Autonomous Agents and Multiagent Systems

Coalition Formation

Francesco Amigoni

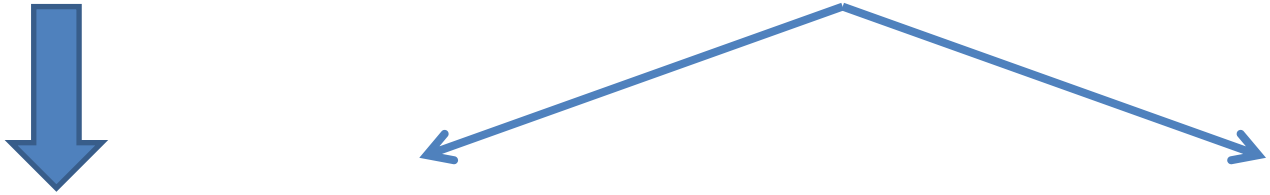
Many of the following slides are taken from the “lecture slides provided by authors”, Chapter 8, available at: http://www.the-mas-book.info/index.html

Chapter 8: Computational Coalition Formation

Edith Elkind
(Nanyang Technological University, Singapore)
Talal Rahwan, Nicholas R. Jennings
(University of Southampton, UK)

Cooperative Games

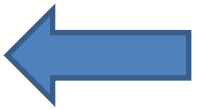
- Cooperative games model scenarios, where
  - agents can benefit by cooperating
  - binding agreements are possible
- In cooperative games, actions are taken by groups of agents



Transferable utility games: payoffs are given to the group and then divided among its members

Non-transferable utility games: group actions result in payoffs to individual group members

Characteristic Function Games vs. Partition Function Games

- In general TU games, the payoff obtained by a coalition depends on the actions chosen by other coalitions
  - these games are also known as partition function games (PFG)
- Characteristic function games (CFG):  the payoff of each coalition only depends on the action of that coalition
  - in such games, each coalition can be identified with the profit it obtains by choosing its best action

Transferable Utility Games Formalized

- A transferable utility game is a pair (N, v), where:
  - $N = \{1, \ldots, n\}$ is the set of players
  - $v: 2^N \rightarrow \mathbb{R}$ is the characteristic function
    - for each subset of players C, $v(C)$ is the amount that the members of C can earn by working together
  - usually it is assumed that v is
    - normalized: $v(\emptyset) = 0$
    - non-negative: $v(C) \geq 0$ for any $C \subseteq N$
    - monotone: $v(C) \leq v(D)$ for any C, D such that $C \subseteq D$
- A coalition is any subset of N;
  N itself is called the grand coalition

Ice-Cream Game: Characteristic Function


C: \$6,

M: \$4,

P: \$3

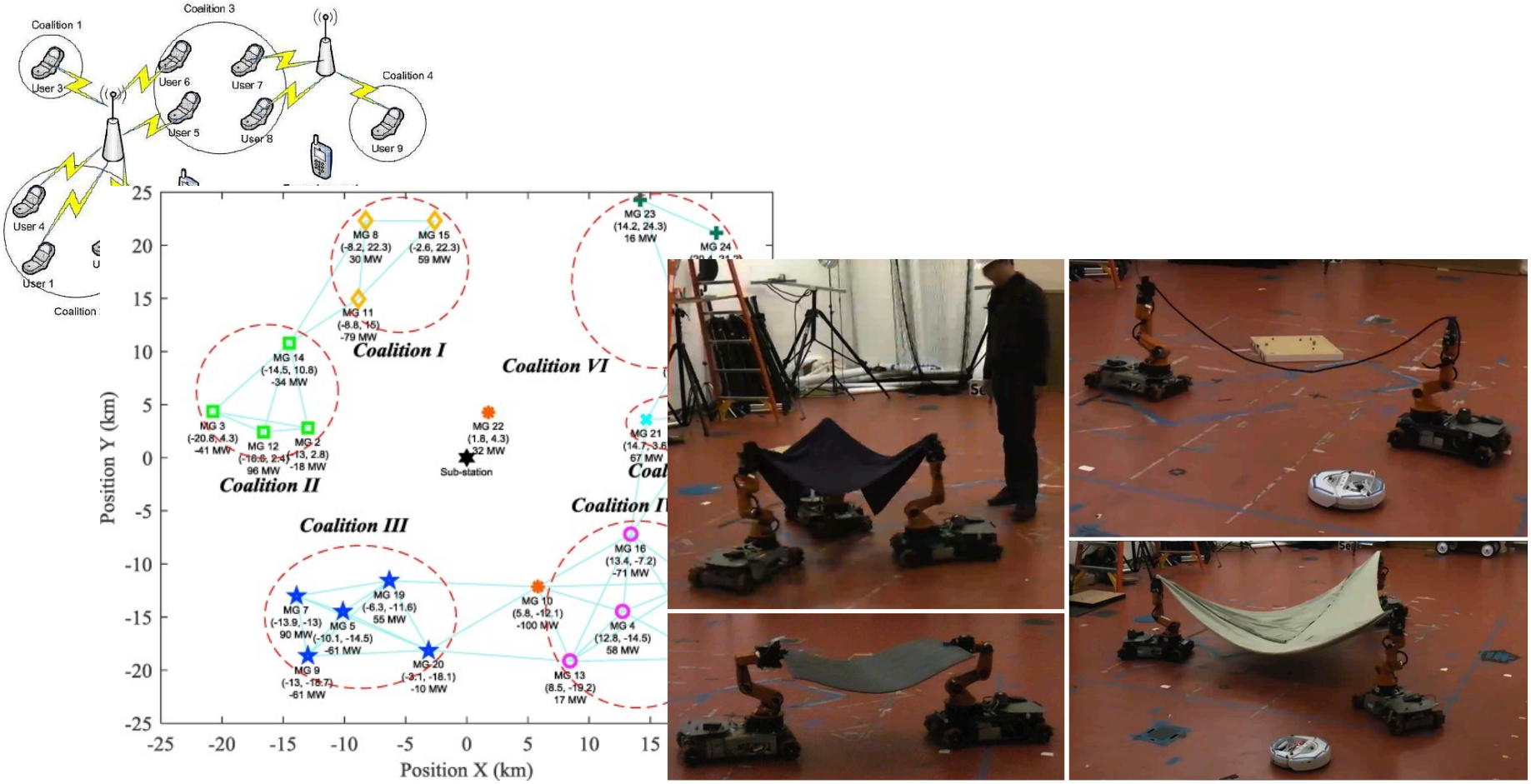
w = 500
p = \$7

w = 750
p = \$9

w = 1000
p = \$11

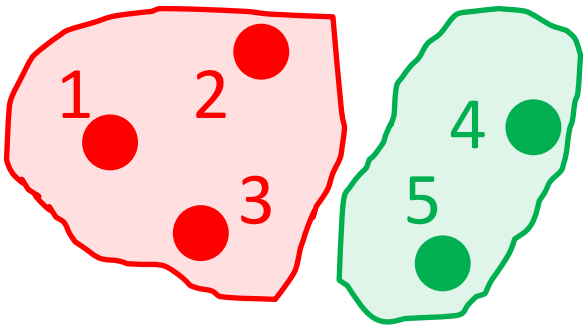
- $v(\emptyset) = v(\{C\}) = v(\{M\}) = v(\{P\}) = 0$
- $v(\{C, M\}) = 750$, $v(\{C, P\}) = 750$, $v(\{M, P\}) = 500$
- $v(\{C, M, P\}) = 1000$

Real-world examples



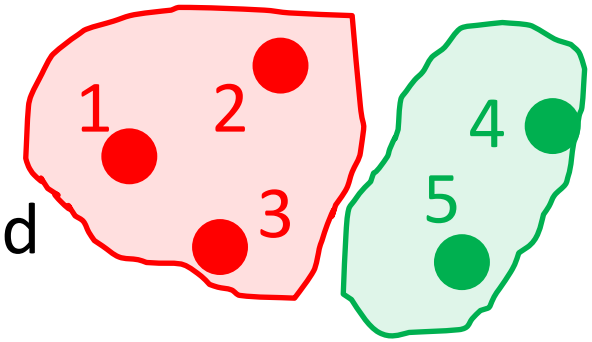
Transferable Utility Games: Outcome

- An outcome of a TU game G =(N, v) is a pair (CS, **x**), where:
  - CS =(C_1, ..., C_k) is a coalition structure, i.e., partition of N into coalitions:
    - $\cup_i C_i = N$, $C_i \cap C_j = \emptyset$ for $i \neq j$
  - **x** = (x_1, ..., x_n) is a payoff vector, which distributes the value of each coalition in CS:
    - $x_i \geq 0$ for all $i \in N$
    - $\Sigma_{i \in C} x_i = v(C)$ for each C is CS



Transferable Utility Games: Outcome

- Example:
  - suppose v({1, 2, 3}) = 9, v({4, 5}) = 4
  - then (({1, 2, 3}, {4, 5}), (3, 3, 3, 3, 1)) is an outcome
  - (({1, 2, 3}, {4, 5}), (2, 3, 2, 3, 3)) is NOT an outcome: transfers between coalitions are not allowed



- An outcome (CS, **x**) is called an imputation if it satisfies individual rationality: $x_i \geq v(\{i\})$ for all $i \in N$
- Notation: we will denote $\Sigma_{i \in C} x_i$ by $x(C)$

Superadditive Games

- Definition: a game $G = (N, v)$ is called superadditive if $v(C \cup D) \geq v(C) + v(D)$ for any two disjoint coalitions C and D
- Example: $v(C) = |C|^2$:
  - $v(C \cup D) = (|C|+|D|)^2 \geq |C|^2+|D|^2 = v(C) + v(D)$
- In superadditive games, two coalitions can always merge without losing money; hence, we can assume that players form the grand coalition

Other kinds of characteristic games

Convex games

$v(C \cup D) + v(C \cap D) \geq v(C) + v(D)$, for all $C, D \subseteq N$

Simple games

$v(C) \in \{0, 1\}$, for all $C \subseteq N$

Example (1)

Alice, *Bob*, *Celine* share a meal:

$$v(c) = \begin{cases} 80, & \text{if } c = \{A\} \\ 56, & \text{if } c = \{B\} \\ 70, & \text{if } c = \{C\} \\ 80, & \text{if } c = \{A, B\} \\ 85, & \text{if } c = \{A, C\} \\ 72, & \text{if } c = \{B, C\} \\ 90, & \text{if } c = \{A, B, C\} \end{cases}$$

Superadditive?

Convex?

Simple?

No

No

No

Example (2)

C	$v(C)$
$\{a\}$	0
$\{b\}$	4
$\{c\}$	6
$\{a,b\}$	11
$\{a,c\}$	6
$\{b,c\}$	13
$\{a,b,c\}$	20

Superadditive?

Convex?

Simple?

Yes

Yes

No

Two problems in coalition formation

(1) How to distribute the gain from cooperation

(2) How to split agents in coalitions (when the game G is not super-additive)

Solution concepts

What Is a Good Outcome?

-  C: \$4,  M: \$3,  P: \$3

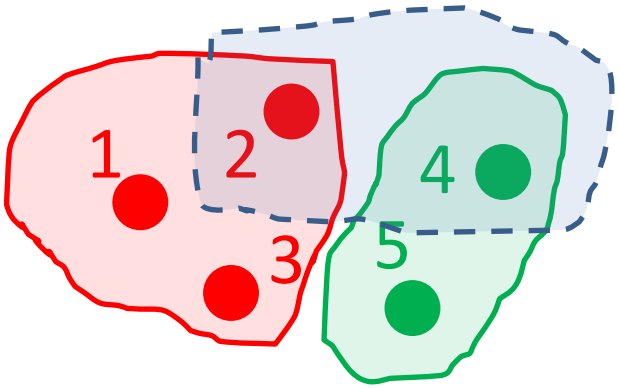
- v(Ø) = v({C}) = v({M}) = v({P}) = 0
- v({C, M}) = 500, v({C, P}) = 500, v({M, P}) = 0
- v({C, M, P}) = 750
- This is a superadditive game
  - grand coalition is formed
- How should the players share the ice-cream?
  - if they share as (200, 200, 350), Charlie and Marcie can get more ice-cream by buying a 500g tub on their own, and splitting it equally
  - the outcome (200, 200, 350) is not stable!

Transferable Utility Games: Stability

- Definition: the core of a game is the set of all stable outcomes, i.e., outcomes that no coalition wants to deviate from

  core(G) = {(CS, **x**) | $\Sigma_{i \in C} x_i \geq v(C)$ for any $C \subseteq N$}

  – each coalition earns at least as much as it can make on its own
- Note that G is not assumed to be superadditive
- Example
  – suppose v({1, 2, 3}) = 9, v({4, 5}) = 4, v({2, 4}) = 7
  – then (({1, 2, 3}, {4, 5}), (3, 3, 3, 3, 1)) is NOT in the core



Ice-Cream Game: Core

-  C: \$4,  M: \$3,  P: \$3

- $v(\emptyset) = v(\{C\}) = v(\{M\}) = v(\{P\}) = 0$, $v(\{C, M, P\}) = 750$
- $v(\{C, M\}) = 500$, $v(\{C, P\}) = 500$, $v(\{M, P\}) = 0$
- (200, 200, 350) is not in the core:
  - $v(\{C, M\}) > x_C + x_M$
- (250, 250, 250) is in the core:
  - no subgroup of players can deviate so that each member of the subgroup gets more
- (750, 0, 0) is also in the core:
  - Marcie and Pattie cannot get more on their own!

Games with Empty Core

- The core is a very attractive solution concept
- However, some games have empty cores
- $G = (N, v)$
  - $N = \{1, 2, 3\}$, $v(C) = 1$ if $|C| > 1$ and $v(C) = 0$ otherwise
  - consider an outcome $(CS, \mathbf{x})$
  - if $CS = (\{1\}, \{2\}, \{3\})$, the grand coalition can deviate
  - if $CS = (\{1, 2\}, \{3\})$, either 1 or 2 gets less than 1, so can deviate with 3
  - same argument for $CS = (\{1, 3\}, \{2\})$ or $CS = (\{2, 3\}, \{1\})$
  - suppose $CS = \{1, 2, 3\}$:

    $x_i > 0$ for some i, so $x(N\backslash\{i\}) < 1$, yet $v(N\backslash\{i\}) = 1$

A property of the core

Convex games have non-empty cores

Marginal Contribution

- A fair payment scheme would reward each agent according to his contribution
- First attempt: given a game $G = (N, v)$, set $x_i = v(\{1, ..., i-1, i\}) - v(\{1, ..., i-1\})$
  - payoff to each player = his marginal contribution to the coalition of his predecessors
- We have $x_1 + ... + x_n = v(N)$
  - $\underline{\mathbf{x}}$ is a payoff vector
- However, payoff to each player depends on the order
- $G = (N, v)$
  - $N = \{1, 2\}$, $v(\emptyset) = 0$, $v(\{1\}) = v(\{2\}) = 5$, $v(\{1, 2\}) = 20$
  - $x_1 = v(1) - v(\emptyset) = 5$, $x_2 = v(\{1, 2\}) - v(\{1\}) = 15$

Average Marginal Contribution

- Idea: to remove the dependence on ordering, can average over all possible orderings
- G = (N, v)
  - N = {1, 2}, v(∅) = 0, v({1}) = v({2}) = 5, v({1, 2}) = 20
  - 1, 2: x_1 = v(1) - v(∅) = 5, x_2 = v({1, 2}) - v({1}) = 15
  - 2, 1: y_2 = v(2) - v(∅) = 5, y_1 = v({1, 2}) - v({2}) = 15
  - $z_1 = (x_1 + y_1)/2 = 10$, $z_2 = (x_2 + y_2)/2 = 10$
  - the resulting outcome is fair!
- Can we generalize this idea?

Shapley Value

- Reminder: a permutation of {1,..., n} is a one-to-one mapping from {1,..., n} to itself
  - let P(N) denote the set of all permutations of N
- Let $S_\pi(i)$ denote the set of predecessors of i in $\pi \in P(N)$

- For $C \subseteq N$, let $\delta_i(C) = v(C \cup \{i\}) - v(C)$
- Definition: the Shapley value of player i in a game G = (N, v) with |N| = n is

$$\phi_i(G) = 1/n! \sum_{\pi:\, \pi \in P(N)} \delta_i(S_\pi(i))$$

- In the previous slide we have $\phi_1 = \phi_2 = 10$

Shapley Value: Properties (1)-(2)

- Proposition: in any game G,

$$\phi_1 + \ldots + \phi_n = v(N)$$

  - $(\phi_1, \ldots, \phi_n)$ is a payoff vector
- Definition: a player i is a dummy in a game $G = (N, v)$ if $v(C) = v(C \cup \{i\})$ for any $C \subseteq N$
- Proposition: if a player i is a dummy in a game $G = (N, v)$ then $\phi_i = 0$

Shapley Value: Properties (3)-(4)

- Definition: given a game $G = (N, v)$, two players i and j are said to be symmetric if $v(C \cup \{i\}) = v(C \cup \{j\})$ for any $C \subseteq N \setminus \{i, j\}$
- Proposition: if i and j are symmetric then $\phi_i = \phi_j$
- Definition: Let $G_1 = (N, u)$ and $G_2 = (N, v)$ be two games with the same set of players. Then $G = G_1 + G_2$ is the game with the set of players N and characteristic function w given by $w(C) = u(C) + v(C)$ for all $C \subseteq N$
- Proposition: $\phi_i(G_1+G_2) = \phi_i(G_1) + \phi_i(G_2)$

Axiomatic Characterization

- Properties of Shapley value:
  1. Efficiency: $\phi_1 + \ldots + \phi_n = v(N)$
  2. Dummy: if i is a dummy, $\phi_i = 0$
  3. Symmetry: if i and j are symmetric, $\phi_i = \phi_j$
  4. Additivity: $\phi_i(G_1+G_2) = \phi_i(G_1) + \phi_i(G_2)$
- Theorem: Shapley value is the only payoff distribution scheme that has properties (1) - (4)

A property of the Shapley values

In a convex game, the payoff vector of the Shapley values is in the core

Computational Issues in Coalitional Games

- We have defined many solution concepts - but can we compute them efficiently?
- Problem: the naive representation of a coalitional game is exponential in the number of players n
  - need to list values of all coalitions (2^n)
- We are usually interested in algorithms whose running time is polynomial in n
- So what can we do?

How to Deal with Representation Issues?

- Restricted classes
  - consider games on combinatorial structures
  - problem: not all games can be represented in this way
- Give up on worst-case succinctness
  - devise complete representation languages that allow for compact representation of interesting games

Coalition structure generation

Coalition Structure Generation

How do we **partition the set of agents** into coalitions to maximize the overall profit?

The Coalition Structure Generation Problem

Example: given 3 agents, the possible **coalitions** are:

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

The possible **coalition structures** are:

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

The **input** is the characteristic function

$v(\{a_1\}) = 30$

$v(\{a_2\}) = 40$

$v(\{a_3\}) = 25$

$v(\{a_1,a_2\}) = 50$

$v(\{a_1,a_3\}) = 60$

$v(\{a_2,a_3\}) = 55$

$v(\{a_1,a_2,a_3\}) = 90$

What we want as **output** is a coalition structure in which the sum of values is maximized

$V(\{\{a_1\},\{a_2\},\{a_3\}\}) = 30+40+25 = 95$

$V(\{\{a_1,a_2\},\{a_3\}\}) = 50+25 = 75$

...

optimal coalition structure

Optimal coalition structure

$$\mathrm{CS}^* = argmax_{\mathrm{CS}\in\mathcal{P}^A} V(\mathrm{CS})$$

$$V(\mathrm{CS}) = \sum_{C\in\mathrm{CS}} v(C)$$

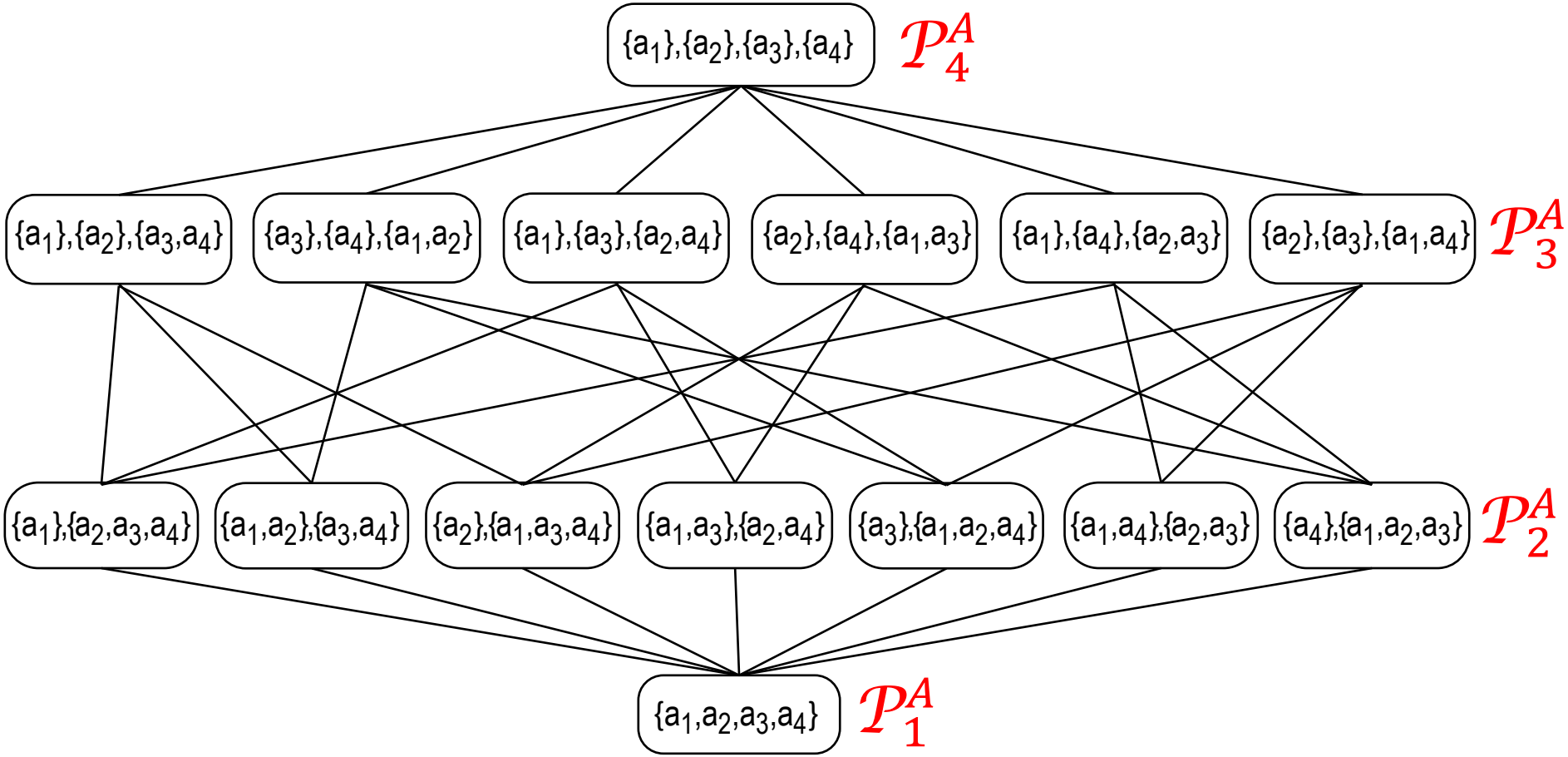
Coalition Structure Generation

How should we solve this problem?

We will present multiple algorithms, but first we need to present the main **representations** of the search space (the set of coalition structures, denoted as $\mathcal{P}^{\mathrm{A}}$)

The Coalition Structure Graph
(example of 4 agents)

$\mathcal{P}_i^A \subseteq \mathcal{P}^A$ contains all coalition structures that consist of exactly i coalitions



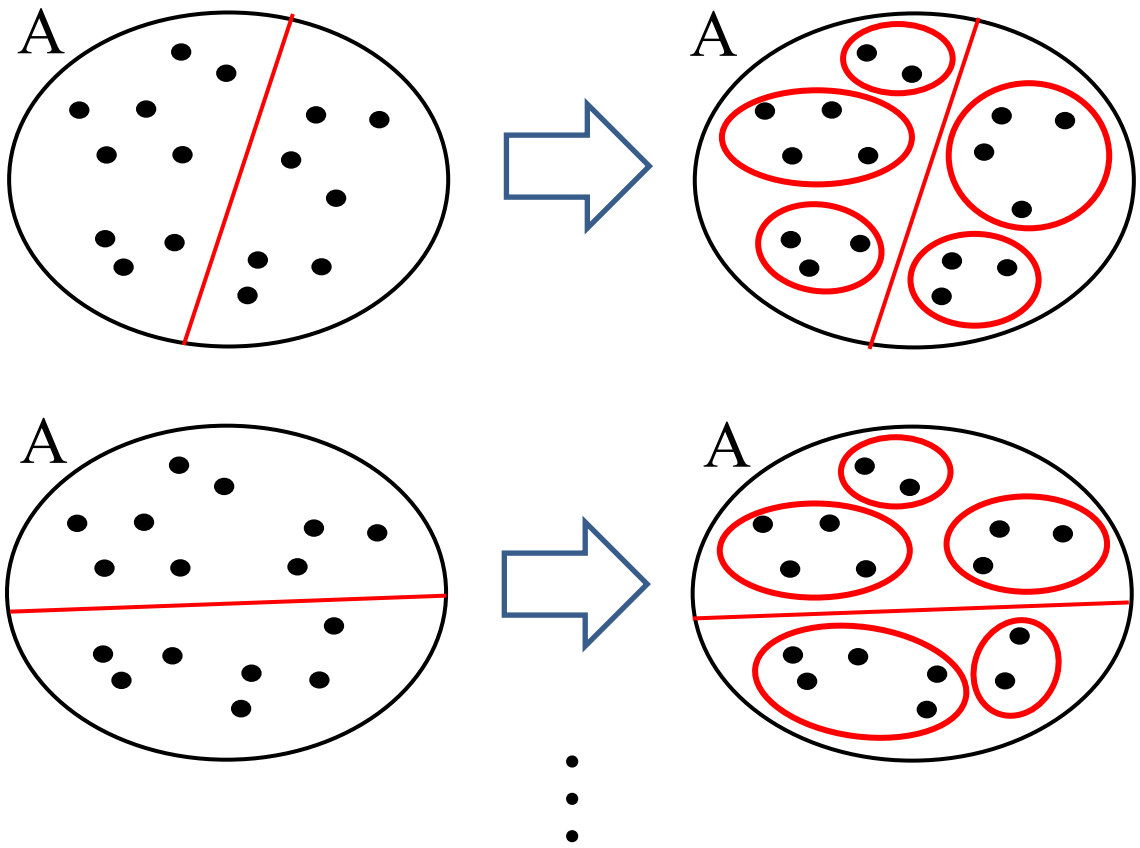
Coalition Structure Generation

Solving the problem using **Dynamic Programming**

The Dynamic Programming (DP) Algorithm

Main observation:
To examine all coalition structure $CS : |CS| \geq 2$, it is sufficient to:

- try the possible ways to split the **set of agents into two sets**, and
- for every half, find **the optimal partition** of that half.



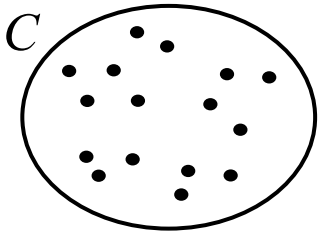
The Dynamic Programming (DP) Algorithm

Main theorem:

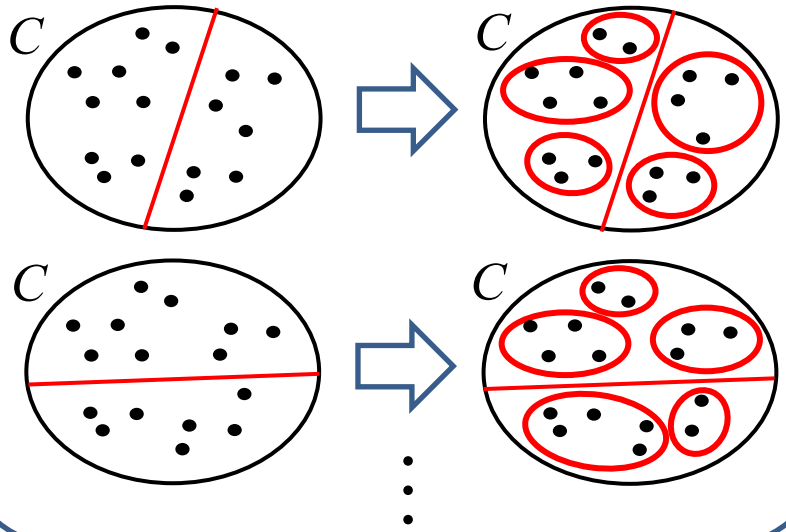
Given a coalition $C \subseteq A$, let $\mathcal{P}^C$ be the set of partitions of C, and let $f(C)$ be the value of an optimal partition of C, i.e., $f(C) = \max_{P \in \mathcal{P}^C} V(P)$. Then,

$$f(C) = \begin{cases} v(C) & \text{if } |C| = 1 \\ \max\{\, v(C),\ \max_{\{C', C''\} \in \mathcal{P}^C} f(C') + f(C'') \,\} & \text{otherwise} \end{cases}$$

the value of the coalition itself (without partitioning)



the maximum value for all such partitions



The Dynamic Programming (DP) Algorithm

Algorithm:

- Iterate over all coalitions $C:|C|=1$, then over all $C:|C|=2$, then all $C:|C|=3$, etc.
- For every coalition, C, compute $f(C)$ using the above equation
- While computing $f(C)$:
  - the algorithm stores in $t(C)$ the best way to split C in two
  - unless it is more beneficial to keep C as it is (i.e., without splitting)
- By the end of this process, $f(A)$ will be computed, which is by definition the value of the optimal coalition structure
- It remains to compute the optimal coalition structure itself, by using $t(A)$

Consider the following example of 4 agents

Example:

input:

v({1}) = 30
v({2}) = 40
v({3}) = 25
v({4}) = 45
v({1,2}) = 50
v({1,3}) = 60
v({1,4}) = 80
v({2,3}) = 55
v({2,4}) = 70
v({3,4}) = 80
v({1,2,3}) = 90
v({1,2,4}) = 120
v({1,3,4}) = 100
v({2,3,4}) = 115
v({1,2,3,4}) = 140

	coalition	evaluations performed before setting f		t	f
step 1	{1}	v({1})=30		{1}	30
	{2}	v({2})=40		{2}	40
	{3}	v({3})=25		{3}	25
	{4}	v({4})=45		{4}	45
step 2	{1,2}	v({1,2})=50	f({1})+f({2})=70	{1} {2}	70
	{1,3}	v({1,3})=60	f({1})+f({3})=55	{1,3}	60
	{1,4}	v({1,4})=80	f({1})+f({4})=75	{1,4}	80
	{2,3}	v({2,3})=55	f({2})+f({3})=65	{2} {3}	65
	{2,4}	v({2,4})=70	f({2})+f({4})=85	{2} {4}	85
	{3,4}	v({3,4})=80	f({3})+f({4})=70	{3,4}	80
step 3	{1,2,3}	v({1,2,3})=90	f({1})+f({2,3})=95	{2} {1,3}	100
		f({2})+f({1,3})=100	f({3})+f({1,2})=95		
	{1,2,4}	v({1,2,4})=120	f({1})+f({2,4})=115	{1,2,4}	120
		f({2})+f({1,4})=110	f({4})+f({1,2})=115		
	{1,3,4}	v({1,3,4})=100	f({1})+f({3,4})=110	{1} {3,4}	110
		f({3})+f({1,4})=105	f({4})+f({1,3})=105		
	{2,3,4}	v({2,3,4})=115	f({2})+f({3,4})=120	{2} {3,4}	120
		f({3})+f({2,4})=110	f({4})+f({2,3})=110		
step 4	{1,2,3,4}	v({1,2,3,4})=140	f({1})+f({2,3,4})=150	{1,2} {3,4}	150
		f({2})+f({1,3,4})=150	f({3})+f({1,2,4})=145	step 5	
		f({4})+f({1,2,3})=145	f({1,2})+f({3,4})=150		
		f({1,3})+f({2,4})=145	f({1,4})+f({2,3})=145		

The Dynamic Programming (DP) Algorithm

Note:

- While DP is guaranteed to find an optimal coalition structure, many of its operations were shown to be redundant

- An improved dynamic programming algorithm (called IDP) was developed that avoids all redundant operations

Advantage:

- IDP is the fastest algorithm that finds an optimal coalition structure in $O(3^n)$

Disadvantage:

- IDP provides no interim solutions before completion, meaning that it is not possible to trade computation time for solution quality.

Metaheuristic Algorithms

Good scalability, "*good*" solution, no guarantees!

Metaheuristic Algorithms

As the number of agents increases, the problem becomes too hard, and the only practical option is to use metaheuristic algorithms.

Advantage:

- Can usually be applied for very large problems.

Disadvantage:

- No guarantees that an optimal solution is ever found
- No guarantees on the quality of their solutions.

Examples:

- Genetic Algorithms [Sen & Dutta, 2000]
- Simulated Annealing [Keinanen, 2009]
- Decentralized greedy algorithm [Shehory & Kraus, 1998]
- Greedy algorithm based on GRASP [Di Mauro *et al*, 2010]

Decentralized greedy algorithm by Shehory and Kraus

Algorithm 4 Shehory and Kraus algorithm

(1) $\mathcal{C}_i \leftarrow$ set of all coalitions that include agent a_i
(2) $C_i^* \leftarrow \text{argmax}_{C \in \mathcal{C}_i} \frac{v(C)}{|C|}$
(3) broadcast (a_i, C_i^*), receive other broadcasts, put received (a_j, C_j^*) in $\mathcal{C}^*$ (including (a_i, C_i^*))
(4) $C_{max} \leftarrow$ largest subset of set of agents A such that, for all $a_j \in C_{max}$, $(a_j, C_{max}) \in \mathcal{C}^*$
(5) **if** $a_i \in C_{max}$, **then** join C_{max} and **return**
(6) delete from $\mathcal{C}_i$ all coalitions that include agents from C_{max}
(7) **if** $\mathcal{C}_i$ is not empty, **then goto** 2
(8) **return**

The version refers to the algorithm run by agent a_i

Sub-optimal, in general

Example

S	{1}	{2}	{3}	{1, 2}	{1, 3}	{2, 3}	{1, 2, 3}
v(S)	0	0	0	90	80	70	120

agent 1	agent 2	agent 3
C_1 = {{1}, {1,2}, {1,3}, {1,2,3}}	C_2 = {{2}, {1,2}, {2,3}, {1,2,3}}	C_3 = {{3}, {1,3}, {2,3}, {1,2,3}}
C_1^* = {1,2}	C_2^* = {1,2}	C_3^* = {1,2,3} (assuming that the largest coalition is chosen when breaking ties)
C_{max} = {1,2}	C_{max} = {1,2}	C_{max} = {1,2}
agents 1 and 2 join coalition {1,2}		
		C_3 = {{3}}
		C_3^* = {3}
		C_{max} = {3}
		agent 3 joins coalition {3}

The coalition structure {{1,2}, {3}} is eventually generated.