesis, University of the Saarland, 2004.



Related Work

- **Soh & Tsatsoulis, 2002 (ST-DCF)**
 - time-constrained CF: hasty one-shot “trial-and-error” forming of coalitions (weighted utility on capability and reliability attributes of agents for task execution); tasks may change
 - valid coalitions cannot be improved; agents may not leave coalitions
- **Kraus, Shehory & Taase, 2003**
 - heuristic-based time-constrained CFA for joint task allocation
 - incomplete information on actual costs of (sub-) tasks but common knowledge on average costs per task (“mean value”)
 - ranking of coalition candidates = cost reduction + reliability (#refusals of subtask allocation proposals in the past)
 - equal share payoff distribution with vague stability concept

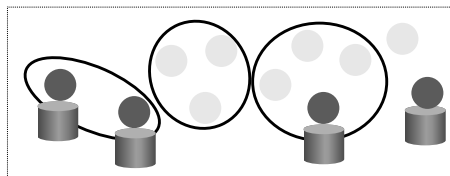


Application: Agent Coalitions In Virtual Malls

Rational agents in e-markets form

- Retailer Agent Coalitions
 - To maximize individual benefits of joint sales to their customers
- Customer Agent Coalitions
 - To maximize individual benefits of joint purchases at retailer site, or
 - To maximize individual brokerage/commission from their users
- Mixed Coalitions

Coalition Games (A, v)
Stable Solutions (S, u)



Example: Retailer Agent Coalitions

Set A of retailer agents form coalitions to improve and share their joint benefits of selling requested items to customer agents.

1. Value $v(C)$ of coalition C in A is the maximum joint benefit of retailer agents in C for selling relevant items to their customer agents.
2. Individual item utility $U(a,p)$ for retailer a of selling item p to its customers.
3. Self-value $v(\{a\})$ of retailer agent a is the maximum gain of sales w/o any cooperation.
4. Retailer agent coalition game (A,v) is the set of all coalition values.
5. Agents solve the game (A,v) : Negotiate coalition structure S w/ *individually rational* and *stable distribution* u of respective coalition values $v(C)$, $C \in S$, such that no agent has an incentive to leave its coalition in S due to its assigned payoff $u(a)$.




Example: Car Retailer Agent Coalition Game

Local items:

$car_{11}, car_{12}, car_{13}$

Item Sales:

$k1$
 $car_{11}: 2$
 $car_{22}: 1.5$
 $car_{32}: 1$



Local items:

$car_{21}, car_{22}, car_{23}$

$car_{21}: 1.5$
 $car_{12}: 1$
 $car_{31}: 1$
 $car_{33}: 2$



Local items:

$car_{31}, car_{32}, car_{33}$

$car_{31}: 1$
 $car_{12}: 2$
 $car_{13}: 2$
 $car_{21}: 2$
 $car_{23}: 2$



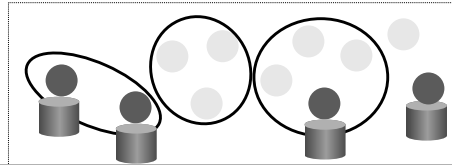
Coalition Game (A,v)

$v(\{a_1\}) = 2, v(\{a_2\}) = 1.5, v(\{a_3\}) = 1$
 $v(\{a_1, a_2\}) = 6, v(\{a_1, a_3\}) = 8, v(\{a_2, a_3\}) = 7$
 $v(\{a_1, a_2, a_3\}) = 15$

Coalition value $v(C)$: Maximum car sales in C



Dynamic and Trusted Coalition Forming



Trust in potential coalition partners?

Agents that dynamically negotiate trusted coalitions shall

- Use *efficient* mechanism for *trust assessment*, and
- Produce a *game-theoretically stable* solution as fast as possible

Fast BSCA-T algorithm for dynamic forming of trusted and bilateral Shapley value stable coalitions

W. Park. Master Thesis, University of the Saarland, 2004.



Recall: Bilateral Shapley Values

- Shapley value of agent in grand coalition $C=A$

- Fair payoff distribution

$$u(a) = \sum_{C \subseteq A} \frac{(|C| - |A|)! (|C| - 1)!}{|A|!} \cdot (v(C) - v(C \setminus \{a\}))$$

- Bilateral Shapley value BSV for arbitrary coalitions C

$$C = C_1 \cup C_2, \quad C_1 \cap C_2 = \emptyset$$

$$BSV_C(C_1) = \frac{1}{2} v(C_1) + \frac{1}{2} (v(C) - v(C_2)), \quad BSV_C(C_2) = \frac{1}{2} v(C_2) + \frac{1}{2} (v(C) - v(C_1))$$

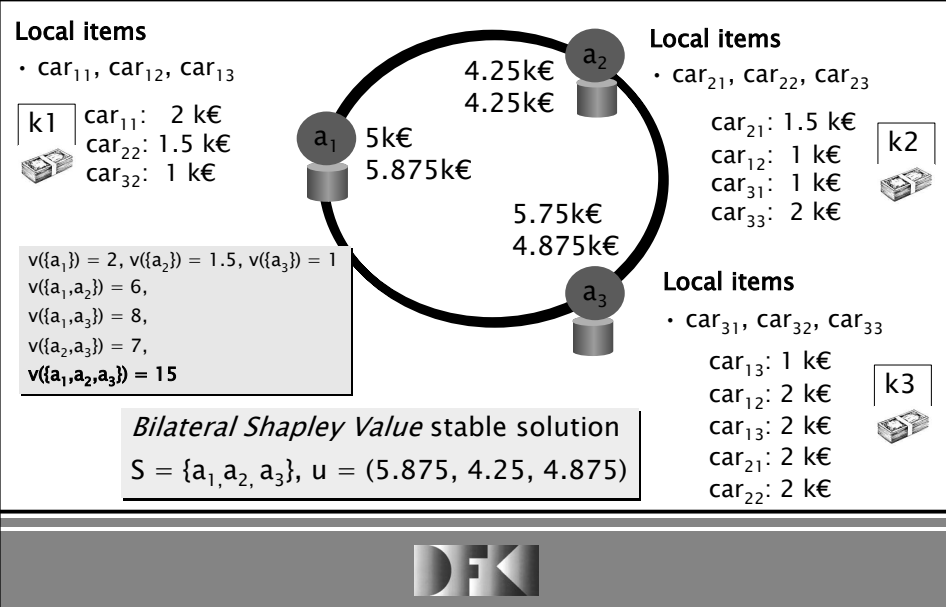
- Only *bilateral* coalition forming of $C+C'$ by *coalition leaders* of C and C'
- Bilateral *coalition formation history tree* $UG(C)$

- Distribute $BSV_{C+C'}(C)$ to individual agents in C

- Recursive computation of $BSV_C(a)$ over $UG(C)$



Example: BSV-Stable Retailer Agent Coalition



BSCA-D: Dynamic Coalition Forming

For each negotiation round, given game (A, v) and configuration (S, u) :

Each coalition (leader of) C bilaterally negotiates w/ other coalitions C'

- Send merger proposals $BSV(C', C+C')$ to limited number of C'
w/ maximum, individual rational joint profit $BSV(C, C+C')$ for C
- Form $C+C'$ with bilaterally accepted maximum profit
 - Distribute $BSV(C, C+C')$ to individual agents a in C ; Inform A on $C+C'$
- Determine preliminary overall configuration (S^*, u^*) w/ $C+C'$ in S^*

Coalition leader of $C+C'$ reacts on dynamic changes that affect $C+C'$

- if agent leaves: Split $C+C'$ to stable coalitions using new $UG(C+C' \setminus \{a\})$
- Inform A on changed S^* and/or u^* , any other type of change
- Determine configuration (S, u) and new game (A', v')

Until (Grand coalition A or Timeout or No change of coalition structure)



Agents *enter* or *leave* negotiation
 Agents change *utilities* of item(s)

Example

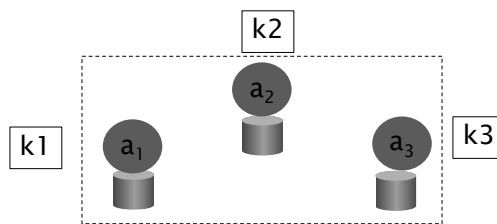
Cf. Retailer Coalition Game Example

Round 1: BSV-stable solution ($\{\{a_1, a_3\}, \{a_2\}\}, (4.5, 1.5, 3.5)$)

K-stable solution ($\{\{a_1, a_2\}, \{a_3\}\}, (3.5, 2.5, 1)$)

Round 2: Final **BSV-stable** solution ($\{a_1, a_2, a_3\}, (5.875, 4.25, 4.875)$)

Final **K-stable** solution ($\{a_1, a_2, a_3\}, (5, 4.25, 5.75)$)



BSCA-T: BSV-Stable Coalitions with Trust

In each bilateral negotiation round, each coalition leader a of C^*

- Determines *BSV-stable joint coalition candidates* C .
- Determines *trustworthiness of* C based on that of its members.

Trust rating of agent a' for a in joint coalition C is the

Sum of changes $SC_a(a')$ of its payoffs $u(a)$ caused by a' in joint coalitions so far, and trust ratings of a' by all other agents in C

- Send proposals to BSV-stable *and* trustworthy candidates C only.

Update of trust rating & probation time of a' :

1. Agent a' breaks joint coalition
2. Utility of items of a' for a changes
3. Negotiation ends

$$T_a(a') = \frac{SC_a(a')}{cn(a, a')}, \text{ if } cn(a, a') > pt_a(a'), \text{ else } 0$$

$$(1) SC_a(a')^{t+1} = SC_a(a') - \frac{v(C)^t - u(a')^t - v(C \setminus \{a'\})^t}{|C| - 1}$$

$$(2) SC_a(a')^{t+1} = SC_a(a') - \frac{d([v(C)]_{[a(a) \neq a']})}{|C|}$$

$$(3) SC_a(a')^{t+1} = SC_a(a') - \frac{v(C)^t - v(C \setminus \{a'\})^t}{2|C|}$$



Trustworthy Agents and Coalitions

- **For agent a**
 - agent a' is trustworthy iff $T_a(a') + R_a(a') \geq 0$
 - coalition C' is
 - Trustworthy if all agents in C' are trustworthy for a
 - Neutral if there are untrustworthy agents for a in C' but the sum of trust and reputation of all a in C' is non-negative.
 - Untrustworthy else
- **For coalition (leader of) C**
 - coalition C' is
 - Trustworthy if C' is trustworthy for each a in C
 - Untrustworthy if C' is untrustworthy for at least one a in C



BSCA-T: Dynamic and Trusted Coalitions

For each negotiation round, given game (A, v) and configuration (S, u) :

Each coalition (leader of) C bilaterally negotiates w/ other coalitions C'

- Send merger proposals $BSV_{C+C'}(C')$ to limited number of C'
 - max joint profit $BSV_{C+C'}(C)$, C' trustworthy or neutral for C
- Form $C+C'$ with bilaterally accepted maximum proposal
 - Distribute $BSV_{C+C'}(C)$ to individual agents a in C ; Inform A on $C+C'$
- Determine preliminary overall configuration (S^*, u^*) w/ $C+C'$ in S^*

Coalition leader of $C+C'$ reacts on dynamic changes that affect $C+C'$

- if agent leaves: Split $C+C'$ to stable coalitions using new $UG(C+C' \setminus \{a\})$
- Inform A on changed S^* and/or u^*
- Determine configuration (S, u) , new game (A', v')

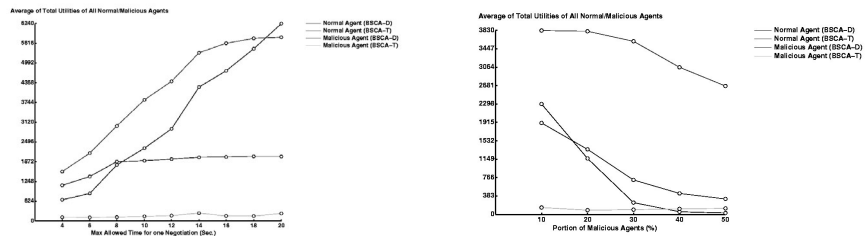
Each agent updates trust/reputation ratings, adapt probation times.

Until (Grand coalition A or Timeout or No change of coalition structure)



BSCA-T: Evaluation Results

Avg. total agent payoffs



Vs. Negotiation round times

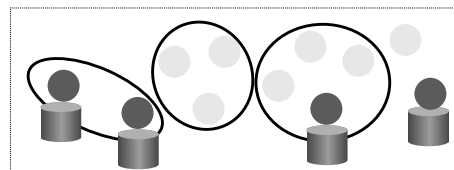
Vs. Number of untrustworthy agents

Each simulation (n=30/40/50/60 agts): n*100 neg., init probation time 4 neg., fixed set of changes

Using the BSCA-T, trustworthy agents get higher payoffs than untrustworthy ones in dynamic BSV-stable coalition negotiations w/ or w/o trust (BSCA-D). Simple but very fast and effective trust rating w/ stable payoff distribution.



Secure Coalition Forming



Secure and Stable Coalitions?

Secure negotiation of stable coalitions shall
 preserve individual agent's **data privacy** (security) and
 yield a **stable payoff distribution** to coalition members (safety)

KCA algorithm for forming secure and Kernel stable coalitions



Recall: KCA Coalition Algorithm

Each agent $a \in C$ in coalition structure S with payoff distribution u :

→ Broadcast tasks, items, and local values $lworth(a, C)$ for all C in S ←

If coalition leader of C then

- Generate and send beneficial Kernel stable proposals (S', u') for $C+C'$
- Evaluate incoming proposals
- Accept most beneficial proposal; broadcast decision to other agents
- Stop if no agent accepted any proposal
- Decide which accepted proposal is the next configuration
- Inform coalition members on new coalition and payoff

Vote for new coalition leader of $C+C'$

$lworth(a, C)$ is the maximum contribution of agent a to C as its member



KCA: Secure K-Stable Coalitions

Using the KCA, any agent a can hide from other agents in K-stable coalition negotiations without loss of benefit for anyone ...

... any set of its local items and local customer purchases -
iff this local data is exclusively used to compute its self-value $v(\{a\})$.

Thrm. (Blankenburg/Klusch 04)

Given K-stable (S, u) of (A, v) , any configuration (S, u^*) for (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 *Kernel stable*. Since $u(a) - u^*(a)$ is constant for *all* pairs of Kernel stable proposals $(S, u)/(S, u^*)$, the KCA is *safe* against non-disclosure of self-values.

B. Blankenburg, M. Klusch, in Proceedings AAMAS 2004 Conference



KCA: Safety and Data Privacy

For any K-stable solution (S, u) of (A, v) , the configuration (S, u^*) with $u^*(a) = u(a) - l_{\text{worth}}(a, \{a\})$ is Kernel stable for (A, v^*) with $v^*(C) = v(C) - l_{\text{worth}}(a, \{a\}), v^*(\{a\})=0$, f.a. $a \in A$.

Consider (A, v) , (S, u) , and assume that agent a in C in S communicates to all C in S :

$l_{\text{worth}}^*(a, C) = l_{\text{worth}}(a, C) - l_{\text{worth}}(a, \{a\})$

- > New game (A, v^*) with $v^*(C) = v(C) - l_{\text{worth}}(a, \{a\}), v^*(\{a\})=0$
- > No change in excesses: $e^*(C) = v^*(C) - u^*(C) = v(C) - u(C) = e(C)$
- > No change in surpluses of a in solution (S, u) of (A, v) resp. (S, u^*) of (A, v^*)
- > Induction over all agents a in A and S yields
equivalent game (A, v^*) modulo self-values



Example

lworth	{a ₁ }	{a ₁ , a ₂ }	{a ₁ , a ₃ }	{a ₂ , a ₃ }	A
a ₁	2	3.5	3	-	4.5
a ₂	1.5	2.5	-	2	4.5
a ₃	1	-	5	5	6
v*		6	8	7	15

$A = \{a_1, a_2, a_3\}$

$v^*(C) =$

$\sum_{a \in C} (l_{\text{worth}}(a, C) - l_{\text{worth}}(a, a))$

K-stable solution $(\{a_1, a_2\}, \{a_3\}, (3.5, 2.5, 1))$ of original game (A, v)

$$s(a_1, a_2) = v(13) - (u(1) + u(3)) = 8 - 3.5 - 1 = 3.5$$

$$s(a_2, a_1) = 7 - 2.5 - 1 = 3.5$$

Hiding of self-values induces a new game (A, v^*) with

new but still K-stable solution $(\{a_1, a_2\}, \{a_3\}, (1.5, 1, 0))$ balancing

$$s^*(a_1, a_2) = (v(13) - v(1) - v(3)) - (u(1) - v(1) + u(3) - v(3)) = 5 - 1.5 = 3.5$$

$$s^*(a_2, a_1) = 4.5 - 1 = 3.5$$



Example: K-Stable Retailer Agent Coalition

Local items:
car₁₁, car₁₂, car₁₃

Car sales:

k1
 car₁₁: 2
 car₂₂: 1.5
 car₃₂: 1

Local items:
car₂₁, car₂₂, car₂₃

k2
 car₂₁: 1.5
 car₁₂: 1
 car₃₁: 1
 car₃₃: 2

$v(\{a_1\}) = 2, v(\{a_2\}) = 1.5, v(\{a_3\}) = 1$
 $v(\{a_1, a_2\}) = 6,$
 $v(\{a_1, a_3\}) = 8,$
 $v(\{a_2, a_3\}) = 7,$
 $v(\{a_1, a_2, a_3\}) = 15$

K-stable solution
 $S = \{a_1, a_2, a_3\}, u = (5, 4.25, 5.75)$

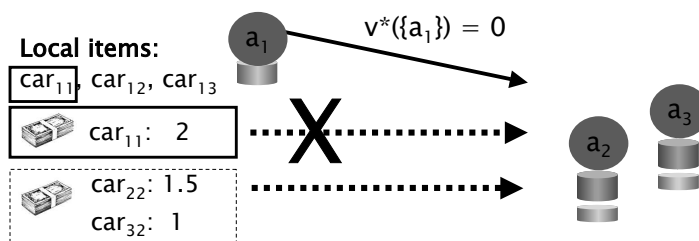
Local items:
car₃₁, car₃₂, car₃₃

k3
 car₃₁: 1
 car₁₂: 2
 car₁₃: 2
 car₂₁: 2
 car₂₃: 2



Example: Secure K-Stable Coalitions

Data privacy: Agent a_1 can prevent other agents from knowing ...
 how many and what kind of own cars it can sell to its customer for what price.

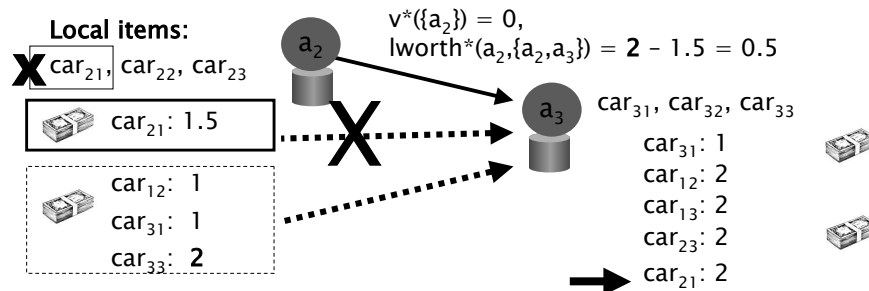


Agents a_2 and a_3 *cannot infer* $v(\{a_1\})$ from their local knowledge,
 since car₁₁ cannot be sold to them to maximize any joint coalition value.



Example: Secure K-Stable Coalitions

If local items can locally *and* remotely be sold then their local sales value can be partially inferred by all agents.



Knowing that car₃₃ can be sold for 2 (instead of car₂₁) by a₂ in coalition {a₂, a₃} to maximize its value, agent a₃ can infer $v(\{a_2\}) \geq 1.5 (= 2 - lworth^*(a_2, \{a_2, a_3\}))$



KCA: Frauds in Coalition Negotiations

Using the KCA, any agent *a* can unfoundedly strengthen its bargaining position, but at high computational costs.

In first negotiation round only:

- Receive all values from other agents but *delay communication of own values*.
- *Compute the game* (A,v) and simulate a KCA run to *predict* the solution (S,u).
- Choose C' *not in S*. Iteratively determine add-on factor **r** for increasing own local value for C' such that $u'(a)$ in (S',u') for games (A,v'), $v'(C') = v(C') + r$ is *maximum*: F.e. (A,v') simulate KCA runs until (S',u'), C' ∈ S', is found.
- Deceive all other agents on own real local value $lworth(a, C')$ for C'

The Fraud: $lworth^*(a, C') = lworth(a, C') + r$

