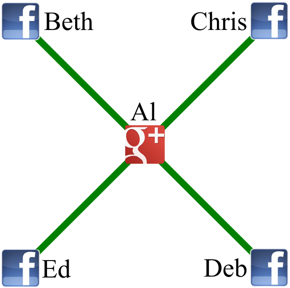
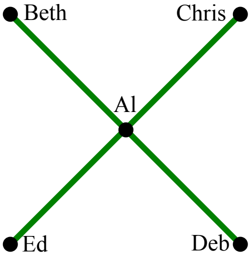
Computing Shapley Value in Supermodular Coalitional Games

David Liben-Nowell (CS, Carleton College),
Alexa Sharp (CS, Oberlin College),
Tom Wexler (CS, Oberlin College),
Kevin Woods (Math, Oberlin College).



A Game



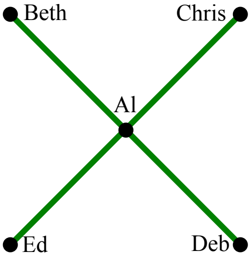
Players are in a social network (edges = friends).

Deciding whether to adopt a social technology (cell phone plan, Google+).

Each player has a cost, c, to adopt.

Each player gets benefit, b, for each friend who has also adopted.

A Game



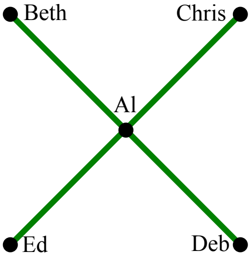
Players are in a social network (edges = friends).

Deciding whether to adopt a social technology (cell phone plan, Google+).

Each player has a cost, c, to adopt.

Each player gets benefit, b, for each friend who has also adopted.

A Game



Players are in a social network (edges = friends).

Deciding whether to adopt a social technology (cell phone plan, Google+).

Each player has a cost, c, to adopt.

Each player gets benefit, b, for each friend who has also adopted.

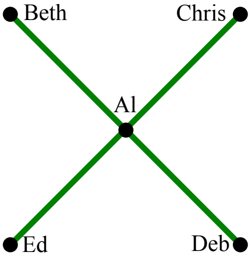
A Game

Example: $b = 9$, $c = 7$.

All players want to adopt.

Leaf players have utility $b - c = 2$.
Al has utility $4b - c = 29$.

total surplus: $8b - 5c = 37$.



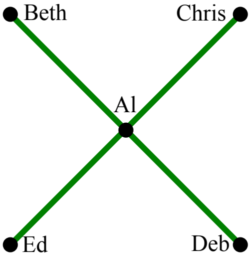
A Game

Example: $b = 9$, $c = 7$.

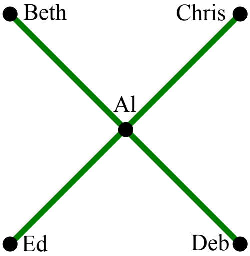
All players want to adopt.

Leaf players have utility $b - c = 2$.
Al has utility $4b - c = 29$.

total surplus: $8b - 5c = 37$.



A Game



Example: $b = 9$, $c = 7$.

All players want to adopt.

Leaf players have utility $b - c = 2$.
Al has utility $4b - c = 29$.

total surplus: $8b - 5c = 37$.

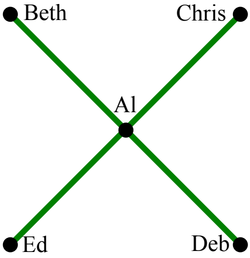
A Game

Example: $b = 9$, $c = 7$.

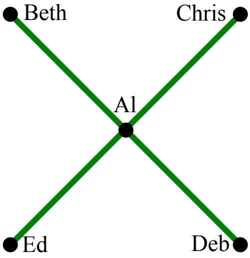
All players want to adopt.

Leaf players have utility $b - c = 2$.
Al has utility $4b - c = 29$.

total surplus: $8b - 5c = 37$.



A Game



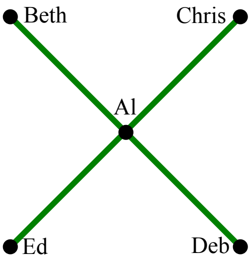
Example: $b = 9$, $c = 10$.

Leaf players do not want to adopt.

But total surplus is: $8b - 5c = 22$.

Al should pay the other players to entice them.

A Game



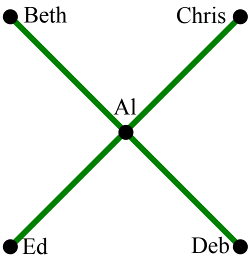
Example: $b = 9$, $c = 10$.

Leaf players do not want to adopt.

But total surplus is: $8b - 5c = 22$.

Al should pay the other players to entice them.

A Game



Example: $b = 9$, $c = 10$.

Leaf players do not want to adopt.

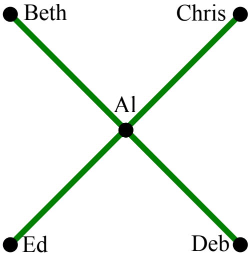
But total surplus is: $8b - 5c = 22$.

Al should pay the other players to entice them.

How much?

How should the total surplus be divided?

Coalitional Games

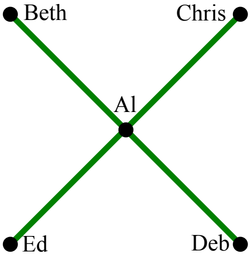


Definition: For any subset, S, of players, associate a value, $v(S)$.

In Example:
$v(ABCDE) = 22$,
$v(ABCD) = 14$,
$v(ABC) = 6$,
$v(BCDE) = 0$,
$v(AB) = 0$,
$v(\emptyset) = 0$, by convention.

That is, $v(S)$ is the maximum total surplus if some subset of S adopts.

Coalitional Games

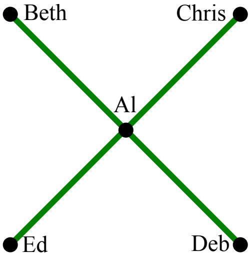


Definition: For any subset, S, of players, associate a value, $v(S)$.

In Example:
$v(ABCDE) = 22$,
$v(ABCD) = 14$,
$v(ABC) = 6$,
$v(BCDE) = 0$,
$v(AB) = 0$,
$v(\emptyset) = 0$, by convention.

That is, $v(S)$ is the maximum total surplus if some subset of S adopts.

Coalitional Games



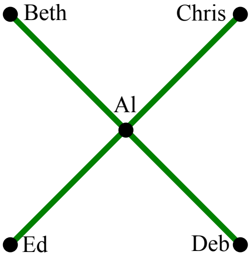
Definition: For any subset, S, of players, associate a value, $v(S)$.

In Example:

$v(ABCDE) = 22$,
$v(ABCD) = 14$,
$v(ABC) = 6$,
$v(BCDE) = 0$,
$v(AB) = 0$,
$v(\emptyset) = 0$, by convention.

That is, $v(S)$ is the maximum total surplus if some subset of S adopts.

Coalitional Games



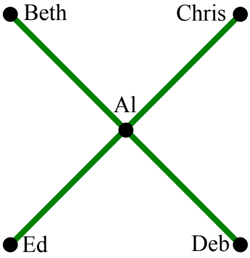
Definition: For any subset, S, of players, associate a value, $v(S)$.

In Example:

$v(ABCDE) = 22$,
$v(ABCD) = 14$,
$v(ABC) = 6$,
$v(BCDE) = 0$,
$v(AB) = 0$,
$v(\emptyset) = 0$, by convention.

That is, $v(S)$ is the maximum total surplus if some subset of S adopts.

Coalitional Games



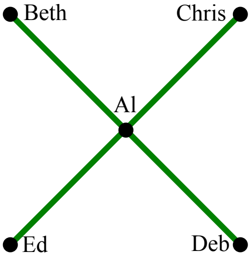
Definition: For any subset, S, of players, associate a value, $v(S)$.

In Example:

$v(ABCDE) = 22$,
$v(ABCD) = 14$,
$v(ABC) = 6$,
$v(BCDE) = 0$,
$v(AB) = 0$,
$v(\emptyset) = 0$, by convention.

That is, $v(S)$ is the maximum total surplus if some subset of S adopts.

Coalitional Games



Definition: For any subset, S, of players, associate a value, $v(S)$.

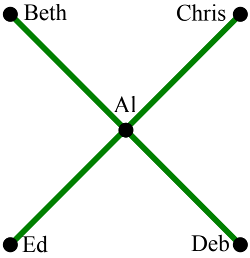
In Example:
$v(ABCDE) = 22$,
$v(ABCD) = 14$,
$v(ABC) = 6$,
$v(BCDE) = 0$,
$v(AB) = 0$,
$v(\emptyset) = 0$, by convention.

That is, $v(S)$ is the maximum total surplus if some subset of S adopts.

Coalitional Games

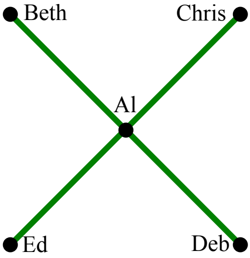
Definition: For any subset, S, of players, associate a value, $v(S)$.

In Example:
$v(ABCDE) = 22$,
$v(ABCD) = 14$,
$v(ABC) = 6$,
$v(BCDE) = 0$,
$v(AB) = 0$,
$v(\emptyset) = 0$, by convention.



That is, $v(S)$ is the maximum total surplus if some subset of S adopts.

Coalitional Games



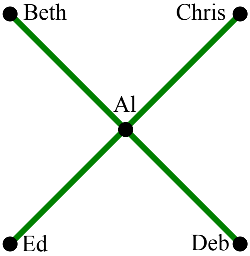
Definition: For any subset, S, of players, associate a value, $v(S)$.

In Example:

$v(ABCDE) = 22$,
$v(ABCD) = 14$,
$v(ABC) = 6$,
$v(BCDE) = 0$,
$v(AB) = 0$,
$v(\emptyset) = 0$, by convention.

That is, $v(S)$ is the maximum total surplus if some subset of S adopts.

Shapley Value

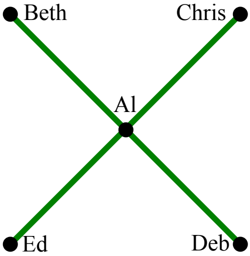


Imagine players arrive in some order, σ.

Player receives marginal contribution:
what he adds to players already present.

Examples:
$\sigma = ABCDE$. m.c. of Chris: $v(ABC) - v(AB) = 6 - 0 = 6$.
$\sigma = ABDCE$. m.c. of Chris: $v(ABDC) - v(ABD) = 14 - 6 = 8$.

Shapley Value



Imagine players arrive in some order, σ.

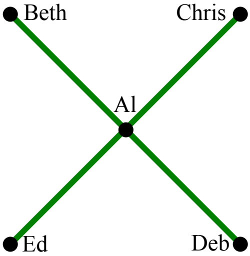
Player receives marginal contribution: what he adds to players already present.

Examples:

$\sigma = ABCDE$. m.c. of Chris: $v(ABC) - v(AB) = 6 - 0 = 6$.

$\sigma = ABDCE$. m.c. of Chris: $v(ABDC) - v(ABD) = 14 - 6 = 8$.

Shapley Value



Imagine players arrive in some order, σ.

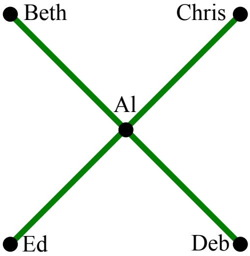
Player receives marginal contribution:
what he adds to players already present.

Examples:

$\sigma = ABCDE$. m.c. of Chris: $v(ABC) - v(AB) = 6 - 0 = 6$.

$\sigma = ABDCE$. m.c. of Chris: $v(ABDC) - v(ABD) = 14 - 6 = 8$.

Shapley Value



Imagine players arrive in some order, σ.

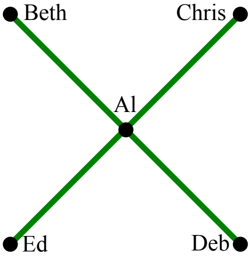
Player receives marginal contribution:
what he adds to players already present.

Examples:
$\sigma = ABCDE$. m.c. of Chris: $v(ABC) - v(AB) = 6 - 0 = 6$.
$\sigma = ABDCE$. m.c. of Chris: $v(ABDC) - v(ABD) = 14 - 6 = 8$.

Shapley Value for player i: the average, over all permutations, of the marginal contribution of player i.

In example, Al gets 8.4, leaf players get 3.4.
Can we efficiently compute Shapley values, in general?

Shapley Value



Imagine players arrive in some order, σ.

Player receives marginal contribution:
what he adds to players already present.

Examples:
$\sigma = ABCDE$. m.c. of Chris: $v(ABC) - v(AB) = 6 - 0 = 6$.
$\sigma = ABDCE$. m.c. of Chris: $v(ABDC) - v(ABD) = 14 - 6 = 8$.

Shapley Value for player i: the average, over all permutations, of the marginal contribution of player i.

In example, Al gets 8.4, leaf players get 3.4.
Can we efficiently compute Shapley values, in general?

Supermodular Games

Definition: For all S, T, $v(S \cup T) + v(S \cap T) \geq v(S) + v(T)$.

or: For all $S \subseteq T$, for all $i \notin T$,

$$v(T \cup \{i\}) - v(T) \geq v(S \cup \{i\}) - v(S).$$

Increasing marginal contributions.

Examples:

- Our game.
- Multicast tree game: building a path to a source.
- Anything with Bandwagon effect.

Given that a game is supermodular, can we efficiently compute its Shapley values?

Supermodular Games

Definition: For all S, T, $v(S \cup T) + v(S \cap T) \geq v(S) + v(T)$.
or: For all $S \subseteq T$, for all $i \notin T$,
$v(T \cup \{i\}) - v(T) \geq v(S \cup \{i\}) - v(S)$.

Increasing marginal contributions.

Examples:

- Our game.
- Multicast tree game: building a path to a source.
- Anything with Bandwagon effect.

Given that a game is supermodular, can we efficiently compute its Shapley values?

Supermodular Games

Definition: For all S, T, $v(S \cup T) + v(S \cap T) \geq v(S) + v(T)$.
or: For all $S \subseteq T$, for all $i \notin T$,
$$v(T \cup \{i\}) - v(T) \geq v(S \cup \{i\}) - v(S).$$

Increasing marginal contributions.

Examples:

- Our game.
- Multicast tree game: building a path to a source.
- Anything with Bandwagon effect.

Given that a game is supermodular, can we efficiently compute its Shapley values?

Supermodular Games

Definition: For all S, T, $v(S \cup T) + v(S \cap T) \geq v(S) + v(T)$.
or: For all $S \subseteq T$, for all $i \notin T$,
$$v(T \cup \{i\}) - v(T) \geq v(S \cup \{i\}) - v(S).$$

Increasing marginal contributions.

Examples:

- Our game.
- Multicast tree game: building a path to a source.
- Anything with Bandwagon effect.

Given that a game is supermodular, can we efficiently compute its Shapley values?

Supermodular Games

Definition: For all S, T, $v(S \cup T) + v(S \cap T) \geq v(S) + v(T)$.
or: For all $S \subseteq T$, for all $i \notin T$,
$$v(T \cup \{i\}) - v(T) \geq v(S \cup \{i\}) - v(S).$$

Increasing marginal contributions.

Examples:

- Our game.
- Multicast tree game: building a path to a source.
- Anything with Bandwagon effect.

Given that a game is supermodular, can we efficiently compute its Shapley values?

Supermodular Games

Definition: For all S, T, $v(S \cup T) + v(S \cap T) \geq v(S) + v(T)$.
or: For all $S \subseteq T$, for all $i \notin T$,
$v(T \cup \{i\}) - v(T) \geq v(S \cup \{i\}) - v(S)$.

Increasing marginal contributions.

Examples:

- Our game.
- Multicast tree game: building a path to a source.
- Anything with Bandwagon effect.

Given that a game is supermodular, can we efficiently compute its Shapley values?

Shapley Value for Supermodular Games

Assume oracle access to $v(\cdot)$.

Bad Example: Let $\mathcal{C}$ be a collection of subsets of $\{1, \ldots, n\}$, each of cardinality $n/2$.

$$v_{\mathcal{C}}(A) = \begin{cases} 0 & \text{if } |A| < n/2, \\ 0 & \text{if } |A| = n/2 \text{ and } A \notin \mathcal{C}, \\ 1 & \text{if } |A| = n/2 \text{ and } A \in \mathcal{C}, \\ 2|A| - n & \text{if } |A| > n/2. \end{cases}$$

Supermodular.

Shapley Value for Supermodular Games

Assume oracle access to $v(\cdot)$.

Bad Example: Let $\mathcal{C}$ be a collection of subsets of $\{1, \ldots, n\}$, each of cardinality $n/2$.

$$v_{\mathcal{C}}(A) = \begin{cases} 0 & \text{if } |A| < n/2, \\ 0 & \text{if } |A| = n/2 \text{ and } A \notin \mathcal{C}, \\ 1 & \text{if } |A| = n/2 \text{ and } A \in \mathcal{C}, \\ 2|A| - n & \text{if } |A| > n/2. \end{cases}$$

Supermodular.

Shapley Value for Supermodular Games

Assume oracle access to $v(\cdot)$.

Bad Example: Let $\mathcal{C}$ be a collection of subsets of $\{1, \ldots, n\}$, each of cardinality $n/2$.

$$v_{\mathcal{C}}(A) = \begin{cases} 0 & \text{if } |A| < n/2, \\ 0 & \text{if } |A| = n/2 \text{ and } A \notin \mathcal{C}, \\ 1 & \text{if } |A| = n/2 \text{ and } A \in \mathcal{C}, \\ 2|A| - n & \text{if } |A| > n/2. \end{cases}$$

Supermodular.

Shapley Value for Supermodular Games

Assume oracle access to $v(\cdot)$.

Bad Example: Let $\mathcal{C}$ be a collection of subsets of $\{1, \ldots, n\}$, each of cardinality $n/2$.

$$v_{\mathcal{C}}(A) = \begin{cases} 0 & \text{if } |A| < n/2, \\ 0 & \text{if } |A| = n/2 \text{ and } A \notin \mathcal{C}, \\ 1 & \text{if } |A| = n/2 \text{ and } A \in \mathcal{C}, \\ 2|A| - n & \text{if } |A| > n/2. \end{cases}$$

Supermodular.

Shapley Value for Supermodular Games

Assume oracle access to $v(\cdot)$.

Bad Example: Let $\mathcal{C}$ be a collection of subsets of $\{1, \ldots, n\}$, each of cardinality $n/2$.

$$v_{\mathcal{C}}(A) = \begin{cases} 0 & \text{if } |A| < n/2, \\ 0 & \text{if } |A| = n/2 \text{ and } A \notin \mathcal{C}, \\ 1 & \text{if } |A| = n/2 \text{ and } A \in \mathcal{C}, \\ 2|A| - n & \text{if } |A| > n/2. \end{cases}$$

Supermodular.

Shapley Value for Supermodular Games

Assume oracle access to $v(\cdot)$.

Bad Example: Let $\mathcal{C}$ be a collection of subsets of $\{1, \ldots, n\}$, each of cardinality $n/2$.

$$v_{\mathcal{C}}(A) = \begin{cases} 0 & \text{if } |A| < n/2, \\ 0 & \text{if } |A| = n/2 \text{ and } A \notin \mathcal{C}, \\ 1 & \text{if } |A| = n/2 \text{ and } A \in \mathcal{C}, \\ 2|A| - n & \text{if } |A| > n/2. \end{cases}$$

Supermodular.

Shapley Value for Supermodular Games

Assume oracle access to $v(\cdot)$.

Bad Example: Let $\mathcal{C}$ be a collection of subsets of $\{1, \ldots, n\}$, each of cardinality $n/2$.

$$v_{\mathcal{C}}(A) = \begin{cases} 0 & \text{if } |A| < n/2, \\ 0 & \text{if } |A| = n/2 \text{ and } A \notin \mathcal{C}, \\ 1 & \text{if } |A| = n/2 \text{ and } A \in \mathcal{C}, \\ 2|A| - n & \text{if } |A| > n/2. \end{cases}$$

Supermodular.

Shapley Value for Supermodular Games

Assume oracle access to $v(\cdot)$.

Bad Example: Let $\mathcal{C}$ be a collection of subsets of $\{1, \ldots, n\}$, each of cardinality $n/2$.

$$v_{\mathcal{C}}(A) = \begin{cases} 0 & \text{if } |A| < n/2, \\ 0 & \text{if } |A| = n/2 \text{ and } A \notin \mathcal{C}, \\ 1 & \text{if } |A| = n/2 \text{ and } A \in \mathcal{C}, \\ 2|A| - n & \text{if } |A| > n/2. \end{cases}$$

Supermodular.
Slightly different $\mathcal{C}$'s yield slightly different Shapley values.
Exact computation needs $> \binom{n}{n/2}$ oracle calls.

Too much wiggle room.

Shapley Value for Supermodular Games

Assume oracle access to $v(\cdot)$.

Bad Example: Let $\mathcal{C}$ be a collection of subsets of $\{1, \ldots, n\}$, each of cardinality $n/2$.

$$v_{\mathcal{C}}(A) = \begin{cases} 0 & \text{if } |A| < n/2, \\ 0 & \text{if } |A| = n/2 \text{ and } A \notin \mathcal{C}, \\ 1 & \text{if } |A| = n/2 \text{ and } A \in \mathcal{C}, \\ 2|A| - n & \text{if } |A| > n/2. \end{cases}$$

Supermodular.
Slightly different $\mathcal{C}$'s yield slightly different Shapley values.
Exact computation needs $> \binom{n}{n/2}$ oracle calls.

Too much wiggle room. How about approximating?

Shapley Value for Supermodular Games

Probabilistic Algorithm: For some m, choose m permutations, uniformly at random, and average the marginal contributions of a player.

Theorem: For supermodular games, oracle access, this gives a fully polynomial-time randomized approximation scheme (FPRAS).

Shapley Value for Supermodular Games

Probabilistic Algorithm: For some m, choose m permutations, uniformly at random, and average the marginal contributions of a player.

Theorem: For supermodular games, oracle access, this gives a fully polynomial-time randomized approximation scheme (FPRAS).

Shapley Value for Supermodular Games

Probabilistic Algorithm: For some m, choose m permutations, uniformly at random, and average the marginal contributions of a player.

Theorem: For supermodular games, oracle access, this gives a fully polynomial-time randomized approximation scheme (FPRAS).

For n players, given $\epsilon > 0$, let $m = 4n(n-1)/\epsilon^2 \in \text{poly}(n, 1/\epsilon)$.

With probability $3/4$, the computed values of all players will be within a $1 \pm \epsilon$ multiplicative factor of the correct values.

To replace $3/4$ with $1 - \delta$, need $m \in \text{poly}(n, 1/\epsilon, \log(1/\delta))$.

Shapley Value for Supermodular Games

Probabilistic Algorithm: For some m, choose m permutations, uniformly at random, and average the marginal contributions of a player.

Theorem: For supermodular games, oracle access, this gives a fully polynomial-time randomized approximation scheme (FPRAS).

For n players, given $\epsilon > 0$, let $m = 4n(n-1)/\epsilon^2 \in \text{poly}(n, 1/\epsilon)$.

With probability 3/4, the computed values of all players will be within a $1 \pm \epsilon$ multiplicative factor of the correct values.

To replace 3/4 with $1 - \delta$, need $m \in \text{poly}(n, 1/\epsilon, \log(1/\delta))$.

Shapley Value for Supermodular Games

Probabilistic Algorithm: For some m, choose m permutations, uniformly at random, and average the marginal contributions of a player.

Theorem: For supermodular games, oracle access, this gives a fully polynomial-time randomized approximation scheme (FPRAS).

For n players, given $\epsilon > 0$, let $m = 4n(n-1)/\epsilon^2 \in \text{poly}(n, 1/\epsilon)$.

With probability $3/4$, the computed values of all players will be within a $1 \pm \epsilon$ multiplicative factor of the correct values.

To replace $3/4$ with $1 - \delta$, need $m \in \text{poly}(n, 1/\epsilon, \log(1/\delta))$.

Shapley Value for Supermodular Games

Key: Supermodularity implies that can't get huge values with tiny probability.

Indeed, largest marginal contribution is when player appears last, which happens with probability $1/n$.

Assumption: $v(\{i\}) \geq 0$, for all i.
Quick Fix: If want to fairly allocate gains from cooperation, game should have $v(\{i\}) = 0$, anyway.

Shapley Value for Supermodular Games

Key: Supermodularity implies that can't get huge values with tiny probability.

Indeed, largest marginal contribution is when player appears last, which happens with probability $1/n$.

Assumption: $v(\{i\}) \geq 0$, for all i.
Quick Fix: If want to fairly allocate gains from cooperation, game should have $v(\{i\}) = 0$, anyway.

Shapley Value for Supermodular Games

Key: Supermodularity implies that can't get huge values with tiny probability.

Indeed, largest marginal contribution is when player appears last, which happens with probability $1/n$.

Assumption: $v(\{i\}) \geq 0$, for all i.
Quick Fix: If want to fairly allocate gains from cooperation, game should have $v(\{i\}) = 0$, anyway.

Shapley Value for Supermodular Games

Key: Supermodularity implies that can't get huge values with tiny probability.

Indeed, largest marginal contribution is when player appears last, which happens with probability $1/n$.

Assumption: $v(\{i\}) \geq 0$, for all i.
Quick Fix: If want to fairly allocate gains from cooperation, game should have $v(\{i\}) = 0$, anyway.

Shapley Value for Supermodular Games

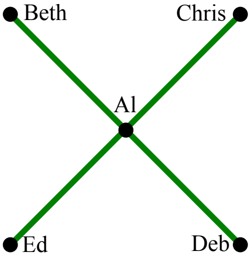
Theorem:

- No deterministic algorithm can do as well as poly$(n, 1/\epsilon)$.
- No other probabilistic algorithm can do better than poly$(n, 1/\epsilon)$.
- Doesn't depend on P $\neq$ NP.

Similar bad example as before shows these facts.

The Kernel

In example, suppose have decided to allocate 6 to Al and 4 to each leaf player.



Beth can threaten Al that she will "go alone",

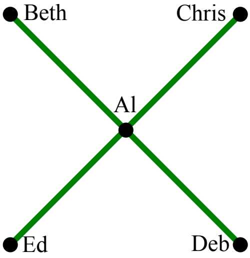
Cost of threat to Beth: $4 - 0 = 4$.

The Kernel

In example, suppose have decided to allocate 6 to Al and 4 to each leaf player.

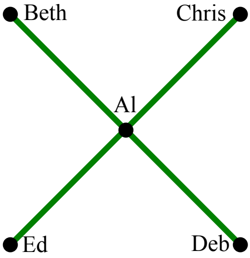
Beth can threaten Al that she will "go alone",

Cost of threat to Beth: $4 - 0 = 4$.



The Kernel

In example, suppose have decided to allocate 6 to Al and 4 to each leaf player.



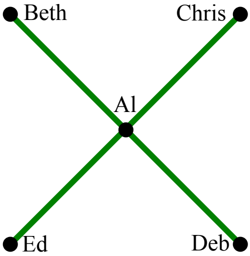
Beth can threaten Al that she will "go alone",

Cost of threat to Beth: $4 - 0 = 4$.

Al can threaten Beth that he will go alone, cost to him: 6.

The Kernel

In example, suppose have decided to allocate 6 to Al and 4 to each leaf player.



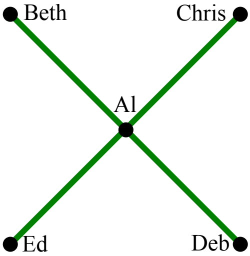
Beth can threaten Al that she will "go alone",

Cost of threat to Beth: $4 - 0 = 4$.

Or Al can threaten that he will only cooperate with C, D, and E.

The Kernel

In example, suppose have decided to allocate 6 to Al and 4 to each leaf player.



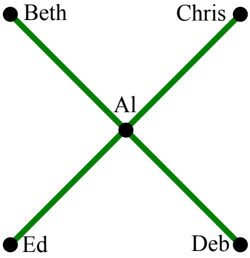
Beth can threaten Al that she will "go alone",

Cost of threat to Beth: $4 - 0 = 4$.

Or Al can threaten that he will only cooperate with C, D, and E.
Then $v(ACDE) = 14$, but Al must continue paying C, D, and E each 4, leaving 2 for himself.
Cost of threat to Al: $6 - 2 = 4$.

The Kernel

In example, suppose have decided to allocate 6 to Al and 4 to each leaf player.



Beth can threaten Al that she will "go alone",

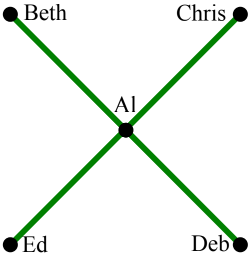
Cost of threat to Beth: $4 - 0 = 4$.

Or Al can threaten that he will only cooperate with C, D, and E. Then $v(ACDE) = 14$, but Al must continue paying C, D, and E each 4, leaving 2 for himself.

Cost of threat to Al: $6 - 2 = 4$.

The Kernel

In example, suppose have decided to allocate 6 to Al and 4 to each leaf player.



Beth can threaten Al that she will "go alone",

Cost of threat to Beth: $4 - 0 = 4$.

Or Al can threaten that he will only cooperate with C, D, and E. Then $v(ACDE) = 14$, but Al must continue paying C, D, and E each 4, leaving 2 for himself.
Cost of threat to Al: $6 - 2 = 4$.

Best threats are in equilibrium. Called the kernel.

The Kernel

Kernel exists and is unique for supermodular games [Shapley].

Kernel = Stable Outcome,
Shapley = Fair Outcome.

Unlike Shapley value, kernel of supermodular games can be exactly computed, in polynomial time [Kuipers].

Key Difference:

- Shapley value depends on the values of all 2^n subsets.
- Kernel only depends on the values of the $n(n-1)$ best threats of player i to player j.

The Kernel

Kernel exists and is unique for supermodular games [Shapley].

Kernel = Stable Outcome,
Shapley = Fair Outcome.

Unlike Shapley value, kernel of supermodular games can be exactly computed, in polynomial time [Kuipers].

Key Difference:

- Shapley value depends on the values of all 2^n subsets.
- Kernel only depends on the values of the $n(n-1)$ best threats of player i to player j.

The Kernel

Kernel exists and is unique for supermodular games [Shapley].

Kernel = Stable Outcome,
Shapley = Fair Outcome.

Unlike Shapley value, kernel of supermodular games can be exactly computed, in polynomial time [Kuipers].

Key Difference:

- Shapley value depends on the values of all 2^n subsets.
- Kernel only depends on the values of the $n(n-1)$ best threats of player i to player j.

The Kernel

Kernel exists and is unique for supermodular games [Shapley].

Kernel = Stable Outcome,
Shapley = Fair Outcome.

Unlike Shapley value, kernel of supermodular games can be exactly computed, in polynomial time [Kuipers].

Key Difference:

- Shapley value depends on the values of all 2^n subsets.
- Kernel only depends on the values of the $n(n-1)$ best threats of player i to player j.

The Kernel

Kernel exists and is unique for supermodular games [Shapley].

Kernel = Stable Outcome,
Shapley = Fair Outcome.

Unlike Shapley value, kernel of supermodular games can be exactly computed, in polynomial time [Kuipers].

Key Difference:

- Shapley value depends on the values of all 2^n subsets.
- Kernel only depends on the values of the $n(n-1)$ best threats of player i to player j.

Open Questions

- For specific supermodular games, like our example, can the Shapley value be computed efficiently?
- For our example game, how is the Shapley value related to the structure of the graph?

Open Questions

- For specific supermodular games, like our example, can the Shapley value be computed efficiently?
- For our example game, how is the Shapley value related to the structure of the graph?

Thank You!