UNIVERSITY OF Southampton
School of Electronics and Computer Science

Cooperative Game Theory & its Application to Multi-Agent Systems

Dr. Talal Rahwan
University of Southampton, UK

Organizations in Multi-Agent Systems

[Horligh and Lesser, 2005]

Definition:

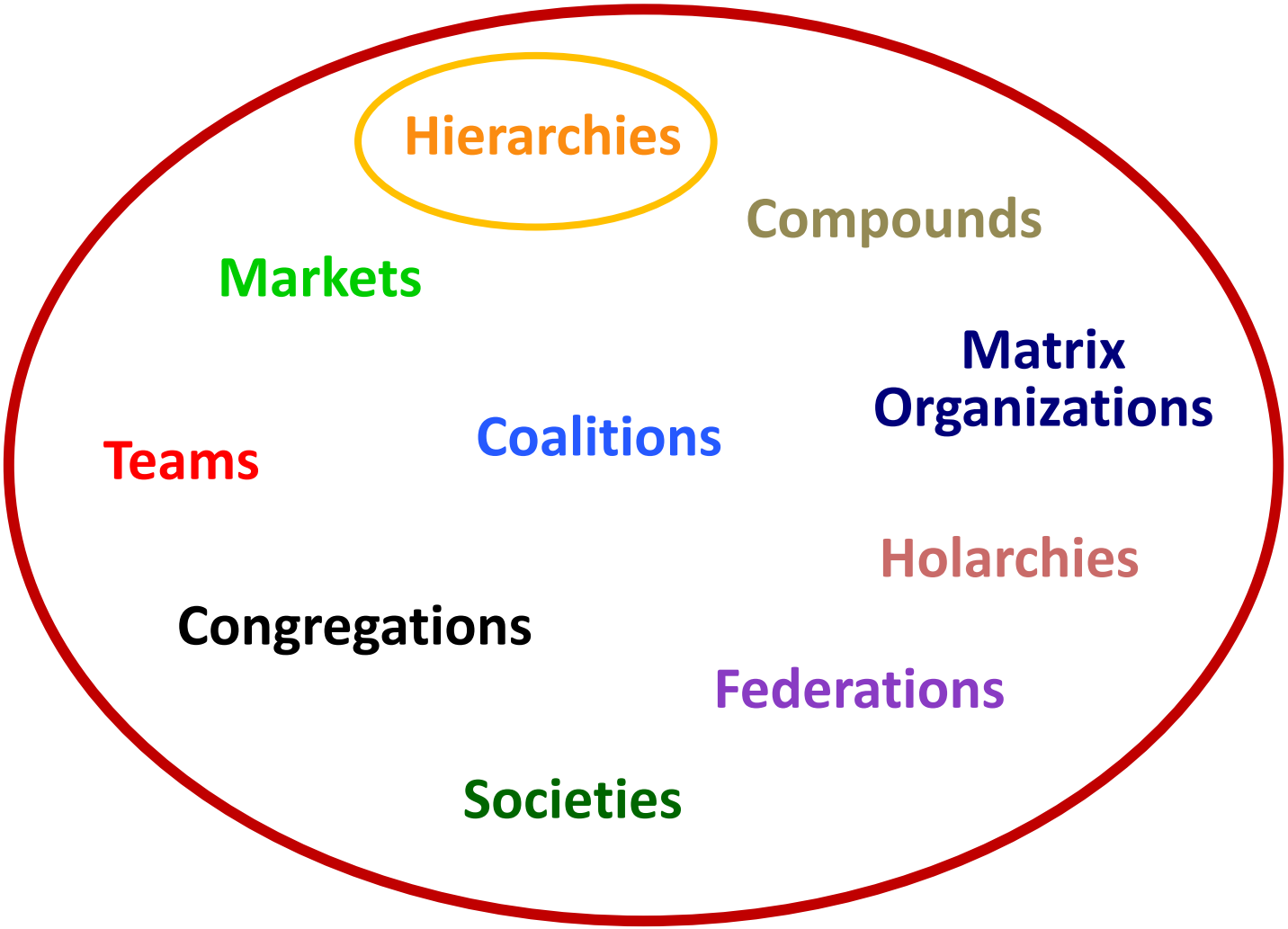
The organization of a multi-agent system is the collection of roles, relationships, and authority structures which govern its behaviour [Horling & Lesser, 2005].

Organizations guide how agents interact with one another. This guidance influences authority relationships, data flow, resource allocation, coordination patterns or any number of other system characteristics

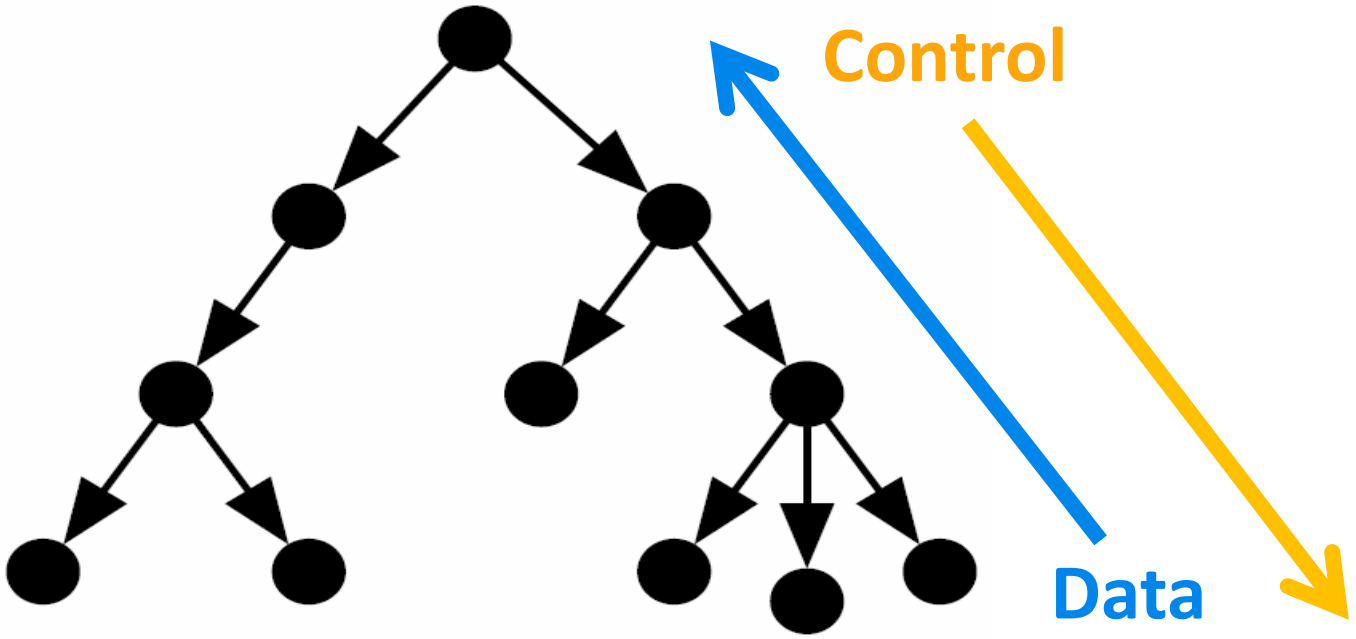
there is no single type of organizations that is suitable for all situations. This is due to the inevitable tradeoffs that must be made, as well as the uncertainty, lack of global coherence and dynamism present in any realistic population

Organizations in Multi-Agent Systems

[Horligh and Lesser, 2005]

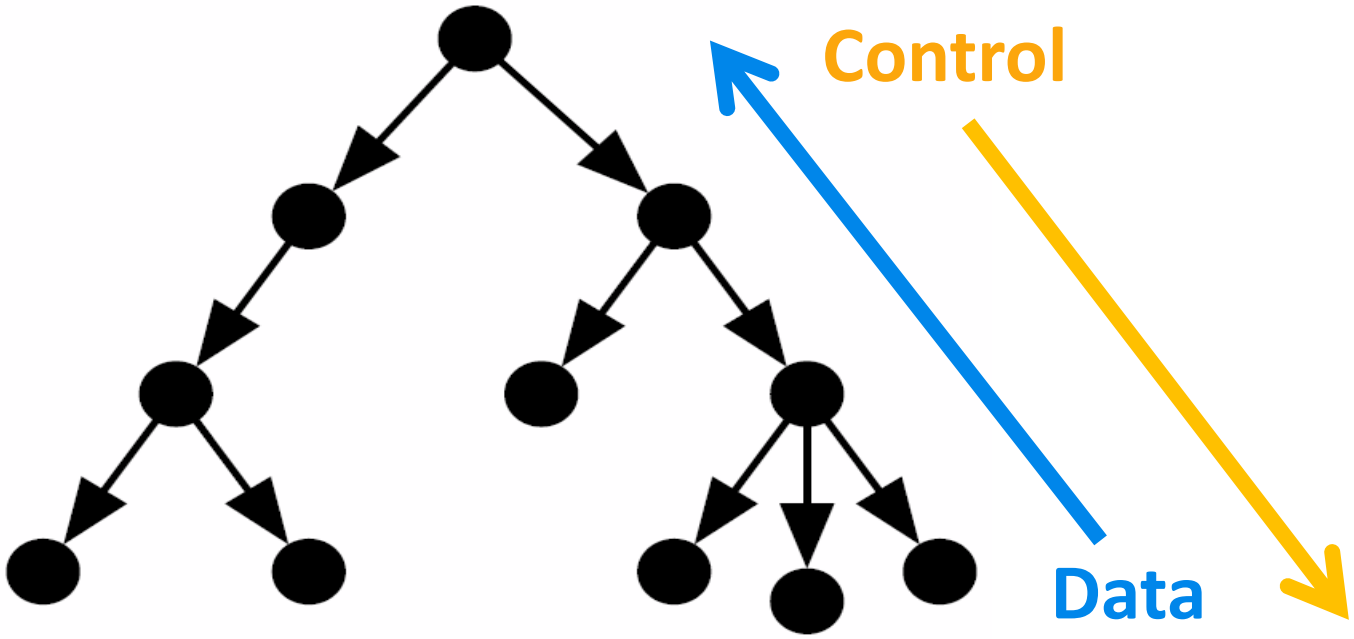


Hierarchies



Agents are arranged in a tree-structure, where agents higher in the tree have a more global view than those below them.

Hierarchies



Benefits:

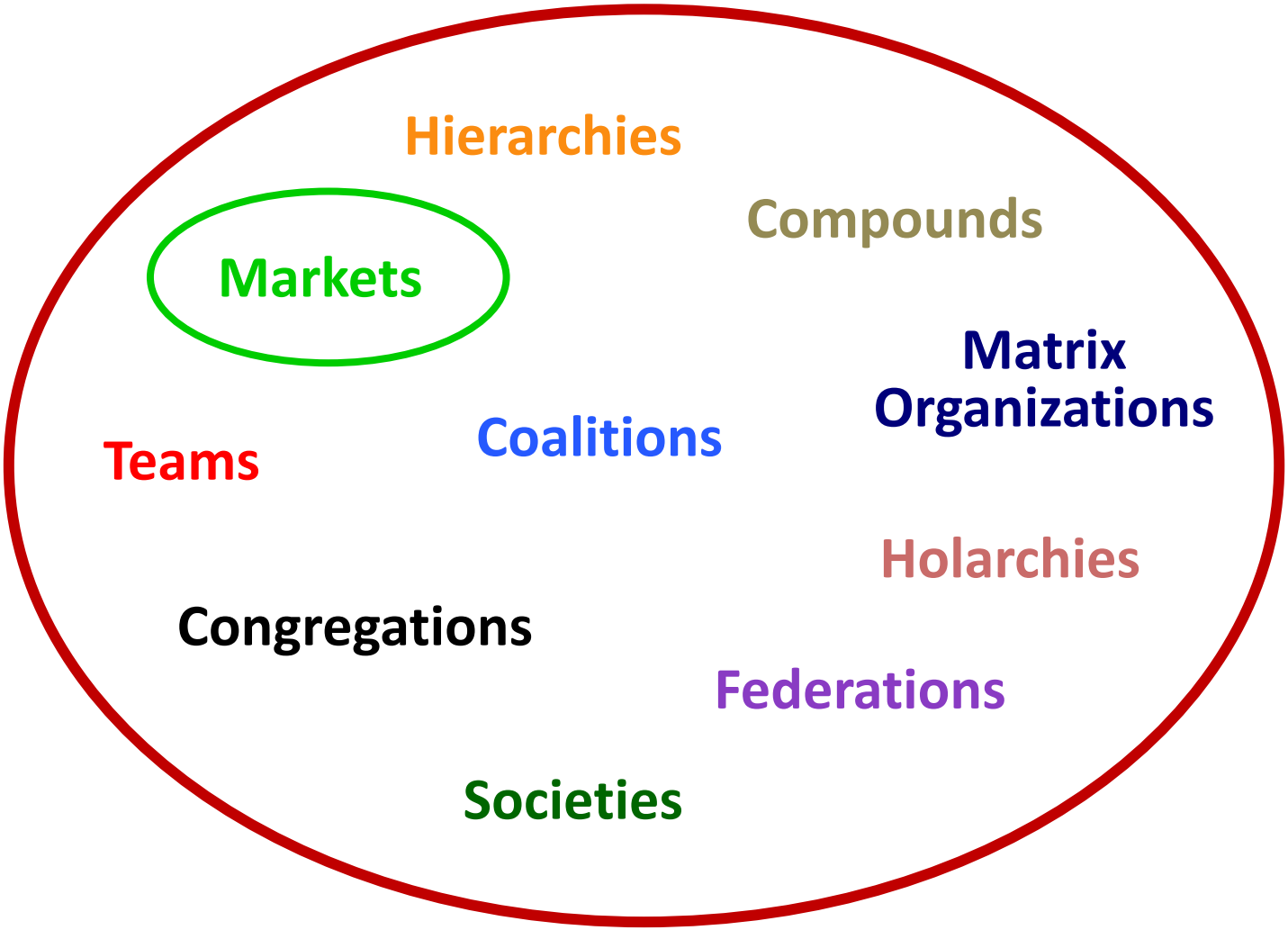
- Maps to many common domains.
- Handles scale well (because it constrains the agents to a number of interactions that is small relative to the total population size).

Drawbacks:

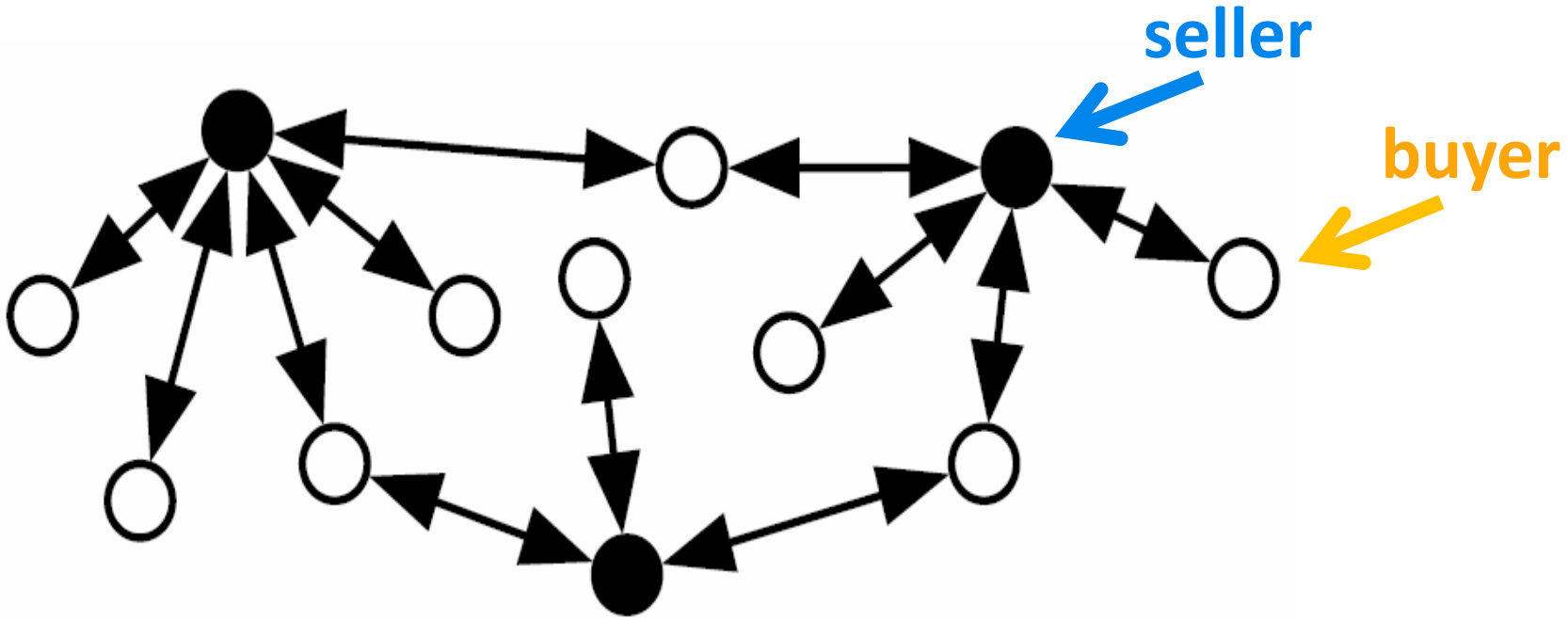
- Prone to single-point failures with potentially global consequences.
- Can lead to overloading (if the structure is too flat) or delays (if the structure is tall).

Organizations in Multi-Agent Systems

[Horligh and Lesser, 2005]



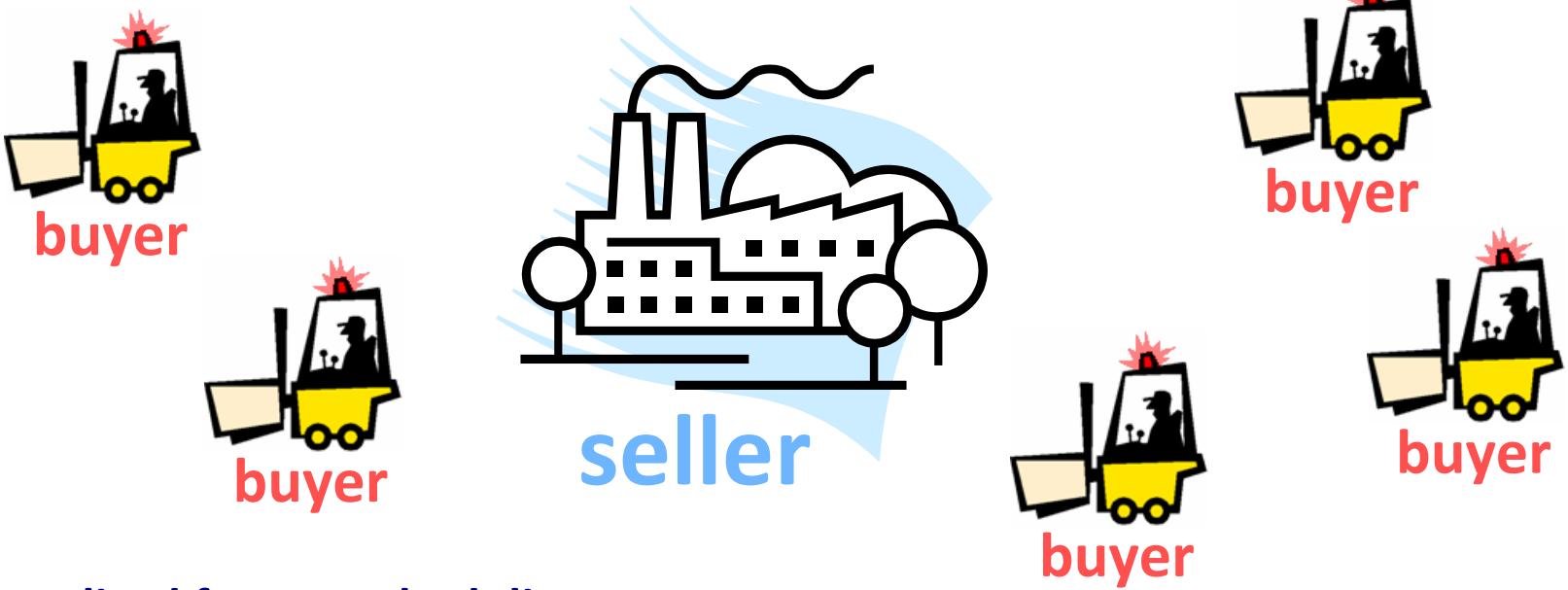
Markets



Buyers request (or place bids for) items, such as shared resources, tasks, services or goods.

Sellers are responsible for processing bids and determining the winner.

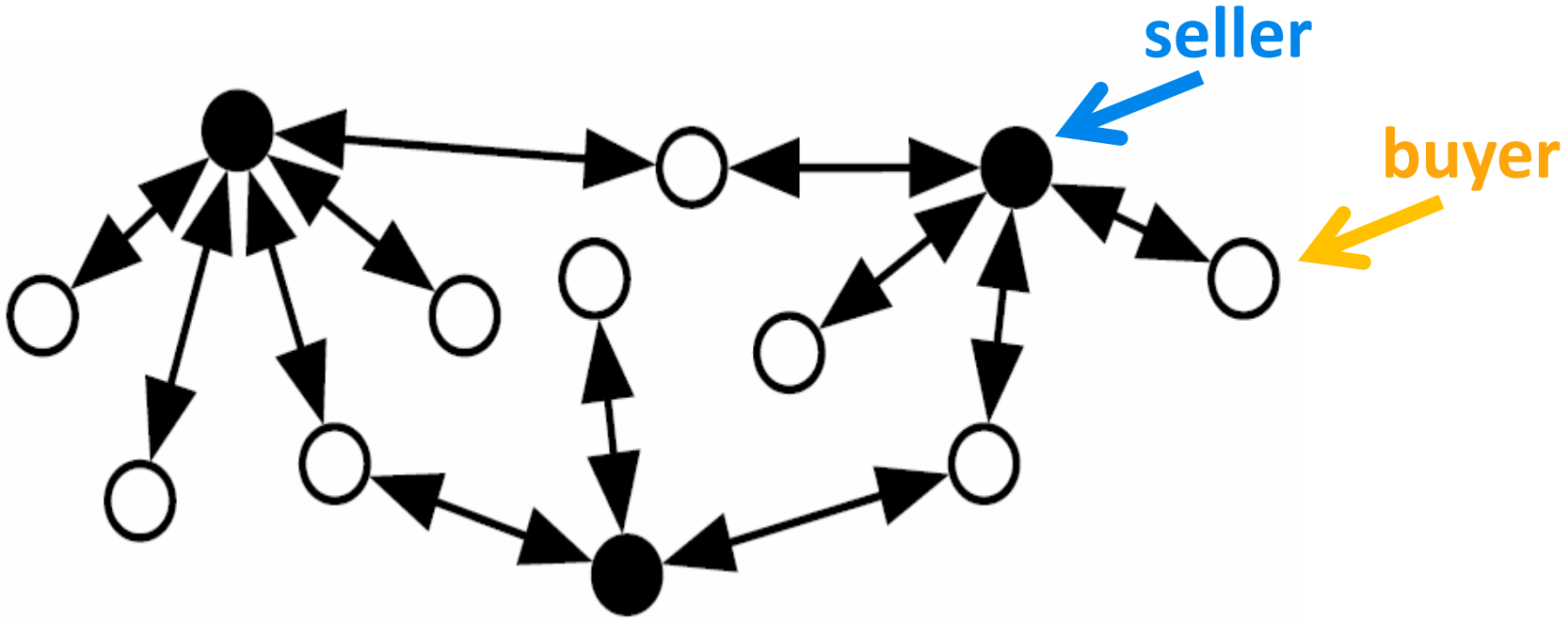
Markets



decentralized factory scheduling:

- Each factory job is associated with a duration, deadline and value.
- The factory has a reserve price for each time slot that it has available.
- Agents bid on a set of slots that has sufficient time to satisfy the job without exceeding the deadline, using the job value as a maximum bid price.
- Market forces cause agents to seek out the most cost-effective time slots, and higher-valued jobs naturally take precedence over lower ones.
- This lead to an efficient allocation of (time) resources, while maximizing the factory's overall utility.

Markets



Benefits:

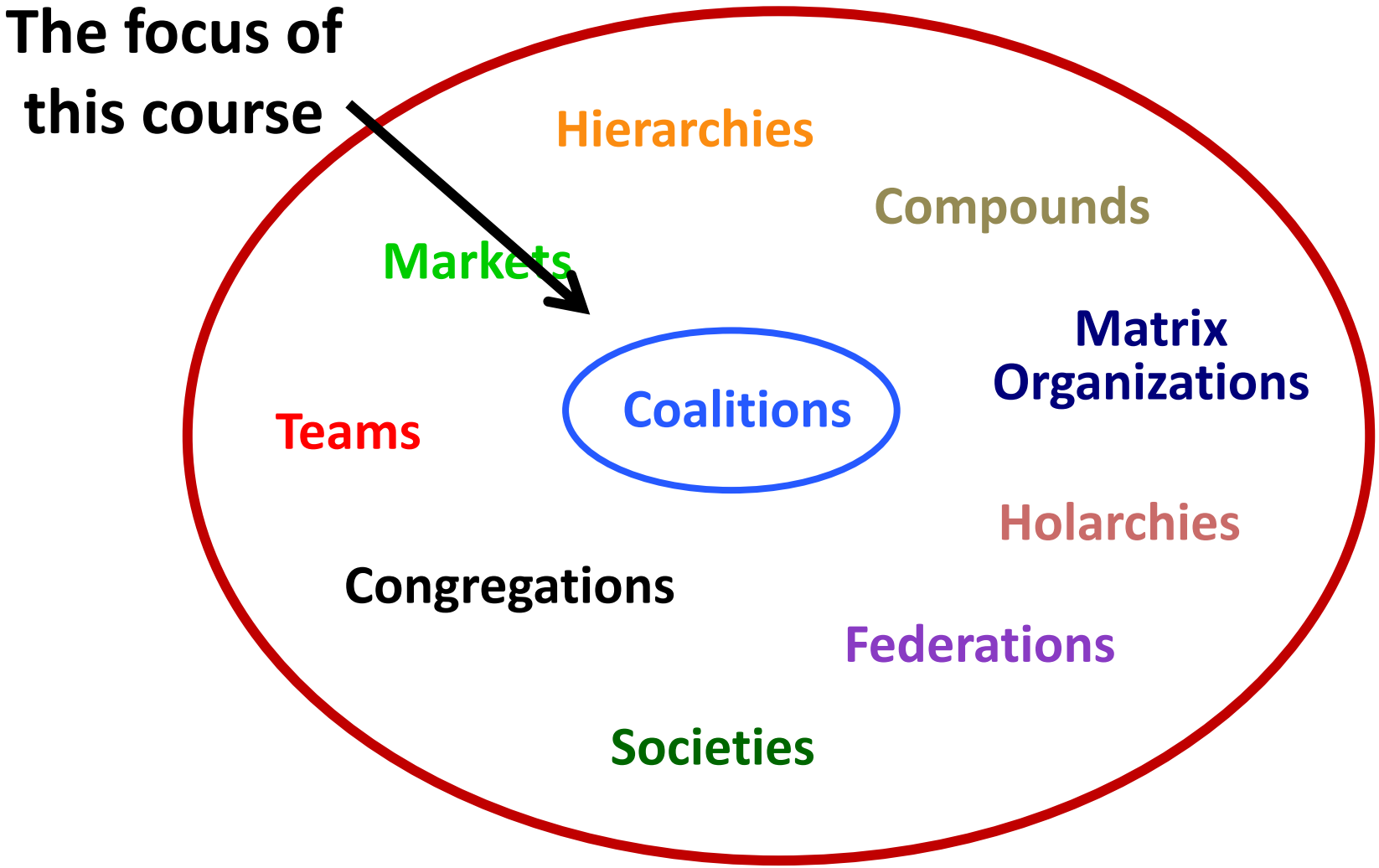
- Efficient allocation and pricing *(if agents bid truthfully)*
- A wealth of results (from human economics & business) can be used
- Increased fairness through bidding

Drawbacks:

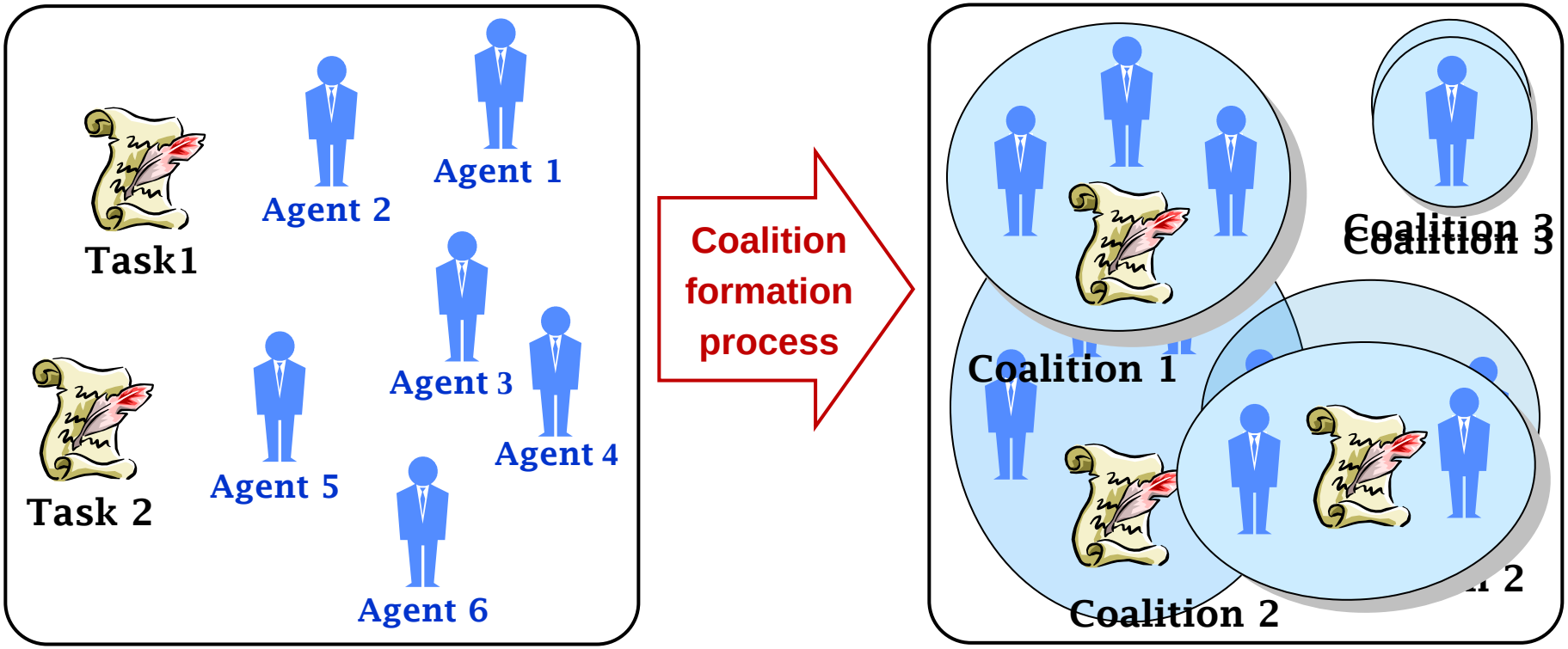
- The potential complexity of reasoning about the bidding process, and determining the auction's outcome.
- Potential for collusion and malicious behaviour

Organizations in Multi-Agent Systems

[Horligh and Lesser, 2005]



Coalition Formation



Main characteristicis

- Coalition in general are goal-directed and short-lived
- No coordination among members of different coalitions
- The organizational structure within each coalition is flat

Applications of Coalition Formation

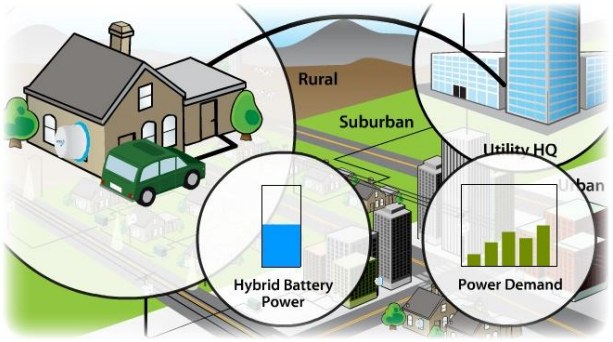
- **e-commerce:** Buyers can form coalitions to purchase a product in bulk and take advantage of price discounts [Tsvetovat et al., 2000].

- **Distributed sensor networks:** Coalitions of sensors can work together to track targets of interest [Dang et al. 2006]

- **Distributed vehicle routing:** Coalitions of delivery companies can be formed to reduce the transportation costs by sharing deliveries [Sandholm and Lesser, 1997].

- **Information gathering:** Several information servers can form coalitions to answer queries [Klusch and Shehory, 1996].

Applications of Coalition Formation

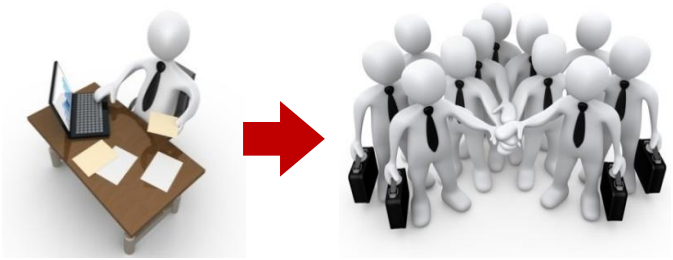


Smart Energy Grids

- Intelligent appliances and energy storage devices coordinate for optimal energy use
- Links with PRI Limited (smart meters).

Electronic-commerce

- Cooperation among buyers to obtain quantity discounts, and sellers to maintain cartel pricing.
- Links with Lostwax (aircraft industry) and with Aroxo (home appliance).

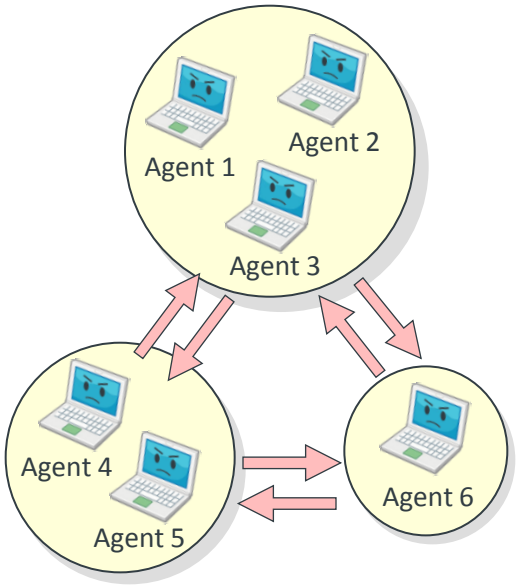


Disaster Management

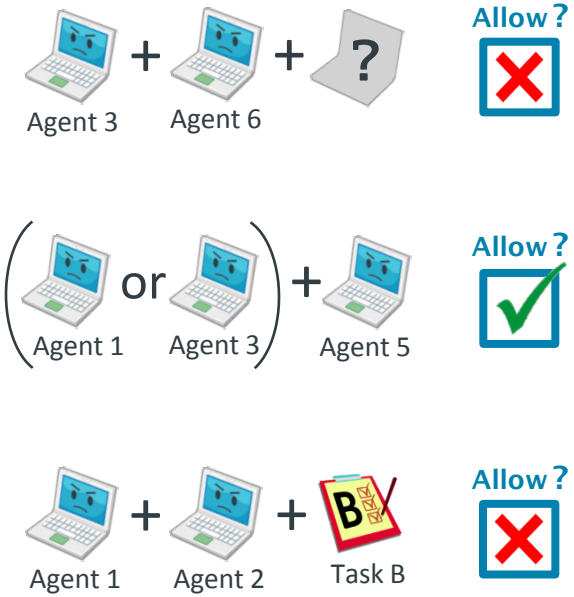
- *"Efforts by the United Nations in Haiti have lacked sufficient coordination"* --- independent UN report.
- Links with BAE Systems (security & aerospace).

Challenges

Inter-dependencies among coalitions

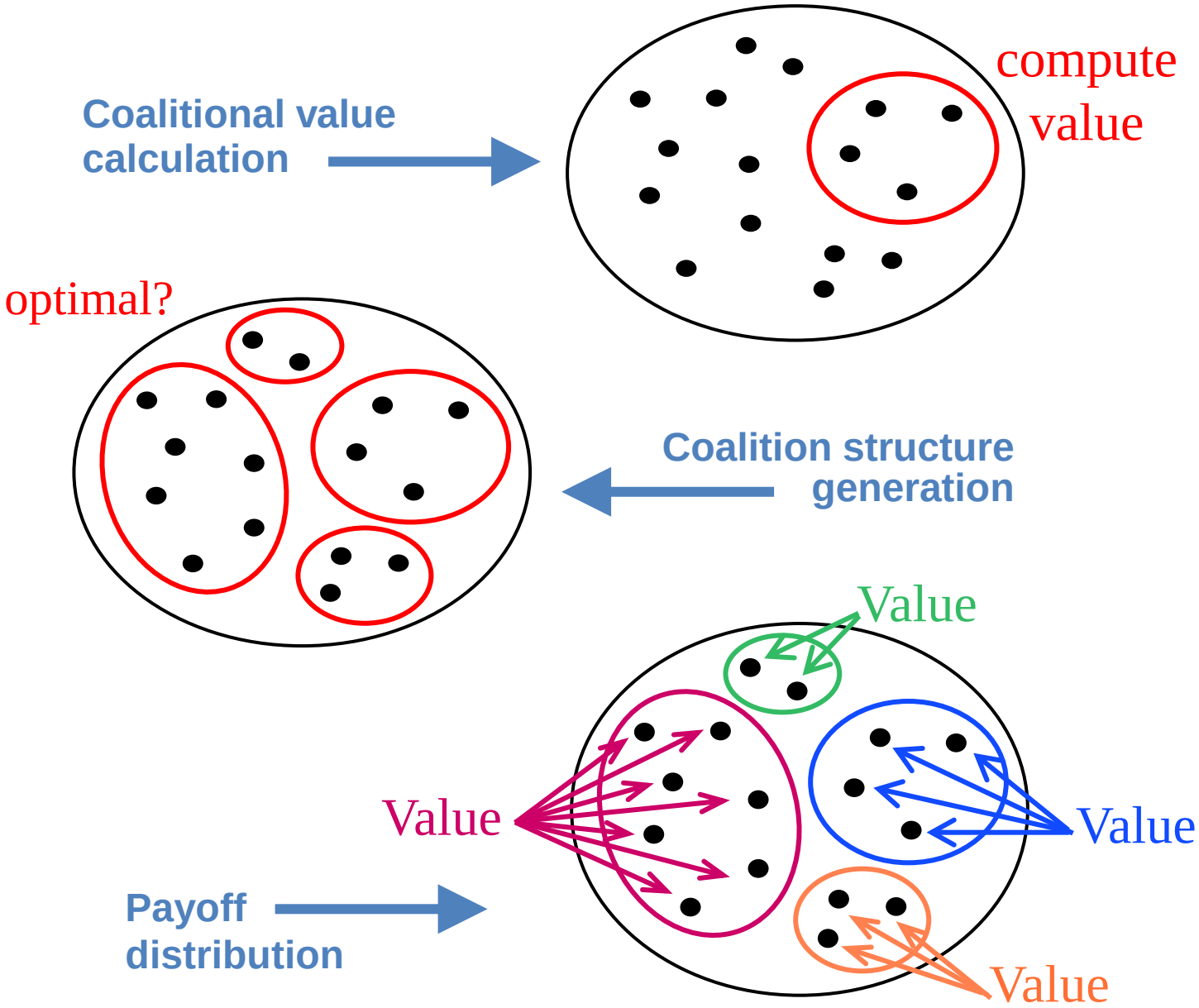


Constraints on the feasible coalitions



Distributing the computations

Coalition formation process



Coalition Value Calculation

The agents should decide which of the potential coalitions to actually form.

To do so, they typically calculate a value for each coalition, known as the ***coalition value***, which indicates how beneficial that coalition would be if it was formed.

For n agents, the number of possible coalitions is $2^n - 1$

Coalition Value Calculation

This calculation is domain dependant:

- In an electronic marketplace, the value of a coalition of buyers can be calculated as the difference between the sum of the reservation costs of the coalition members and the minimum cost needed to satisfy the requests of all the members [Li & Sycara, 2002].
- In wireless networks the value of a coalition of transmitters corresponds to the maximum sum-rate achievable by that coalition [Mathur et al., 2006b].
- In information gathering systems, the coalition value can be designed to represent a measure of how closely the information agents' domains are related [Klusch & Shehory, 1996]

Distributing the Coalition Value Calculations

The following properties for a distribution algorithm:

- The distribution process should be decentralized
- Communication between the agents should be minimal
- Minimize the number of calculations that are redundantly carried out
- The calculations should be distributed equally among the agents
- The amount of memory each agent requires for performing the computations should be minimized.

Distributing the Coalition Value Calculations

In the following example, we have:

- Number of Agents: $n = 5$
- Coalitions may contain up to 3 agents

Distributing the Coalition Value Calculations

In the following e ave:

- Number of Agents: $n = 5$
- Coalitions may contain up to 3 agents

coalitions in which Agent **2** is a member	Long-term commitment list
2	
1 , 2	
2 , 3	
2 , 4	
2 , 5	
1 , 2 , 3	
1 , 2 , 4	
1 , 2 , 5	
2 , 3 , 4	
2 , 3 , 5	

coalitions in which Agent **3** is a member	Long-term commitment list
3	
1 , 3	
2 , 3	
3 , 4	
3 , 5	
1 , 2 , 3	
1 , 3 , 4	
1 , 3 , 5	
2 , 3 , 4	
2 , 3 , 5	

Distributing the Coalition Value Calculations

coalitions in which Agent **1** is a member	Long-term commitment list
1	
1 , 2	
1 , 3	
1 , 4	
1 , 5	
1 , 2 , 3	
1 , 2 , 4	
1 , 2 , 5	
1 , 3 , 4	
1 , 3 , 5	

Commit

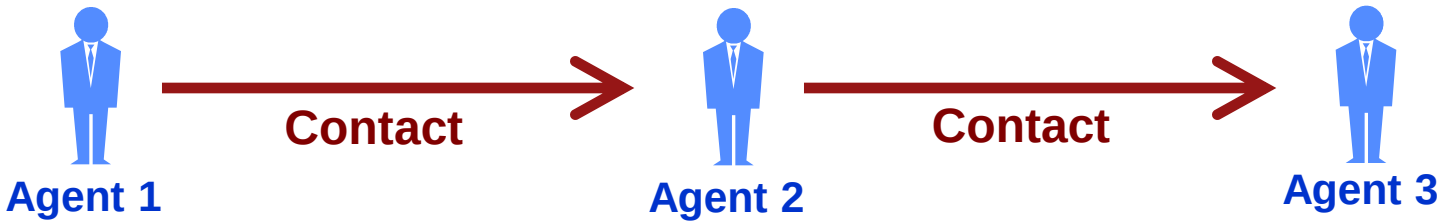
Commit

coalitions in which Agent **2** is a member	Long-term commitment list
2	
1 , 2	
	2 , 3
2 , 4	
2 , 5	
	1 , 2 , 3
1 , 2 , 4	
1 , 2 , 5	
2 , 3 , 4	
2 , 3 , 5	

Delete

Delete

coalitions in which Agent **3** is a member	Long-term commitment list
3	
1 , 3	
3 , 4	
3 , 5	
1 , 3 , 4	
1 , 3 , 5	
2 , 3 , 4	
2 , 3 , 5	

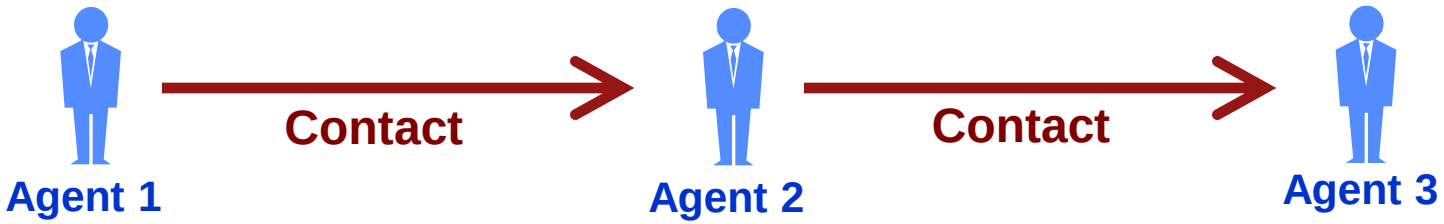


Distributing the Coalition Value Calculations

coalitions in which Agent **1** is a member	Long-term commitment list
1	
Commit →	1 , 2
1 , 3	
1 , 4	
1 , 5	
1 , 2 , 3	
1 , 2 , 4	
Commit →	1 , 2 , 5
1 , 3 , 4	
1 , 3 , 5	

coalitions in which Agent **2** is a member	Long-term commitment list
2	
Delete →	
Commit →	2 , 3
2 , 4	
2 , 5	
Commit →	1 , 2 , 3
1 , 2 , 4	
Delete →	
2 , 3 , 4	
2 , 3 , 5	

coalitions in which Agent **3** is a member	Long-term commitment list
3	
1 , 3	
Delete →	
3 , 4	
3 , 5	
Delete →	
1 , 3 , 4	
1 , 3 , 5	
2 , 3 , 4	
2 , 3 , 5	

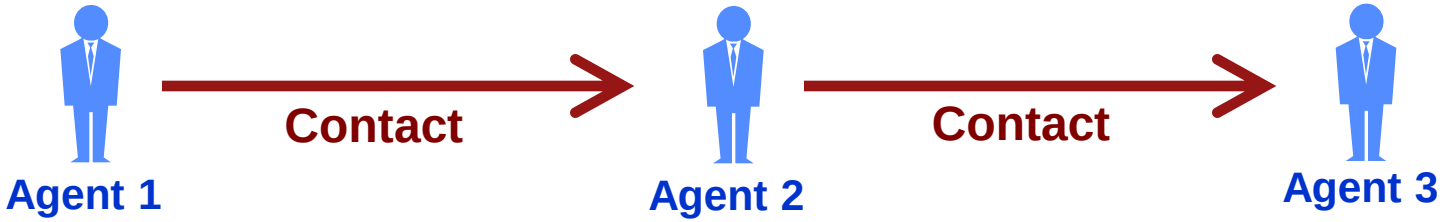


Distributing the Coalition Value Calculations

coalitions in which Agent **1** is a member	Long-term commitment list
1	
Commit →	1 , 2
1 , 3	
1 , 4	
1 , 5	
1 , 2 , 3	
1 , 2 , 4	
Commit →	1 , 2 , 5
1 , 3 , 4	
1 , 3 , 5	

coalitions in which Agent **2** is a member	Long-term commitment list
Commit →	2
Delete →	
Commit →	2 , 3
Delete →	
Commit →	2 , 5
Commit →	1 , 2 , 3
Commit →	1 , 2 , 4
Delete →	
Delete →	
Commit →	2 , 3 , 5

coalitions in which Agent **3** is a member	Long-term commitment list
3	
1 , 3	
Delete →	
3 , 4	
3 , 5	
Delete →	
1 , 3 , 4	
1 , 3 , 5	
2 , 3 , 4	
2 , 3 , 5	



Distributing the Coalition Value Calculations

The aforementioned algoritm, due to Shehory and Kraus (*Artificial intelligence journal*, *1998*) suffers from the following limitations:

- Some agents may calculate significantly more values than others
- Some values may be calculated more than once
- Requires an exponentially increasing communication overhead

DCVC

We developed a novel algorithm (called **DCVC**) for Distributing Coalitional Value Calculations among the cooperative agents

DCVC has the following advantages:

- It is a *distributed, decentralized* algorithm
- Requires *no communication* between the agents
- Results in *no redundant* calculations
- Distributes the calculations *equally* among the agents
- Has *minimal* memory requirements

  *Be careful! Given 40 agents, storing the values of all possible coalitions requires **5120 Giga Bytes** of memory* **.**

DCVC

The set of possible coalitions can be divided into subsets, each containing the coalitions of a particular size.

In DCVC, the distribution of all possible coalitions is carried out by distributing each of these subsets equally among the agents

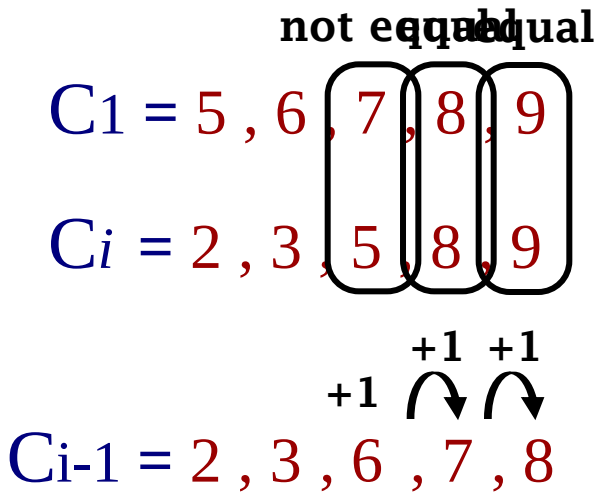
L1	L2	L3	L4	L5	L6
1	1 , 2	1 , 2 , 3	1 , 2 , 3 , 4	1 , 2 , 3 , 4 , 5	1, 2, 3, 4, 5, 6
2	1 , 3	1 , 2 , 4	1 , 2 , 3 , 5	1 , 2 , 3 , 4 , 6	
3	1 , 4	1 , 2 , 5	1 , 2 , 3 , 6	1 , 2 , 3 , 5 , 6	
4	1 , 5	1 , 2 , 6	1 , 2 , 4 , 5	1 , 2 , 4 , 5 , 6	
5	1 , 6	1 , 3 , 4	1 , 2 , 4 , 6	1 , 3 , 4 , 5 , 6	
6	2 , 3	1 , 3 , 5	1 , 2 , 5 , 6	2 , 3 , 4 , 5 , 6	
	2 , 4	1 , 3 , 6	1 , 3 , 4 , 5		
	2 , 5	1 , 4 , 5	1 , 3 , 4 , 6		
	2 , 6	1 , 4 , 6	1 , 3 , 5 , 6		
	3 , 4	1 , 5 , 6	1 , 4 , 5 , 6		
	3 , 5	2 , 3 , 4	2 , 3 , 4 , 5		
	3 , 6	2 , 3 , 5	2 , 3 , 4 , 6		
	4 , 5	2 , 3 , 6	2 , 3 , 5 , 6		
	4 , 6	2 , 4 , 5	2 , 4 , 5 , 6		
	5 , 6	2 , 4 , 6	3 , 4 , 5 , 6		
		2 , 5 , 6			
		3 , 4 , 5			
		3 , 4 , 6			
		3 , 5 , 6			
		4 , 5 , 6			

DCVC

For any permitted size s , the list L_s of possible coalitions of size s, should be ordered as follows:

Example: For 9 agents, the list L_5 should be ordered as follows:

- The first coalition in the list is: { 5 , 6 , 7 , 8 , 9 }
- The last coalition in the list is: { 1 , 2 , 3 , 4 , 5 }
- Given any coalition that is located at index i in L_s , for example: C_i = { 2 , 3 , 5 , 8 , 9 }, the agent can find C_{i-1} as follows:



DCVC

L 4

For ***n*** agents, the number of coalitions of size ***s*** is:

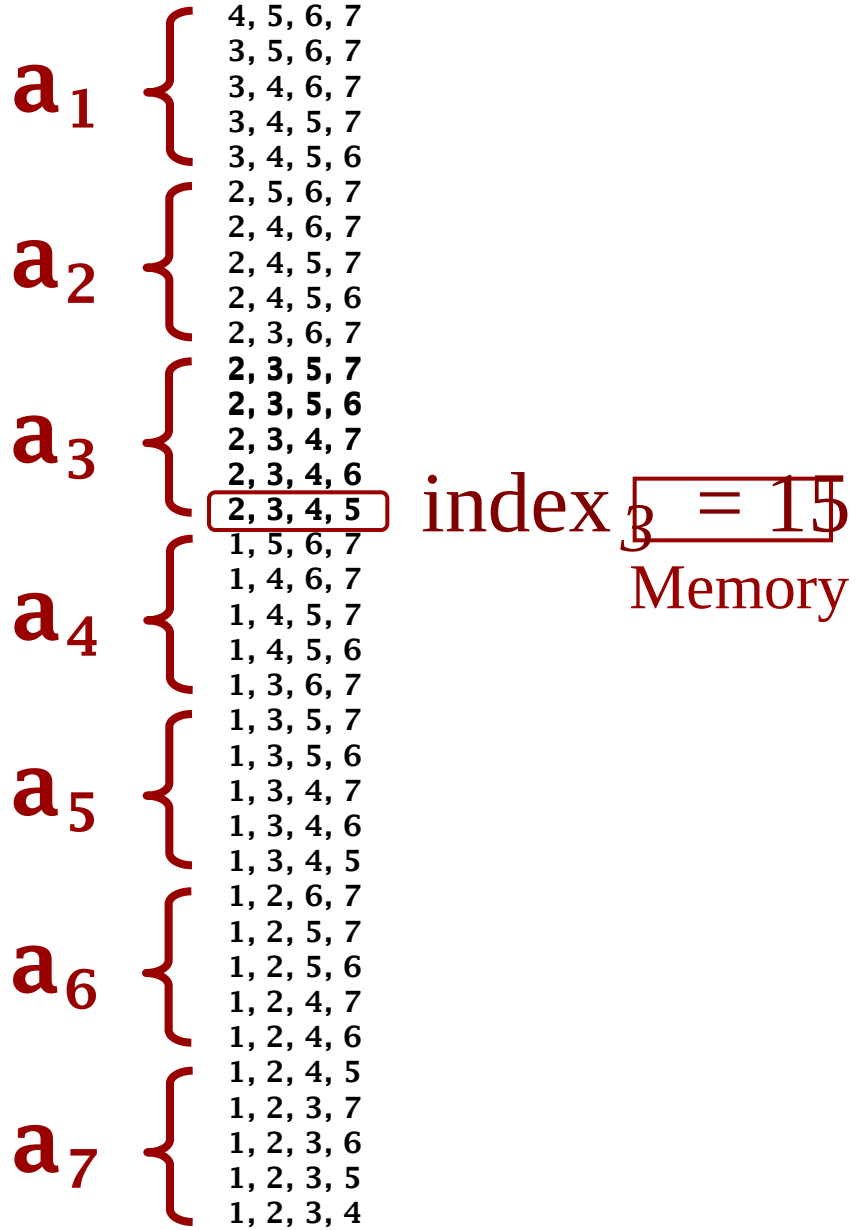
$$C_s^n = \binom{n}{s} = \frac{n!}{(n-s)! \times s!}$$

The number of coalitions in agent a_i 's share is :

$$n_i = \frac{C_s^n}{n}$$

and the share ends with the coalition located at:

$$\text{index} = i \times n_i$$



DCVC

L 4

For ***n*** agents, the number of coalitions of size ***s*** is:

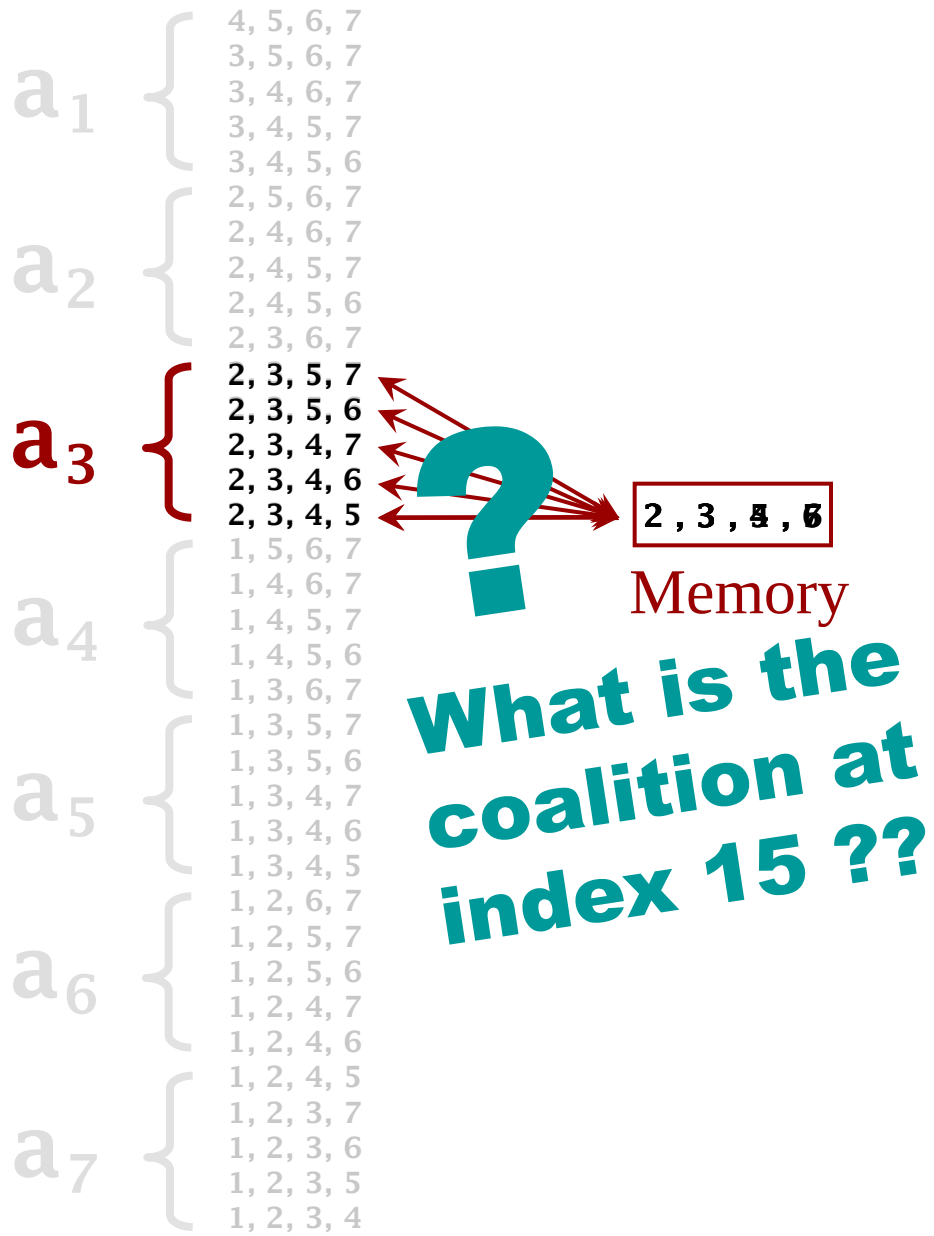
$$C_s^n = \binom{n}{s} = \frac{n!}{(n-s)!\ \text{x}\ s!}$$

The number of coalitions in agent a_i 's share is :

$$n_i = \frac{C_s^n}{n}$$

and the share ends with the coalition located at:

$$\text{index} = i \ \text{x}\ n_i$$



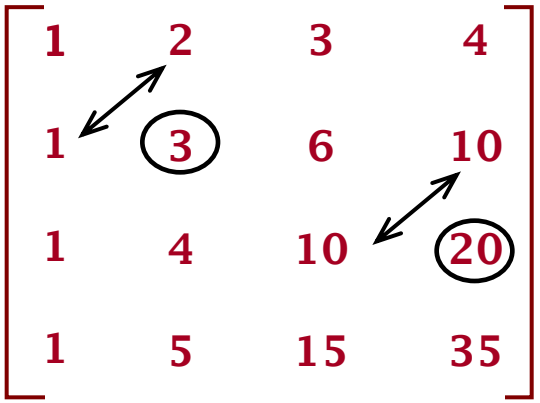
DCVC

By ordering L_S as described earlier, we show that the following holds:

Any coalition which starts with $(n - s + 1) - i + 1$ must have an index k

$$\sum_{j=1}^{i} |s + j - 2, s - 1| \;<\; k \;\leq\; \sum_{j=1}^{i+1} |s + j - 2, s - 1|$$

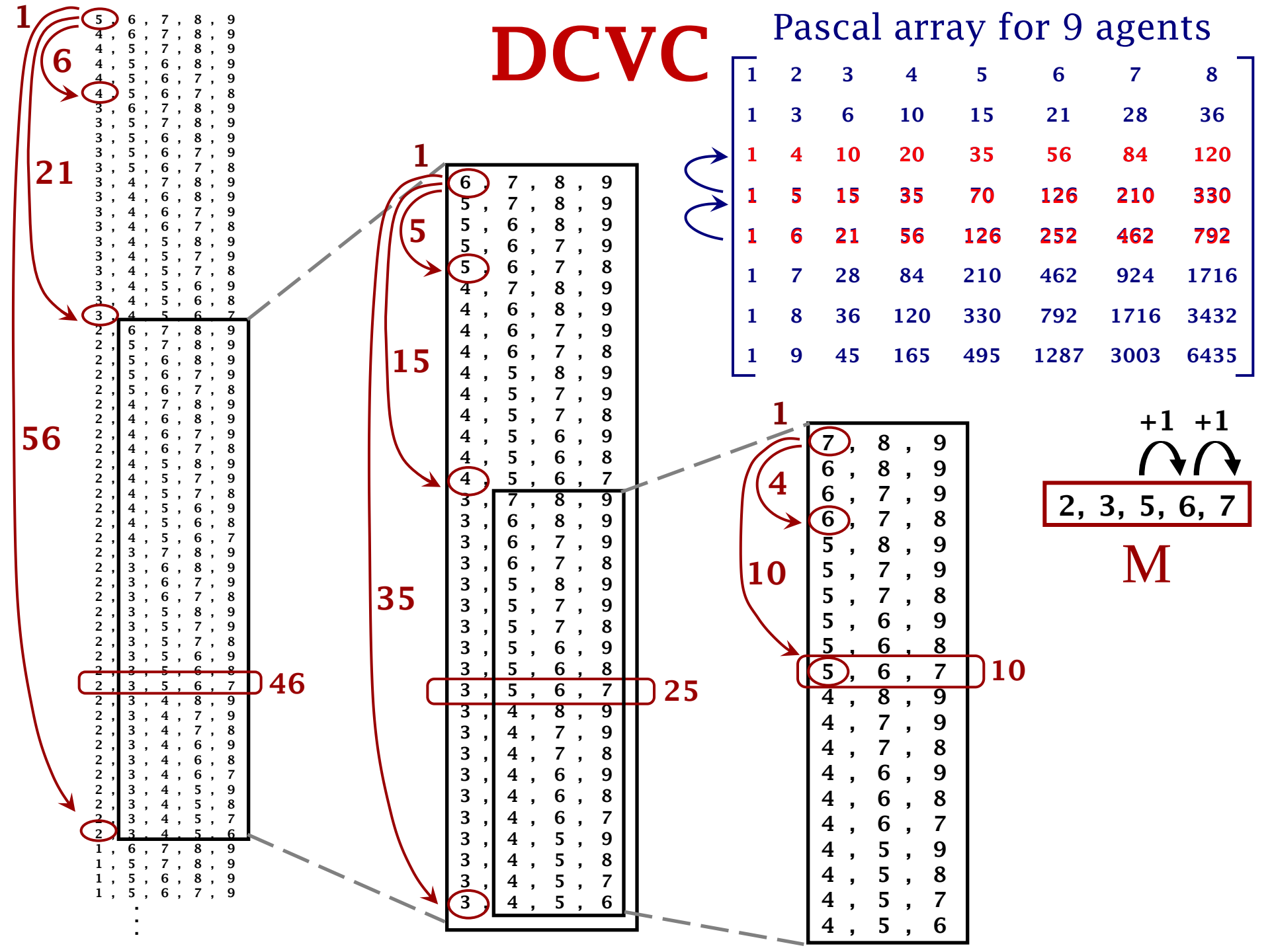
Each agent forms what we call a Pascal array (which is of size $n - 1 * n - 1$)



Pascal array for 4 agents

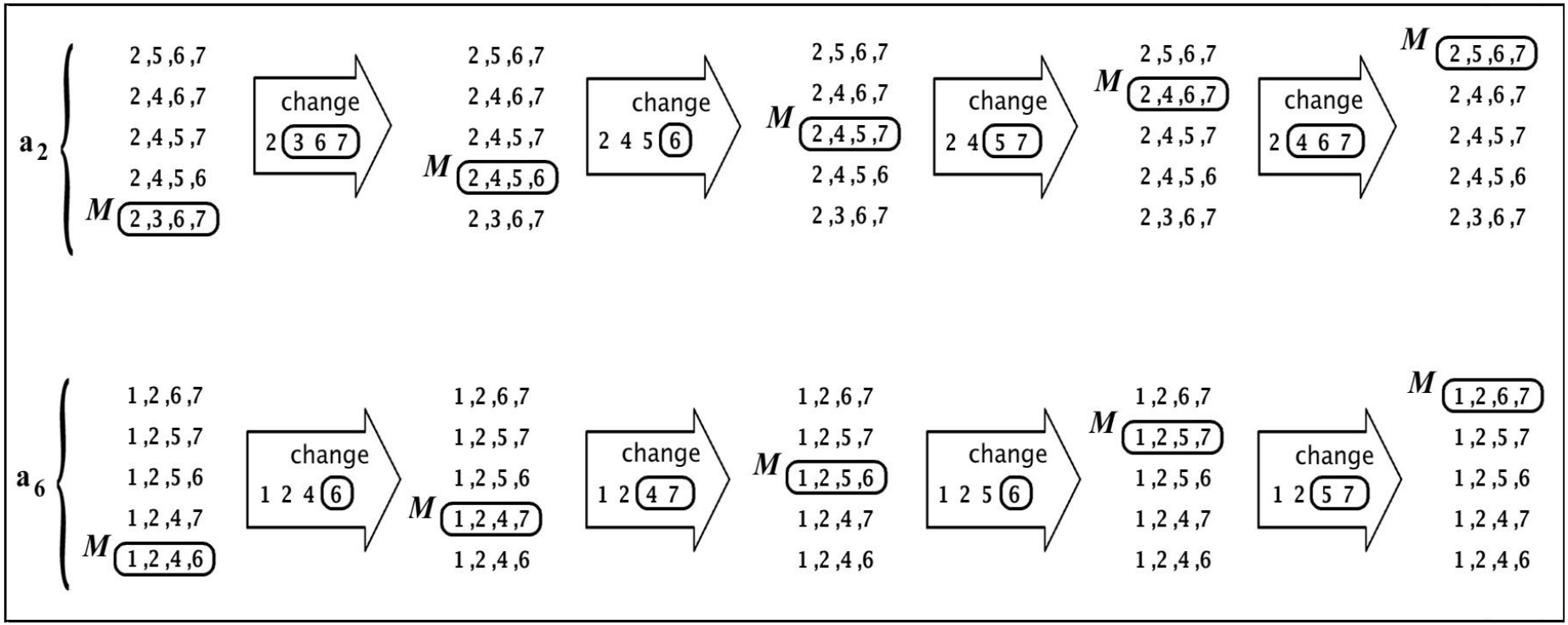
By this, the following equation holds:

$$Pascal[s, i] = \sum_{j=1}^{i} |s + j - 2, s - 1|$$

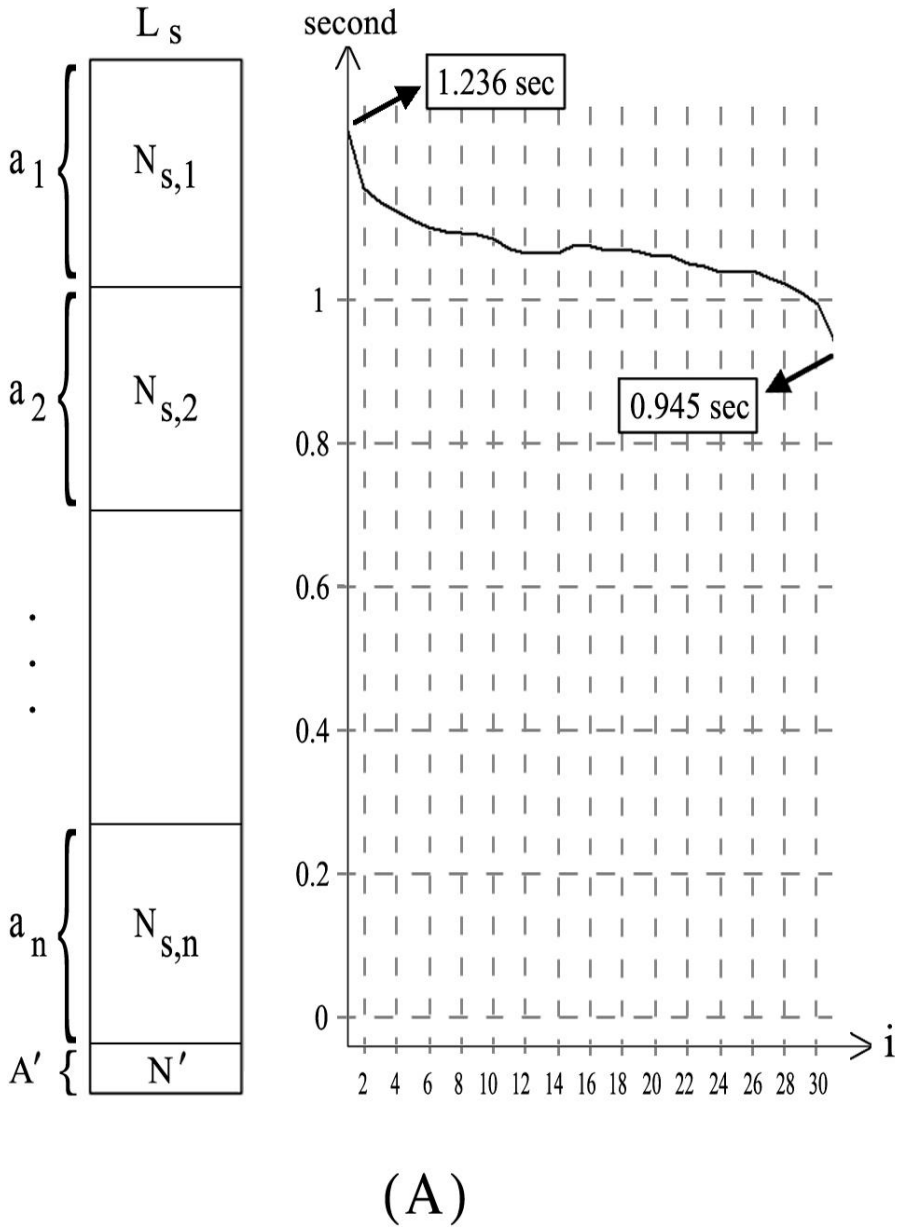

DCVC
Pascal array for 9 agents
1 2 3 4 5 6 7 8
1 3 6 10 15 21 28 36
1 4 10 20 35 56 84 120
1 5 15 35 70 126 210 330
1 6 21 56 126 252 462 792
1 7 28 84 210 462 924 1716
1 8 36 120 330 792 1716 3432
1 9 45 165 495 1287 3003 6435
+1 +1
2, 3, 5, 6, 7
M
1
6
21
56
46
1
5
15
35
25
1
4
10
10

DCVC

Example: Given 7 agents, let us compare the shares of agents a_2 and a_6, and that is for the coalitions of size 4

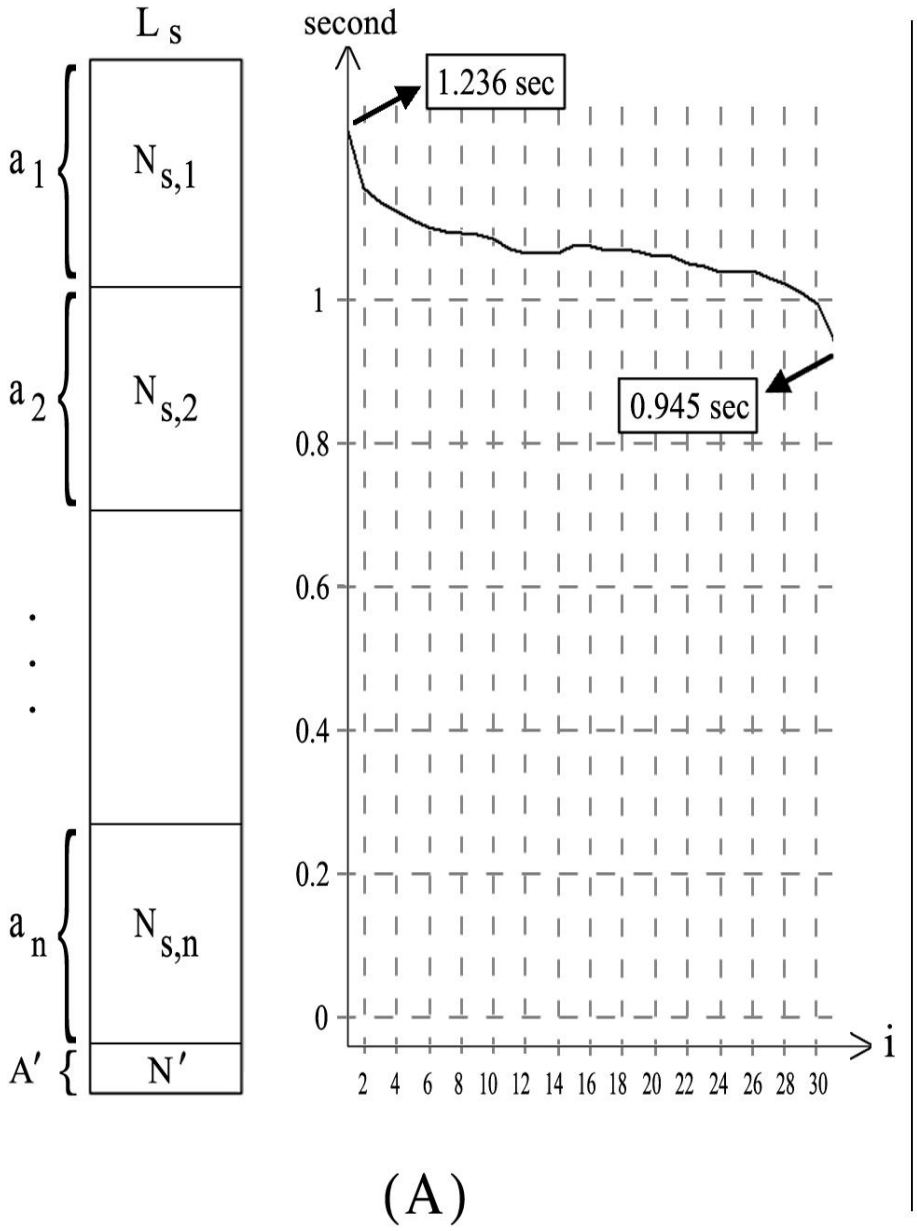


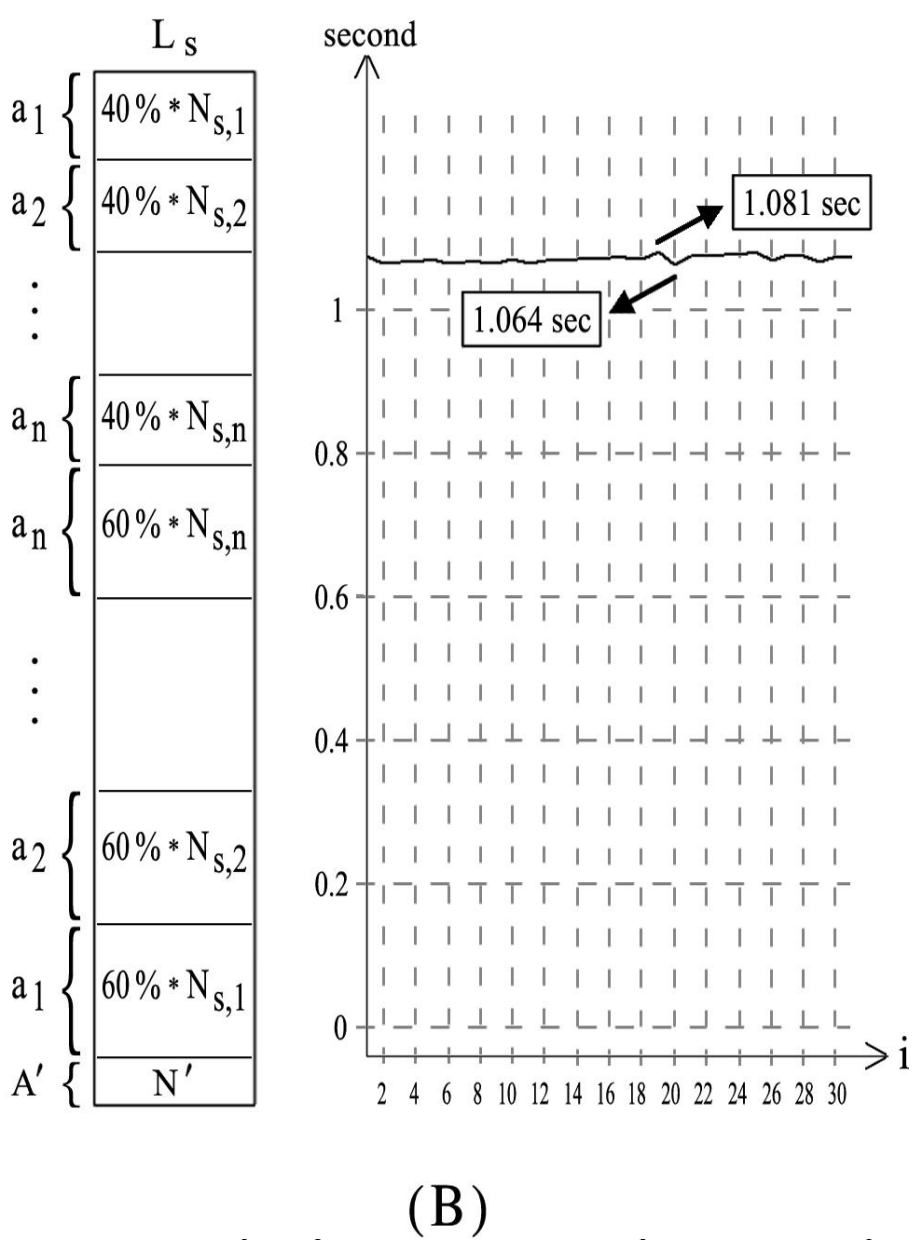
DCVC


L s
a 1
N s,1
a 2
N s,2
a n
N s,n
A′
N′
second
1.236 sec
0.945 sec
1
0.8
0.6
0.4
0.2
0
i
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30
(A)


How can we reduce the difference ?

DCVC


L s
a 1
N s,1
a 2
N s,2
a n
N s,n
A′
N′
second
1.236 sec
0.945 sec
1
0.8
0.6
0.4
0.2
0
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30
i
(A)


L s
a 1
40 % * N s,1
a 2
40 % * N s,2
a n
40 % * N s,n
a n
60 % * N s,n
a 2
60 % * N s,2
a 1
60 % * N s,1
A′
N′
second
1.081 sec
1.064 sec
1
0.8
0.6
0.4
0.2
0
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30
i
(B)

Evaluating DCVC

We compared DCVC against *Shehory & Kraus*'s algorithm, and that is based on the following metrics:

- The time required for the distribution process
- The total number of redundant calculations
- The communication (i.e., total number of bytes that were sent between the agents)
- Fairness (i.e., the difference between the agent that had the biggest share and the one that had the smallest)
- Memory (i.e., the minimum number of bytes that are required per agent to carry out the calculations)

Evaluating DCVC

Time:

The time required (in seconds) for the distribution process

	DCVC	Shehory
16 agents	< 0.01	2.77
17 agents	< 0.01	6.11
18 agents	< 0.01	13.76
19 agents	< 0.01	32.29
20 agents	< 0.01	72.27
21 agents	< 0.01	159.89
22 agents	< 0.01	372.58
23 agents	< 0.01	881.64
24 agents	0.01	2280.43
25 agents	0.02	5298.52

Evaluating DCVC

Redundancy:

The total number of redundant values that were calculated

	DCVC	Shehory
16 agents	0	513452
17 agents	0	1208715
18 agents	0	2583828
19 agents	0	5506420
20 agents	0	11659720
21 agents	0	24605666
22 agents	0	52170535
23 agents	0	108933551
24 agents	0	201504067
25 agents	0	477826101

Evaluating DCVC

Communication:

The total number of bytes that were sent between the agents

	DCVC	Shehory
16 agents	0	735408
17 agents	0	2350481
18 agents	0	4974743
19 agents	0	10538129
20 agents	0	22152227
21 agents	0	46512635
22 agents	0	97957698
23 agents	0	204911555
24 agents	0	429009502
25 agents	0	1188779705

Evaluating DCVC

Fairness:

The difference between the agent that had the biggest share and the one that had the smallest

	DCVC	Shehory
16 agents	1	8424
17 agents	1	12886
18 agents	1	26071
19 agents	1	52890
20 agents	1	103484
21 agents	1	208931
22 agents	1	454812
23 agents	1	880428
24 agents	1	2191528
25 agents	1	3043149

Evaluating DCVC

Memory:

The minimum number of bytes required per agent to carry out the necessary calculations

	DCVC	Shehory
16 agents	2	65536
17 agents	3	196608
18 agents	3	393216
19 agents	3	786432
20 agents	3	1572864
21 agents	3	3145728
22 agents	3	6291456
23 agents	3	12582912
24 agents	3	25165824
25 agents	4	67108864

Evaluating DCVC

To summarize: When comparing the performance of DCVC with that of *Shehory and Kraus*'s algorithm, given 25 agents, we find that:

- The distribution process of DCVC took **0.0005%** of the time
- The values were calculated using **0.000006%** of the memory
- The calculation redundancy dropped **from 353579392 to 0**
- The communication was reduced **from 674047872 to 0** bytes
- These improvements become **exponentially better** for larger numbers of agents

Coalition Value *re*-Calculation

The agents are ***continuously forming coalitions*** whenever necessary, and any of the formed coalitions can be dissolved whenever it is necessary or beneficial.

After a coalition is formed, the ***coalitional values might need to be re-calculated*** before the agents can form another.

The re-calculation process might differ from the initial calculation process since ***some agents might no longer be able to join other coalitions.***

Coalition Value *re*-Calculation

Let **A** be the set of agents, and **A*** be the subset of **A** in which every agent can join a coalition

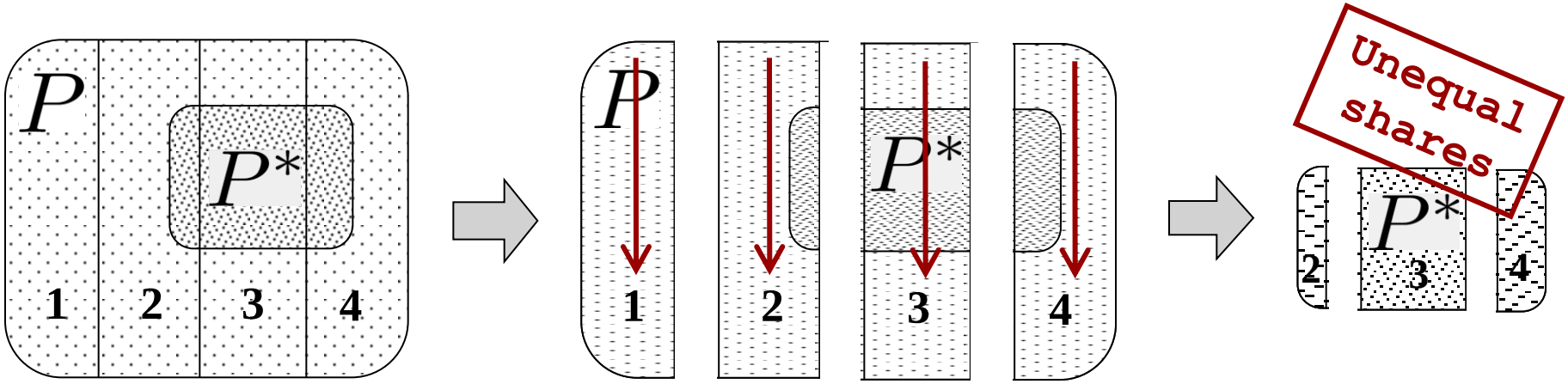
Let **P** be the set of all possible coalitions, and **P*** be the subset of **P** in which every coalition contains only members of **A***

Based on this, every time the agents need to form a coalition, they only need to consider the coalitions that belong to **P***

Coalition Value *re*-Calculation

(Distributing P* by Searching through P)

By having each agent search through its share of ***P***, the re-calculation is carried out in a distributed manner, However…

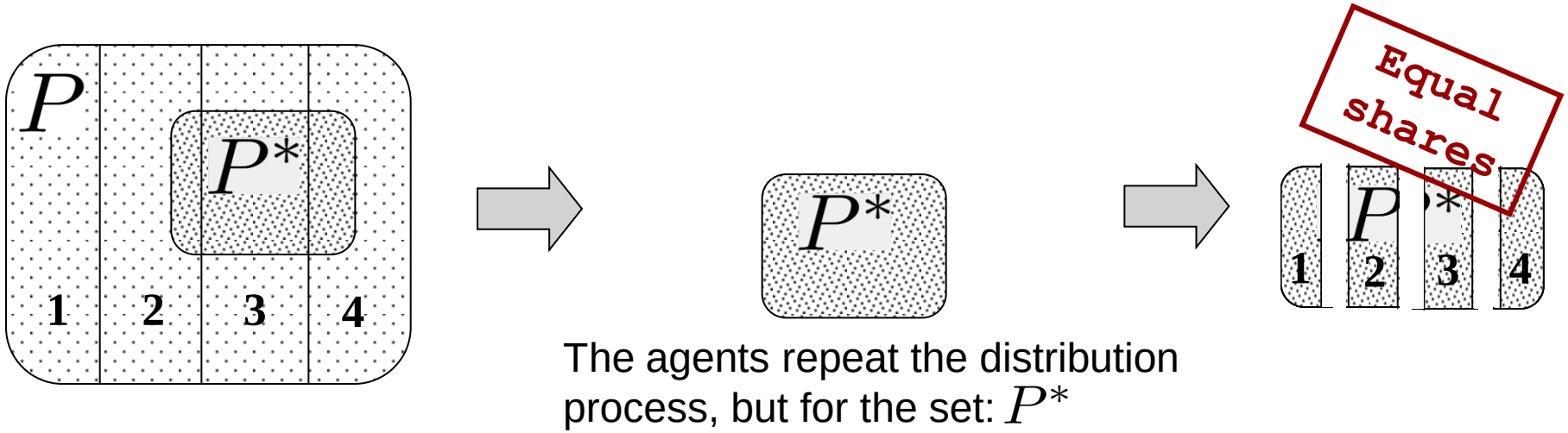


Every agent searches its share of: P

Coalition Value *re*-Calculation

(repeat distribution process, but for P instead of P*)

The only way to obtain an optimal distribution of **P*** is to initially distribute **P*** (instead of **P**), and then repeat the entire distribution process of **P*** whenever **A*** is changed.



The agents repeat the distribution process, but for the set: P^*

When using Shehory & Kraus's algorithm, this is inapplicable due to:

- The time required for each agent to re-build its share in memory.
- The communication that is required every time the distribution process is performed.

Coalition Value *re*-Calculation

(repeat distribution process, but for P instead of P*)

DCVC is the first distribution algorithm available in which:

- No communication between the agents is necessary.
- The agents do not need to build any lists in memory.

This makes repeating the distribution process applicable.

Evaluation: we compare the two methods (i.e., searching through P, and repeating the entire distribution process using DCVC), and that is based on the following metrics:

- Equality of the agents' shares
- Number of operations performed
- Memory requirements

Coalition Value *re*-Calculation

(evaluation)

Equality of the agents' shares:

The difference between the agent that had the biggest share of calculations, and the one that had the smallest

Size of A*	Repeat the distribution	Search through P
29	1	4,445,182
28	1	5,105,232
27	1	3,808,067
26	1	1,482,807
25	1	679,789
24	1	382,358
23	1	213,289
22	1	116,533
21	1	62,869
20	1	31,964
19	1	16,638
18	1	8,290
17	1	4,431
16	1	2,627
15 . .	. . .	. . .

Coalition Value *re*-Calculation

(evaluation)

Memory requirements:

The table shows the results given different No. of agents

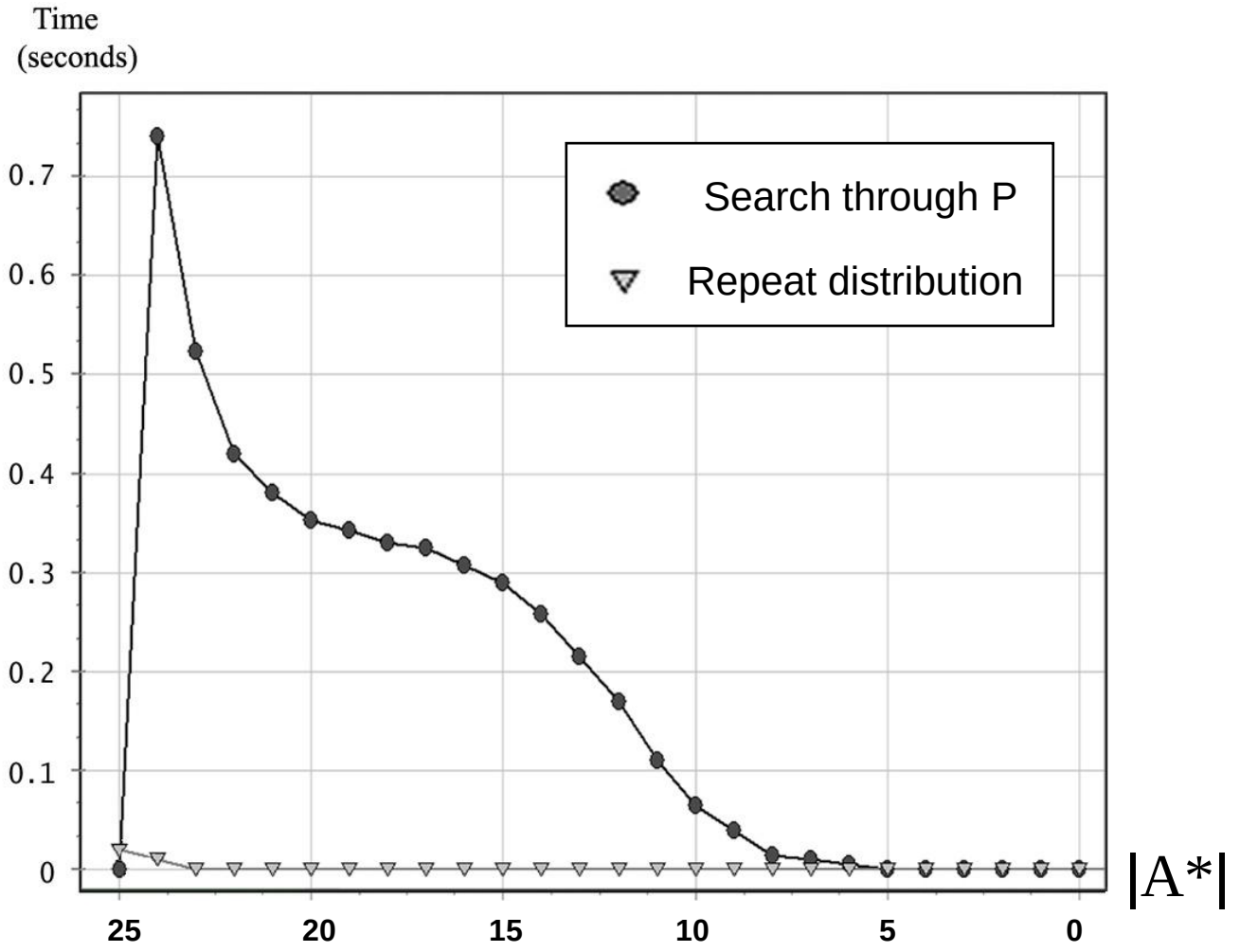
Number of agents	DCVC	Search through P
10	2	206
11	2	374
12	2	684
13	2	1,262
14	2	2,342
15	2	4,370
16	2	8,192
17	3	23,133
18	3	43,692
19	3	82,785
20	3	157,287
21	3	299,595
22	3	571,953
23	3	1,094,169
24	3	2,097,153
25	4	5,368,712

Coalition Value *re*-Calculation

(evaluation)

Run time:

Given 25 agents with different sizes of A*



Question

What if we have identical agents?

For example, what if the set of agents was as follows: **A = {4,4,4,5,5,6}**

What are the possible coalitions of {4,4,4,5,5,6} ?

The possible subsets of {1,2,3,4,5,6}:

1	1,2	1,2,3	1,2,3,4	1,2,3,4,5	1,2,3,4,5,6
2	1,3	1,2,4	1,2,3,5	1,2,3,4,6	
3	1,4	1,2,5	1,2,3,6	1,2,3,5,6	
4	1,5	1,2,6	1,2,4,5	1,2,4,5,6	
5	1,6	1,3,4	1,2,4,6	1,3,4,5,6	
6	2,3	1,3,5	1,2,5,6	2,3,4,5,6	
	2,4	1,3,6	1,3,4,5		
	2,5	1,4,5	1,3,4,6		
	2,6	1,4,6	1,3,5,6		
	3,4	1,5,6	1,4,5,6		
	3,5	2,3,4	2,3,4,5		
	3,6	2,3,5	2,3,4,6		
	4,5	2,3,6	2,3,5,6		
	4,6	2,4,5	2,4,5,6		
	5,6	2,4,6	3,4,5,6		
		2,5,6			
		3,4,5			
		3,4,6			
		3,5,6			
		4,5,6			

The possible subsets of {4,4,4,5,5,6}:

4	4,4	4,4,4	4,4,4,5	4,4,4,5,5	4,4,4,5,5,6
~~4~~	~~4,4~~	4,4,5	~~4,4,4,5~~	4,4,4,5,6	
~~4~~	4,5	4,4,5	4,4,4,6	~~4,4,4,5,6~~	
5	~~4,5~~	4,4,6	4,4,5,5	4,4,5,5,6	
~~5~~	4,6	~~4,4,5~~	4,4,5,6	~~4,4,5,5,6~~	
6	~~4,4~~	~~4,4,5~~	~~4,4,5,6~~	~~4,4,5,5,6~~	
	~~4,5~~	~~4,4,6~~	~~4,4,5,5~~		
	~~4,5~~	4,5,5	~~4,4,5,6~~		
	~~4,6~~	4,5,6	~~4,4,5,6~~		
	~~4,5~~	~~4,5,6~~	4,5,5,6		
	~~4,5~~	~~4,4,5~~	~~4,4,5,5~~		
	~~4,6~~	~~4,4,5~~	~~4,4,5,6~~		
	5,5	~~4,4,6~~	~~4,4,5,6~~		
	5,6	~~4,5,5~~	~~4,5,5,6~~		
	~~5,6~~	~~4,5,6~~	~~4,5,5,6~~		
		~~4,5,6~~			
		~~4,5,5~~			
		~~4,5,6~~			
		~~4,5,6~~			
		5,5,6			

What we really need is this:

4	4,4	4,4,4	4,4,4,5	4,4,4,5,5	4,4,4,5,5,6
5	4,5	4,4,5	4,4,4,6	4,4,4,5,6	
6	4,6	4,4,6	4,4,5,5	4,4,5,5,6	
	5,5	4,5,5	4,4,5,6		
	5,6	4,5,6	4,5,5,6		
		5,5,6			

What are the possible coalitions of {4,4,4,5,5,6} ?

4
4
4,4
4,4,4

5
5
5,5

6
6

4,5
4,5
4,5,5
4,4,5
4,4,5,5
4,4,4,5
4,4,4,5,5

4,6
4,6
4,4,6
4,4,4,6

5,6
5,6
5,5,6

4,5,6
4,5,6
4,5,5,6
4,4,5,6
4,4,5,5,6
4,4,4,5,6
4,4,4,5,5,6

Algorithm

- **Get the underlying set of {4,4,4,5,5,6}**

  The result is: {4,5,6}

- **Get the possible coalitions of {4,5,6}**

  {4}, {5}, {6}, {4,5}, {4,6}, {5,6}, {4,5,6}

- **If element "i" is repeated "x" times, then put "i" once, then twice, then 3 times, ..., then "x" times.**

  For example, "4" is repeated 3 times, so we put "4", then "4,4", then "4,4,4"

What we really need is this:

4 5 6	4,4 4,5 4,6 5,5 5,6	4,4,4 4,4,5 4,4,6 4,5,5 4,5,6 5,5,6	4,4,4,5 4,4,4,6 4,4,5,5 4,4,5,6 4,5,5,6	4,4,4,5,5 4,4,4,5,6 4,4,5,5,6	4,4,4,5,5,6

Exercise

What are the possible coalitions of {5,5,6,6,7,7} ?

What are the possible coalitions of {5,5,6,6,7,7} ?

5
5
5,5

6
6
6,6

7
7
7,7

5,6
5,6
5,6,6
5,5,6
5,5,6,6

5,7
5,7
5,7,7
5,5,7
5,5,7,7

6,7
6,7
6,7,7
6,6,7
6,6,7,7

5,6,7
5,6,7
5,6,7,7
5,6,6,7
5,6,6,7,7
5,5,6,7
5,5,6,7,7
5,5,6,6,7
5,5,6,6,7,7

Algorithm

- **Get the underlying set of {5,5,6,6,7,7}**

  The result is: {5,6,7}

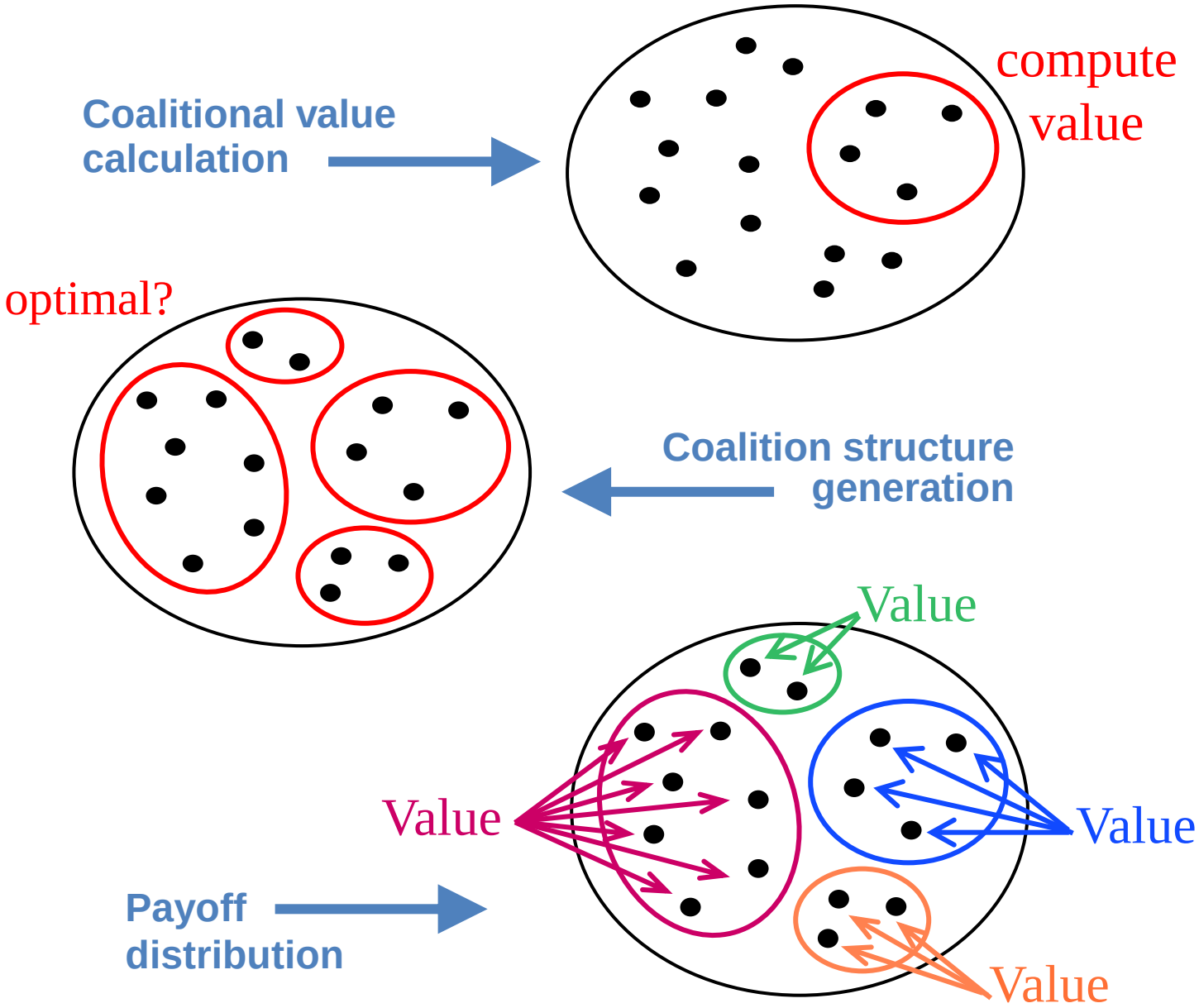
- **Get the possible coalitions of {5,6,7}**

  {5}, {6}, {7}, {5,6}, {5,7}, {6,7}, {5,6,7}

- **If element "i" is repeated "x" times, then put "i" once, then twice, then 3 times, ..., then "x" times.**

  For example, "5" is repeated 3 times, so we put "5", then "5,5", then "5,5,5"

Coalition formation process



Coalition Structure Generation

(in Characteristic Function Games)

Given 3 agents, the **set of agents** is:

$\{a_1,a_2,a_3\}$

The possible **coalitions** are:

$\{a_1\}$	$\{a_2\}$	$\{a_3\}$	$\{a_1,a_2\}$	$\{a_1,a_3\}$	$\{a_2,a_3\}$	$\{a_1,a_2,a_3\}$
20	40	30	70	40	65	95

The possible **coalition structures**:

$\{\{a_1\},\{a_2\},\{a_3\}\}$	$\{\{a_1,a_2\},\{a_3\}\}$	$\{\{a_2\},\{a_1,a_3\}\}$	$\{\{a_1\},\{a_2,a_3\}\}$	$\{\{a_1,a_2,a_3\}\}$
20+40+30=90	70+30=100	40+40=80	20+65=85	95

Input: a value of every possible coalition

Output: a coalition structure in which the sum of values is maximized

L_1	value	L_2	value	L_3	value	L_4	value
{1}	30	{1, 2}	50	{1, 2, 3}	90	{1,2,3,4}	140
{2}	40	{1, 3}	60	{1, 2, 4}	120		
{3}	25	{1, 4}	80	{1, 3, 4}	100		
{4}	45	{2, 3}	55	{2, 3, 4}	115		
		{2, 4}	70				
		{3, 4}	80				

Exercise

What is the optimal coalition structure ?

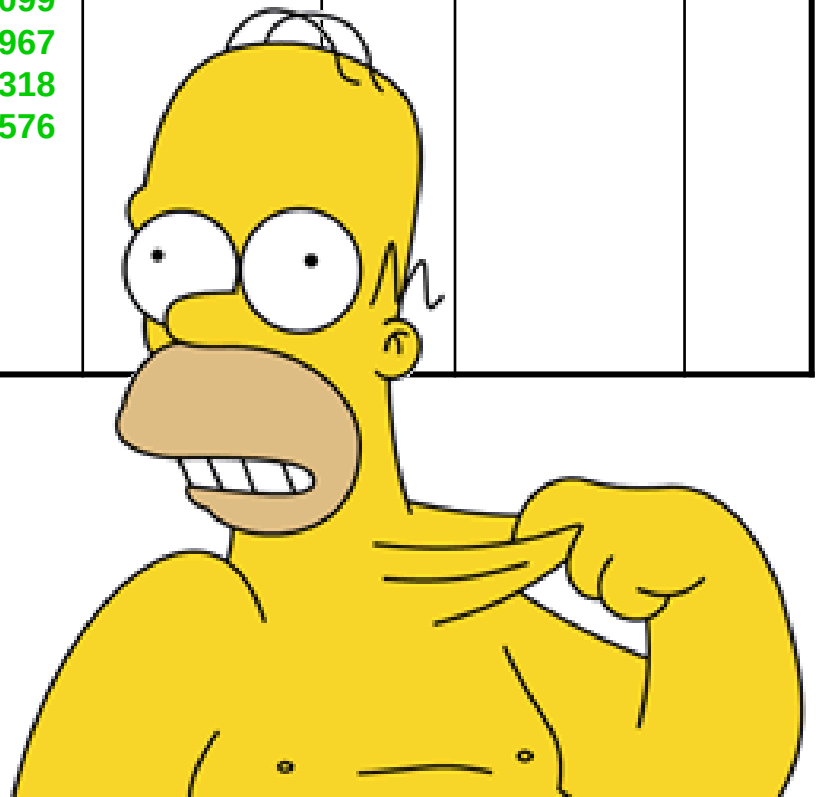
Answer

{ {1}, {2}, {3,4} }

L_1	value	L_2	value	L_3	value	L_4	value	L_5	value	L_6	value
1	826	1, 2	1750	1, 2, 3	3352	1, 2, 3, 4	4355	1, 2, 3, 4, 5	4804	1,2, 3, 4, 5, 6	6217
2	1108	1, 3	1670	1, 2, 4	3102	1, 2, 3, 5	4373	1, 2, 3, 4, 6	5657		
3	1065	1, 4	1989	1, 2, 5	3301	1, 2, 3, 6	3770	1, 2, 3, 5, 6	4609		
4	890	1, 5	1664	1, 2, 6	3119	1, 2, 4, 5	3528	1, 2, 4, 5, 6	4829		
5	907	1, 6	2023	1, 3, 4	3287	1, 2, 4, 6	3967	1, 3, 4, 5, 6	5650		
6	1024	2, 3	2083	1, 3, 5	2696	1, 2, 5, 6	3647	2, 3, 4, 5, 6	5852		
		2, 4	2272	1, 3, 6	2950	1, 3, 4, 5	4142				
		2, 5	2082	1, 4, 5	3324	1, 3, 4, 6	3875				
		2, 6	1995	1, 4, 6	2460	1, 3, 5, 6	3905				
		3, 4	1807	1, 5, 6	3134	1, 4, 5, 6	3645				
		3, 5	2529	2, 3, 4	2943	2, 3, 4, 5	3850				
		3, 6	2045	2, 3, 5	2956	2, 3, 4, 6	4099				
		4, 5	1683	2, 3, 6	2950	2, 3, 5, 6	3967				
		4, 6	2115	2, 4, 5	3661	2, 4, 5, 6	3318				
		5, 6	1956	2, 4, 6	2618	3, 4, 5, 6	3576				
				2, 5, 6	2906						
				3, 4, 5	2769						
				3, 4, 6	3070						
				3, 5, 6	3135						
				4, 5, 6							

Exercise

What is the optimal coalition structure ?



Related Work

- Dynamic Programming techniques

IDP [Rahwan & Jennings N. R., *An Improved Dynamic Programming Algorithm for Coalition Structure Generation*, AAMAS 2008].

- Anytime with guarantees on solution quality

IP [Rahwan et al., *An Anytime Algorithm for Optimal Coalition Structure Generation*, AAAI 2007, JAIR 2009].

Property \ Algorithm	IDP	IP
Worst case performance	$O(3^n)$ ✓	$O(n^n)$ ✗
Return solutions anytime	False ✗	True ✓
Time to return optimal solution	Slow ✗	Fast ✓

The Dynamic Programming (DP) Algorithm

The Dynamic Programming (DP) algorithm is based on the following observation:

if $CS^* = \{ C_1, C_2, C_3, C_4, C_5, C_6, C_7 \}$ is an optimal CS

then $\{ \quad , \quad , \quad \}$ is an optimal partition of: $C_2 \cup C_5 \cup C_6$

DP performs a number of evaluations, and stores the results in a table. In the next slide, we show an example of 4 agents

size	coalition	Evaluations performed before setting f		Best split	f
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
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
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		
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		
		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		

L_1	value	L_2	value	L_3	value	L_4	value
{1}	50	{1, 2}	90	{1, 2, 3}	125	{1,2,3,4}	160
{2}	30	{1, 3}	80	{1, 2, 4}	120		
{3}	45	{1, 4}	65	{1, 3, 4}	135		
{4}	35	{2, 3}	90	{2, 3, 4}	115		
		{2, 4}	70				
		{3, 4}	60				

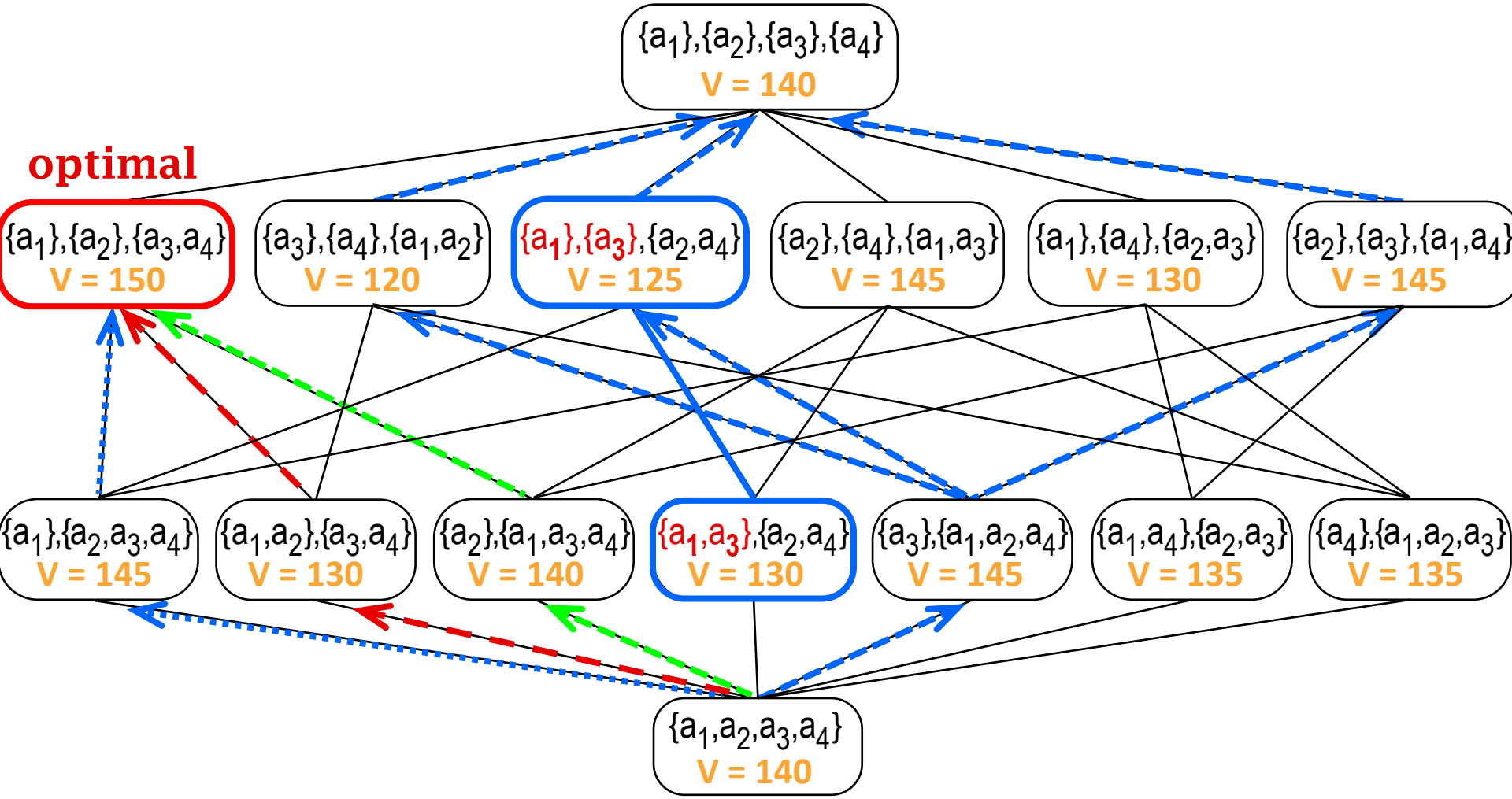
Exercise

Use Dynamic Programming to find the optimal coalition structure

size	coalition	Evaluations performed before setting f		Best Split	f
1	{1}	v({1})=50		{1}	50
	{2}	v({2})=30		{2}	30
	{3}	v({3})=45		{3}	45
	{4}	v({4})=35		{4}	35
2	{1,2}	v({1,2})=90	f({1})+f({2})=80	{1,2}	90
	{1,3}	v({1,3})=80	f({1})+f({3})=95	{1} {3}	95
	{1,4}	v({1,4})=65	f({1})+f({4})= ?	?	?
	{2,3}	v({2,3})=90	f({2})+f({3})= ?	?	?
	{2,4}	v({2,4})=70	f({2})+f({4})= ?	?	?
	{3,4}	v({3,4})=60	f({3})+f({4})= ?	?	?
3	{1,2,3}	v({1,2,3})=125 f({2})+f({1,3})=125	f({1})+f({2,3})=140 f({3})+f({1,2})=135	{1} {2,3}	140
	{1,2,4}	v({1,2,4})=120 f({2})+f({1,4})=115	f({1})+f({2,4})=120 f({4})+f({1,2})=125	{4} {1,2}	125
	{1,3,4}	v({1,3,4})=135 f({3})+f({1,4})= ?	f({1})+f({3,4})= ? f({4})+f({1,3})= ?	?	?
	{2,3,4}	v({2,3,4})=115 f({3})+f({2,4})= ?	f({2})+f({3,4})= ? f({4})+f({2,3})= ?	?	?
4	{1,2,3,4}	v({1,2,3,4})=160 f({2})+f({1,3,4})=160 f({4})+f({1,2,3})= ? f({1,3})+f({2,4})= ?	f({1})+f({2,3,4})=175 f({3})+f({1,2,4})=170 f({1,2})+f({3,4})= ? f({1,4})+f({2,3})= ?	?	?

size	coalition	Evaluations performed before setting f		Best Split	f
1	{1}	v({1})=50		{1}	50
	{2}	v({2})=30		{2}	30
	{3}	v({3})=45		{3}	45
	{4}	v({4})=35		{4}	35
2	{1,2}	v({1,2})=90	f({1})+f({2})=80	{1,2}	90
	{1,3}	v({1,3})=80	f({1})+f({3})=95	{1} {3}	95
	{1,4}	v({1,4})=65	f({1})+f({4})=85	{1} {4}	85
	{2,3}	v({2,3})=90	f({2})+f({3})=75	{2,3}	90
	{2,4}	v({2,4})=70	f({2})+f({4})=65	{2,4}	70
	{3,4}	v({3,4})=60	f({3})+f({4})=80	{3} {4}	80
3	{1,2,3}	v({1,2,3})=125 f({2})+f({1,3})=125	f({1})+f({2,3})=140 f({3})+f({1,2})=135	{1} {2,3}	140
	{1,2,4}	v({1,2,4})=120 f({2})+f({1,4})=115	f({1})+f({2,4})=120 f({4})+f({1,2})=125	{4} {1,2}	125
	{1,3,4}	v({1,3,4})=135 f({3})+f({1,4})=130	f({1})+f({3,4})=130 f({4})+f({1,3})=130	{1,3,4}	135
	{2,3,4}	v({2,3,4})=115 f({3})+f({2,4})=115	f({2})+f({3,4})=110 f({4})+f({2,3})=125	{4} {2,3}	125
4	{1,2,3,4}	v({1,2,3,4})=160 f({2})+f({1,3,4})=160 f({4})+f({1,2,3})=175 f({1,3})+f({2,4})=165	f({1})+f({2,3,4})=175 f({3})+f({1,2,4})=170 f({1,2})+f({3,4})=170 f({1,4})+f({2,3})=175	{4} {1,2,3}	175

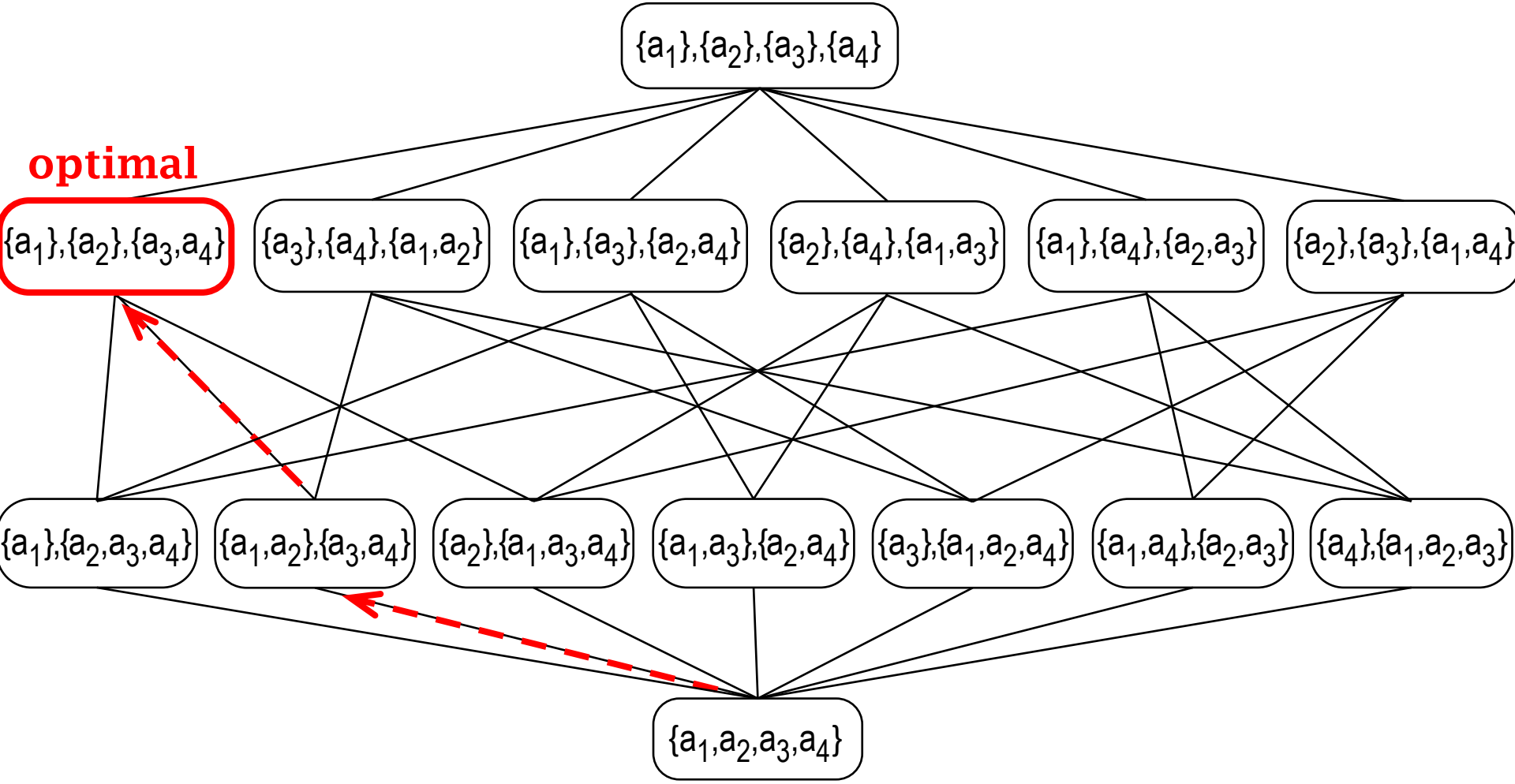
The Coalition Structure Graph



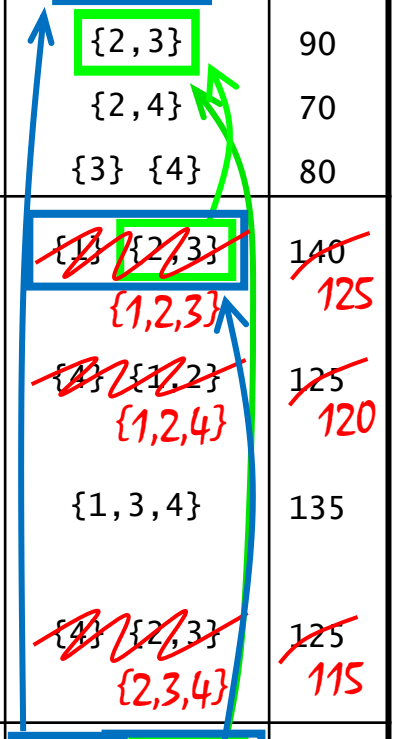
The Improved Dynamic Programming Algorithm (IDP)

- We define a subset of edges **E***
- We prove that the edges in **E*** are sufficient to form a path to every node in the graph
- We modify the original algorithm such that it only evaluates the movements through the edges in **E***

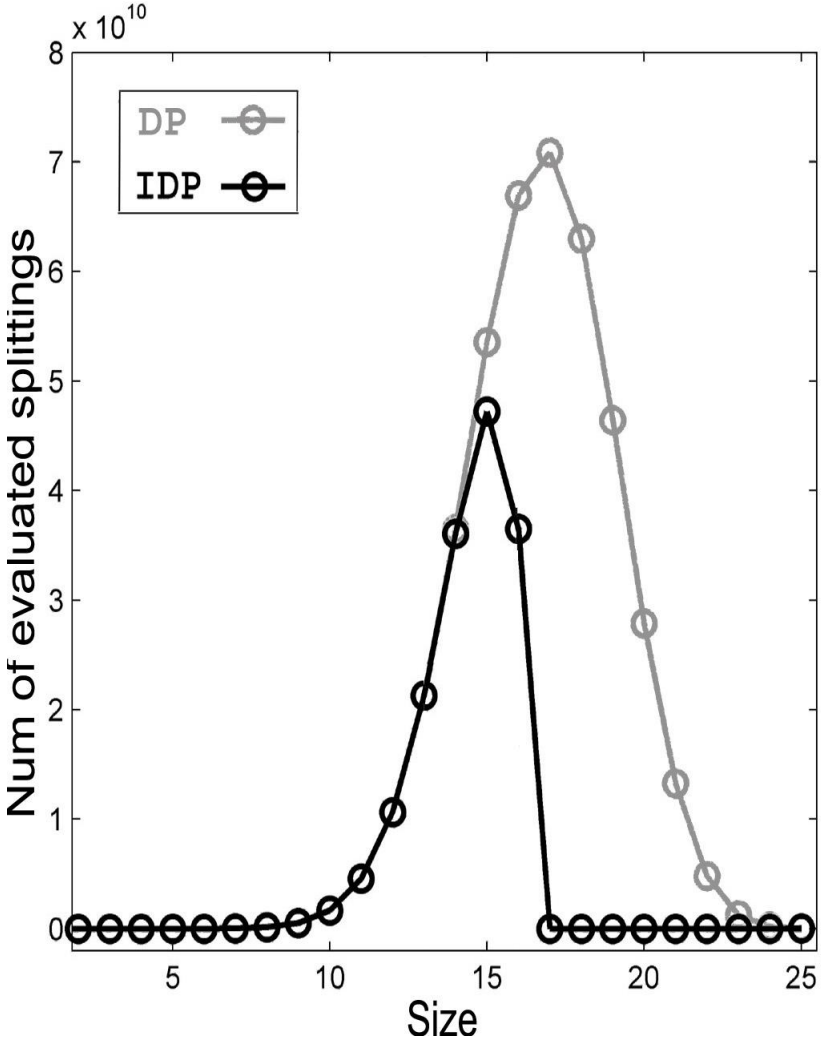
The Coalition Structure Graph

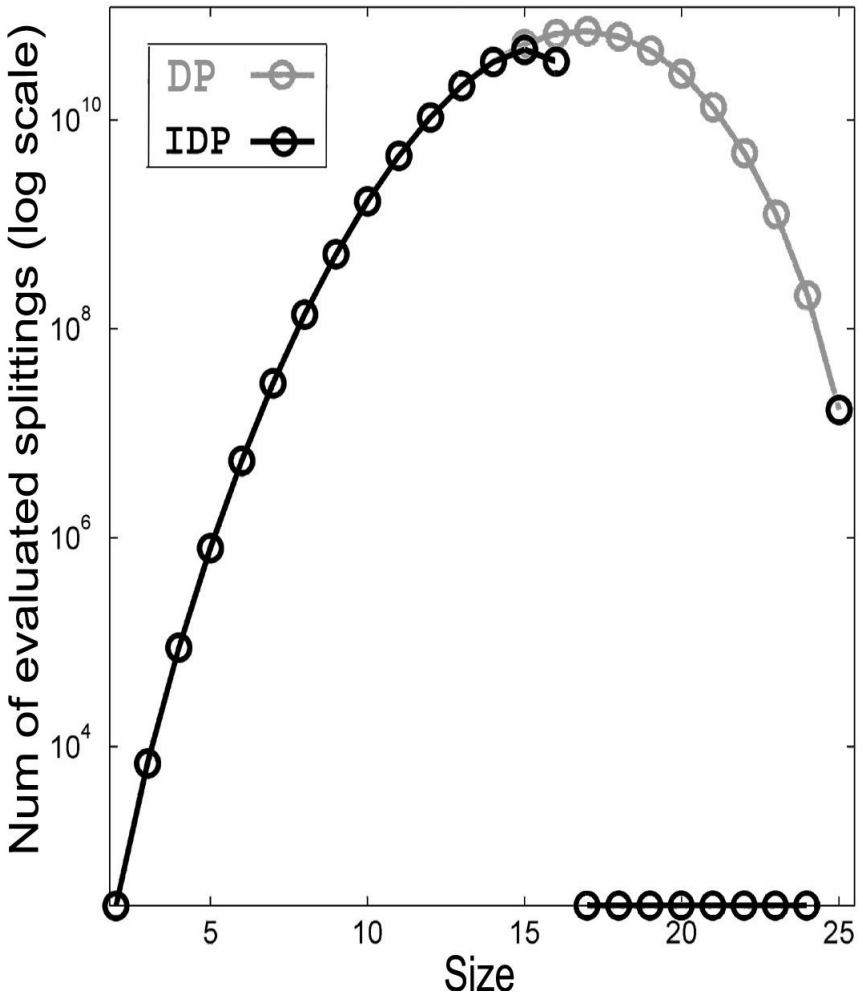


size	Coalition	Evaluations performed before setting f		Best split	f
1	{1}	v({1})=50		{1}	50
	{2}	v({2})=30		{2}	30
	{3}	v({3})=45		{3}	45
	{4}	v({4})=35		{4}	35
2	{1,2}	v({1,2})=90	f({1})+f({2})=80	{1,2}	90
	{1,3}	v({1,3})=80	f({1})+f({3})=95	{1} {3}	95
	{1,4}	v({1,4})=65	f({1})+f({4})=85	{1} {4}	85
	{2,3}	v({2,3})=90	f({2})+f({3})=75	{2,3}	90
	{2,4}	v({2,4})=70	f({2})+f({4})=65	{2,4}	70
	{3,4}	v({3,4})=60	f({3})+f({4})=80	{3} {4}	80
3	{1,2,3}	**v({1,2,3})=125** f({2})+f({1,3})=125	f({1})+f({2,3})=140 f({3})+f({1,2})=135	{1} {2,3}	140 → 125
	{1,2,4}	**v({1,2,4})=120** f({2})+f({1,4})=115	f({1})+f({2,4})=120 f({4})+f({1,2})=125	{4} {1,2}	125 → 120
	{1,3,4}	v({1,3,4})=135 f({3})+f({1,4})=130	f({1})+f({3,4})=130 f({4})+f({1,3})=130	{1,3,4}	135
	{2,3,4}	v({2,3,4})=115 f({3})+f({2,4})=115	f({2})+f({3,4})=110 f({4})+f({2,3})=125	{4} {2,3}	125 → 115
4	{1,2,3,4}	**v({1,2,3,4})=160** f({2})+f({1,3,4})=160 f({4})+f({1,2,3})=175 **f({1,3})+f({2,4})=165**	f({1})+f({2,3,4})=175 f({3})+f({1,2,4})=170 **f({1,2})+f({3,4})=170** f({1,4})+f({2,3})=175	{1}{4} {2,3}	175



Evaluation of IDP



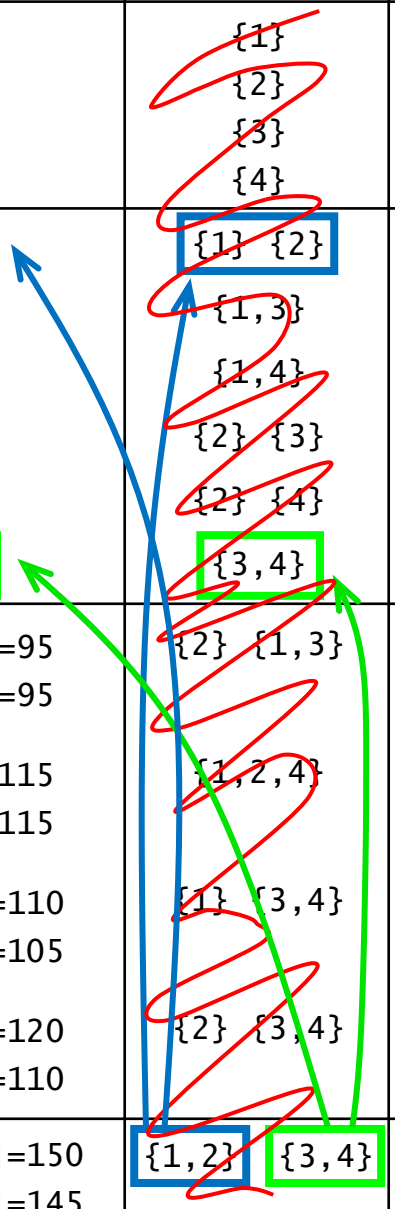


The total number of evaluations performed by IDP is only **38.7%** of those performed by the original dynamic programming (DP) algorithm

Question

How can we reduce the memory requirements ?

size	Coalition	Evaluations performed before setting f	Best split	f
1		v[{1}]=30 v[{2}]=40 v[{3}]=25 v[{4}]=45	{1} {2} {3} {4}	30 40 25 45
2		v[{1,2}]=50 f[{1}]+f[{2}]=70 v[{1,3}]=60 f[{1}]+f[{3}]=55 v[{1,4}]=80 f[{1}]+f[{4}]=75 v[{2,3}]=55 f[{2}]+f[{3}]=65 v[{2,4}]=70 f[{2}]+f[{4}]=85 v[{3,4}]=80 f[{3}]+f[{4}]=70	{1} {2} {1,3} {1,4} {2} {3} {2} {4} {3,4}	70 60 80 65 85 80
3		v[{1,2,3}]=90 f[{1}]+f[{2,3}]=95 f[{2}]+f[{1,3}]=100 f[{3}]+f[{1,2}]=95 v[{1,2,4}]=120 f[{1}]+f[{2,4}]=115 f[{2}]+f[{1,4}]=110 f[{4}]+f[{1,2}]=115 v[{1,3,4}]=100 f[{1}]+f[{3,4}]=110 f[{3}]+f[{1,4}]=105 f[{4}]+f[{1,3}]=105 v[{2,3,4}]=115 f[{2}]+f[{3,4}]=120 f[{3}]+f[{2,4}]=110 f[{4}]+f[{2,3}]=110	{2} {1,3} {1,2,4} {1} {3,4} {2} {3,4}	100 120 110 120
4		v[{1,2,3,4}]=140 f[{1}]+f[{2,3,4}]=150 f[{2}]+f[{1,3,4}]=150 f[{3}]+f[{1,2,4}]=145 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	{1,2} {3,4}	150



Related Work

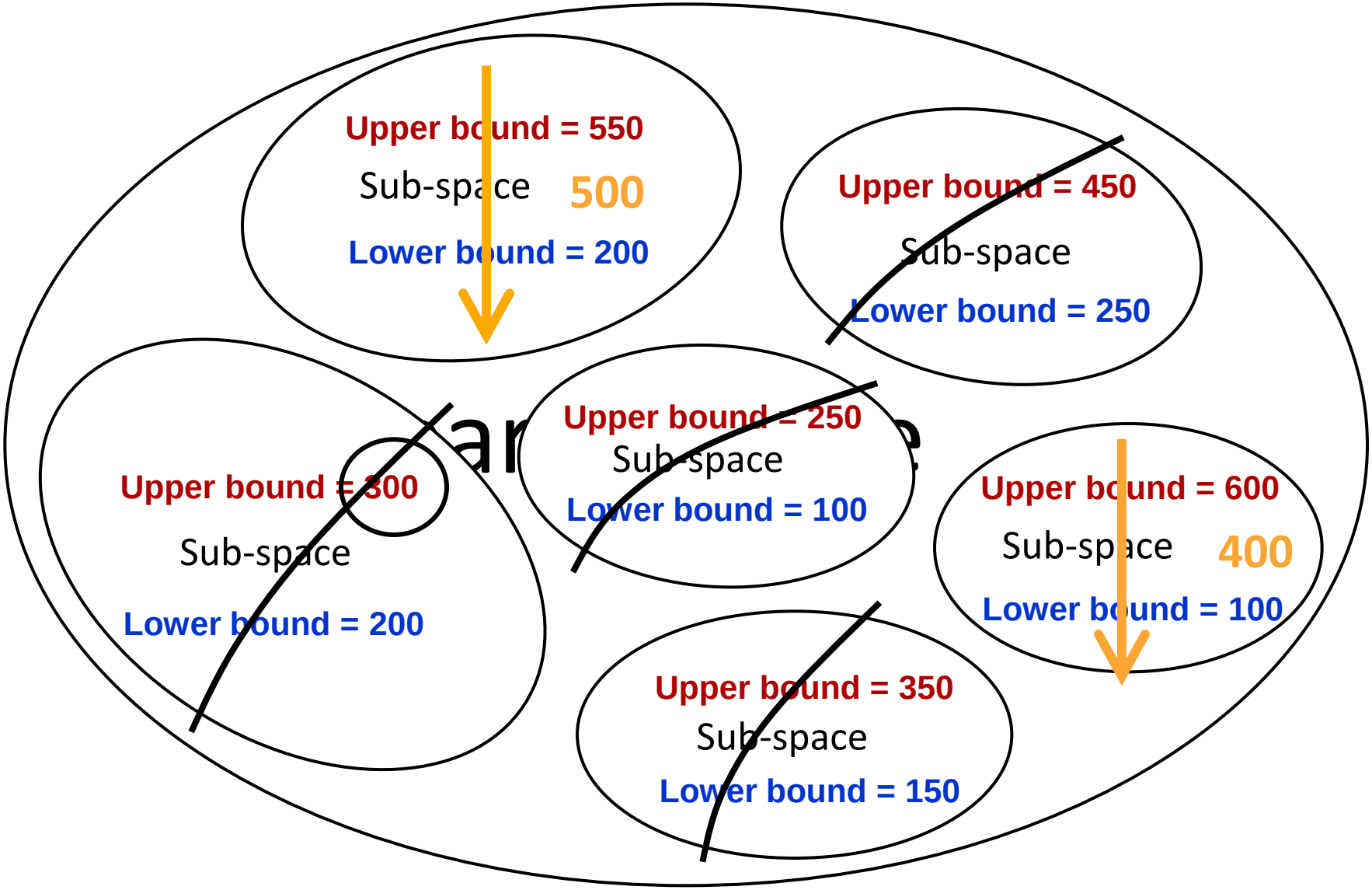
- Dynamic Programming techniques

  IDP [Rahwan & Jennings N. R., *An Improved Dynamic Programming Algorithm for Coalition Structure Generation*, AAMAS 2008].

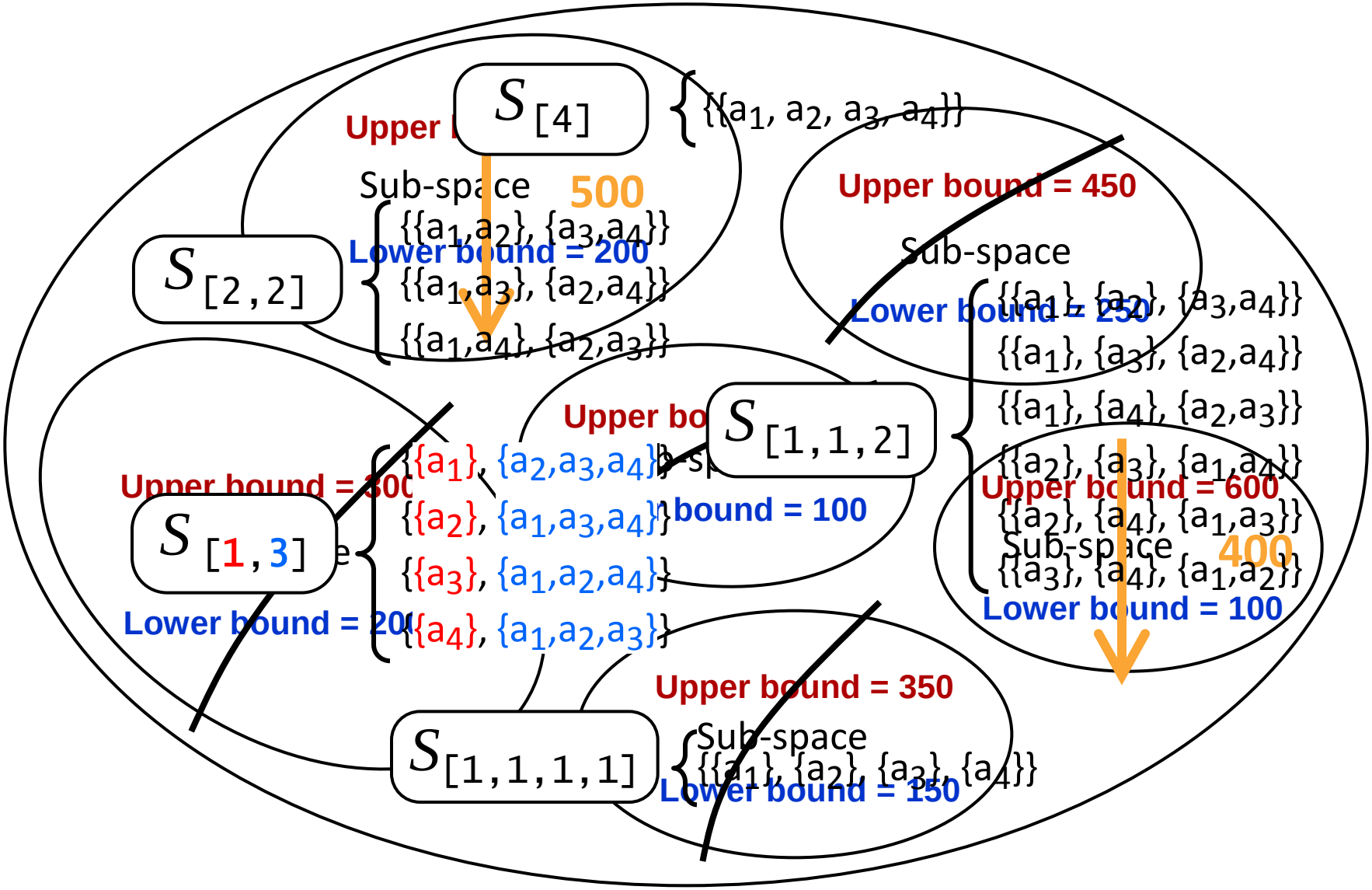
- Anytime with guarantees on solution quality

  IP [Rahwan et al., *An Anytime Algorithm for Optimal Coalition Structure Generation*, AAAI 2007, JAIR 2009].

Basic Idea of **IP**



Basic Idea of **IP**



How the Bounds are Computed

L_1	value	L_2	value	L_3	value	L_4	value
1	125	{1, 2}	175	{1, 2, 3}	200	{1, 2, 3, 4}	425
2	50	{1, 3}	150	{1, 2, 4}	150		
3	75	{1, 4}	100	{1, 3, 4}	300		
4	150	{2, 3}	150	{2, 3, 4}	150		
	Max1 = 150	{2, 4}	200		Max3 = 300		Max4 = 425
	Avg1 = 100	{3, 4}	125		Avg3 = 200		Avg4 = 425
			Max2 = 200				
			Avg2 = 150				

$S_{[4]}$ Avg=425 UB=425 { $\{a_1, a_2, a_3, a_4\}$ }

$S_{[2,2]}$ Avg=300 UB=400 { $\{a_1,a_2\}, \{a_3,a_4\}$ }, { $\{a_1,a_3\}, \{a_2,a_4\}$ }, { $\{a_1,a_4\}, \{a_2,a_3\}$ }

$S_{[1,3]}$ Avg=300 UB=450

{ {a1}, {a1, a2, a3} }
{ {a1}, {a1, a2, a4} }
{ {a1}, {a1, a3, a4} }
{ {a1}, {a2, a3, a4} }

{ {a2}, {a1, a2, a3} }
{ {a2}, {a1, a2, a4} }
{ {a2}, {a1, a3, a4} }
{ {a2}, {a2, a3, a4} }

{ {a3}, {a1, a2, a3} }
{ {a3}, {a1, a2, a4} }
{ {a3}, {a1, a3, a4} }
{ {a3}, {a2, a3, a4} }

{ {a4}, {a1, a2, a3} }
{ {a4}, {a1, a2, a4} }
{ {a4}, {a1, a3, a4} }
{ {a4}, {a2, a3, a4} }

$S_{[1,1,2]}$ Avg=350 UB=500 { $\{a_1\}, \{a_2\}, \{a_3,a_4\}$ }, { $\{a_1\}, \{a_3\}, \{a_2,a_4\}$ }, { $\{a_1\}, \{a_4\}, \{a_2,a_3\}$ }, { $\{a_2\}, \{a_3\}, \{a_1,a_4\}$ }, { $\{a_2\}, \{a_4\}, \{a_1,a_3\}$ }, { $\{a_3\}, \{a_4\}, \{a_1,a_2\}$ }

$S_{[1,1,1,1]}$ Avg=400 UB=600 { $\{a_1\}, \{a_2\}, \{a_3\}, \{a_4\}$ }

Exercise: find the optimal solution using IP

L_1	Value	L_2	Value	L_3	Value	L_4	value	L_5	value
1	20	1, 2	40	1, 2, 3	65	1, 2, 3, 4	85	1, 2, 3, 4, 5	165
2	15	1, 3	30	1, 2, 4	55	1, 2, 3, 5	140		Avg=165
3	25	1, 4	20	1, 2, 5	70	1, 2, 4, 5	90		
4	30	1, 5	45	1, 3, 4	60	1, 3, 4, 5	75		
5	10	2, 3	40	1, 3, 5	75	2, 3, 4, 5	110		
	Avg=20	2, 4	65	1, 4, 5	55		Avg=100		
		2, 5	20	2, 3, 4	70				
		3, 4	30	2, 3, 5	75				
		3, 5	45	2, 4, 5	65				
		4, 5	15	3, 4, 5	60				
			Avg=35		Avg=65				

[5]
Avg=165 UB=165

[1,4]
Avg=120 UB=170

[2,3]
Avg=100 UB=140

[1,1,3]
Avg=105 UB=135

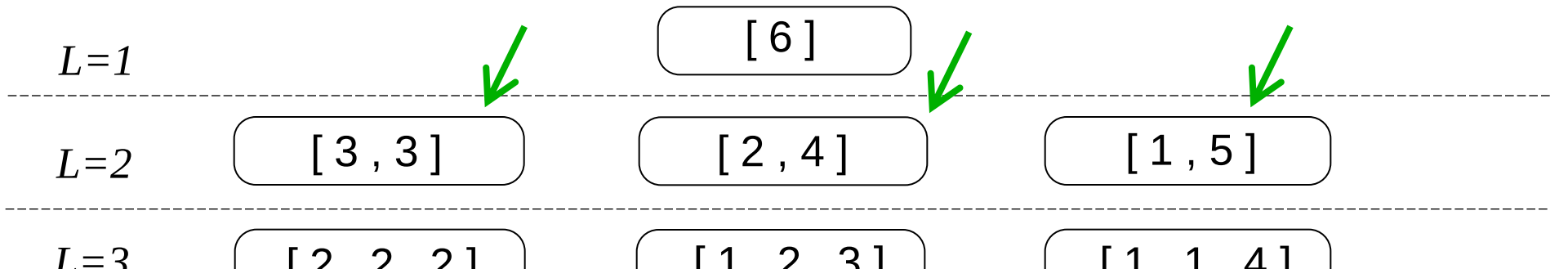
[1,2,2]
Avg=90 UB=160

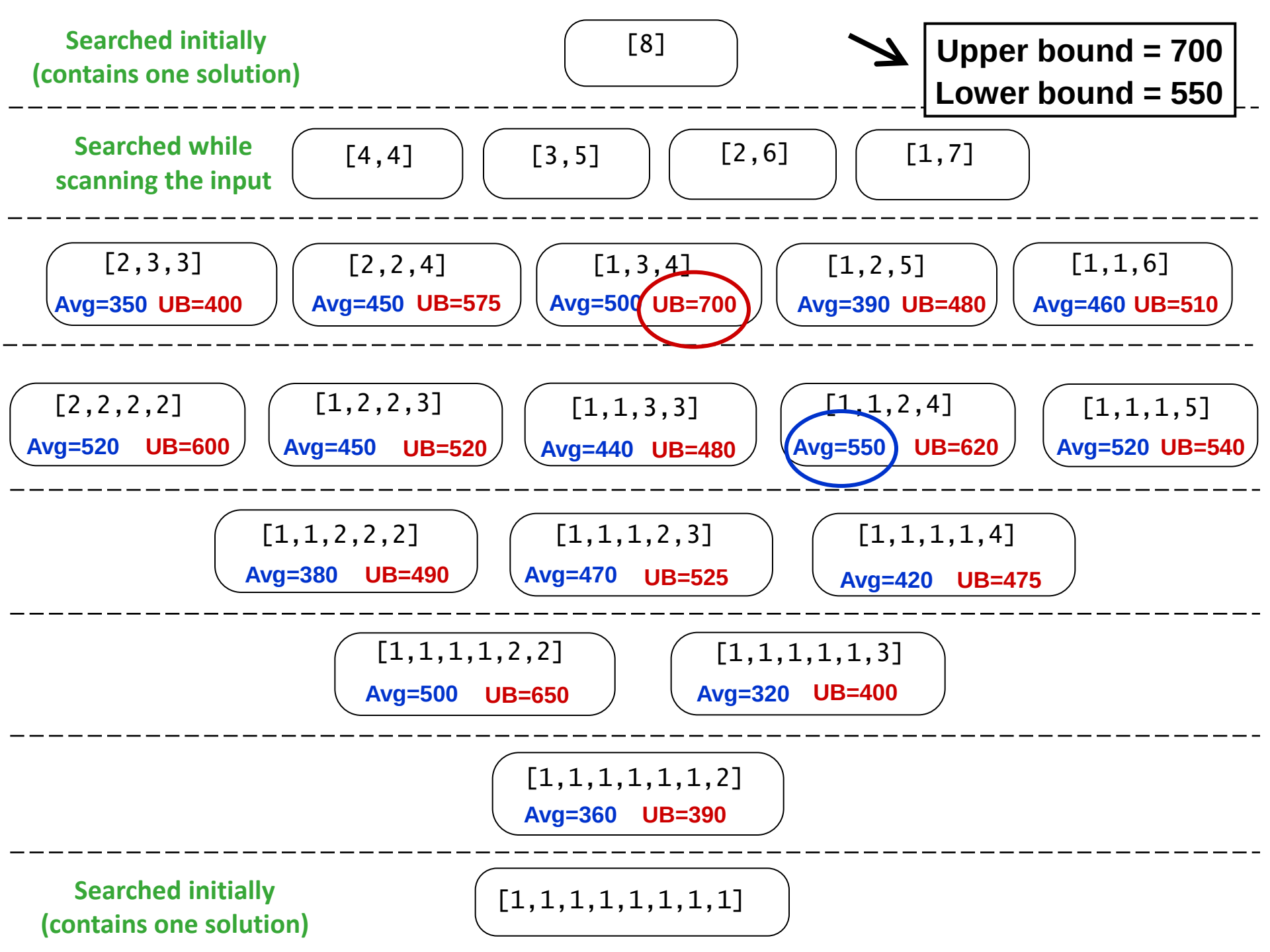
[1,1,1,2]
Avg=95 UB=155

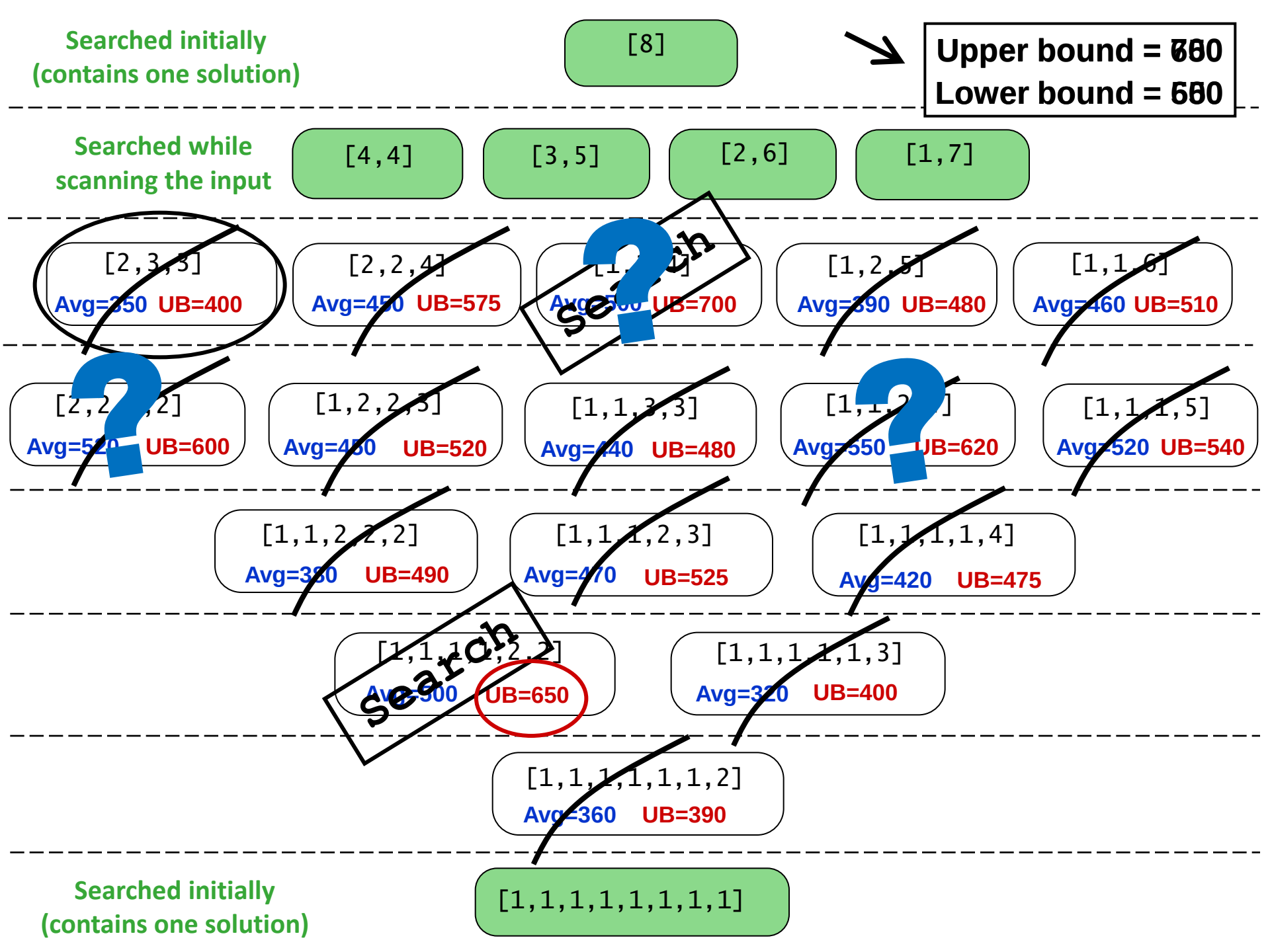
[1,1,1,1,1]
Avg=100 UB=150

Scanning the input

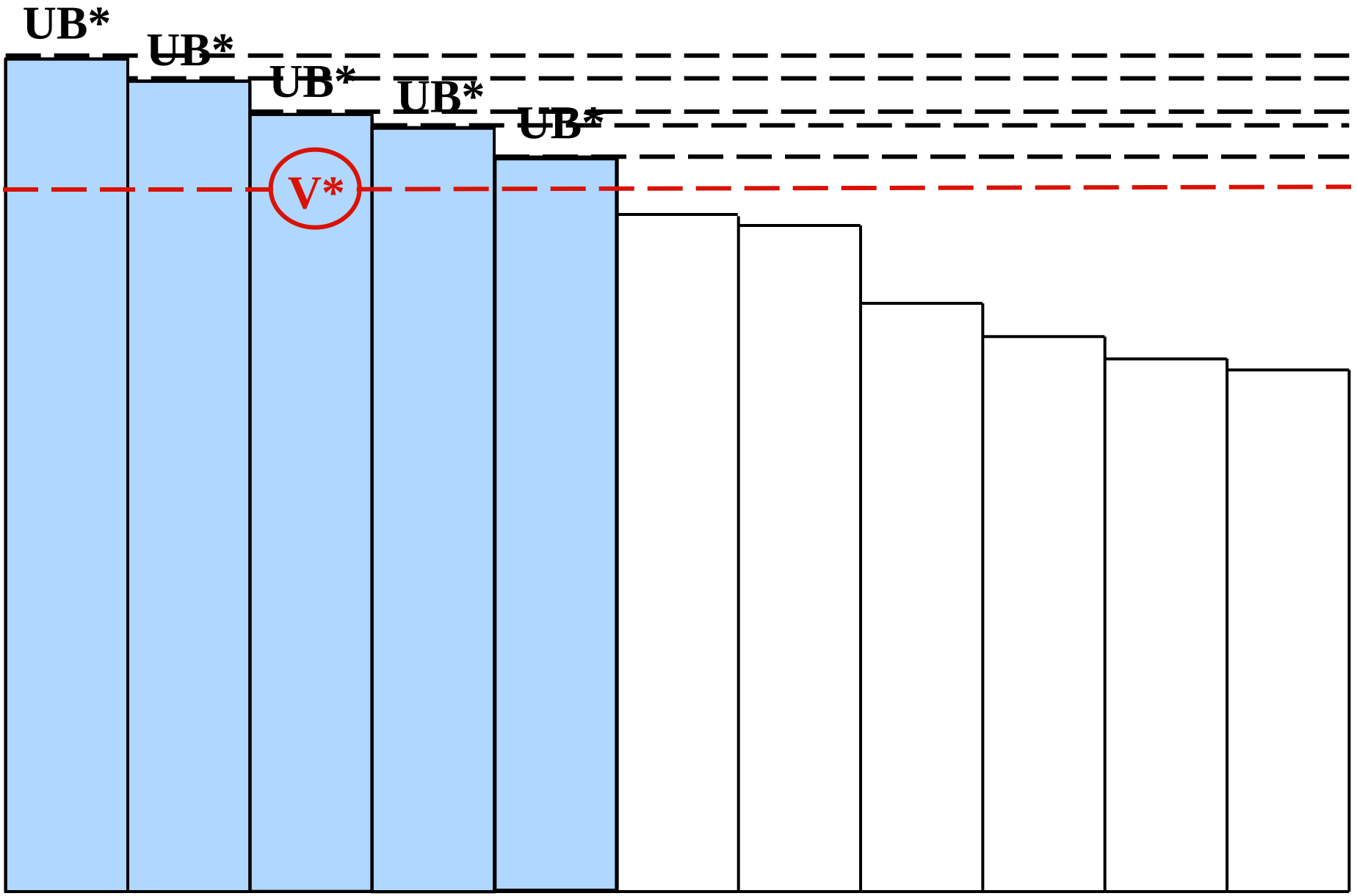
L_1	*value*	L_2	*value*	L_3	*value*	L_4	*value*	L_5	*value*	L_6	*value*
1	8[illegible]6	1, 2	1742	1, 2, 3	3352	1, 2, 3, 4	4[illegible]55	1, 2, 3, 4, 5	48[illegible]4	1,2, 3, 4, 5, 6	6217
2	11[illegible]08	1, 3	1667	1, 2, 4	3102	1, 2, 3, 5	43[illegible]73	1, 2, 3, 4, 6	56[illegible]7		
3	10[illegible]65	1, 4	1989	1, 2, 5	3301	1, 2, 3, 6	37[illegible]70	1, 2, 3, 5, 6	46[illegible]9		
4	8[illegible]0	1, 5	1664	1, 2, 6	3119	1, 2, 4, 5	35[illegible]28	1, 2, 4, 5, 6	48[illegible]9		
5	9[illegible]7	1, 6	2023	1, 3, 4	3287	1, 2, 4, 6	39[illegible]67	1, 3, 4, 5, 6	56[illegible]0		
6	1[illegible]4	2, 3	2083	1, 3, 5	2696	1, 2, 5, 6	36[illegible]47	2, 3, 4, 5, 6	5[illegible]2		
		2, 4	2272	1, 3, 6	2950	1, 3, 4, 5	41[illegible]42				
		2, 5	2082	1, 4, 5	3[illegible]24	1, 3, 4, 6	38[illegible]75				
		2, 6	1995	1, 4, 6	2[illegible]50	1, 3, 5, 6	39[illegible]05				
		3, 4	1807	1, 5, 6	3134	1, 4, 5, 6	36[illegible]45				
		3, 5	2529	2, 3, 4	2[illegible]3	2, 3, 4, 5	38[illegible]50				
		3, 6	2045	2, 3, 5	2956	2, 3, 4, 6	40[illegible]99				
		4, 5	1683	2, 3, 6	2950	2, 3, 5, 6	39[illegible]67				
		4, 6	2215	2, 4, 5	3661	2, 4, 5, 6	33[illegible]18				
		5, 6	1[illegible]6	2, 4, 6	2618	3, 4, 5, 6	3[illegible]6				
				2, 5, 6	2906						
				3, 4, 5	2769						
				3, 4, 6	3[illegible]70						
				3, 5, 6	3135						
				4, 5, 6							




Searched initially
(contains one solution)
[8]
Upper bound = 700
Lower bound = 550
Searched while
scanning the input
[4,4]
[3,5]
[2,6]
[1,7]
[2,3,3]
Avg=350 UB=400
[2,2,4]
Avg=450 UB=575
[1,3,4]
Avg=500 UB=700
[1,2,5]
Avg=390 UB=480
[1,1,6]
Avg=460 UB=510
[2,2,2,2]
Avg=520 UB=600
[1,2,2,3]
Avg=450 UB=520
[1,1,3,3]
Avg=440 UB=480
[1,1,2,4]
Avg=550 UB=620
[1,1,1,5]
Avg=520 UB=540
[1,1,2,2,2]
Avg=380 UB=490
[1,1,1,2,3]
Avg=470 UB=525
[1,1,1,1,4]
Avg=420 UB=475
[1,1,1,1,2,2]
Avg=500 UB=650
[1,1,1,1,1,3]
Avg=320 UB=400
[1,1,1,1,1,1,2]
Avg=360 UB=390
Searched initially
(contains one solution)
[1,1,1,1,1,1,1,1]

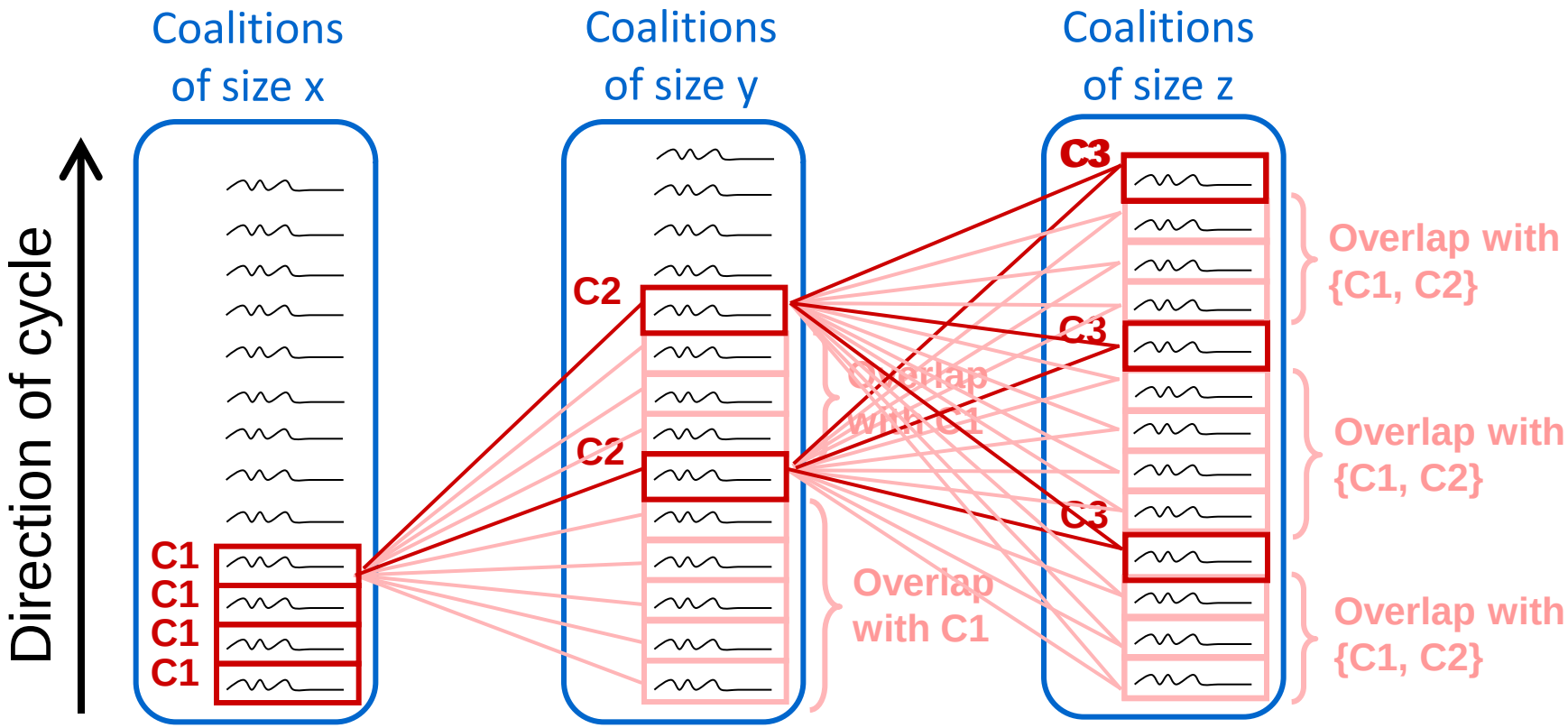

Searched initially
(contains one solution)
[8]
Upper bound = 680
Lower bound = 680
Searched while
scanning the input
[4,4]
[3,5]
[2,6]
[1,7]
[2,3,3]
Avg=350 UB=400
[2,2,4]
Avg=450 UB=575
Search
[1,2,5]
Avg=390 UB=480
[1,1,6]
Avg=460 UB=510
[2,2,2,2]
Avg=520 UB=600
[1,2,2,3]
Avg=450 UB=520
[1,1,3,3]
Avg=440 UB=480
[1,1,2,4]
Avg=550 UB=620
[1,1,1,5]
Avg=520 UB=540
[1,1,2,2,2]
Avg=380 UB=490
[1,1,1,2,3]
Avg=470 UB=525
[1,1,1,1,4]
Avg=420 UB=475
Search
[1,1,1,2,2,2]
Avg=500 UB=650
[1,1,1,1,1,3]
Avg=320 UB=400
[1,1,1,1,1,1,2]
Avg=360 UB=390
Searched initially
(contains one solution)
[1,1,1,1,1,1,1,1]

Why select sub-space with highest UB ?



Searching a Sub-space

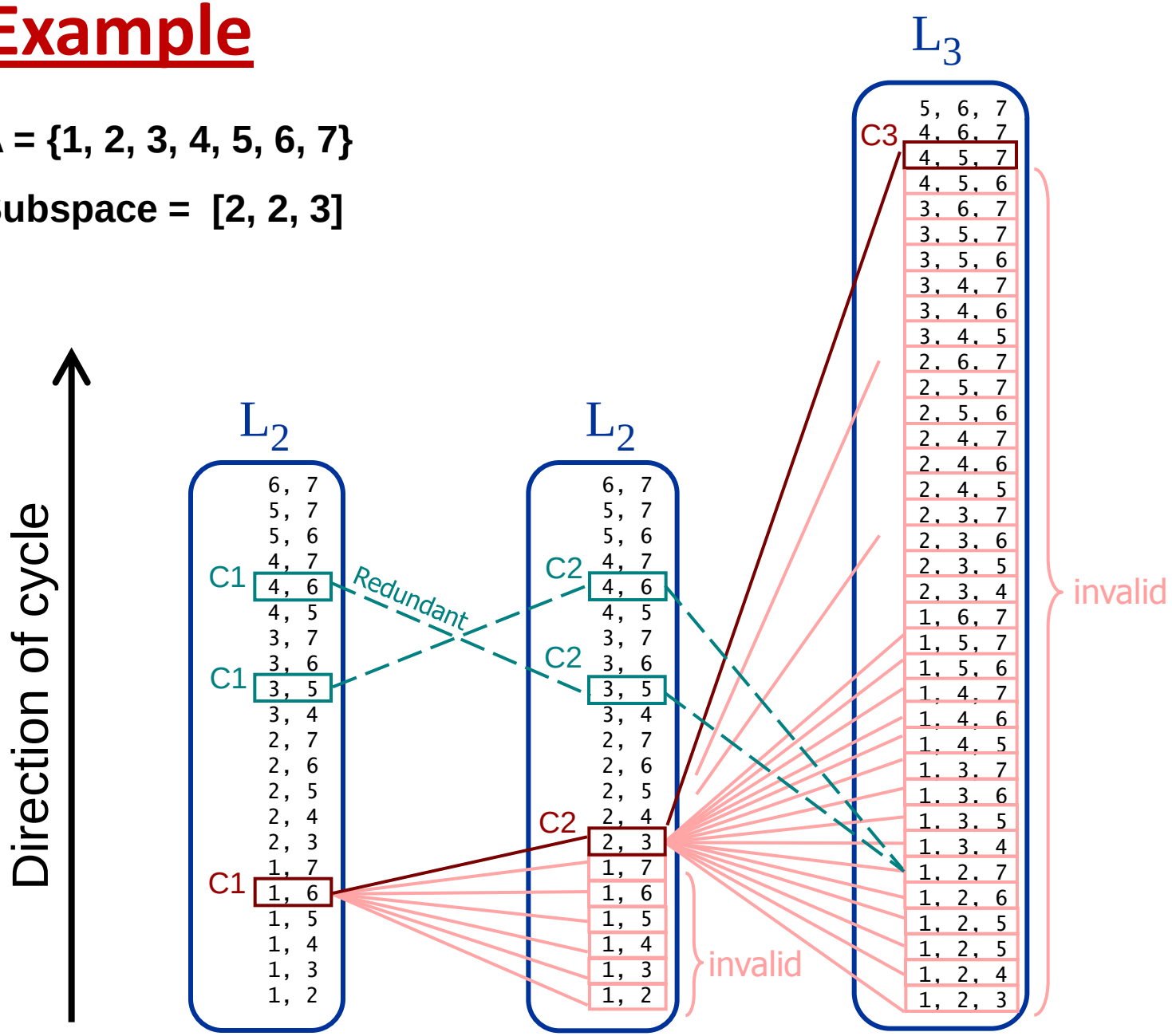
Example: Search the following sub-space [x, y, z]



Example

A = {1, 2, 3, 4, 5, 6, 7}

Subspace = [2, 2, 3]

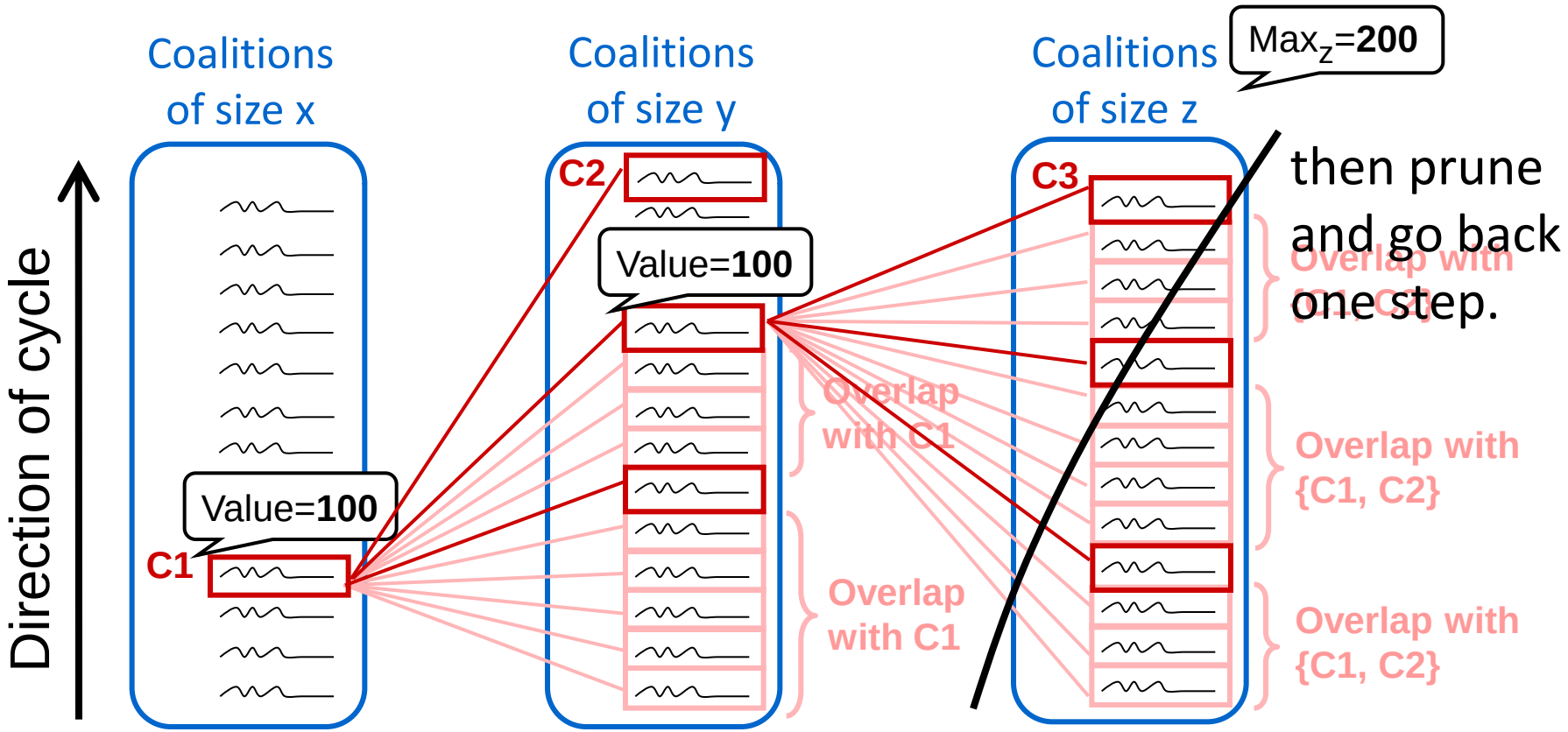


Searching a Sub-space

Example: Search the following sub-space [x, y, z]

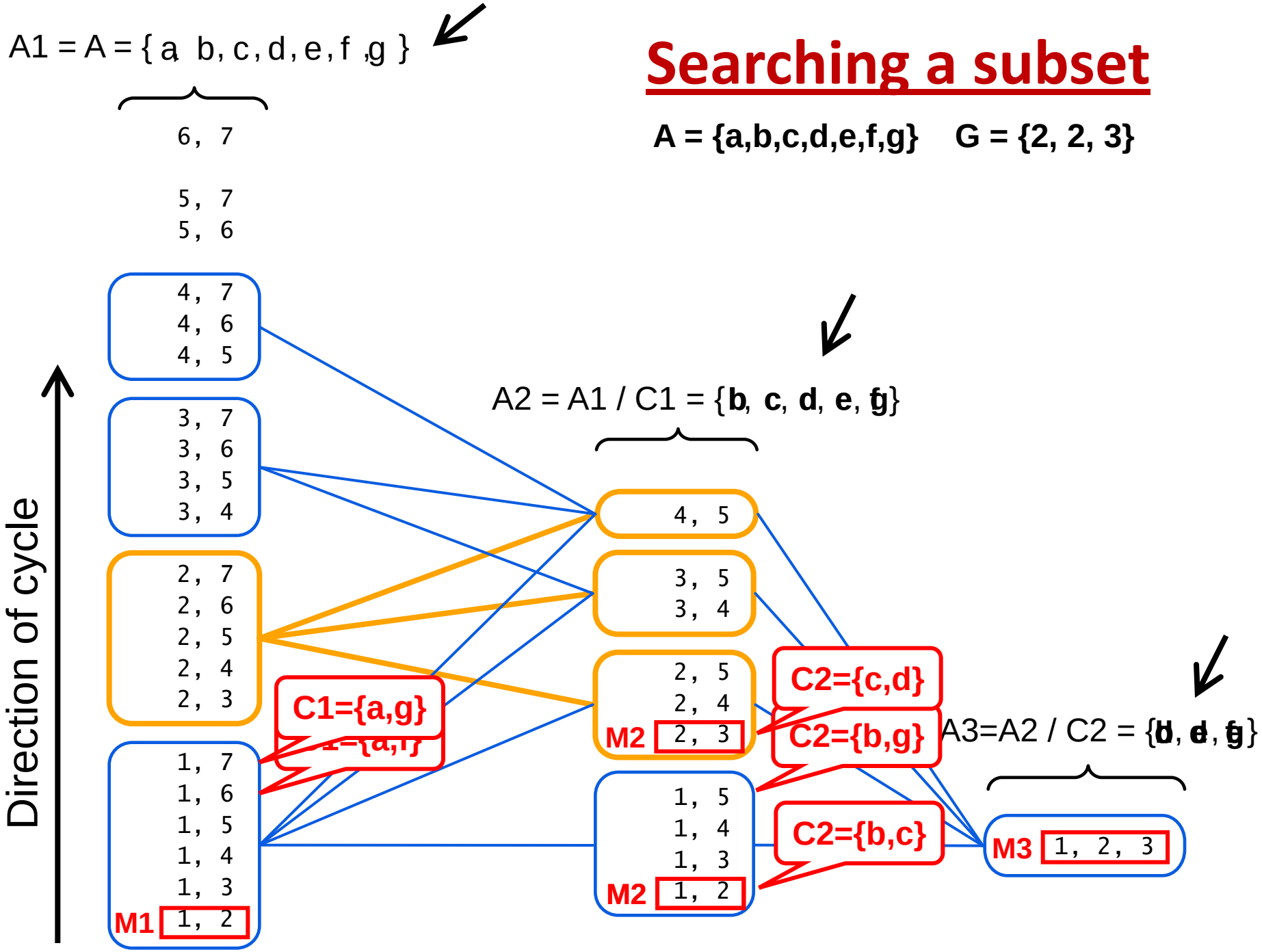
Applying Branch and Bound:

Example: If value of best solution found so far is **500**, then if:



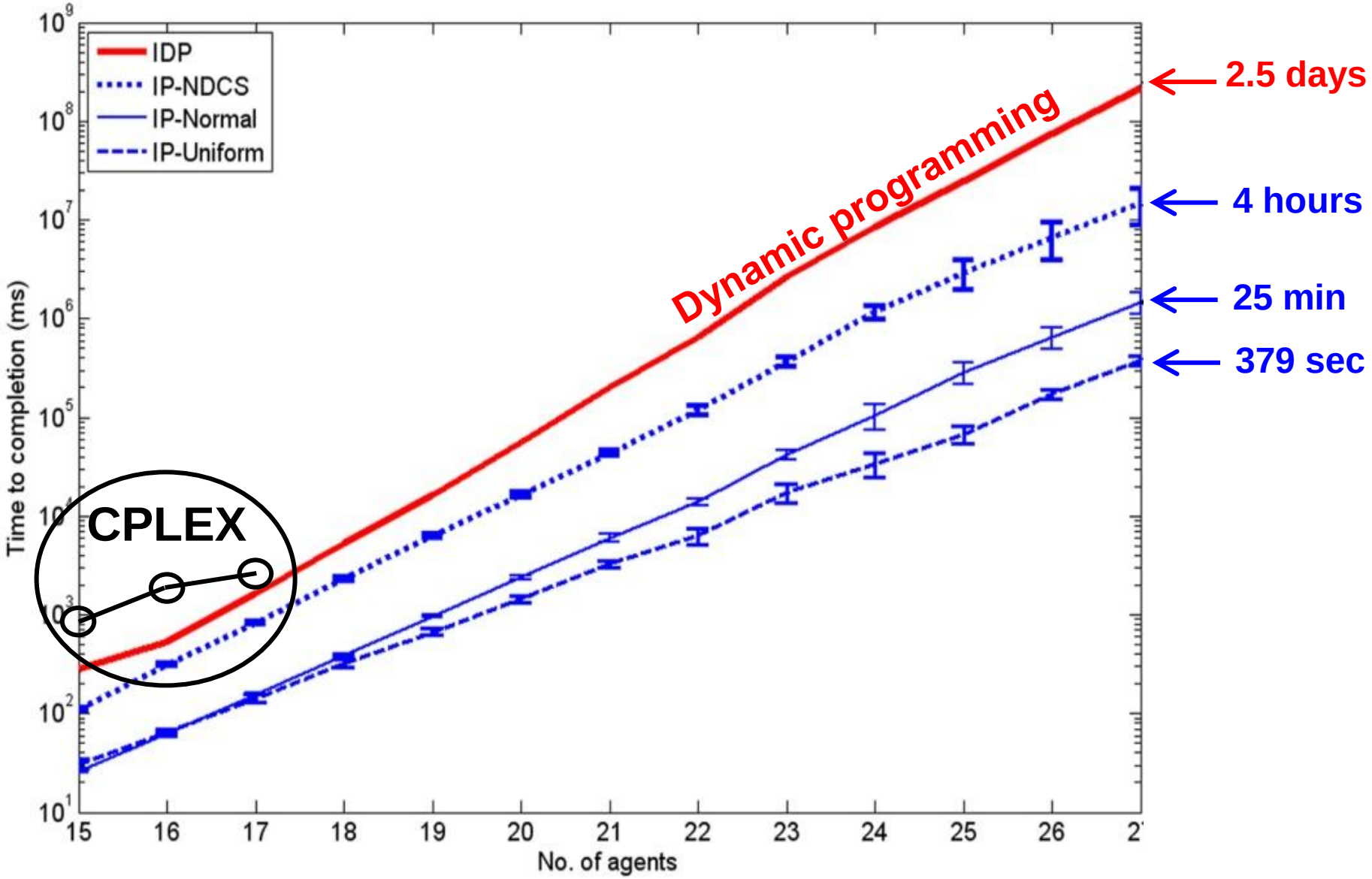
Searching a subset

A = {a,b,c,d,e,f,g} G = {2, 2, 3}

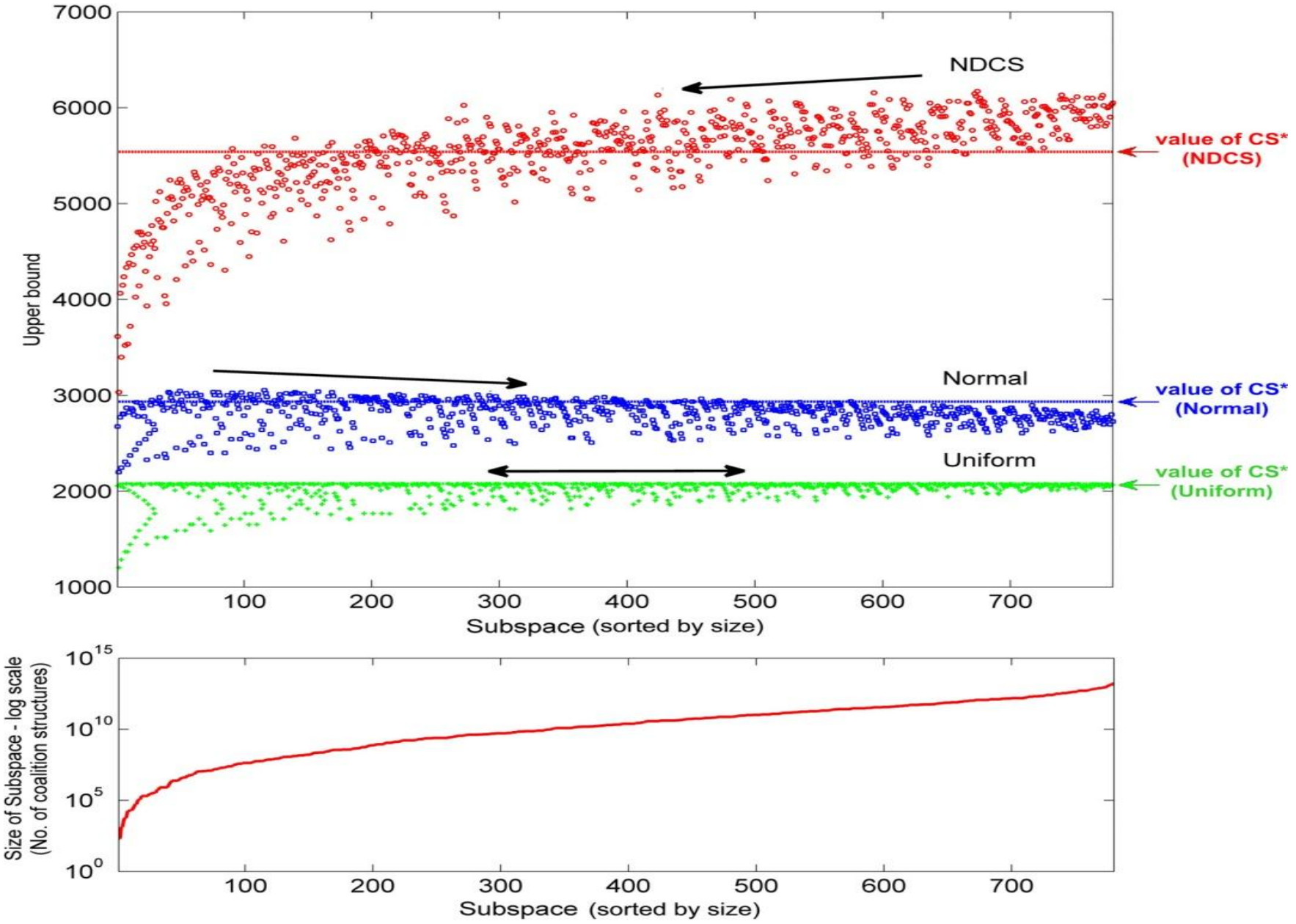


Evaluation

(Time to Terminate)

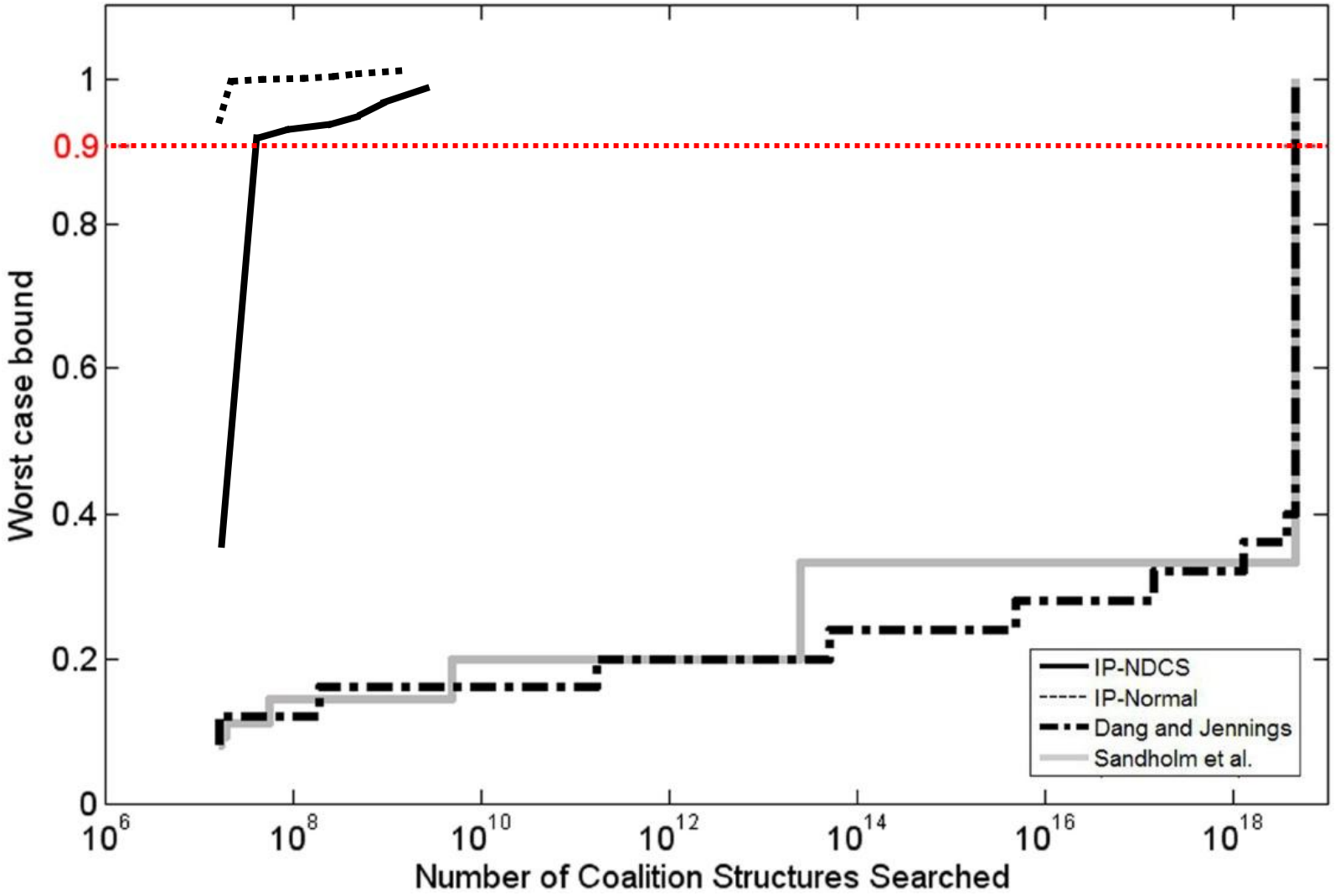


Upper Bounds of Different subspaces

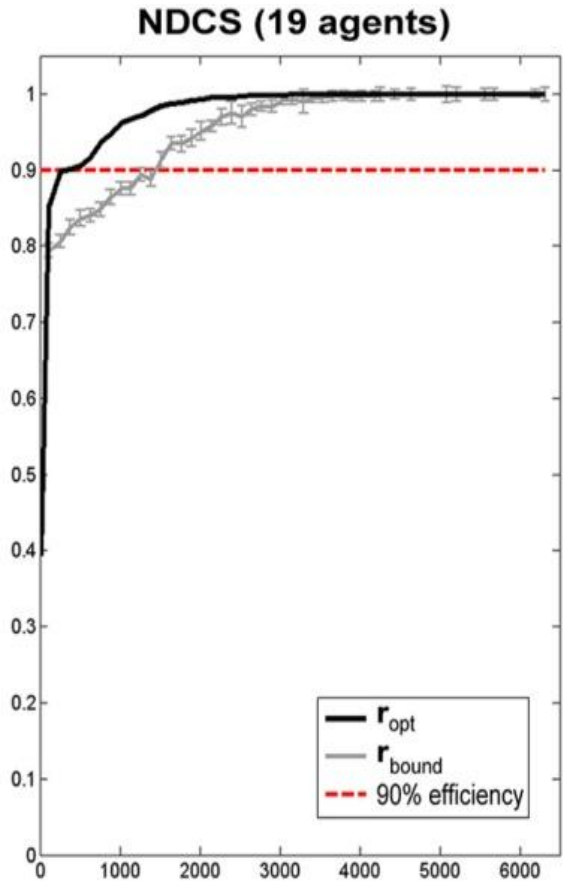


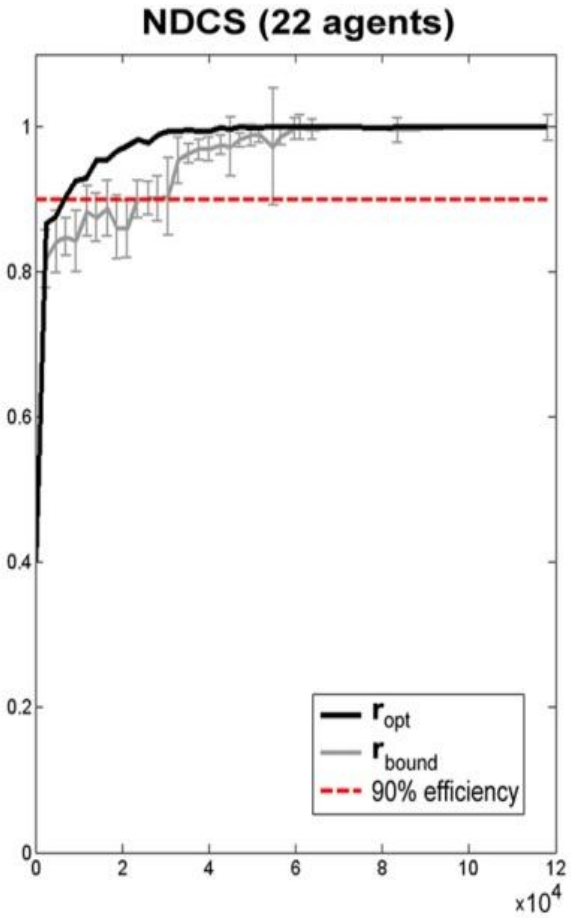
Evaluation

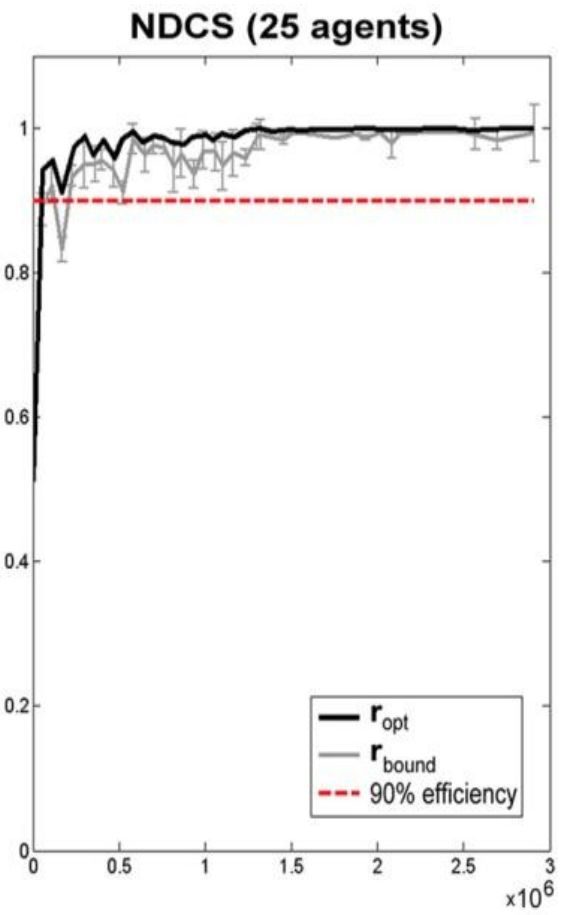
(Worst-Case Guarantees)



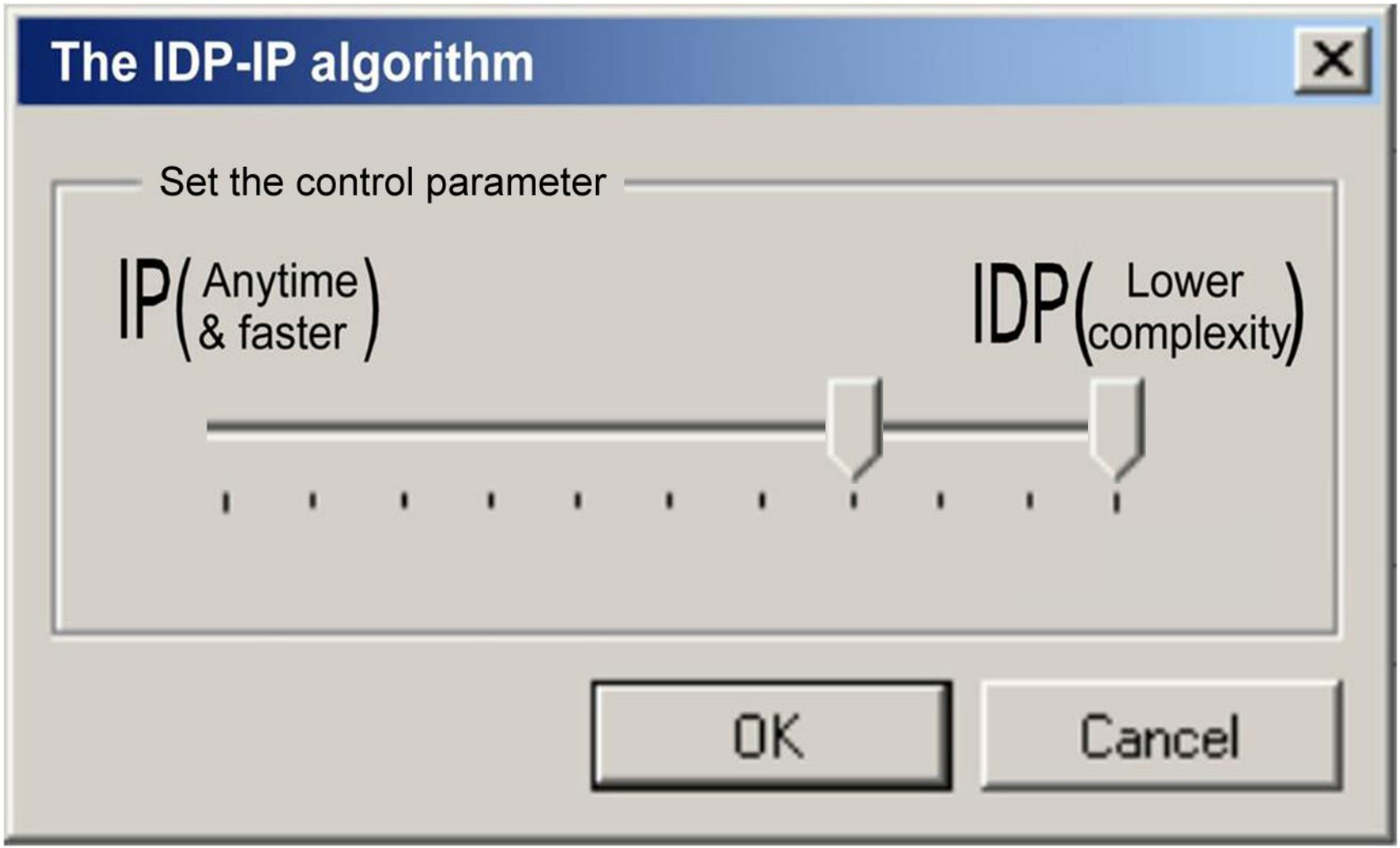
Evaluation
(Solution Quality)



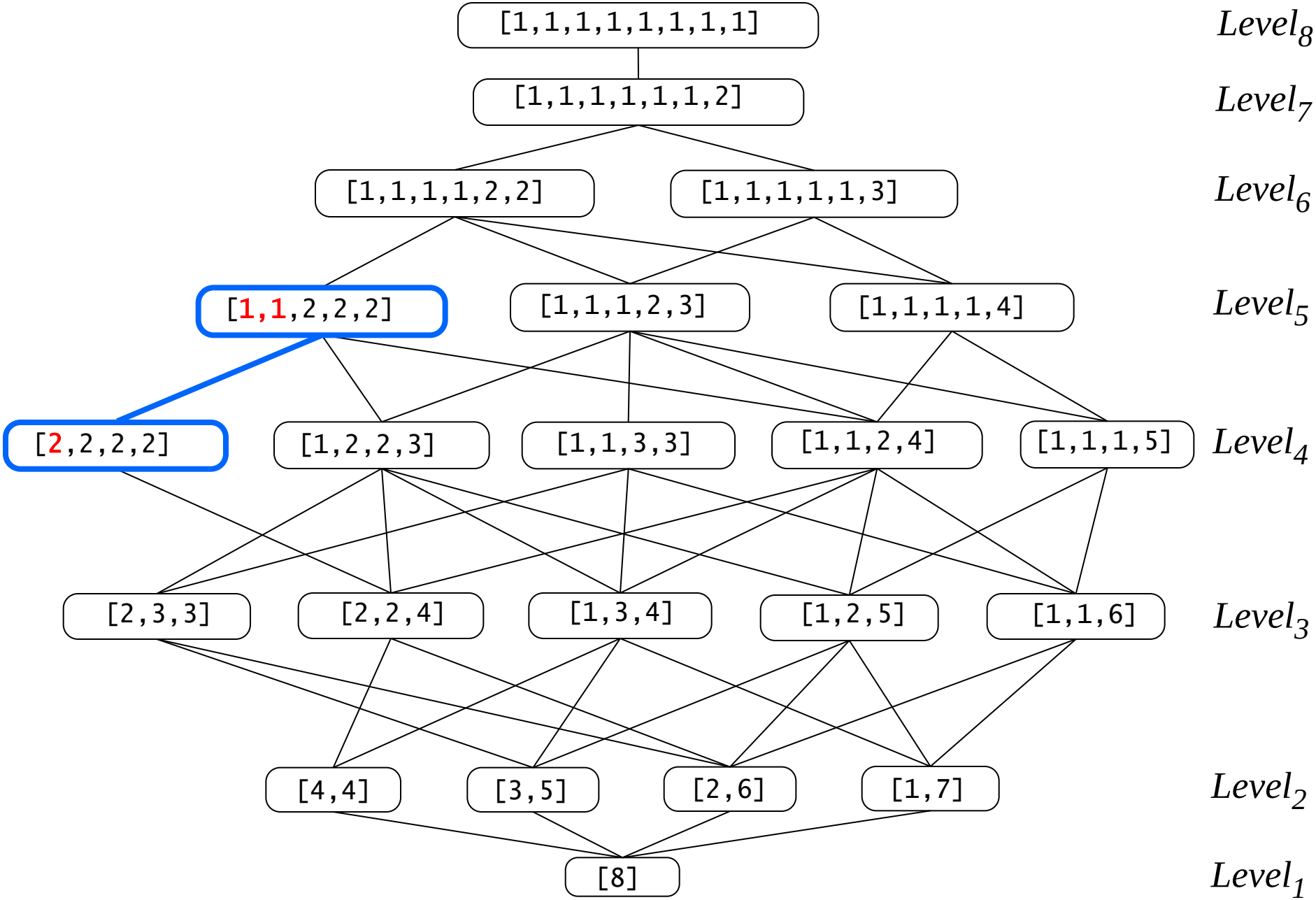


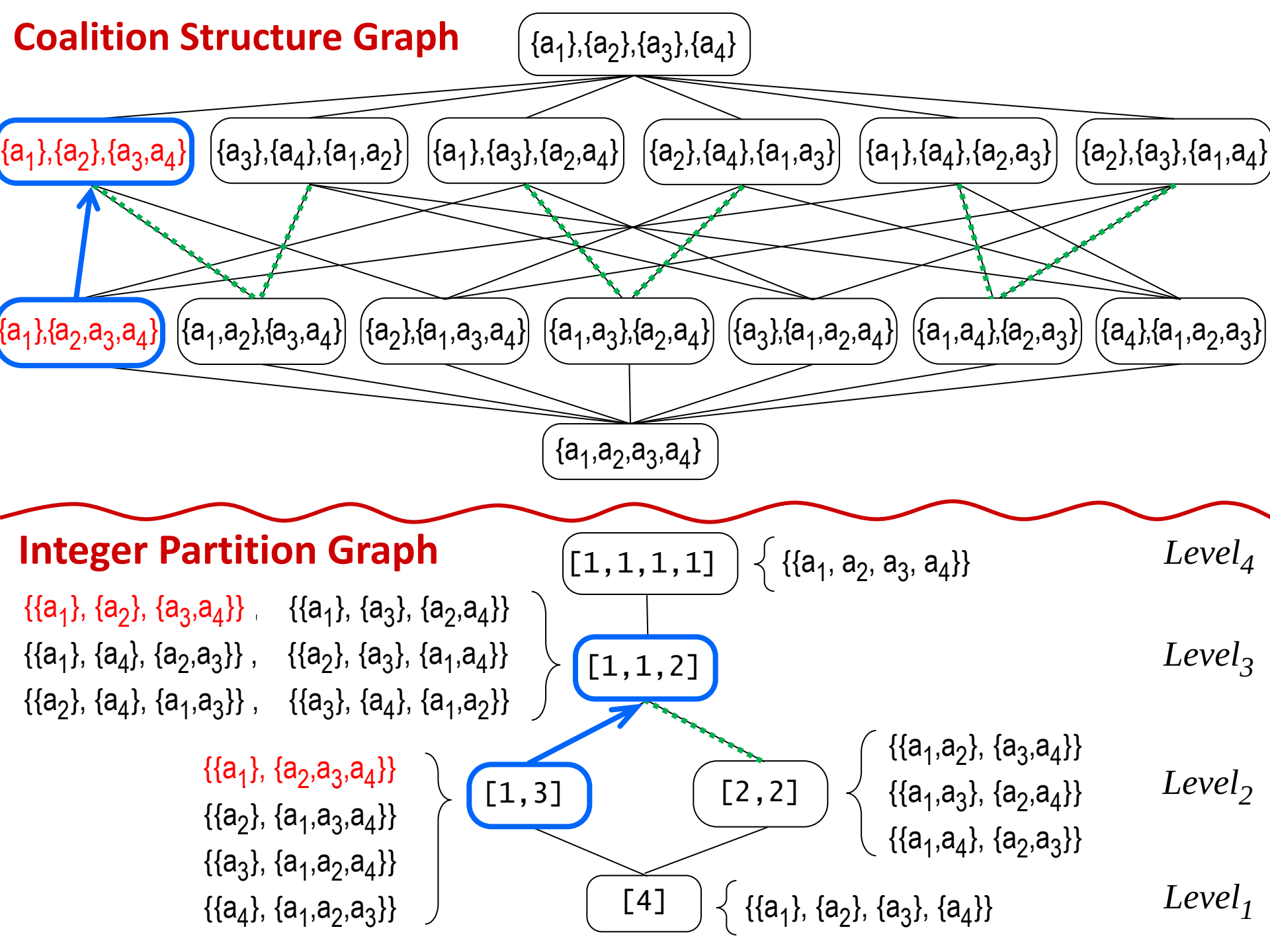


IDP-IP

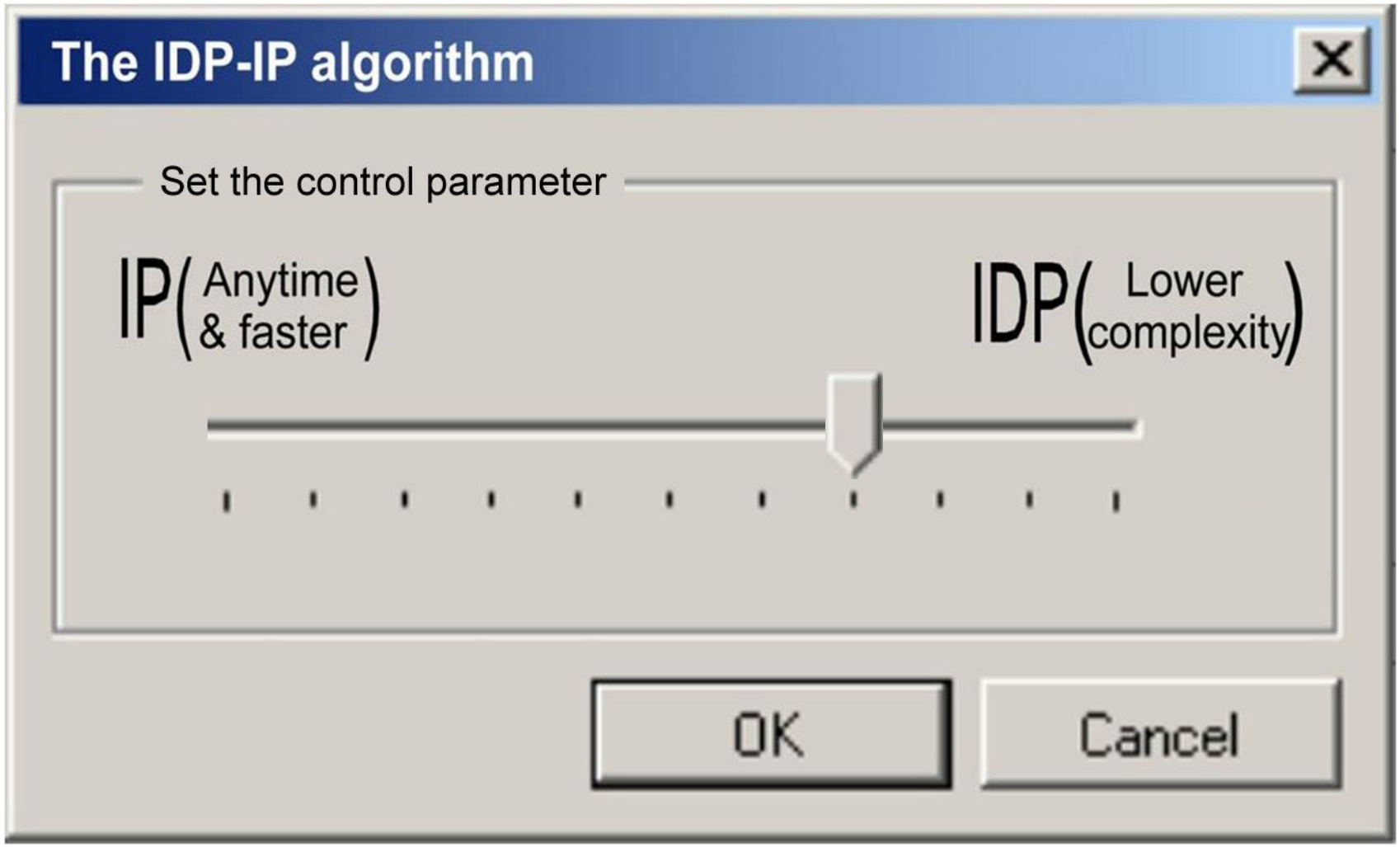

The IDP-IP algorithm
Set the control parameter
IP(Anytime & faster)
IDP(Lower complexity)
OK
Cancel

The Integer-Partition Graph

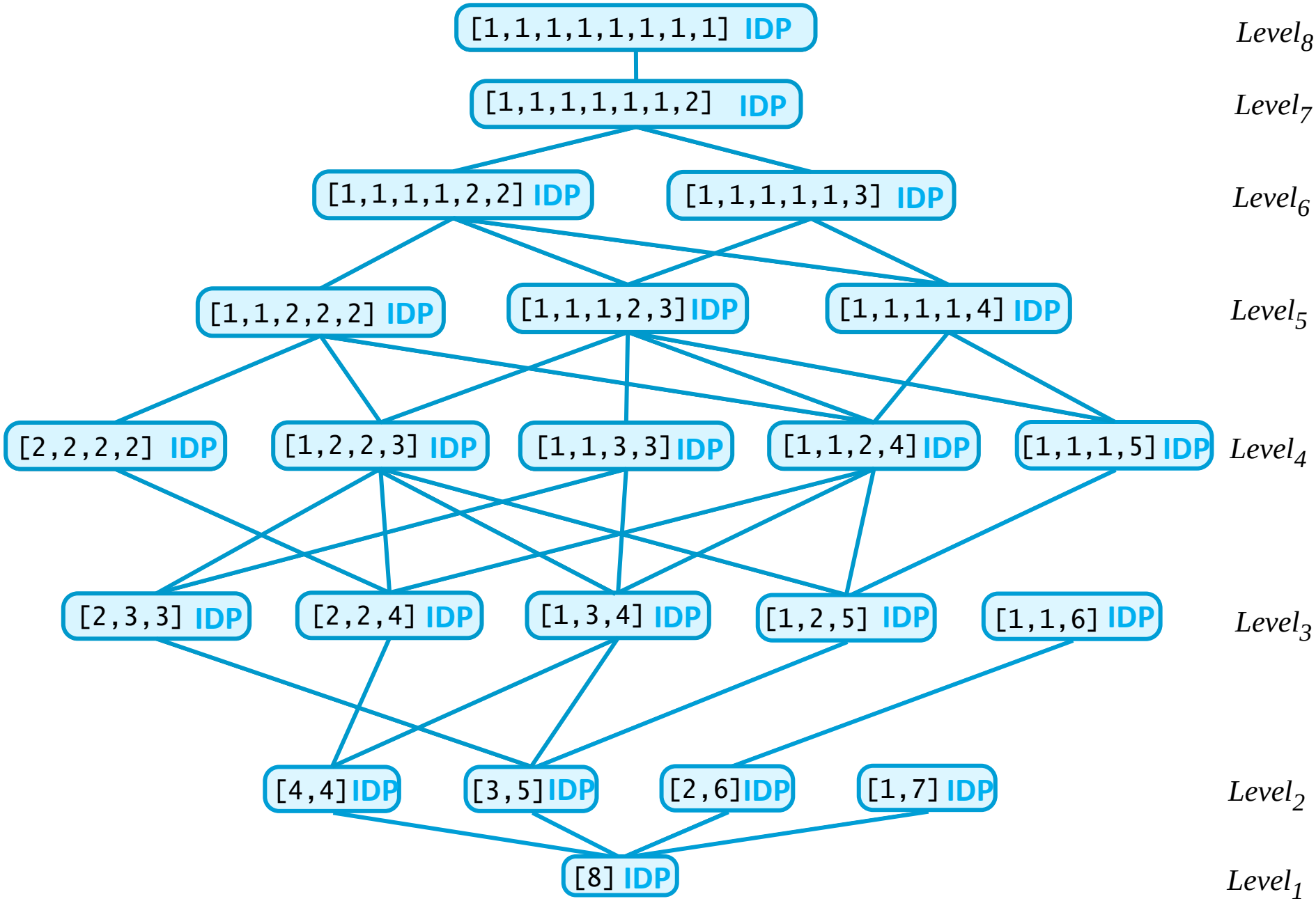



Coalition Structure Graph
{a1},{a2},{a3},{a4}
{a1},{a2},{a3,a4}
{a3},{a4},{a1,a2}
{a1},{a3},{a2,a4}
{a2},{a4},{a1,a3}
{a1},{a4},{a2,a3}
{a2},{a3},{a1,a4}
{a1},{a2,a3,a4}
{a1,a2},{a3,a4}
{a2},{a1,a3,a4}
{a1,a3},{a2,a4}
{a3},{a1,a2,a4}
{a1,a4},{a2,a3}
{a4},{a1,a2,a3}
{a1,a2,a3,a4}
Integer Partition Graph
[1,1,1,1]
{{a1, a2, a3, a4}}
Level4
{{a1}, {a2}, {a3,a4}}
{{a1}, {a3}, {a2,a4}}
{{a1}, {a4}, {a2,a3}}
{{a2}, {a3}, {a1,a4}}
{{a2}, {a4}, {a1,a3}}
{{a3}, {a4}, {a1,a2}}
[1,1,2]
Level3
{{a1}, {a2,a3,a4}}
{{a2}, {a1,a3,a4}}
{{a3}, {a1,a2,a4}}
{{a4}, {a1,a2,a3}}
[1,3]
[2,2]
{{a1,a2}, {a3,a4}}
{{a1,a3}, {a2,a4}}
{{a1,a4}, {a2,a3}}
Level2
[4]
{{a1}, {a2}, {a3}, {a4}}
Level1

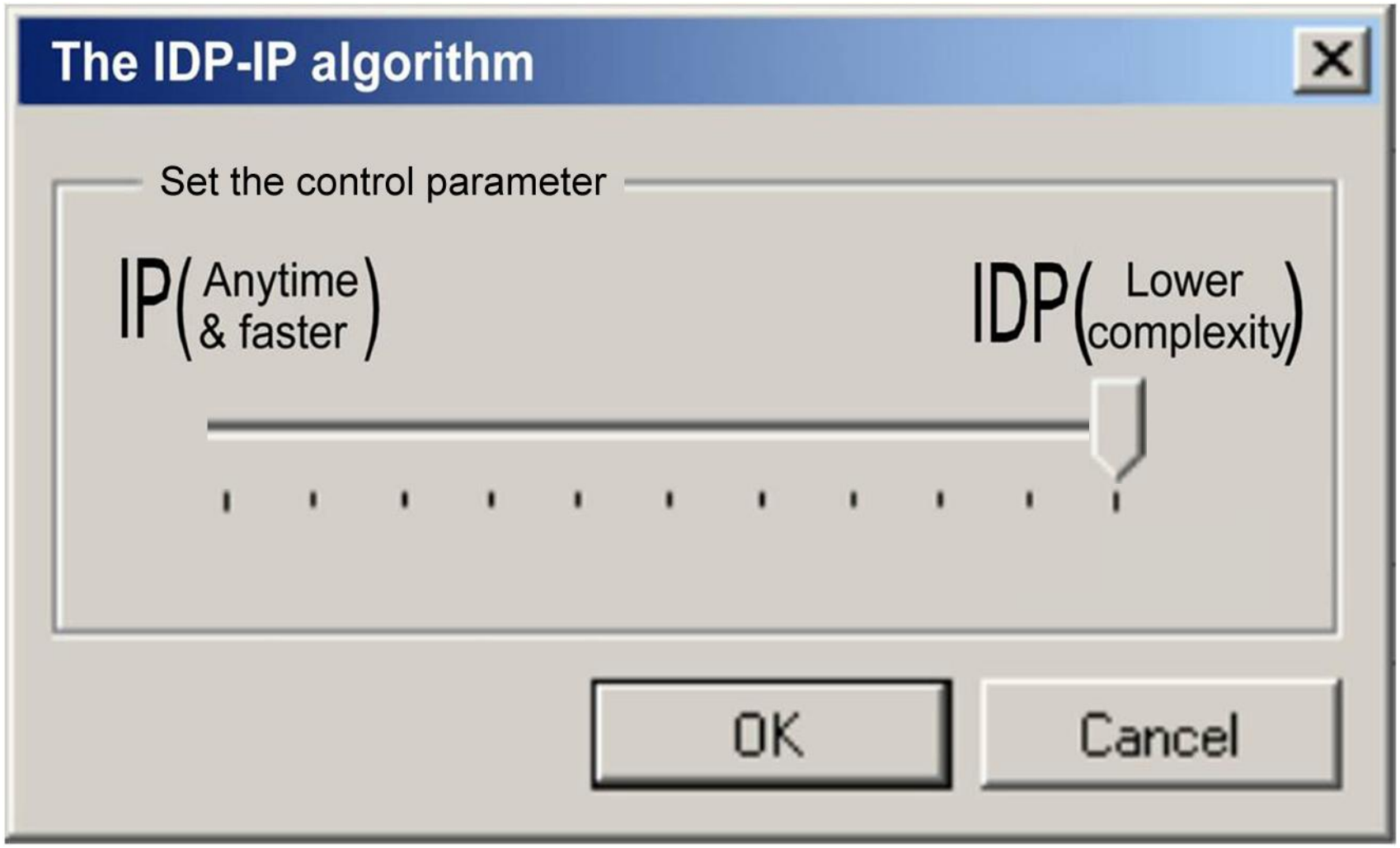
IDP-IP


The IDP-IP algorithm
Set the control parameter
IP (Anytime & faster)
IDP (Lower complexity)
OK
Cancel

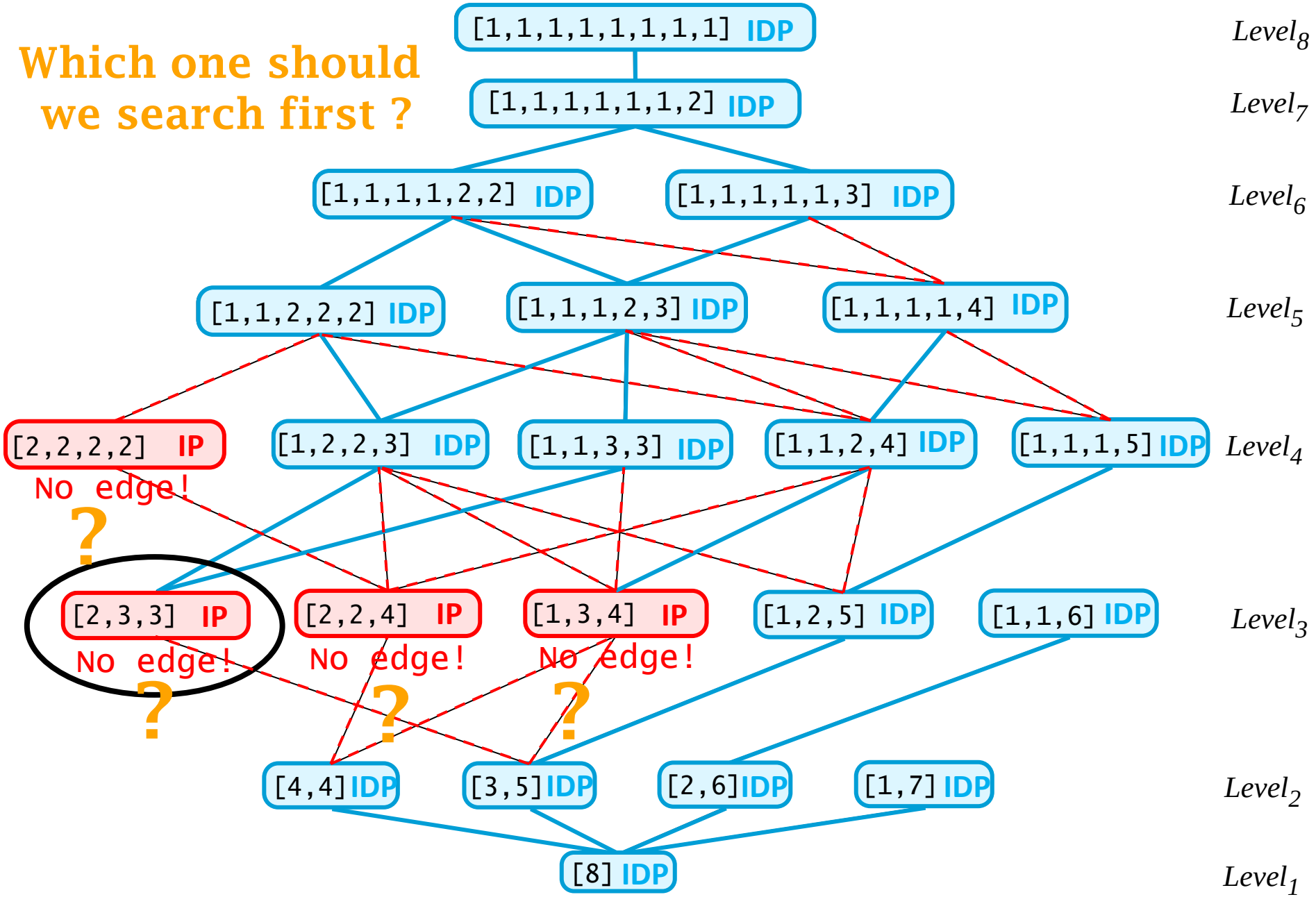
The Integer-Partition Graph



IDP-IP

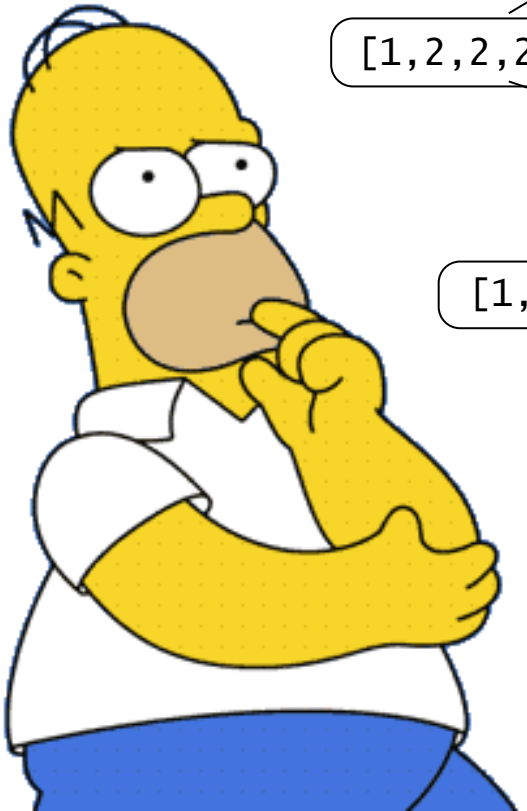

The IDP-IP algorithm
Set the control parameter
IP (Anytime & faster)
IDP (Lower complexity)
OK
Cancel

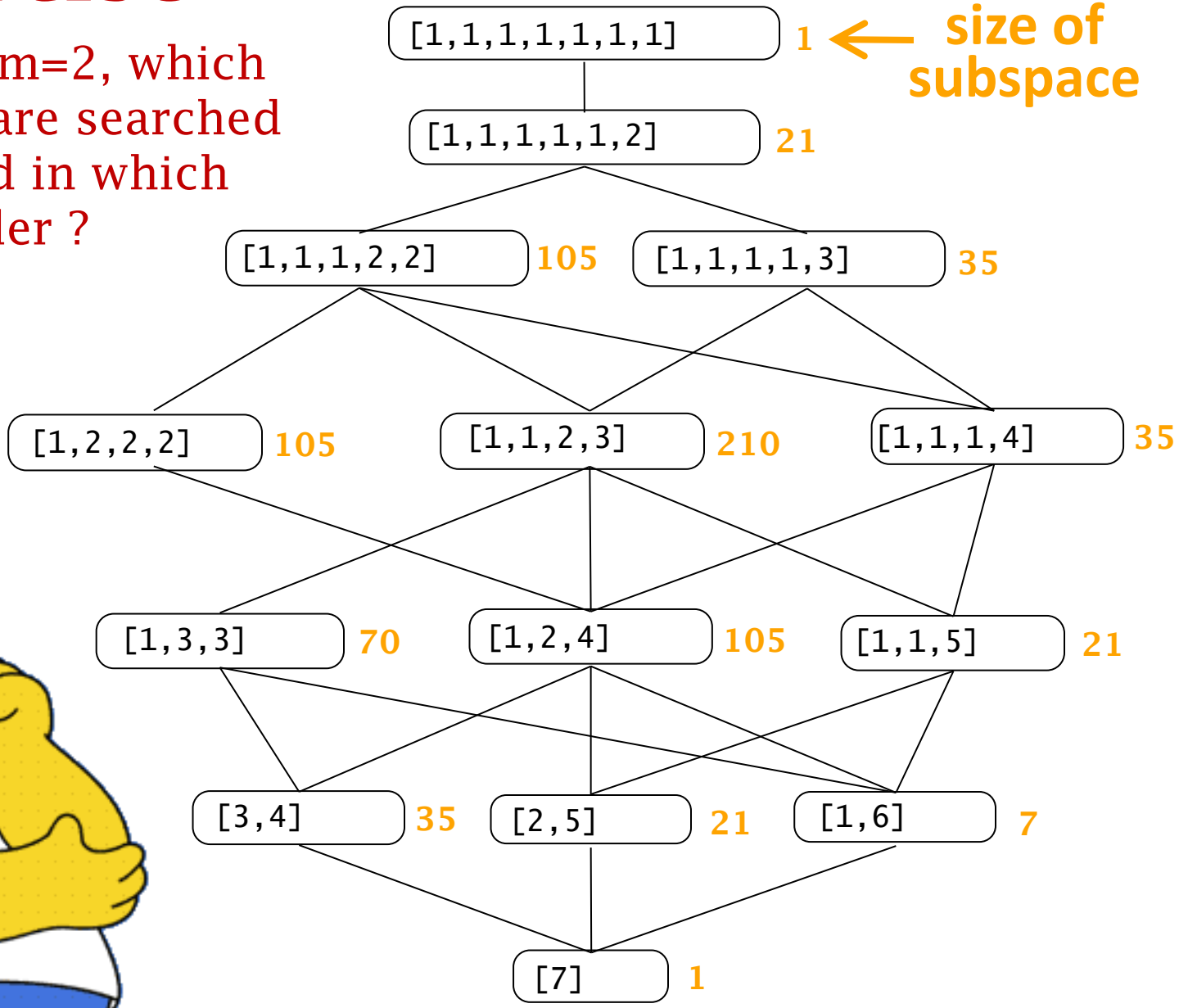
The Integer-Partition Graph



Exercise

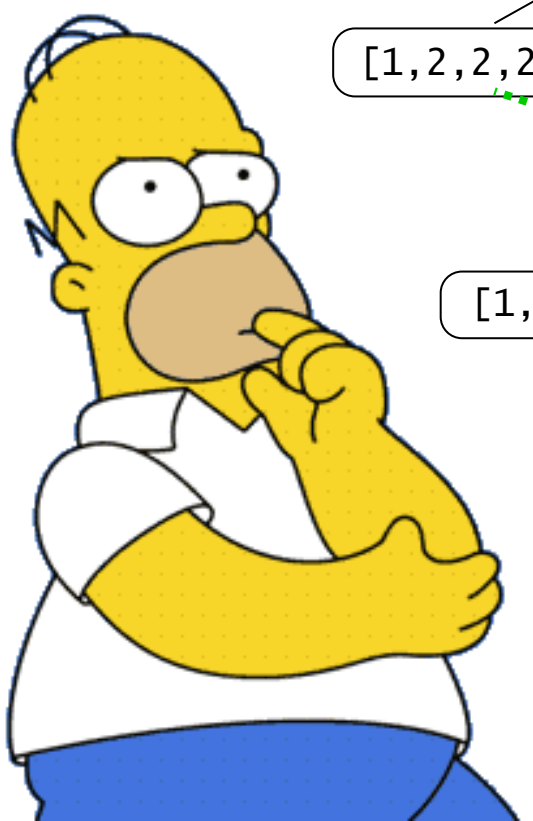
By setting m=2, which subspaces are searched by IP, and in which order ?

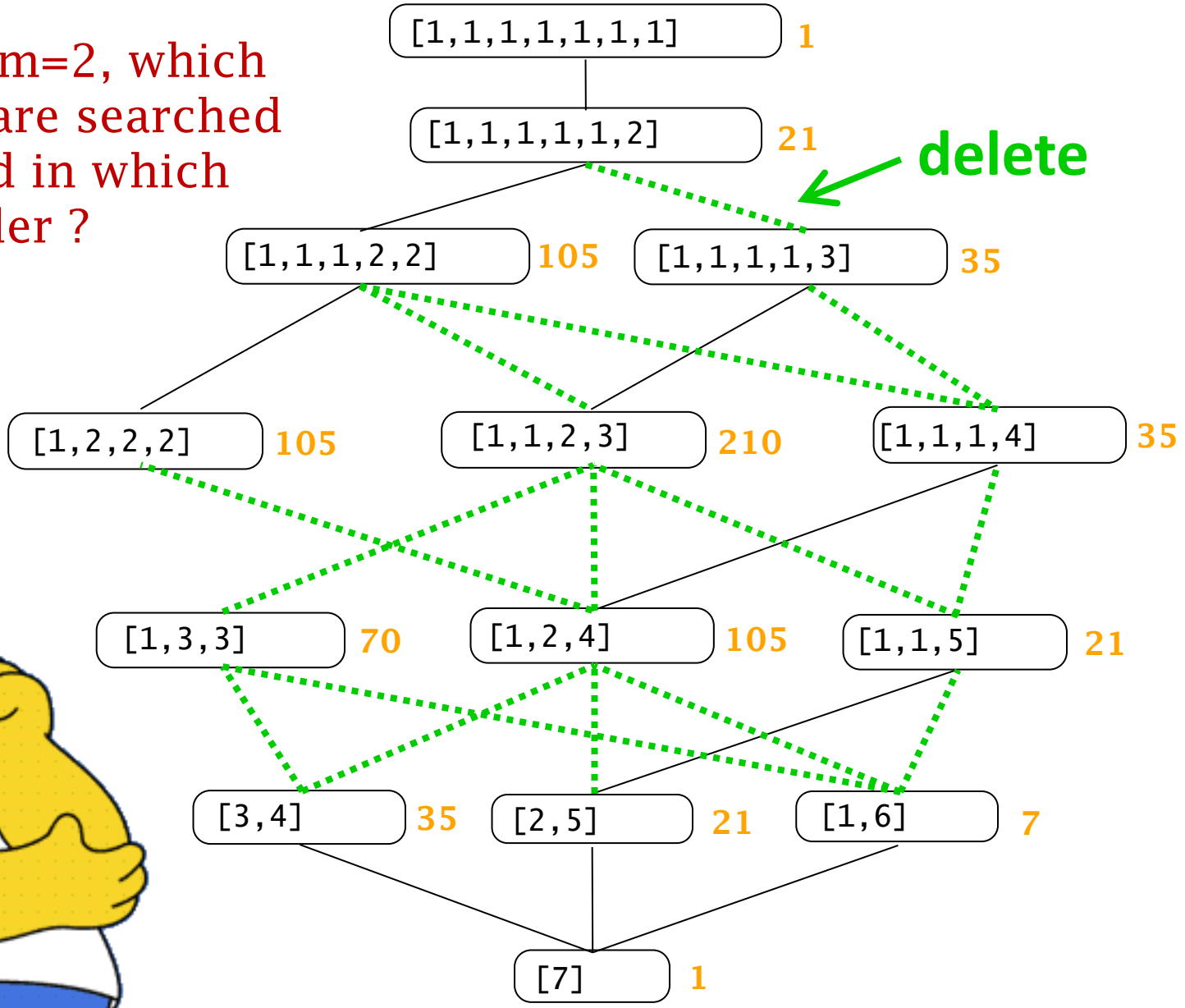




Exercise

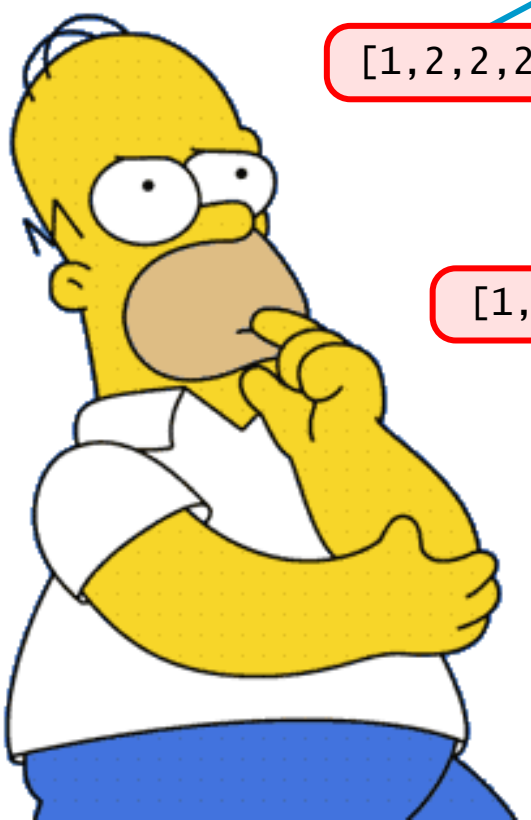
By setting m=2, which subspaces are searched by IP, and in which order ?

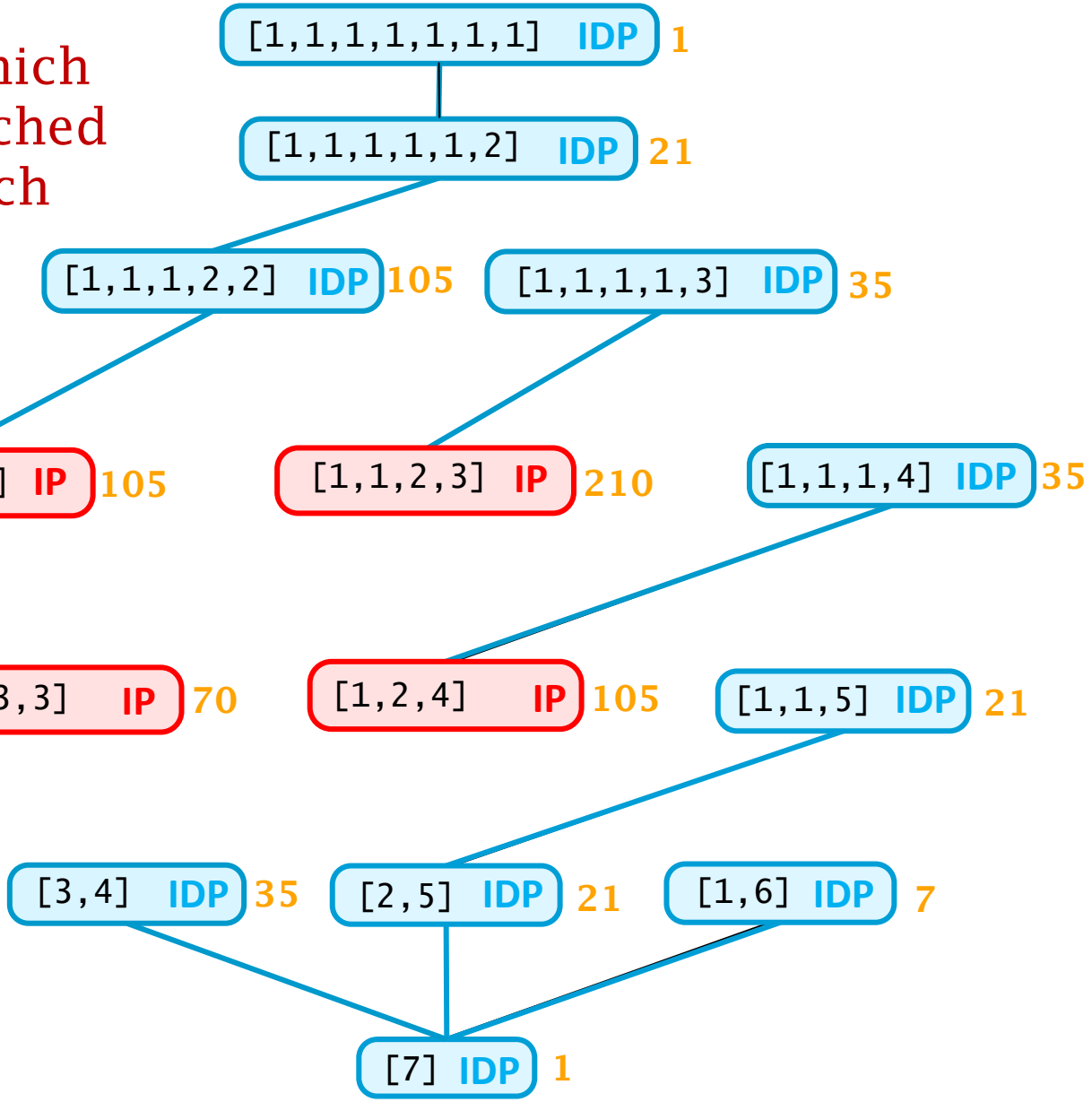




Exercise

By setting m=2, which subspaces are searched by IP, and in which order ?





Exercise

By setting m=2, which subspaces are searched by IP, and in which order ?

[1,1,1,1,1,1,1] IDP 1

[1,1,1,1,1,2] IDP 21

[1,1,1,2,2] IDP 105

[1,1,1,1,3] IDP 35

[1,2,2,2] IP 105

$$\frac{105+105+21+1}{105} = 2.21$$

[1,1,2,3] IP 210

$$\frac{210+35}{210} = 1.16$$

[1,1,1,4] IDP 35

[1,3,3] IP 70

$$\frac{70}{70} = 1$$

[1,2,4] IP 105

$$\frac{105+35}{105} = 1.33$$

[1,1,5] IDP 21

[3,4] IDP 35

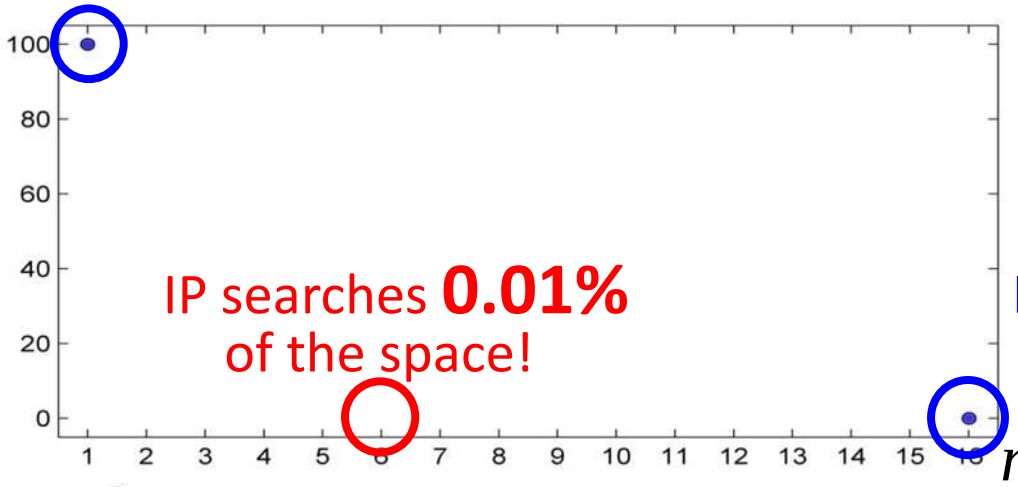
[2,5] IDP 21

[1,6] IDP 7

[7] IDP 1

Worst-case perfomance

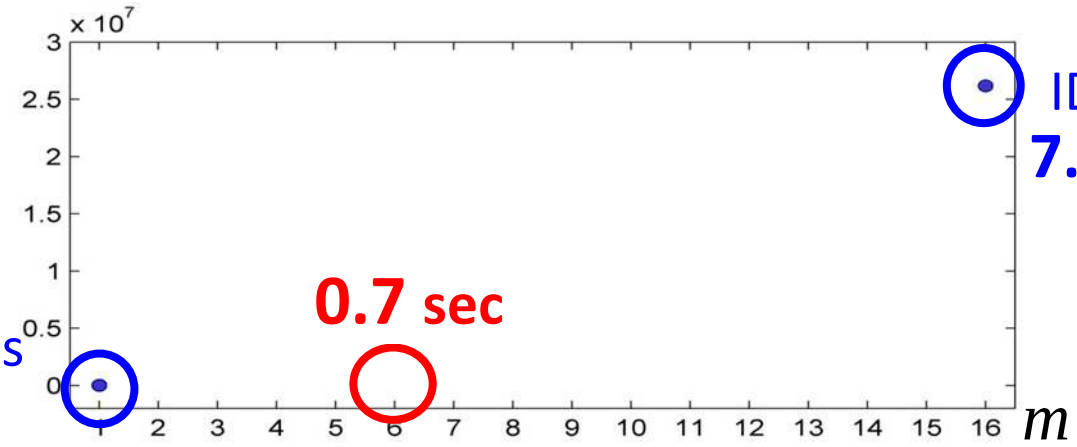
IP searches **100%** of the space



IP searches **0.01%** of the space!

IP searches **0%** of the space

Time to return initial solution

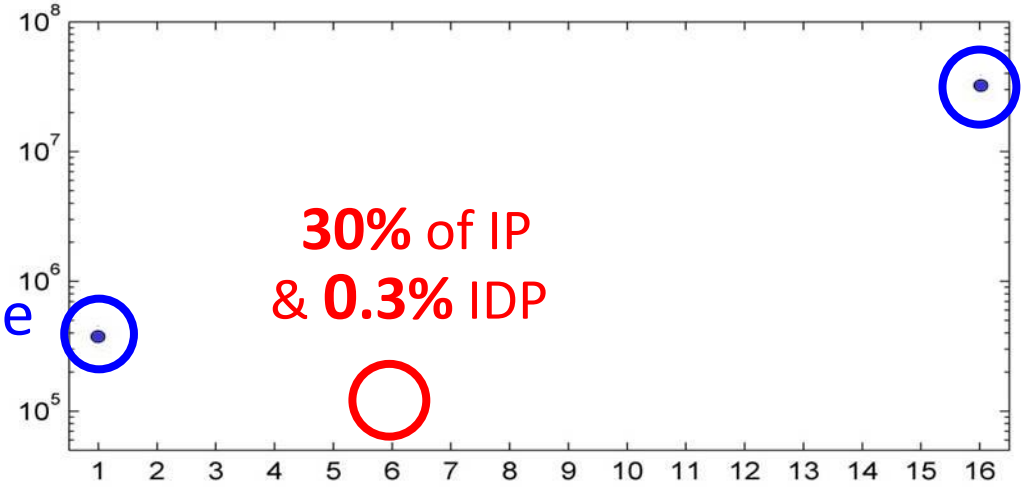


IDP takes **7.2 hours**

0.7 sec

IP takes **0 sec**

Time to return optimal solution



IDP's time

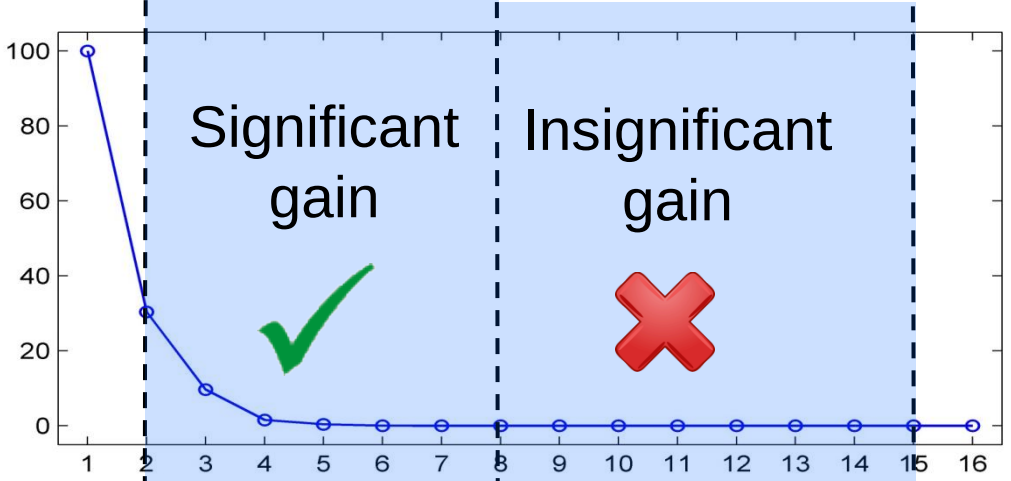
30% of IP & **0.3%** IDP

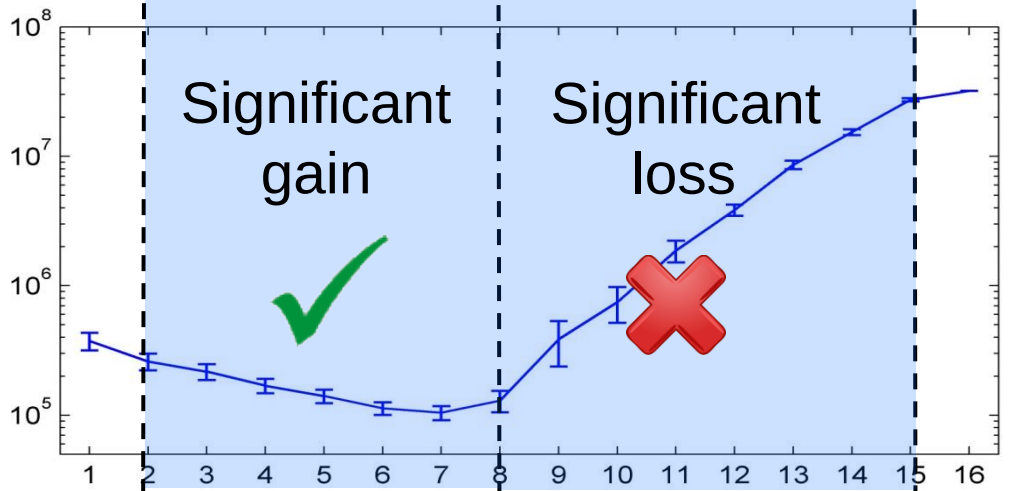
IP's time

Worst-case perfomance

Time to return initial solution

Time to return optimal solution





The number of pruned subspaces and coalition structures

m	Num of evaluated splittings	Num of pruned subspaces	Percentage of pruned subspace	Num of pruned solutions	Percentage of pruned solutions
2	16777515	1268	64.8%	3230863621973843192	69.7%
3	16784415	1478	75.5%	4189923777773485292	90.3%
4	16872965	1703	87.0%	4564993844653940392	98.4%
5	17669915	1776	90.7%	4619890309966821892	99.6%
6	23160015	1843	94.1%	4635658723330786692	99.9%
7	53444115	1874	95.7%	4637889492635473692	>99.9%
8	190804140	1900	97.0%	4638465143123614992	>99.9%
9	711762765	1915	97.8%	4638563634445793142	>99.9%
10	2382099125	1927	98.4%	4638581351908188038	>99.9%
11	6942019325	1935	98.8%	4638588823705171238	>99.9%
12	17587033425	1942	99.1%	4638590208602264538	>99.9%
13	38882261925	1948	99.5%	4638590229863691088	>99.9%
14	74924798325	1953	99.7%	4638590265094980688	>99.9%
15	122135499005	1956	99.9%	4638590307397638228	>99.9%
16	158653677130	1958	100%	4638590332229999353	100%

- By evaluating 17669915 splittings (which takes **0.2 seconds** on our PC), IDP prunes 4.6 x 10^{18} coalition structures (i.e. **99.6% of the space**).
- Increasing m from 14 to 15 causes IDP to evaluate an additional 4.7 x 10^{10} splittings (which takes **2.3 hours**) only to prune **0.0000009% of the space**!

Coalition Structure Generation in Multi-Agent Systems with Positive and Negative Externalities

CSG in Characteristic Function Games

Given 3 agents: the set of agents is:

$\{a_1,a_2,a_3\}$

The possible **coalitions** are:

INPUT
a value for every coalition

$\{a_1\}$	$\{a_2\}$	$\{a_3\}$	$\{a_1,a_2\}$	$\{a_1,a_3\}$	$\{a_2,a_3\}$	$\{a_1,a_2,a_3\}$
20	40	30	70	40	65	95

The possible **coalition structures**:

OUTPUT
an optimal coalition structure

$\{\{a_1\},\{a_2\},\{a_3\}\}$	$\{\{a_1,a_2\},\{a_3\}\}$	$\{\{a_2\},\{a_1,a_3\}\}$	$\{\{a_1\},\{a_2,a_3\}\}$	$\{\{a_1,a_2,a_3\}\}$
20+40+30=90	70+30=100	40+40=80	20+65=85	95

Assumption: The value of a coalition structure is **the sum** of the values of the coalitions within that structure

CSG in ~~Characteristic~~ Partition Function Games

Given 3 agents: the set of agents is:

$\{a_1,a_2,a_3\}$

The possible **coalitions** are:

INPUT
a value for every coalition

$\{a_1\}$	$\{a_2\}$	$\{a_3\}$	$\{a_1,a_2\}$	$\{a_1,a_3\}$	$\{a_2,a_3\}$	$\{a_1,a_2,a_3\}$
~~20~~	40	30	70	40	65	95

The possible **coalition structures**:

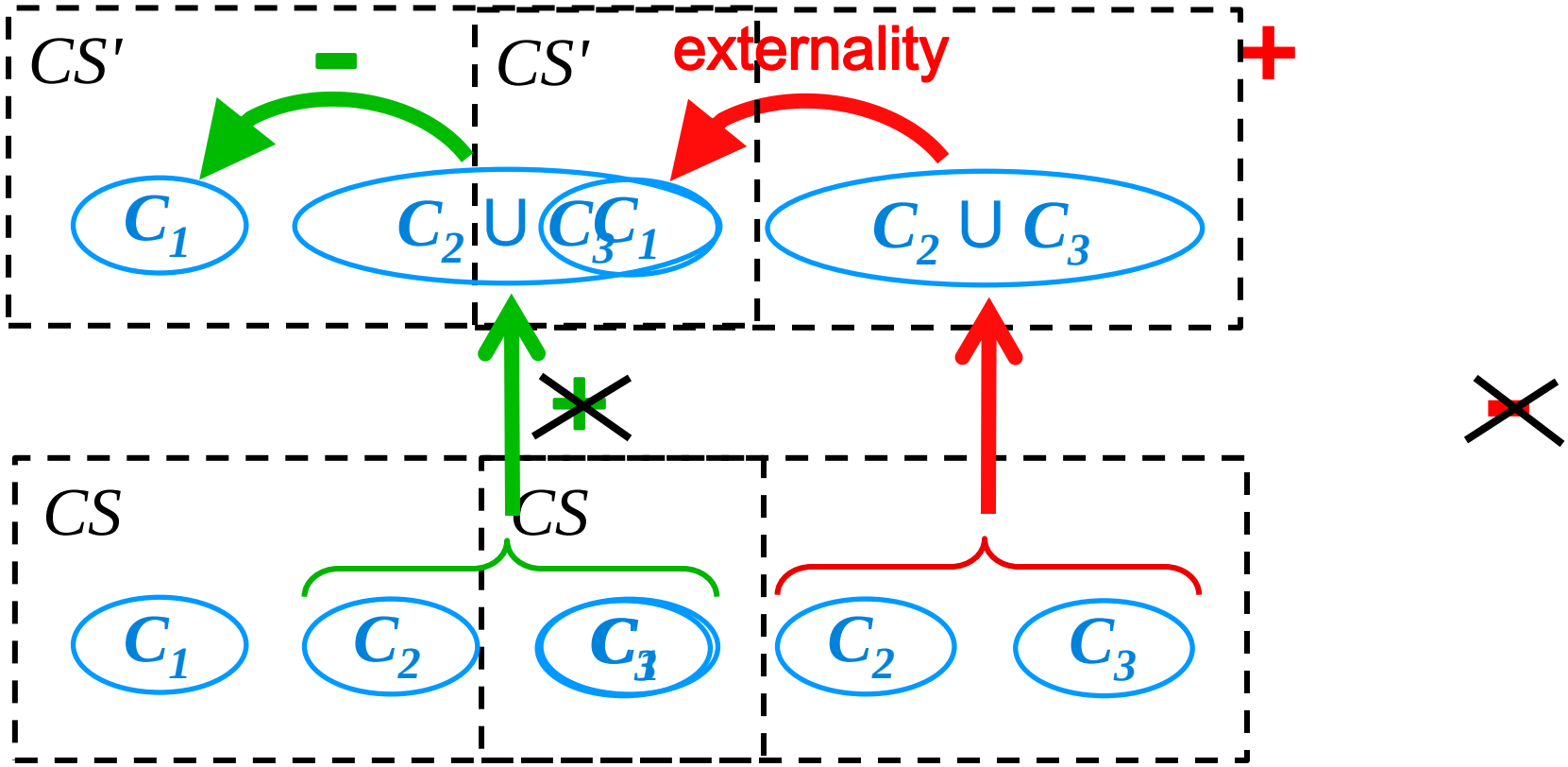
OUTPUT
an optimal coalition structure

$\{\{a_1\},\{a_2\},\{a_3\}\}$	$\{\{a_1,a_2\},\{a_3\}\}$	$\{\{a_2\},\{a_1,a_3\}\}$	$\{\{a_1\},\{a_2,a_3\}\}$	$\{\{a_1,a_2,a_3\}\}$
~~20~~+40+30= ~~90~~ 95	70+30=100	40+40=80	~~20~~ 25+65=~~85~~ 90	95

Assumption: The value of a coalition structure is **the sum** of the values of the coalitions within that structure

The merging of two coalitions may affect the values of other coalitions in the structure!

Michalak et. al. [ECAI-2008] focused on the following classes:



~~Super-additive~~ values,
negative externalities

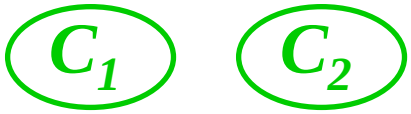
~~Sub-additive~~ values,
positive externalities

Games with externalities are known as *Partition Function Games*

Computing Upper & Lower Bounds

We need to compute upper and lower bounds on the value of any combination of disjoint coalitions

Example:



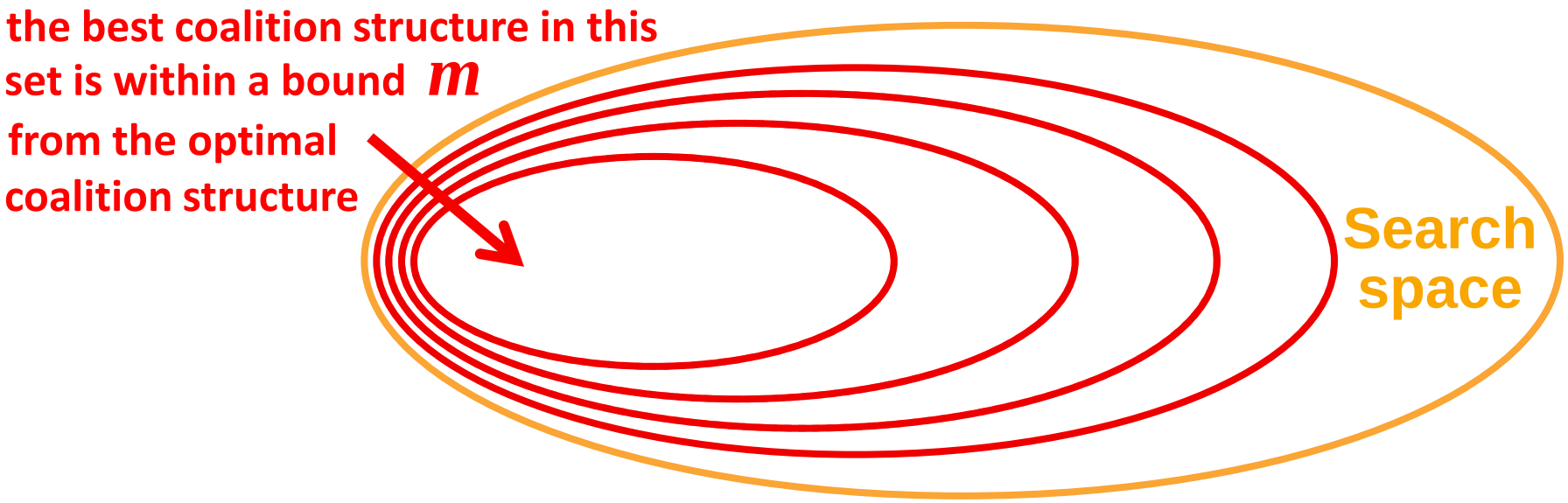
The ***maximum*** possible value of this combination is **500**, and the ***minimum*** possible value is **100**

Theorem : *Consider a* PF^- (PF^+) *setting. Given a coalition* $C \subseteq A$, *a partition* $P \in \mathcal{P}_C$, *and a coalition structure* $CS \supseteq P$, *the following holds, where* $\overline{C} = \{a_1, ..., a_{|\overline{C}|}\}$:

$$V(P, \{\{a_1\}, ..., \{a_{|\overline{C}|}\}\} \cup P) \leq (\geq)\ V(P, CS) \leq (\geq)\ V(P, \{\overline{C}\} \cup P\})$$

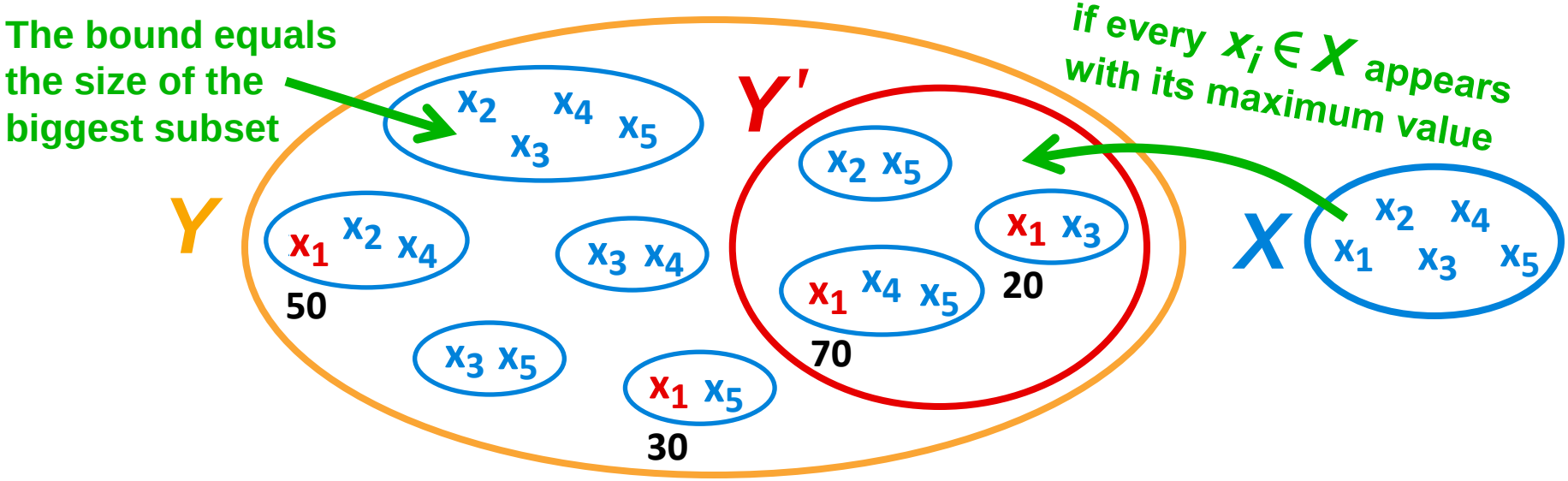
Establishing Worst-Case Guarantees

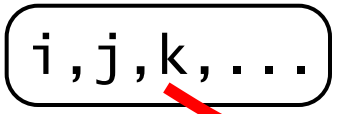
- We need to identify the minimum set of coalition structures that must be searched in order to establish a worst-case guarantee
- We need to develop a novel algorithm for improving the worst-case bound with further search



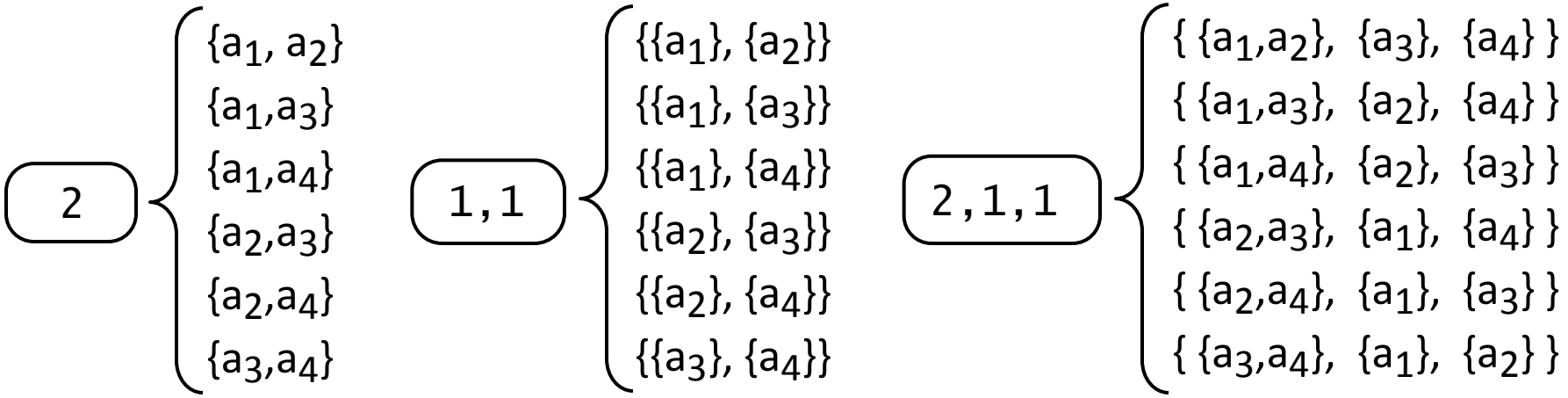
Main Theorem

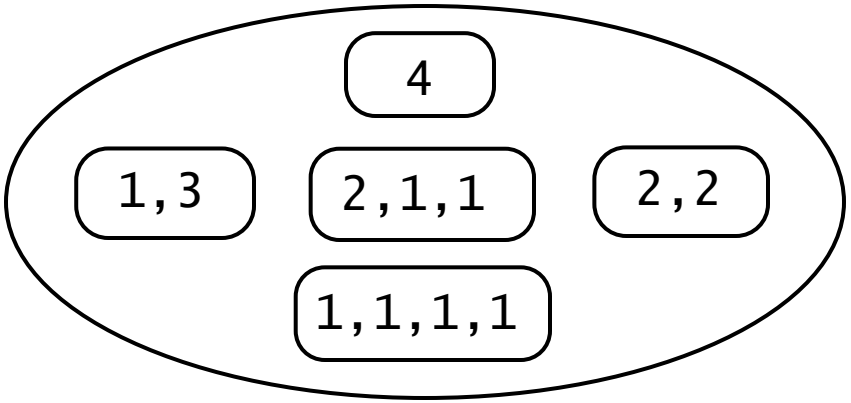
- X is a set of elements
- Y is a set of subsets of X
- Every $x_i \in X$ may have different values in different subsets
- Let Y' be a subset of Y such that every element in X appears with its maximum value in Y'
- then the value of the best subset in Y' is within a bound m from the optimal, where m is the size of the biggest subset in Y



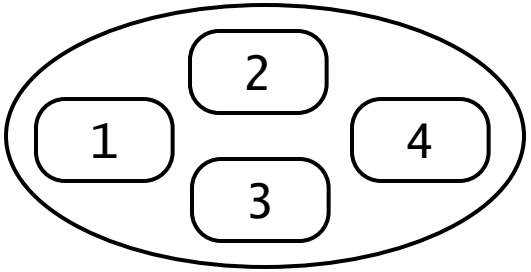
In the next few slides, we will be using the following graphical representation: 

integers represent coalition sizes



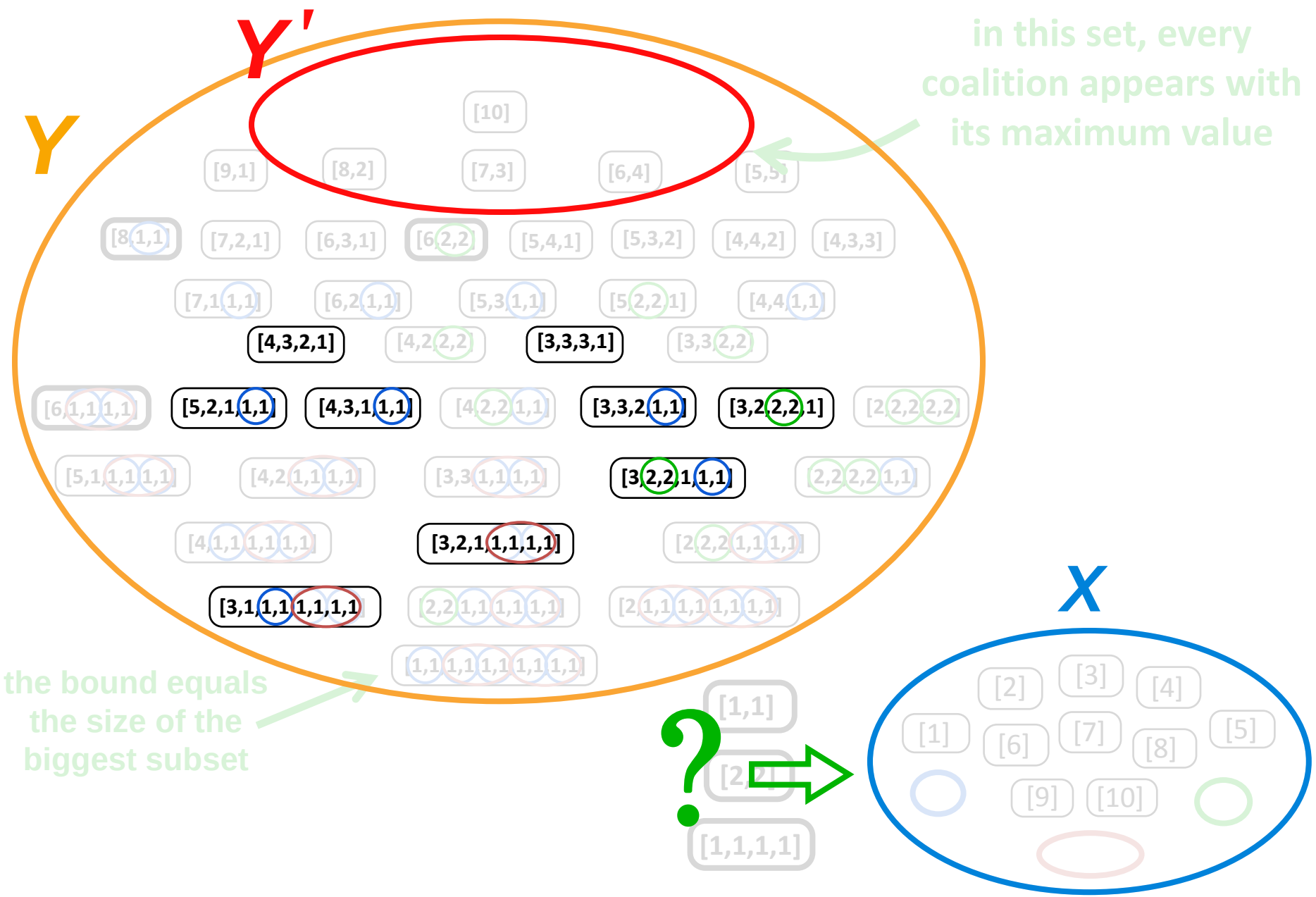


Set of coalition structures



Set of coalitions

Establishing the Worst-Case Bound



Solve Using Integer Programming

$$Minimize \sum_{c=1}^{|columns|} \lambda[c] \times cost[c]$$

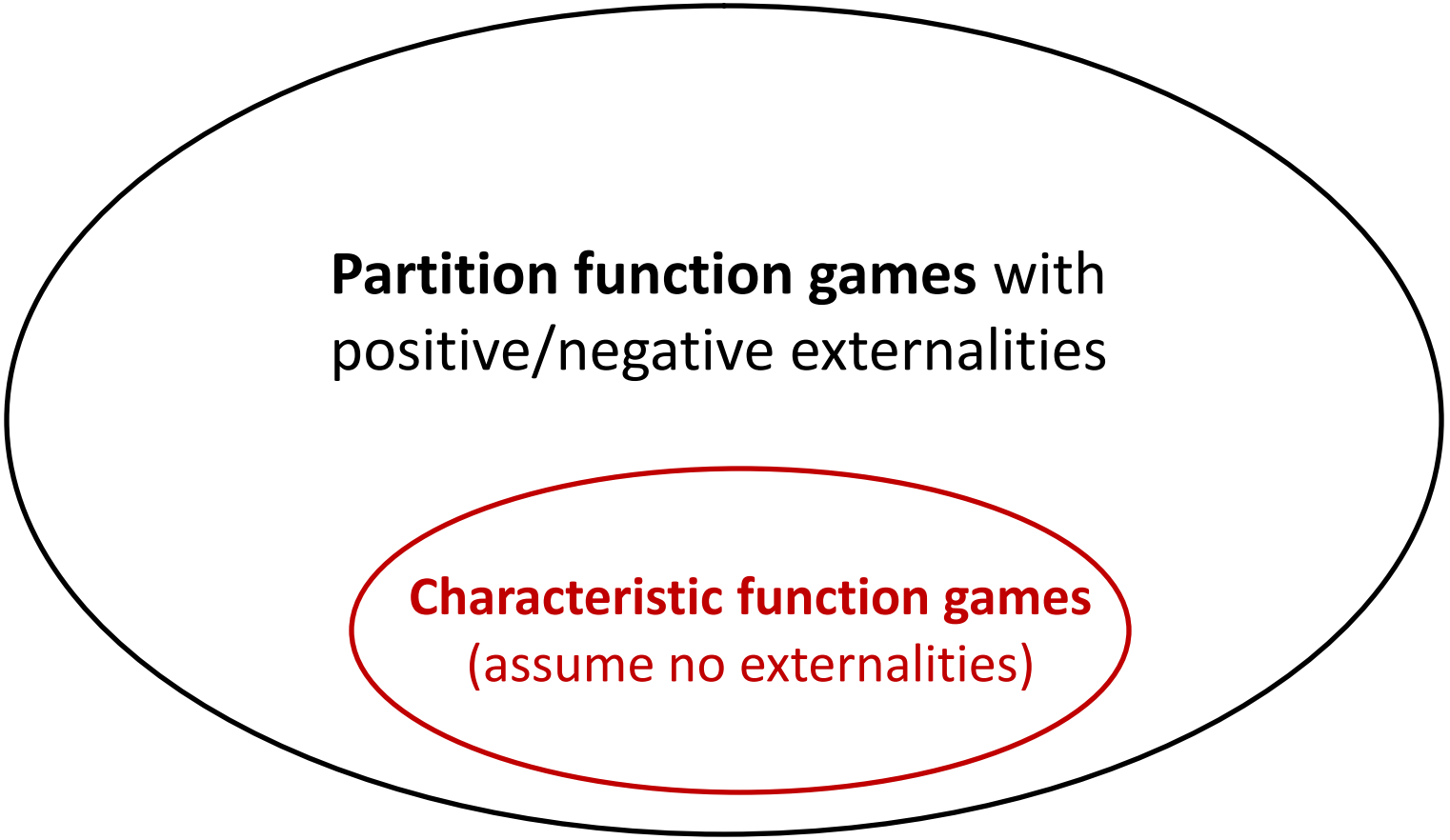
$$s.t. \quad \forall r \in \{1, \cdots, |rows|\}, \sum_{c=1}^{|columns|} \lambda[c] \times matrix[r][c] \geq 1$$

$$\forall c \in \{1, \cdots, |columns|\}, \lambda[c] \in \{0, 1\}$$

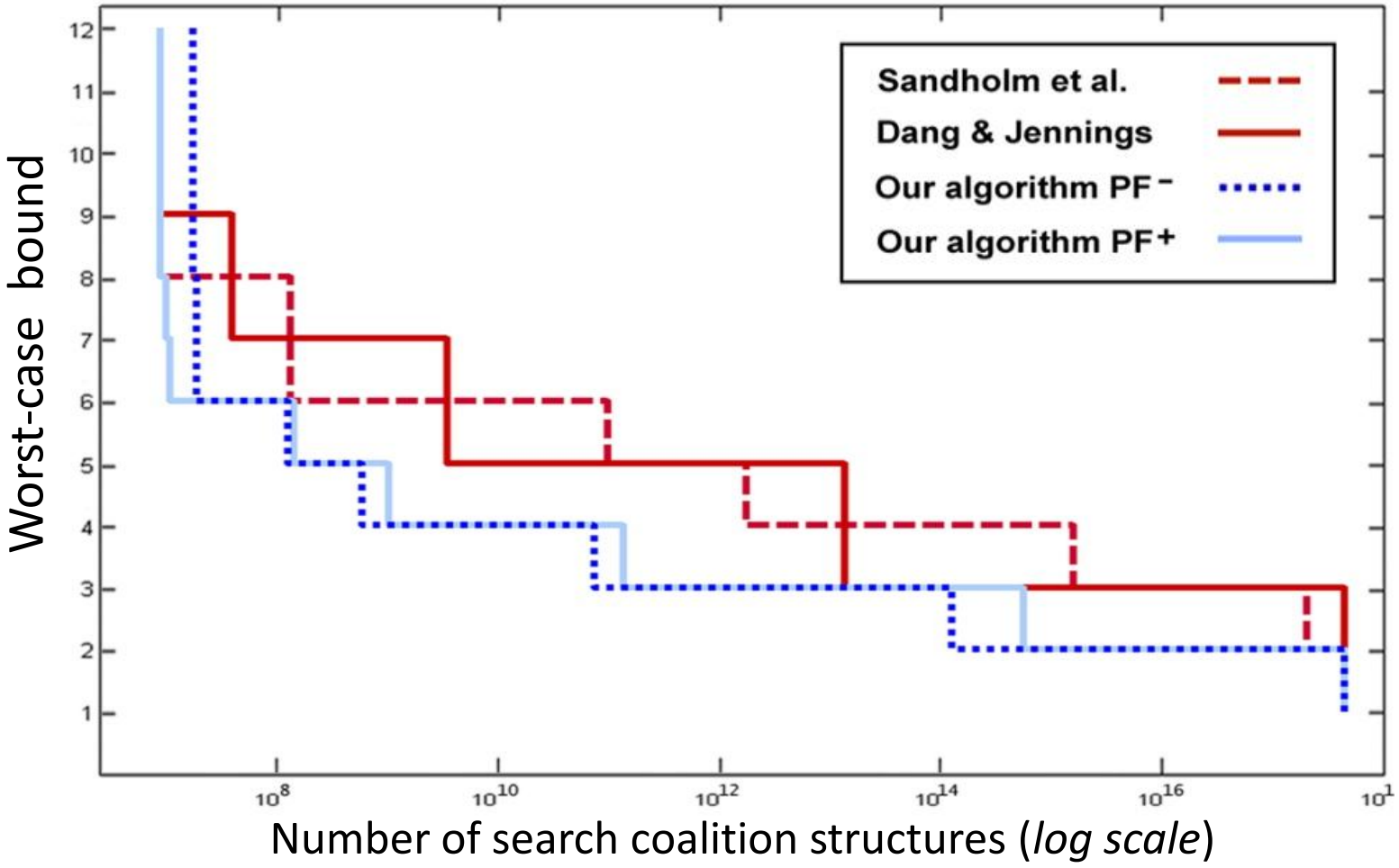
$\lambda =$	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
$cost =$	2520	1260	2520	360	1260	120	2100	1575	1260	2520	840	1575	2520	2100	3150	252	2100	6300	3780	9450	3150	210

matrix	5,2	5,1	5,1,1	2,1	2,1,1	1,1,1	4,3	4,2	4,1	3,2	3,1	4,1,1	3,1,1	3,1,1,1,1	2,1,1,1,1	1,1,1,1,1	3,3	3,2,2	2,2,1	2,2,1,1	2,2,2	1,1,1,1,1,1
[5,2,1(1,1)]	1	1	1	1	1	1																
[4,3,2,1]				1			1	1	1	1	1											
[4,3,1(1,1)]						1	1		1		1	1	1									
[3,2,1(1,1,1,1)]				1						1	1			1	1	1						
[3,3,3,1]											1						1					
[3,3,2(1,1)]					1					1			1				1					
[3(2,2)1(1,1)]						1					1		1					1	1	1		
[3(2,2)2,1]				1						1	1							1	1		1	
[3(1,1,1(1,1,1,1)]						1					1		1	1		1						1

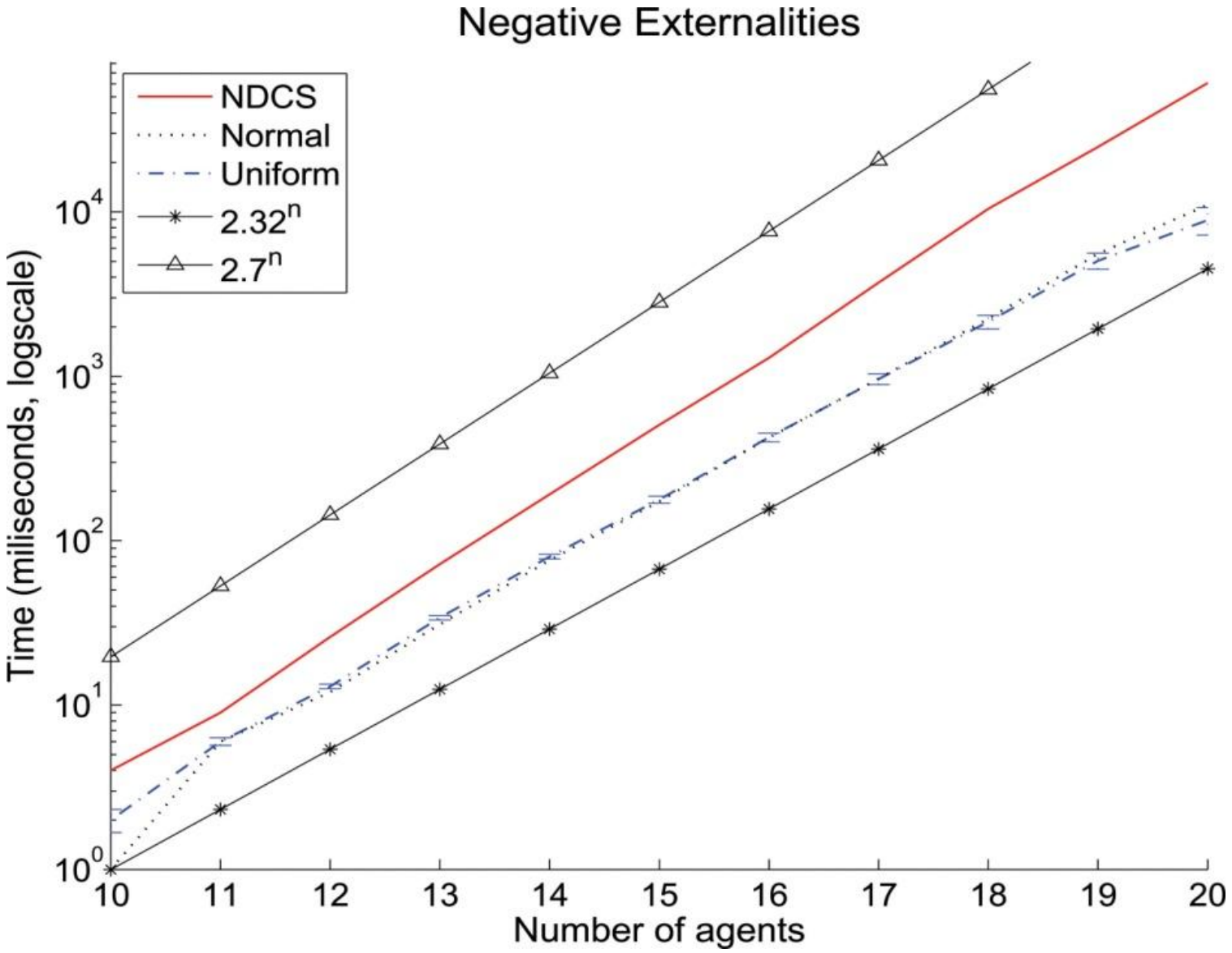
Evaluation

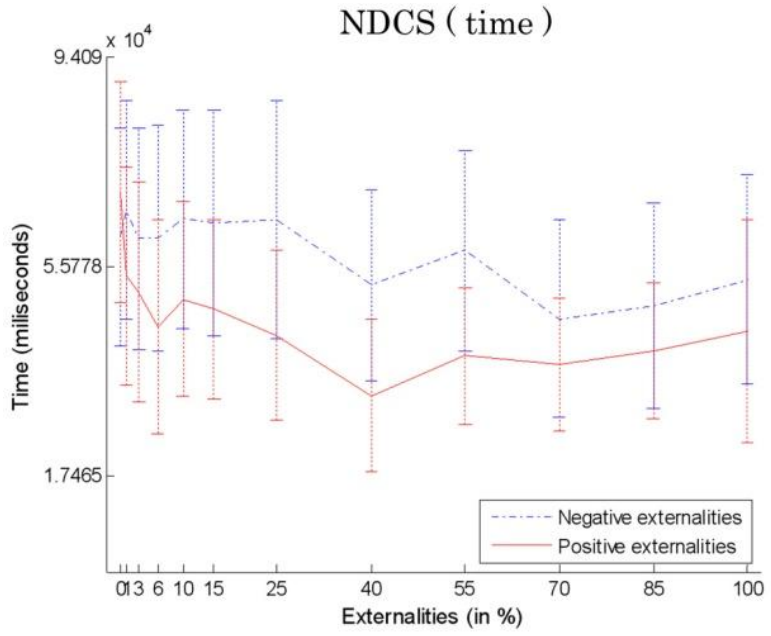

Partition function games with
positive/negative externalities
Characteristic function games
(assume no externalities)

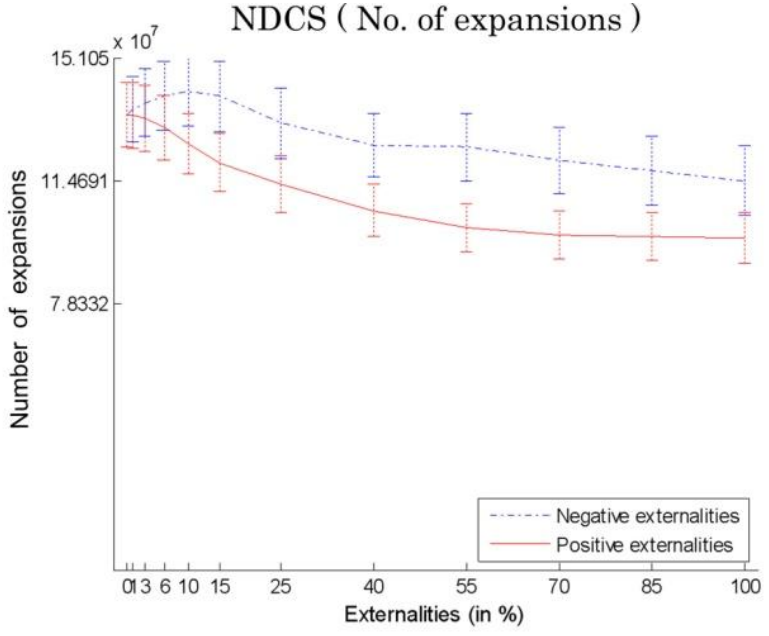
This figure shows, *on a log scale*, how the **worst-case bound** drops (*i.e. improves*) as we search more coalition structures

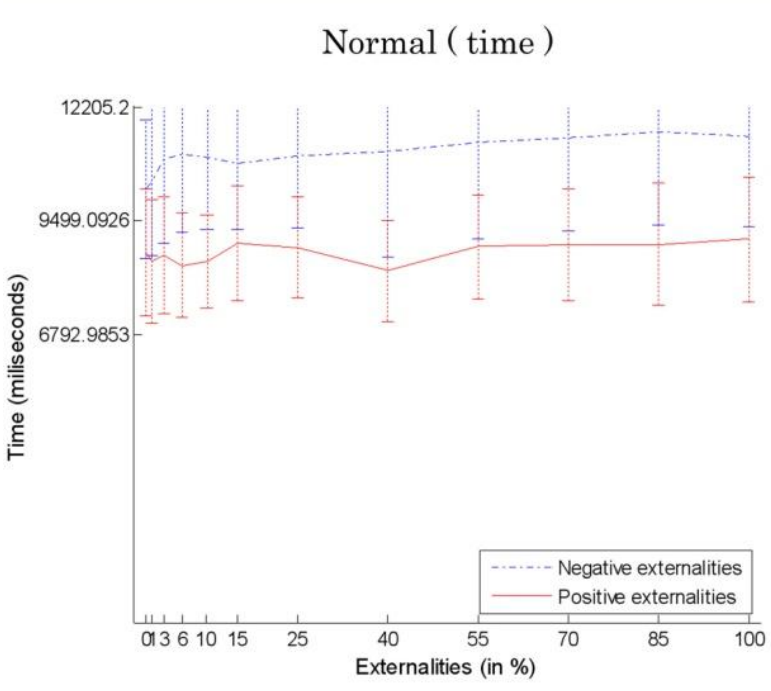


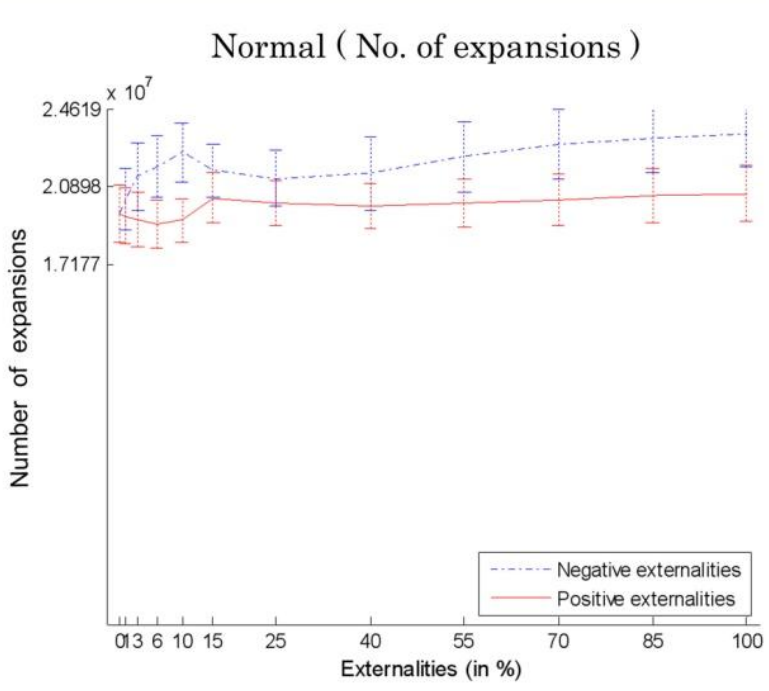
e.g. to reach a bound of 3, we take **0.00002%** compared to Sandholm et al., and **0.5%** compared to Dang and Jennings

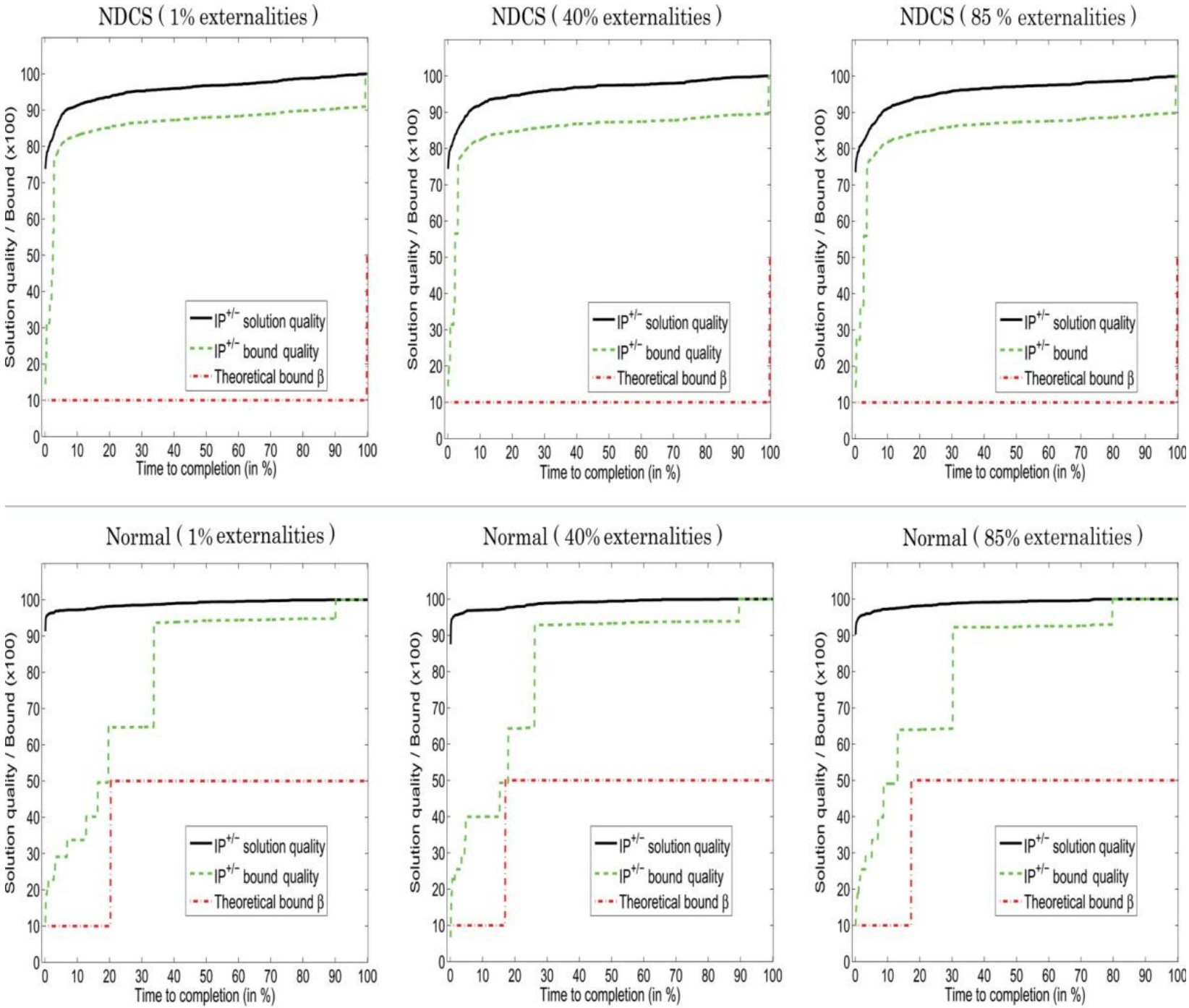

Negative Externalities
NDCS
Normal
Uniform
2.32^n
2.7^n
Time (miliseconds, logscale)
10^0
10^1
10^2
10^3
10^4
10
11
12
13
14
15
16
17
18
19
20
Number of agents

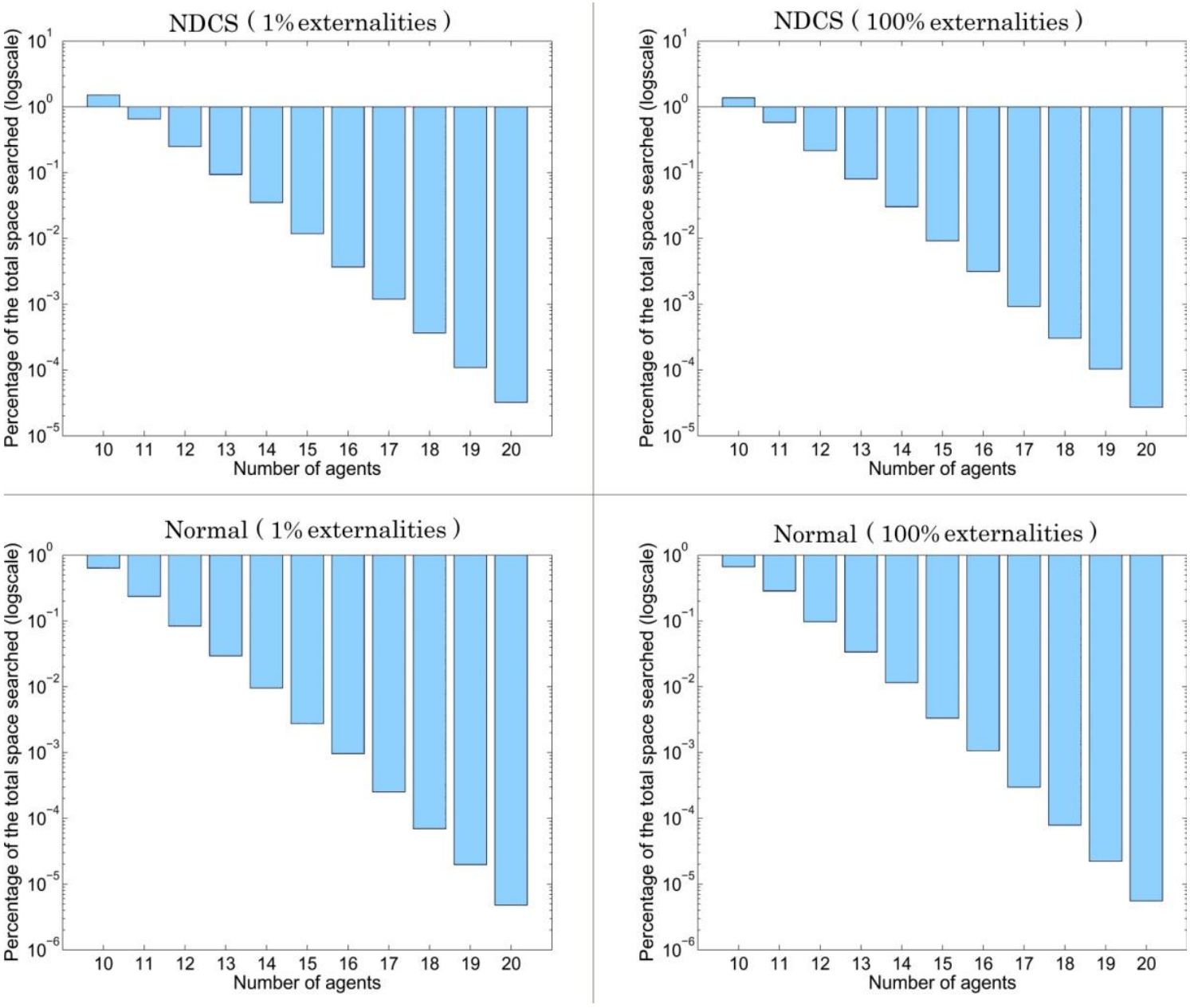

NDCS (time)
x 10^4
9.409
5.5778
1.7465
Time (miliseconds)
Negative externalities
Positive externalities
0 1 3 6 10 15 25 40 55 70 85 100
Externalities (in %)


NDCS (No. of expansions)
x 10^7
15.105
11.4691
7.8332
Number of expansions
Negative externalities
Positive externalities
0 1 3 6 10 15 25 40 55 70 85 100
Externalities (in %)


Normal (time)
12205.2
9499.0926
6792.9853
Time (miliseconds)
Negative externalities
Positive externalities
0 1 3 6 10 15 25 40 55 70 85 100
Externalities (in %)


Normal (No. of expansions)
x 10^7
2.4619
2.0898
1.7177
Number of expansions
Negative externalities
Positive externalities
0 1 3 6 10 15 25 40 55 70 85 100
Externalities (in %)


NDCS (1% externalities)
NDCS (40% externalities)
NDCS (85 % externalities)
Normal (1% externalities)
Normal (40% externalities)
Normal (85% externalities)
Solution quality / Bound (×100)
Time to completion (in %)
IP+/- solution quality
IP+/- bound quality
IP+/- bound
Theoretical bound β


NDCS (1% externalities)
NDCS (100% externalities)
Normal (1% externalities)
Normal (100% externalities)
Percentage of the total space searched (logscale)
Number of agents
10 11 12 13 14 15 16 17 18 19 20