KERNEL STABLE COALITION FORMATION

Bardia Khalesi, Magnus Nord

May 17, 2010
1 General
   Introduction
   KCA algorithm

2 On safe kernel stable coalition forming among agents
   Introduction
   Properties of the KCA

3 Feasible Formation of Coalitions in NonSuperAdditive Environments
   Introduction
   DEK-CFM Protocol
   Truncated Transfer Scheme
   DNPK-CFM Protocol

4 Conclusion
Overview

• Formation of kernel stable coalitions
Overview

- Formation of kernel stable coalitions
- KCA algorithm
Overview

- Local value or $lworth_a(C)$ of an agent $a$ is the sum of the self-worth and the marginal contribution to the coalition $C$. 
Overview

- Local value or \( lworth_a(C) \) of an agent \( a \) is the sum of the self-worth and the marginal contribution to the coalition \( C \).
Overview

- Local value or $lworth_a(C)$ of an agent $a$ is the sum of the self-worth and the marginal contribution to the coalition $C$.
- Non-super-additive Games
Overview

- Local value or $lworth_a(C)$ of an agent $a$ is the sum of the self-worth and the marginal contribution to the coalition $C$.
- Non-super-additive Games
  - At least one pair of potential coalitions are not better off by merging
Overview

- Local value or $lworth_a(C)$ of an agent $a$ is the sum of the self-worth and the marginal contribution to the coalition $C$.
- Non-super-additive Games
  - At least one pair of potential coalitions are not better off by merging
  - Costly to increase the amount of members in a coalition
Kernel

- Why the kernel?
  - Will not be empty

Kernel Stable Coalition Formation

Bardia Khalesi, Magnus Nord
Kernel

• Why the kernel?
  • Will not be empty
  • Handles agent symmetry
Kernel

- Why the kernel?
  - Will not be empty
  - Handles agent symmetry
  - Significantly smaller than bargaining set
Kernel

- Why the kernel?
  - Will not be empty
  - Handles agent symmetry
  - Significantly smaller than bargaining set
  - Easier to compute
Kernel

• Why the kernel?
  • Will not be empty
  • Handles agent symmetry
  • Significantly smaller than bargaining set
  • Easier to compute
  • Compatible with other agents
Kernel

- Why the kernel?
  - Will not be empty
  - Handles agent symmetry
  - Significantly smaller than bargaining set
  - Easier to compute
  - Compatible with other agents

- Kernel consists of configurations which are in equilibrium
Kernel

- Why the kernel?
  - Will not be empty
  - Handles agent symmetry
  - Significantly smaller than bargaining set
  - Easier to compute
  - Compatible with other agents

- Kernel consists of configurations which are in equilibrium
  - Equilibrium Conditions:
    - $s_{ij} = s_{ji}$
Kernel

- Why the kernel?
  - Will not be empty
  - Handles agent symmetry
  - Significantly smaller than bargaining set
  - Easier to compute
  - Compatible with other agents

- Kernel consists of configurations which are in equilibrium
  - Equilibrium Conditions:
    - $s_{ij} = s_{ji}$
    - $s_{ij} > s_{ji}, u_{j} = v(A_{j})$
Kernel

- Why the kernel?
  - Will not be empty
  - Handles agent symmetry
  - Significantly smaller than bargaining set
  - Easier to compute
  - Compatible with other agents

- Kernel consists of configurations which are in equilibrium
  - Equilibrium Conditions:
    - \( s_{ij} = s_{ji} \)
    - \( s_{ij} > s_{ji}, u_j = v(A_j) \)
    - \( s_{ji} < s_{ij}, u_i = v(A_i) \)
Kernel

- Why the kernel?
  - Will not be empty
  - Handles agent symmetry
  - Significantly smaller than bargaining set
  - Easier to compute
  - Compatible with other agents

- Kernel consists of configurations which are in equilibrium
  - Equilibrium Conditions:
    - $s_{ij} = s_{ji}$
    - $s_{ij} > s_{ji}, u_j = v(A_j)$
    - $s_{ji} < s_{ij}, u_i = v(A_i)$

- Pareto optimality is insufficient for the evaluation of possible coalitions.
KCA algorithm

1. Communication
   1. Every agent is coalition leader in its singleton coalition
KCA algorithm

1 Communication
   1 Every agent is coalition leader in its singelton coalition
   2 Send and receive tasks from all other agents
KCA algorithm

1 Communication
   1 Every agent is coalition leader in its singleton coalition
   2 Send and receive tasks from all other agents
   3 Send and receive evaluations of the tasks
KCA algorithm

1. Communication
   1. Every agent is coalition leader in its singleton coalition
   2. Send and receive tasks from all other agents
   3. Send and receive evaluations of the tasks
   4. For all possible coalitions: evaluate \( lworth_a(C) \) and send to all agents
KCA algorithm

1 Communication
   1. Every agent is coalition leader in its singleton coalition
   2. Send and receive tasks from all other agents
   3. Send and receive evaluations of the tasks
   4. For all possible coalitions: evaluate $lworth_a(C)$ and send to all agents
   5. Receive all local values from all other agents
KCA algorithm

1 Communication
   1. Every agent is coalition leader in its singleton coalition
   2. Send and receive tasks from all other agents
   3. Send and receive evaluations of the tasks
   4. For all possible coalitions: evaluate $lworth_a(C)$ and send to all agents
   5. Receive all local values from all other agents

2 Generating Proposals
   1. If the agent is not leader of the coalition, 4.3.
KCA algorithm

1. Communication
   1. Every agent is coalition leader in its singleton coalition
   2. Send and receive tasks from all other agents
   3. Send and receive evaluations of the tasks
   4. For all possible coalitions: evaluate $lworth_a(C)$ and send to all agents
   5. Receive all local values from all other agents

2. Generating Proposals
   1. If the agent is not leader of the coalition, 4.3.
   2. For each other coalition, compute a Kernel-stable configuration. Send proposal to strictly dominating coalitional configuration.
KCA algorithm

3 Evaluating proposal

1 Evaluate received proposals, choose the most beneficial
KCA algorithm

1. Evaluating proposal
   1. Evaluate received proposals, choose the most beneficial
   2. Inform all leaders about accepted proposal

2. Deciding coalition configuration
KCA algorithm

3. Evaluating proposal
   1. Evaluate received proposals, choose the most beneficial
   2. Inform all leaders about accepted proposal

4. Deciding coalition configuration
   1. Receive accepted proposals: if none was accepted, then stop
KCA algorithm

3 Evaluating proposal
   1 Evaluate received proposals, choose the most beneficial
   2 Inform all leaders about accepted proposal

4 Deciding coalition configuration
   1 Receive accepted proposals: if none was accepted, then stop
   2 Choose one configuration, by considering the order of preferences: bilateral > unilateral, biggest payoff distribution than greatest computational power.
KCA algorithm

1. Evaluating proposal
   1. Evaluate received proposals, choose the most beneficial
   2. Inform all leaders about accepted proposal

2. Deciding coalition configuration
   1. Receive accepted proposals: if none was accepted, then stop
   2. Choose one configuration, by considering the order of preferences: bilateral > unilateral, biggest payoff distribution than greatest computational power.
   3. Inform all coalition members about new configuration
KCA algorithm

1. Evaluating proposal
   a. Evaluate received proposals, choose the most beneficial
   b. Inform all leaders about accepted proposal

2. Deciding coalition configuration
   a. Receive accepted proposals: if none was accepted, then stop
   b. Choose one configuration, by considering the order of preferences: bilateral > unilateral, biggest payoff distribution than greatest computational power.
   c. Inform all coalition members about new configuration
   d. New coalition leader is the agent with the highest computational power. The other coalition leaders are informed about the new leader
KCA algorithm

3 Evaluating proposal
   1 Evaluate received proposals, choose the most beneficial
   2 Inform all leaders about accepted proposal

4 Deciding coalition configuration
   1 Receive accepted proposals: if none was accepted, then stop
   2 Choose one configuration, by considering the order of preferences: bilateral > unilateral, biggest payoff distribution than greatest computational power.
   3 Inform all coalition members about new configuration
   4 New coalition leader is the agent with the highest computational power. The other coalition leaders are informed about the new leader
   5 If grand coalition is formed, or time ends: stop. Else go back to Generating proposals
KCA algorithm

- Some elements which can affect the performance
KCA algorithm

- Some elements which can affect the performance
  - Distribution
KCA algorithm

- Some elements which can affect the performance
  - Distribution
  - Communication cost
KCA algorithm

- Some elements which can affect the performance
  - Distribution
  - Communication cost
  - Limited computation time
KCA algorithm

• Some elements which can affect the performance
  • Distribution
  • Communication cost
  • Limited computation time

• Therefore there is a trade-off between quality of solution, and speed
KCA algorithm

- Some elements which can affect the performance
  - Distribution
  - Communication cost
  - Limited computation time
- Therefore there is a trade-off between quality of solution, and speed
  - Quality: payoff maximization and stability
KCA algorithm

- Some elements which can affect the performance
  - Distribution
  - Communication cost
  - Limited computation time
- Therefore there is a trade-off between quality of solution, and speed
  - Quality: payoff maximization and stability
  - Speed: efficiency and anytime algorithm
On safe kernel stable coalition forming among agents

- Investigates several properties of the KCA algorithm
On safe kernel stable coalition forming among agents

- Investigates several properties of the KCA algorithm
  - Incomplete information
On safe kernel stable coalition forming among agents

- Investigates several properties of the KCA algorithm
  - Incomplete information
  - Changing agent set
On safe kernel stable coalition forming among agents

- Investigates several properties of the KCA algorithm
  - Incomplete information
  - Changing agent set
  - Privacy
On safe kernel stable coalition forming among agents

- Investigates several properties of the KCA algorithm
  - Incomplete information
  - Changing agent set
  - Privacy
  - Fraud
On safe kernel stable coalition forming among agents

- Investigates several properties of the KCA algorithm
  - Incomplete information
  - Changing agent set
  - Privacy
  - Fraud

- Paper contains many examples, but will skip these due to time constraints
On safe kernel stable coalition forming among agents

• Investigates several properties of the KCA algorithm
  • Incomplete information
  • Changing agent set
  • Privacy
  • Fraud

• Paper contains many examples, but will skip these due to time constraints

• Fairly general: Considers both superadditive and non-superadditive games
Incomplete information

- Example: an agent does not receive the coalition value for some coalition
Incomplete information

- Example: an agent does not receive the coalition value for some coalition
- The agent has to estimate the coalition value
Properties of the KCA

Incomplete information

- Example: an agent does not receive the coalition value for some coalition
- The agent has to estimate the coalition value
- This estimation can lead to the agent solving a different game, compared to the agents with complete information
Properties of the KCA

Incomplete information

- Example: an agent does not receive the coalition value for some coalition
- The agent has to estimate the coalition value
- This estimation can lead to the agent solving a different game, compared to the agents with complete information
- Thus reaching different outcomes
Properties of the KCA

Incomplete information

- Example: an agent does not receive the coalition value for some coalition
- The agent has to estimate the coalition value
- This estimation can lead to the agent solving a different game, compared to the agents with complete information
- Thus reaching different outcomes
- However this still leads to a kernel-stable solution, but the agent might get less payoff.
Properties of the KCA

Incomplete information

- Example: an agent does not receive the coalition value for some coalition
- The agent has to estimate the coalition value
- This estimation can lead to the agent solving a different game, compared to the agents with complete information
- Thus reaching different outcomes
- However this still leads to a kernel-stable solution, but the agent might get less payoff.
- Coalition negotiations are safe with respect to unknown coalition values
Changing agent set

- During the negotiations agents might become unavailable (for example, network connection breaking down)
Properties of the KCA

Changing agent set

- During the negotiations agents might become unavailable (for example, network connection breaking down)
- The agent will therefore not send out messages required by the protocol
Properties of the KCA

Changing agent set

- During the negotiations agents might become unavailable (for example, network connection breaking down)
- The agent will therefore not send out messages required by the protocol
- The severity depends on if it is a coalition leader or member which becomes unavailable
Changing agent set

- During the negotiations agents might become unavailable (for example, network connection breaking down)
- The agent will therefore not send out messages required by the protocol
- The severity depends on if it is a coalition leader or member which becomes unavailable
- If a leader drops out, the coalition will not be send out, or receive, any proposals. In addition the other members in the coalition will not be informed about the new configuration
Properties of the KCA

Changing agent set

- During the negotiations agents might become unavailable (for example, network connection breaking down)
- The agent will therefore not send out messages required by the protocol
- The severity depends on if it is a coalition leader or member which becomes unavailable
- If a leader drops out, the coalition will not be send out, or receive, any proposals. In addition the other members in the coalition will not be informed about the new configuration
- The coalition negotiation will therefore not be safe if the agent set is changing
Privacy (security)

- A surprising property of the KCA algorithm, is the ability of the agents to hide their local information without any profit loss in the final configuration.
Properties of the KCA

Privacy (security)

- A surprising property of the KCA algorithm, is the ability of
  the agents to hide their local information without any profit
  loss in the final configuration
- This is caused by an inherent property of the definition of
  kernel stability: an amount added to the valuation function
  for a coalition will be subtracted when calculating the
  surplus.
Privacy (security)

- A surprising property of the KCA algorithm, is the ability of the agents to hide their local information without any profit loss in the final configuration.
- This is caused by an inherent property of the definition of kernel stability: an amount added to the valuation function for a coalition will be subtracted when calculating the surplus.
- Therefore the local values are not required to be communicated between the agents to reach a kernel stable solution.
Privacy (security)

- A surprising property of the KCA algorithm, is the ability of the agents to hide their local information without any profit loss in the final configuration.
- This is caused by an inherent property of the definition of kernel stability: an amount added to the valuation function for a coalition will be subtracted when calculating the surplus.
- Therefore the local values are not required to be communicated between the agents to reach a kernel stable solution.
- If and only if, the local information is exclusively used to compute its self-value.
Privacy (security)

- A surprising property of the KCA algorithm, is the ability of the agents to hide their local information without any profit loss in the final configuration.
- This is caused by an inherent property of the definition of kernel stability: an amount added to the valuation function for a coalition will be subtracted when calculating the surplus.
- Therefore the local values are not required to be communicated between the agents to reach a kernel stable solution.
- If and only if, the local information is exclusively used to compute its self-value.
- Therefore the coalition negotiations are safe with respect to privacy.
Properties of the KCA

Privacy (security)

- A surprising property of the KCA algorithm, is the ability of the agents to hide their local information without any profit loss in the final configuration.
- This is caused by an inherent property of the definition of kernel stability: an amount added to the valuation function for a coalition will be subtracted when calculating the surplus.
- Therefore the local values are not required to be communicated between the agents to reach a kernel stable solution.
- If and only if, the local information is exclusively used to compute its self-value.
- Therefore the coalition negotiations are safe with respect to privacy.
Fraud

• Waiting to communicate $lworth_a(C)$ until it has received all the other agent’s $lworth$
Fraud

- Waiting to communicate \( lworth_a(C) \) until it has received all the other agent’s \( lworth \)
- The fraudulent agent will be the only one which can compute all coalition values
Fraud

- Waiting to communicate $lworth_a(C)$ until it has received all the other agent’s $lworth$
- The fraudulent agent will be the only one which can compute all coalition values
- Can not be prevented or detected
Fraud

- Waiting to communicate $lworth_a(C)$ until it has received all the other agent’s $lworth$
- The fraudulent agent will be the only one which can compute all coalition values
- Can not be prevented or detected
- However this is computational complex: the fraudulent agent has to check all ($O(2^n)$) possible coalitions
Properties of the KCA

Fraud

- Waiting to communicate $lworth_a(C)$ until it has received all the other agent’s $lworth$
- The fraudulent agent will be the only one which can compute all coalition values
- Can not be prevented or detected
- However this is computational complex: the fraudulent agent has to check all $O(2^n)$ possible coalitions
- Other agents might become suspicious because of the delay in the deceiving agent’s communication
Environment Description

- Distributed AI:
  - Cooperative Distributed Problem Solving (CDPS) → distribution of required effort for solving a particular problem among a number of modules (or nodes).
Environment Description

- Distributed AI:
  - Cooperative Distributed Problem Solving (CDPS) → distribution of required effort for solving a particular problem among a number of modules (or nodes).
  - Multiagent Systems (MAS) → coordinating intelligent behavior among autonomous, heterogeneous, intelligent agents.
Environment Description

- **Distributed AI:**
  - Cooperative Distributed Problem Solving (CDPS) → distribution of required effort for solving a particular problem among a number of modules (or nodes).
  - Multiagent Systems (MAS) → coordinating intelligent behavior among autonomous, heterogeneous, intelligent agents.

- **Protocols:**
  - Any interaction among agents requires some protocols. As more protocols are enforced on the agents, communication usually decreases. Yet the protocols may be contradictory to the rationality of an individual agent.
  - Any deviation from the protocols must be revealable and penalizable, or the protocols must be self-enforced.
  - Some constrains are needed to avoid an endless loop of rejected proposals for coalition formation.
Environment Description

- **Strategies:**
  - the method that agents employ to handle proposals, such as increasing the payoff or satisfying an equilibrium requirement.
Environment Description

• Strategies:
  • the method that agents employ to handle proposals, such as increasing the payoff or satisfying an equilibrium requirement.

• Equilibrium:
  • Nash Equilibrium
  • Approach 1: High computations, thus vast increase in the complexity of the model.
Environment Description

- **Strategies:**
  - the method that agents employ to handle proposals, such as increasing the payoff or satisfying an equilibrium requirement.

- **Equilibrium:**
  - Nash Equilibrium
  - Approach 1: High computations, thus vast increase in the complexity of the model.
  - Approach 2: Bounded rationality leads to approximations which is not satisfactory due to existence of better decisions.
Environment Description

- **Strategies:**
  - the method that agents employ to handle proposals, such as increasing the payoff or satisfying an equilibrium requirement.

- **Equilibrium:**
  - Nash Equilibrium
  - Approach 1: High computations, thus vast increase in the complexity of the model.
  - Approach 2: Bounded rationality leads to approximations which is not satisfactory due to existence of better decisions.
  - Approach 3: Time-bounded equilibrium → By belief of maximizing the expected utility with respect to a bounded computation time of a strategy.
Definitions

- Payment Configuration ($\text{PC}(\mathbf{U}, \mathbf{C})$):
  - $\mathbf{U} = < u_1, u_2, ..., u_n >$, where $u_i$ is the payoff to $A_i$
Definitions

• Payment Configuration \( \textbf{PC}(U,C) \):
  - \( U = < u_1, u_2, ..., u_n > \), where \( u_i \) is the payoff to \( A_i \)
  - \( C = \{ C_i \} \), where \( \bigcup_i C_i = N; \forall C_i, C_j; C_i \neq C_j; C_i \cap C_j = \emptyset \)

Feasible Formation of Coalitions in NonSuperAdditive Environments
Definitions

- Payment Configuration \( \text{PC}(U, C) \):
  - \( U = \langle u_1, u_2, \ldots, u_n \rangle \), where \( u_i \) is the payoff to \( A_i \)
  - \( C = \{ C_i \} \), where \( \bigcup_i C_i = N; \forall C_i, C_j; C_i \neq C_j; C_i \cap C_j = \emptyset \)

- Coalitional Configuration Space (CCS):
  \( \{ C | \forall C_i \in C, V(C_i) \geq \sum_{A_i \in C_i} V(A_i) \} \)
Definitions

- Payment Configuration ($PC(U,C)$):
  - $U = \langle u_1, u_2, ..., u_n \rangle$, where $u_i$ is the payoff to $A_i$
  - $C = \{C_i\}$, where $\bigcup C_i = N; \forall C_i, C_j; C_i \neq C_j; C_i \cap C_j = \emptyset$

- Coalitional Configuration Space ($CCS$):
  \[
  \{C|\forall C_i \in C, V(C_i) \geq \sum_{A_i \in C_i} V(A_i)\}
  \]

- Payment Configuration Space ($PCS$) consists of pairs ($U, C$) where $U$ is individually rational and $C \in CCS$
Definitions

- Payment Configuration ($PC(\mathbf{U}, \mathbf{C})$):
  - $\mathbf{U} = < u_1, u_2, ..., u_n >$, where $u_i$ is the payoff to $A_i$
  - $\mathbf{C} = \{ C_i \}$, where $\bigcup_i C_i = N$; $\forall C_i, C_j; C_i \neq C_j; C_i \cap C_j = \emptyset$

- Coalitional Configuration Space ($CCS$):
  \[ \{ \mathbf{C} | \forall C_i \in \mathbf{C}, V(C_i) \geq \sum_{A_i \in C_i} V(A_i) \} \]

- Payment Configuration Space ($PCS$) consists of pairs $(\mathbf{U}, \mathbf{C})$ where $\mathbf{U}$ is individually rational and $\mathbf{C} \in CCS$

- PC-Error: $e = \max_{i,j}(s_{ij} - s_{ji})$
Definitions

- Payment Configuration ($\text{PC}(\textbf{U}, \textbf{C})$):
  - $\textbf{U} = < u_1, u_2, ..., u_n >$, where $u_i$ is the payoff to $A_i$
  - $\textbf{C} = \{ C_i \}$, where $\bigcup_i C_i = N; \forall C_i, C_j; C_i \neq C_j; C_i \cap C_j = \emptyset$

- Coalitional Configuration Space ($\text{CCS}$):
  $\{ \textbf{C} | \forall C_i \in \textbf{C}, V(C_i) \geq \sum_{A_i \in C_i} V(A_i) \}$

- Payment Configuration Space ($\text{PCS}$) consists of pairs ($\textbf{U}, \textbf{C}$) where $\textbf{U}$ is individually rational and $\textbf{C} \in \text{CCS}$

- PC-Error: $e = \max_{i,j}(s_{ij} - s_{ji})$

- PC relative error: $e_r = \frac{e}{\sum u_i}$
DEK-CFM Protocol

- Distributed, Exponential, Kernel-oriented Coalition-Formation Model (DEK-CFM) leads to a PC that is Pareto optimal and K-stable.
DEK-CFM Protocol

• Distributed, Exponential, Kernel-oriented Coalition-Formation Model (DEK-CFM) leads to a PC that is Pareto optimal and K-stable.

• In cases where time, communications, and computation are cheap or costless, or in cases where there is a small number of agents, DEK-CFM is adequate.
DEK-CFM Protocol

• Protocol:
  1. compute all of the coalitions and corresponding coalitional values and transmit all of them to other agents.
DEK-CFM Protocol

- **Protocol:**
  1. Compute all of the coalitions and corresponding coalitional values and transmit all of them to other agents.
  2. $A_i$ is assigned to a unique random integer $z_i \in [1, n]$. 
DEK-CFM Protocol

- Protocol:
  1. compute all of the coalitions and corresponding coalitional values and transmit all of them to other agents.
  2. $A_i$ is assigned to a unique random integer $z_i \in [1, n]$
  3. compute CCs consist of $z_i$ coalitions.
DEK-CFM Protocol

- Protocol:
  1. compute all of the coalitions and corresponding coalitional values and transmit all of them to other agents.
  2. \( A_i \) is assigned to a unique random integer \( z_i \in [1, n] \)
  3. compute CCs consist of \( z_i \) coalitions.
  4. find a K-stable PCs of each computed CCs.
DEK-CFM Protocol

- Protocol:
  1. compute all of the coalitions and corresponding coalitional values and transmit all of them to other agents.
  2. $A_i$ is assigned to a unique random integer $z_i \in [1, n]$.
  3. compute CCs consist of $z_i$ coalitions.
  4. find a $K$-stable PCs of each computed CCs.
  5. list personally Pareto optimal PCs from computed PCs.
DEK-CFM Protocol

- Protocol:
  1. compute all of the coalitions and corresponding coalitional values and transmit all of them to other agents.
  2. $A_i$ is assigned to a unique random integer $z_i \in [1, n]$
  3. compute CCs consist of $z_i$ coalitions.
  4. find a K-stable PCs of each computed CCs.
  5. list personally Pareto optimal PCs from computed PCs.
  6. merge lists of all agents in one list and find the Pareto optimal PCs:
    - iteration $j$: $A_i$ s.t. $z_i \text{mod} 2^j = 1$ merge its list with $A_k$ s.t. $z_k = z_i + 2^j - 1$
    - find the locally Pareto optimal PCs from the merged list and hold by the $A_i$ with $z_i \text{mod} 2^j = 1$
    - stop iteration if all of the agents have been approached.
  7. choose one of the found Pareto optimal PCs by the decision-making method
DEK-CFM Protocol

- Protocol:
  1. Compute all of the coalitions and corresponding coalitional values and transmit all of them to other agents.
  2. $A_i$ is assigned to a unique random integer $z_i \in [1, n]$
  3. Compute CCs consist of $z_i$ coalitions.
  4. Find a K-stable PCs of each computed CCs.
  5. List personally Pareto optimal PCs from computed PCs.
  6. Merge lists of all agents in one list and find the Pareto optimal PCs:
     - Iteration $j$: $A_i$ s.t. $z_i \mod 2^j = 1$ merge its list with $A_k$ s.t. $z_k = z_i + 2^j - 1$
DEK-CFM Protocol

- Protocol:
  1. compute all of the coalitions and corresponding coalitional values and transmit all of them to other agents.
  2. $A_i$ is assigned to a unique random integer $z_i \in [1, n]$
  3. compute CCs consist of $z_i$ coalitions.
  4. find a K-stable PCs of each computed CCs.
  5. list personally Pareto optimal PCs from computed PCs.
  6. merge lists of all agents in one list and find the Pareto optimal PCs:
     - iteration $j$: $A_i$ s.t. $z_i \mod 2^j = 1$ merge its list with $A_k$ s.t. $z_k = z_i + 2^j - 1$
     - find the locally Pareto optimal PCs from the merged list and hold by the $A_i$ with $z_i \mod 2^j = 1$
DEK-CFM Protocol

- **Protocol:**
  1. compute all of the coalitions and corresponding coalfitional values and transmit all of them to other agents.
  2. $A_i$ is assigned to a unique random integer $z_i \in [1, n]$.
  3. compute CCs consist of $z_i$ coalitions.
  4. find a K-stable PCs of each computed CCs.
  5. list personally Pareto optimal PCs from computed PCs.
  6. merge lists of all agents in one list and find the Pareto optimal PCs:
     - iteration $j$: $A_i$ s.t. $z_i \mod 2^j = 1$ merge its list with $A_k$ s.t. $z_k = z_i + 2^j - 1$
     - find the locally Pareto optimal PCs from the merged list and hold by the $A_i$ with $z_i \mod 2^j = 1$
     - stop iteration if all of the agents have been approached.
DEK-CFM Protocol

- **Protocol:**
  1. compute all of the coalitions and corresponding coalitional values and transmit all of them to other agents.
  2. $A_i$ is assigned to a unique random integer $z_i \in [1, n]$
  3. compute CCs consist of $z_i$ coalitions.
  4. find a **K-stable PCs** of each computed CCs.
  5. list personally *Pareto optimal PCs* from computed PCs.
  6. merge lists of all agents in one list and find the *Pareto optimal PCs*:
     - iteration $j$: $A_i$ s.t. $z_i \mod 2^j = 1$ merge its list with $A_k$ s.t. $z_k = z_i + 2^{j-1}$
     - find the locally *Pareto optimal PCs* from the merged list and hold by the $A_i$ with $z_i \mod 2^j = 1$
     - stop iteration if all of the agents have been approached.
  7. choose one of the found *Pareto optimal PCs* by the decision-making method
DEK-CFM Protocol

⑧ transmit all the details of the calculations to other agents
transmit all the details of the calculations to other agents

any deceitful PC can be detected and canceled by the received calculations.
transmit all the details of the calculations to other agents

any deceitful PC can be detected and canceled by the received calculations.

complexity of the computation of coalitional values and configurations is $O(n^n)$
Truncated Transfer Scheme

- To calculate the K-\(\varepsilon\)-stable PCs:
  1. Start with a \(U_0\)
Truncated Transfer Scheme

- To calculate the K-\(\varepsilon\)-stable \(\text{PCs}\):
  1. Start with a \(\mathbf{U}_0\)
  2. If \(\sum u_i > \sum V(\mathbf{C})\) Then use the n-correction of Wu(1977)
Truncated Transfer Scheme

- To calculate the K-$\varepsilon$-stable PCs:
  1. Start with a $U_0$
  2. If $\sum u_i > \sum V(C)$ Then use the n-correction of Wu(1977)
  3. Calculate the demand functions with respect to $U_i$
Truncated Transfer Scheme

To calculate the K-$\varepsilon$-stable PCs:

1. Start with a $U_0$
2. If $\sum u_i > \sum V(C)$ Then use the n-correction of Wu(1977)
3. Calculate the demand functions with respect to $U_i$
4. Find the greatest $d_{ij}$
Truncated Transfer Scheme

- To calculate the K-\(\varepsilon\)-stable PCs:
  1. Start with a \(U_0\)
  2. If \(\sum u_i > \sum V(C)\) Then use the n-correction of Wu(1977)
  3. Calculate the demand functions with respect to \(U_i\)
  4. Find the greatest \(d_{ij}\)
  5. Pass part \(\alpha, 0 < \alpha \leq d_{ij}\) of \(U_i\) of one agent to another agents
Truncated Transfer Scheme

To calculate the K-$\varepsilon$-stable PCs:

1. Start with a $U_0$
2. If $\sum u_i > \sum V(C)$ then use the n-correction of Wu(1977)
3. Calculate the demand functions with respect to $U_i$
4. Find the greatest $d_{ij}$
5. Pass part $\alpha, 0 < \alpha \leq d_{ij}$ of $U_i$ of one agent to another agent
6. Form $U_{i+1}$
Truncated Transfer Scheme

To calculate the $K$-$\varepsilon$-stable PCs:

1. Start with a $U_0$
2. If $\sum u_i > \sum V(C)$ Then use the $n$-correction of Wu(1977)
3. Calculate the demand functions with respect to $U_i$
4. Find the greatest $d_{ij}$
5. Pass part $\alpha, 0 < \alpha \leq d_{ij}$ of $U_i$ of one agent to another agents
6. Form $U_{i+1}$
7. If $e_r \leq \varepsilon$, Then stop and return $U_{i+1}$ as the result
Truncated Transfer Scheme

- To calculate the K-ε-stable PCs:
  1. Start with a $U_0$
  2. If $\sum u_i > \sum V(C)$ Then use the n-correction of Wu(1977)
  3. Calculate the demand functions with respect to $U_i$
  4. Find the greatest $d_{ij}$
  5. Pass part $\alpha, 0 < \alpha \leq d_{ij}$ of $U_i$ of one agent to another agents
  6. Form $U_{i+1}$
  7. If $e_r \leq \varepsilon$, Then stop and return $U_{i+1}$ as the result
  8. If not, do next iteration
Truncated Transfer Scheme

- **To calculate the K-$\varepsilon$-stable PCs:**
  1. Start with a $U_0$
  2. If $\sum u_i > \sum V(C)$ Then use the n-correction of Wu(1977)
  3. Calculate the demand functions with respect to $U_i$
  4. Find the greatest $d_{ij}$
  5. Pass part $\alpha, 0 < \alpha \leq d_{ij}$ of $U_i$ of one agent to another agents
  6. Form $U_{i+1}$
  7. If $e_r \leq \varepsilon$, Then stop and return $U_{i+1}$ as the result
  8. If not, do next iteration

- complexity of the computation of the K-$\varepsilon$-stable and *Pareto optimal* PCs is $O(n2^n)$
Truncated Transfer Scheme

- To calculate the K-ε-stable PCs:
  1. Start with a $U_0$
  2. If $\sum u_i > \sum V(C)$ Then use the n-correction of Wu(1977)
  3. Calculate the demand functions with respect to $U_i$
  4. Find the greatest $d_{ij}$
  5. Pass part $\alpha, 0 < \alpha \leq d_{ij}$ of $U_i$ of one agent to another agents
  6. Form $U_{i+1}$
  7. If $e_r \leq \varepsilon$, Then stop and return $U_{i+1}$ as the result
  8. If not, do next iteration

- complexity of the computation of the K-ε-stable and Pareto optimal PCs is $O(n2^n)$
- Thus, DEK-CFM has $O(n2^n n^n)$
The Negotiation-oriented CFM

- Distributed, Negotiation-based, Polynomial, Kernel-oriented Coalition-Formation Model (*DNPK-CFM*) is a reduced-cost CFM based on negotiation.
The Negotiation-oriented CFM

- Distributed, Negotiation-based, Polynomial, Kernel-oriented Coalition-Formation Model (*DNPK-CFM*) is a reduced-cost CFM based on negotiation.
- It is an anytime Algorithm due to reaching a steady state:
  1. the agents have reached a **K-stable** and **Pareto optimal** PC, or
  2. the agents have not reached a **PC** as in 1, but have no more possible beneficial proposals (allowed by the protocols) to be transmitted to others.
The Negotiation-oriented CFM

- Distributed, Negotiation-based, Polynomial, Kernel-oriented Coalition-Formation Model (DNPK-CFM) is a reduced-cost CFM based on negotiation.
- It is an anytime Algorithm due to reaching a steady state:
  1. the agents have reached a K-stable and Pareto optimal PC, or
  2. the agents have not reached a PC as in 1, but have no more possible beneficial proposals (allowed by the protocols) to be transmitted to others.
- protocols must be agreed on that will direct the agents to a well-defined polynomial set of coalitions.
  - thus, only coalitions of sizes in the ranges $[K_1; K_2]$ are allowed to be considered for excess calculations.
The Negotiation-oriented CFM

- Preliminary Stage: Prior to negotiation, the agents must calculate the values of coalitions in the range of sizes $K_1$ to $K_2$, using the calculation methods of the DEK-CFM.
The Negotiation-oriented CFM

- Preliminary Stage: Prior to negotiation, the agents must calculate the values of coalitions in the range of sizes $K_1$ to $K_2$, using the calculation methods of the DEK-CFM.
- First Stage:
  1. agents receive proposals as a member of a coalition
  2. coalitions coordinate their actions either via a representative or by voting (or both)
  3. coalitions perform iteratively as follow:
     - transmit a proposal to a target coalition; wait for responses
     - accept $P_{rp}$ only if $P_{rp} = P_{pr}$
     - if $P_{rp}$ was accepted and mutually confirmed, form $C_{r+p}$; if necessary, choose the representative.
     - send acceptance of $P_{rp}$ to other coalitions and reject other proposals.
The Negotiation-oriented CFM

- **First Stage:**
  1. The above sequence should be repeated until a steady state is reached, or when the time-period ends.
  2. Announce the status (if there are any more proposals to transmit)
The Negotiation-oriented CFM

- First Stage:
  4 The above sequence should be repeated until a steady state is reached, or when the time-period ends.
  5 Announce the status (if there are any more proposals to transmit)

- Second (optional) Stage:
The Negotiation-oriented CFM

- **First Stage:**
  4. The above sequence should be repeated until a steady state is reached, or when the time-period ends.
  5. Announce the status (if there are any more proposals to transmit)

- **Second (optional) Stage:**
  6. Following the same sequence of steps in the first stage, proposals that involve destruction are allowed. (Proposals addressed to single agents)
The Negotiation-oriented CFM

- **First Stage:**
  1. The above sequence should be repeated until a steady state is reached, or when the time-period ends.
  2. Announce the status (if there are any more proposals to transmit)

- **Second (optional) Stage:**
  6. Following the same sequence of steps in the first stage, proposals that involve destruction are allowed. (Proposals addressed to single agents)
  7. Agents can leave their coalitions due to changes of the coalition’s payoff vectors. Thus, these coalitions will destruct.
The Negotiation-oriented CFM

- First Stage:
  4. The above sequence should be repeated until a steady state is reached, or when the time-period ends.
  5. Announce the status (if there are any more proposals to transmit)

- Second (optional) Stage:
  6. Following the same sequence of steps in the first stage, proposals that involve destruction are allowed. (Proposals addressed to single agents)
  7. Agents can leave their coalitions due to changes of the coalition’s payoff vectors. Thus, these coalitions will destruct.
  8. If a steady state is reached or time ends, stop the iteration
The Negotiation-oriented CFM

- **First Stage:**
  4. The above sequence should be repeated until a steady state is reached, or when the time-period ends.
  5. Announce the status (if there are any more proposals to transmit)

- **Second (optional) Stage:**
  6. Following the same sequence of steps in the first stage, proposals that involve destruction are allowed. (Proposals addressed to single agents)
  7. Agents can leave their coalitions due to changes of the coalition’s payoff vectors. Thus, these coalitions will destruct.
  8. If a steady state is reached or time ends, stop the iteration
  9. Announce the status
DNPK-CFM Protocol

- DNPK-CFM is enforceable because deviation from it is revealable.
DNPK-CFM Protocol

- DNPK-CFM is enforceable because deviation from it is revealable.
- Complexity:
  \[ n_{coalitions} = \sum_{i=K_1}^{K_2} \frac{n!}{i!(n-i)!} \]
  - preliminary stage: \( O(n^2 \times n_{coalitions}) \)
DNPK-CFM Protocol

- DNPK-CFM is enforceable because deviation from it is revealable.
- Complexity:
  \[ n_{coalitions} = \sum_{i=K_1}^{K_2} \frac{n!}{i!(n-i)!} \]

  - preliminary stage: \( O(n^2 \times n_{coalitions}) \)
  - computation: \( O(n^6 \times n_{coalitions}) \) in case of less bounded time, and \( O(n^3 \times n_{coalitions}) \) in case of strictly bounded time
DNPK-CFM Protocol

- DNPK-CFM is enforceable because deviation from it is revealable.
- Complexity:

\[ n_{\text{coalitions}} = \sum_{i=K_1}^{K_2} \frac{n!}{i!(n-i)!} \]

- preliminary stage: \( O(n^2 \times n_{\text{coalitions}}) \)
- computation: \( O(n^6 \times n_{\text{coalitions}}) \) in case of less bounded time, and \( O(n^3 \times n_{\text{coalitions}}) \) in case of strictly bounded time
- communication: \( O(n^2 \times n_{\text{coalitions}}) \)
DNPK-CFM Protocol

- DNPK-CFM is enforceable because deviation from it is revealable.
- Complexity:
  \[ n_{\text{coalitions}} = \sum_{i=K_1}^{K_2} \frac{n!}{i!(n-i)!} \]
  - preliminary stage: \( O(n^2 \times n_{\text{coalitions}}) \)
  - computation: \( O(n^6 \times n_{\text{coalitions}}) \) in case of less bounded time, and \( O(n^3 \times n_{\text{coalitions}}) \) in case of strictly bounded time
  - communication: \( O(n^2 \times n_{\text{coalitions}}) \)
  - thus, the upper limit is of order \( O(n^n) \)
Conclusion

- Using the KCA algorithm one can achieve kernel stable solutions
Conclusion

- Using the KCA algorithm one can achieve kernel stable solutions
  - Safe with respect to privacy
Conclusion

- Using the KCA algorithm one can achieve kernel stable solutions
  - Safe with respect to privacy
  - Safe with respect to incomplete information
  - Not safe with respect to changing agent sets

- Fraud is possible, however this is impractical because of computational complexity
- By modifying KCA two approaches can be achieved
  - DEK-CFM: pareto optimal and k-stable coalition
  - DNPK-CFM: polynomial and k-\(\varepsilon\)-stable
- These algorithms are enforceable and thus deviations are revealable
Conclusion

- Using the KCA algorithm one can achieve kernel stable solutions
  - Safe with respect to privacy
  - Safe with respect to incomplete information
  - Not safe with respect to changing agent sets
  - Fraud is possible, however this is impractical because of computational complexity
Conclusion

- Using the KCA algorithm one can achieve kernel stable solutions
  - Safe with respect to privacy
  - Safe with respect to incomplete information
  - Not safe with respect to changing agent sets
  - Fraud is possible, however this is impractical because of computational complexity
- By modifying KCA two approaches can be achieved
Conclusion

- Using the KCA algorithm one can achieve kernel stable solutions
  - Safe with respect to privacy
  - Safe with respect to incomplete information
  - Not safe with respect to changing agent sets
  - Fraud is possible, however this is impractical because of computational complexity
- By modifying KCA two approaches can be achieved
  - DEK-CFM: pareto optimal and k-stable coalition
Conclusion

- Using the KCA algorithm one can achieve kernel stable solutions
  - Safe with respect to privacy
  - Safe with respect to incomplete information
  - Not safe with respect to changing agent sets
  - Fraud is possible, however this is impractical because of computational complexity
- By modifying KCA two approaches can be achieved
  - DEK-CFM: pareto optimal and k-stable coalition
  - DNPK-CFM: polynomial and k-ε-stable
Conclusion

- Using the KCA algorithm one can achieve kernel stable solutions
  - Safe with respect to privacy
  - Safe with respect to incomplete information
  - Not safe with respect to changing agent sets
  - Fraud is possible, however this is impractical because of computational complexity
- By modifying KCA two approaches can be achieved
  - DEK-CFM: pareto optimal and k-stable coalition
  - DNPK-CFM: polynomial and k-ε-stable
  - These algorithms are enforceable and thus deviations are revealable