Cooperative Games for the Interpretation of ML Models

Some Methodological and Computational Aspects

[1]**Département de mathématiques** - *Université du Québec à Montréal*
[2]**Institut Intelligence et Données** - *Université Laval*
[3]**Nexialog Consulting**

Séminaire PSPP
EDF Lab Chatou - Chatou, France
October 28, 2025

Marouane Il Idrissi[12], Lucas Offroy[13]

Introduction

In this talk:

Revisit **post-hoc model-agnostic** XAI methods based on **cooperative game theory**

Introduction

In this talk:
Revisit **post-hoc model-agnostic** XAI methods based on **cooperative game theory**

They promise a variable **influence quantification** of **non-linear** ML models

Introduction

In this talk:

Revisit **post-hoc model-agnostic** XAI methods based on **cooperative game theory**

They promise a variable **influence quantification** of **non-linear** ML models

They heavily rely on the notion of **the Shapley values**

☞ **Often misunderstood and misused in XAI**

Introduction

In this talk:

Revisit **post-hoc model-agnostic** XAI methods based on **cooperative game theory**

They promise a variable **influence quantification** of **non-linear** ML models

They heavily rely on the notion of **the Shapley values**

☞ **Often misunderstood and misused in XAI**

In this presentation:

How can we leverage the theory of cooperative games for the interpretation of black-box machine learning models?

Two parts:

1. **Methodological aspects** (Marouane) ☞ Not a lot of new stuff
2. **Computational aspects** (Lucas) ☞ A lot of new stuff

Part I
Methodological aspects

Cooperative Game Theory

"Cooperative game theory = The art of sharing a cake"

"Cooperative game theory = The art of sharing a cake"

Two ingredients:

- $D = \{1, \ldots, d\}$, a **set of players**
  The power-set $\mathcal{P}_D$ is the **set of coalitions of players**
- $v : \mathcal{P}_D \to \mathbb{R}$, a **value function**
  It **assigns a value to each coalition**

☞ (D, v) formally defines a **cooperative game**
☞ $v(D)$ is the value of the "grand coalition" (the cake)

Cooperative Game Theory

"Cooperative game theory = The art of sharing a cake"

Two ingredients:

- $D = \{1, \ldots, d\}$, a **set of players**
  The power-set $\mathcal{P}_D$ is the **set of coalitions of players**
- $v : \mathcal{P}_D \to \mathbb{R}$, a **value function**
  It **assigns a value to each coalition**

☞ (D, v) formally defines a **cooperative game**
☞ $v(D)$ is the value of the "grand coalition" (the cake)

Main concern of the theory of cooperative games:

How can to redistribute $v(D)$ to each of the d players?

A famous example - LMG indices

Example: Lindeman, Merenda, and Gold (1980) indices

A famous example - LMG indices

Example: Lindeman, Merenda, and Gold (1980) indices

Data: covariates $X = (X_1, \ldots, X_d)$, and target Y
Model: estimated linear regression model $\widehat{f}(X) = \widehat{\beta}_0 + X^\top \widehat{\beta}$

A famous example - LMG indices

Example: Lindeman, Merenda, and Gold (1980) indices

Data: covariates $X = (X_1, \ldots, X_d)$, and target Y
Model: estimated linear regression model $\widehat{f}(X) = \widehat{\beta}_0 + X^\top \widehat{\beta}$

Cooperative game:
Players: the covariates $(X_1, \ldots, X_d)$ (rather, their indices D)
Coalitions: for $A \in \mathcal{P}_D$, the subset of covariates X_A
Value function: $v(A) = R^2_Y(X_A)$ (the R^2 coefficient of the linear model only using X_A)

A famous example - LMG indices

Example: Lindeman, Merenda, and Gold (1980) indices

Data: covariates $X = (X_1, \ldots, X_d)$, and target Y
Model: estimated linear regression model $\widehat{f}(X) = \widehat{\beta}_0 + X^\top \widehat{\beta}$

Cooperative game:
Players: the covariates $(X_1, \ldots, X_d)$ (rather, their indices D)
Coalitions: for $A \in \mathcal{P}_D$, the subset of covariates X_A
Value function: $v(A) = R^2_Y(X_A)$ (the R^2 coefficient of the linear model only using X_A)

☞ **The cake:** $v(D)$, the R^2 of the **full model**

A famous example - LMG indices

Example: Lindeman, Merenda, and Gold (1980) indices

Data: covariates $X = (X_1, \ldots, X_d)$, and target Y
Model: estimated linear regression model $\widehat{f}(X) = \widehat{\beta}_0 + X^\top \widehat{\beta}$

Cooperative game:
Players: the covariates $(X_1, \ldots, X_d)$ (rather, their indices D)
Coalitions: for $A \in \mathcal{P}_D$, the subset of covariates X_A
Value function: $v(A) = R^2_Y(X_A)$ (the R^2 coefficient of the linear model only using X_A)

☞ **The cake:** $v(D)$, the R^2 of the **full model**

Evaluate the **value function** $\forall A \in \mathcal{P}_D \iff$ The R^2 of all the 2^d nested linear models
For 20 covariate: more than a million R^2 coefficients

A famous example - LMG indices

Example: Lindeman, Merenda, and Gold (1980) indices

Data: covariates $X = (X_1, \ldots, X_d)$, and target Y
Model: estimated linear regression model $\widehat{f}(X) = \widehat{\beta}_0 + X^\top \widehat{\beta}$

Cooperative game:
Players: the covariates $(X_1, \ldots, X_d)$ (rather, their indices D)
Coalitions: for $A \in \mathcal{P}_D$, the subset of covariates X_A
Value function: $v(A) = R^2_Y(X_A)$ (the R^2 coefficient of the linear model only using X_A)

☞ **The cake:** $v(D)$, the R^2 of the **full model**

Evaluate the **value function** $\forall A \in \mathcal{P}_D \iff$ The R^2 of all the 2^d nested linear models
For 20 covariate: more than a million R^2 coefficients

☞ **This is way too much information to process...**

A famous example - LMG indices

Example: Lindeman, Merenda, and Gold (1980) indices

Data: covariates $X = (X_1, \ldots, X_d)$, and target Y
Model: estimated linear regression model $\widehat{f}(X) = \widehat{\beta}_0 + X^\top \widehat{\beta}$

Cooperative game:
Players: the covariates $(X_1, \ldots, X_d)$ (rather, their indices D)
Coalitions: for $A \in \mathcal{P}_D$, the subset of covariates X_A
Value function: $v(A) = R^2_Y(X_A)$ (the R^2 coefficient of the linear model only using X_A)

☞ **The cake:** $v(D)$, the R^2 of the **full model**

Evaluate the **value function** $\forall A \in \mathcal{P}_D \iff$ The R^2 of all the 2^d nested linear models
For 20 covariate: more than a million R^2 coefficients

☞ **This is way too much information to process...**

Is it possible to aggregate this information (2^d coefficients) into something more manageable?

Allocations as an aggregation

This is **exactly the role of an allocation**!
It summarizes the 2^d evaluations of v into **one quantity for each player**

Allocations as an aggregation

This is **exactly the role of an allocation**!
It summarizes the 2^d evaluations of v into **one quantity for each player**

It is a mapping $\phi : D \to \mathbb{R}$, that must ideally respect **one** criteria:

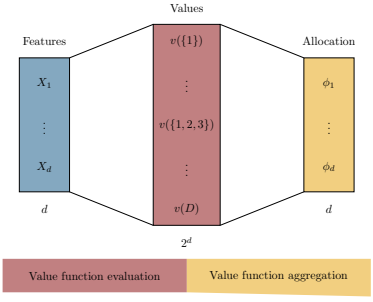
- **Efficiency:** $\sum_{i \in D} \phi(i) = v(D)$
  ☞ Ensures that **the we actually redistribute the cake**

Allocations as an aggregation

This is **exactly the role of an allocation**!
It summarizes the 2^d evaluations of v into **one quantity for each player**

It is a mapping $\phi : D \to \mathbb{R}$, that must ideally respect **one** criteria:

- **Efficiency:** $\sum_{i \in D} \phi(i) = v(D)$
  - ☞ Ensures that **the we actually redistribute the cake**



In a nuthsell:

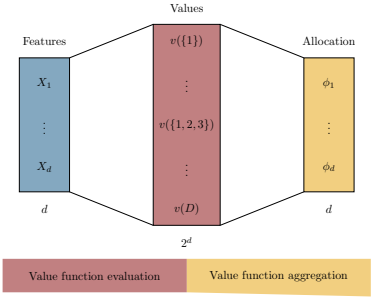
- Start with a learned model with d input features
- Chose a **value function** resulting in 2^d quantities
- Aggregate the 2^d quantities into d quantities using an **efficient allocation**

Allocations as an aggregation

This is **exactly the role of an allocation**!
It summarizes the 2^d evaluations of v into **one quantity for each player**

It is a mapping $\phi : D \to \mathbb{R}$, that must ideally respect **one** criteria:

- **Efficiency:** $\sum_{i \in D} \phi(i) = v(D)$
  - ☞ Ensures that **the we actually redistribute the cake**



In a nuthsell:

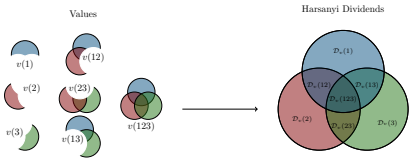
- Start with a learned model with d input features
- Chose a **value function** resulting in 2^d quantities
- Aggregate the 2^d quantities into d quantities using an **efficient allocation**

Is there a way to define efficient allocations?

Allocations as a dividend sharing mechanism

The **Harsanyi (1963) dividends** of a cooperative game (D, v) are defined as:

$$\mathcal{D}_v(A) = \sum_{B \in \mathcal{P}_A} (-1)^{|A|-|B|} v(B), \quad \text{or equivalently,} \quad \mathcal{D}_v(A) = v(A) - \sum_{B \in \mathcal{P}_A} \mathcal{D}_v(B)$$

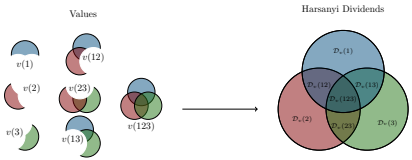


They quantify **the added-value of a coalition**:
$\mathcal{D}_v(12) = v(12) - v(1) - v(2)$

Allocations as a dividend sharing mechanism

The **Harsanyi (1963) dividends** of a cooperative game (D, v) are defined as:

$$\mathcal{D}_v(A) = \sum_{B \in \mathcal{P}_A} (-1)^{|A|-|B|} v(B), \quad \text{or equivalently,} \quad \mathcal{D}_v(A) = v(A) - \sum_{B \in \mathcal{P}_A} \mathcal{D}_v(B)$$



They quantify **the added-value of a coalition**:
$\mathcal{D}_v(12) = v(12) - v(1) - v(2)$

They are the **Möbius inverse** of the **value function**

Proposition *(Möbius inversion on power-sets (Rota 1964; Kung, Rota, and Hung Yan 2012)).*

For any two set functions $v : \mathcal{P}_D \to \mathbb{R}$, $\mathcal{D} : \mathcal{P}_D \to \mathbb{R}$, the following equivalence holds:

$$\forall A \in \mathcal{P}_D, \quad v(A) = \sum_{B \in \mathcal{P}_A} \mathcal{D}(B), \quad \iff \quad \forall A \in \mathcal{P}_D, \quad \mathcal{D}(A) = \sum_{B \in \mathcal{P}_A} (-1)^{|A|-|B|} v(B).$$

(i.e., generalized inclusion-exclusion principle)

Shapley values as the egalitarian dividend sharing mechanism

The **Harsanyi set** is a **family of efficient allocations** that **aggregate of the Harsanyi dividends**:

$$\phi(i) = \sum_{A \in \mathcal{P}_D \,:\, i \in A} \lambda_i(A) \mathcal{D}_v(A), \quad \text{where} \quad \begin{cases} \forall i \in D, \forall A \in \mathcal{P}_D,\ \lambda_i(A) \geq 0, \\ \forall A \in \mathcal{P}_D,\ \sum_{i \in D} \lambda_i(A) = 1 \end{cases}$$

parametrized by the **weight system** $\lambda : D \times \mathcal{P}_D \to \mathbb{R}$

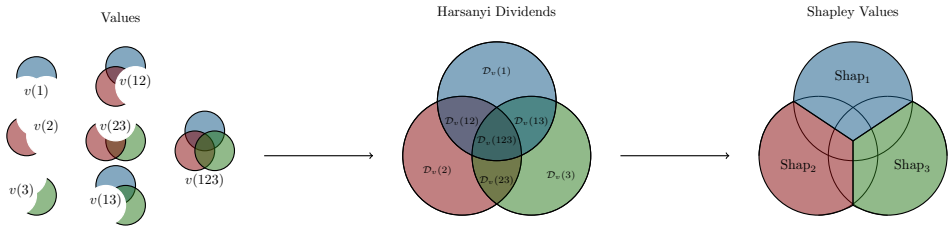
Shapley values as the egalitarian dividend sharing mechanism

The **Harsanyi set** is a **family of efficient allocations** that **aggregate of the Harsanyi dividends**:

$$\phi(i) = \sum_{A \in \mathcal{P}_D \,:\, i \in A} \lambda_i(A) \mathcal{D}_v(A), \quad \text{where} \quad \begin{cases} \forall i \in D, \forall A \in \mathcal{P}_D,\ \lambda_i(A) \geq 0, \\ \forall A \in \mathcal{P}_D,\ \sum_{i \in D} \lambda_i(A) = 1 \end{cases}$$

parametrized by the **weight system** $\lambda : D \times \mathcal{P}_D \to \mathbb{R}$

In this setting, the **Shapley values are the egalitarian redistribution**, i.e., $\lambda_i(A) = 1/|A|$



Allocations using random orders

The **Weber (1988) set of allocations** relies on the notion of **random orders**

Allocations using random orders

The **Weber (1988) set of allocations** relies on the notion of **random orders**

Let $\mathcal{S}_D$ be the **set of permutations** $\pi = (\pi_1, \ldots, \pi_d)$ (i.e., orders) of players
For any $i \in D$, denote $\pi(i)$ the **position of player i in the permutation** π (i.e., $\pi_{\pi(i)} = i$)

Allocations using random orders

The **Weber (1988) set of allocations** relies on the notion of **random orders**

Let $\mathcal{S}_D$ be the **set of permutations** $\pi = (\pi_1, \ldots, \pi_d)$ (i.e., orders) of players
For any $i \in D$, denote $\pi(i)$ the **position of player i in the permutation** π (i.e., $\pi_{\pi(i)} = i$)

The **Weber (1988) set** is a family of **efficient allocations** as an average over the orderings

$$\begin{aligned}\phi(i) &= \mathbb{E}_{\pi \sim p}\left[v\left(\{\pi_1, \ldots, \pi_{\pi(i)}\}\right) - v\left(\{\pi_1, \ldots, \pi_{\pi(i)-1}\}\right)\right] \\ &= \sum_{\pi \in \mathcal{S}_D} p(\pi)\left[v\left(\{\pi_1, \ldots, \pi_{\pi(i)}\}\right) - v\left(\{\pi_1, \ldots, \pi_{\pi(i)-1}\}\right)\right]\end{aligned}$$

parametrized by a **probability mass function** over the permutations $\mathcal{S}_D$.

Allocations using random orders

The **Weber (1988) set of allocations** relies on the notion of **random orders**

Let $\mathcal{S}_D$ be the **set of permutations** $\pi = (\pi_1, \ldots, \pi_d)$ (i.e., orders) of players
For any $i \in D$, denote $\pi(i)$ the **position of player i in the permutation** π (i.e., $\pi_{\pi(i)} = i$)

The **Weber (1988) set** is a family of **efficient allocations** as an average over the orderings

$$\begin{aligned}\phi(i) &= \mathbb{E}_{\pi \sim p}\left[v\left(\{\pi_1, \ldots, \pi_{\pi(i)}\}\right) - v\left(\{\pi_1, \ldots, \pi_{\pi(i)-1}\}\right)\right] \\ &= \sum_{\pi \in \mathcal{S}_D} p(\pi)\left[v\left(\{\pi_1, \ldots, \pi_{\pi(i)}\}\right) - v\left(\{\pi_1, \ldots, \pi_{\pi(i)-1}\}\right)\right]\end{aligned}$$

parametrized by a **probability mass function** over the permutations $\mathcal{S}_D$.

In this setting, the **Shapley values are the uniform distribution over the permutations**, i.e., $p(\pi) = 1/d!$

$$\mathrm{Shap}(i) = \frac{1}{d!} \sum_{\pi \in \mathcal{S}_D} \left[v\left(\{\pi_1, \ldots, \pi_{\pi(i)}\}\right) - v\left(\{\pi_1, \ldots, \pi_{\pi(i)-1}\}\right)\right]$$

The recipe

Overall blueprint for using cooperative games for XAI:
(I., Charpentier, and Fernandes Machado 2025)

1. **Step 1: Identify a quantity of interest**
   Choose **a cake worth cutting**, e.g., point predictions $f(x)$, model variance $\mathbb{V}(f(X))$...
   ☞ Guides the **interpretation of the extracted insights**

The recipe

Overall blueprint for using cooperative games for XAI:
(I., Charpentier, and Fernandes Machado 2025)

1. **Step 1: Identify a quantity of interest**
   Choose **a cake worth cutting**, e.g., point predictions $f(x)$, model variance $\mathbb{V}(f(X))$...
   ☞ Guides the **interpretation of the extracted insights**

2. **Step 2: Pick a value function v**
   And make sure that $v(D)$ **is equal to the quantity of interest**, e.g.,
   $\mathbb{E}[f(X) \mid X_A = x_A]$ for $f(x)$, $\mathbb{V}(\mathbb{E}[f(X) \mid X_A])$ for $\mathbb{V}(f(X))$...
   ☞ **This step is the most important** (garbage in - garbage out)

The recipe

Overall blueprint for using cooperative games for XAI:
(I., Charpentier, and Fernandes Machado 2025)

1. **Step 1: Identify a quantity of interest**
   Choose **a cake worth cutting**, e.g., point predictions $f(x)$, model variance $\mathbb{V}(f(X))$...
   ☞ Guides the **interpretation of the extracted insights**

2. **Step 2: Pick a value function v**
   And make sure that $v(D)$ **is equal to the quantity of interest**, e.g.,
   $\mathbb{E}[f(X) \mid X_A = x_A]$ for $f(x)$, $\mathbb{V}(\mathbb{E}[f(X) \mid X_A])$ for $\mathbb{V}(f(X))$...
   ☞ **This step is the most important** (garbage in - garbage out)

3. **Step 3: Pick an efficient allocation**
   In order to summarize the information of the 2^d evaluations of v
   ☞ Less crucial and can **highlight some model behavior**

Conclusion

Key take-aways:

- **Three ingredients** to using cooperative game for XAI:
  ☞ The **quantity of interest**, the **value function**, and the **allocation**

Conclusion

Key take-aways:

- **Three ingredients** to using cooperative game for XAI:
  ☞ The **quantity of interest**, the **value function**, and the **allocation**
- The **Shapley values** are **one example of allocation**
  ☞ **Egalitarian redistribution** (Harsanyi set), **uniformly likely orders** (Weber set)

It's all fine in theory...

... but how can we compute everything that is needed efficiently?

Conclusion

Key take-aways:

- **Three ingredients** to using cooperative game for XAI:
  ☞ The **quantity of interest**, the **value function**, and the **allocation**
- The **Shapley values** are **one example of allocation**
  ☞ **Egalitarian redistribution** (Harsanyi set), **uniformly likely orders** (Weber set)

It's all fine in theory...

... but how can we compute everything that is needed efficiently?

Part II
Computational Aspects

It's always harder in practice

- **Thank you Marouane, but...**

  ☞ $\sum_{i=0}^{d} \binom{d}{i} = 2^d$, How can we deal with that amount of models ?

It's always harder in practice

- **Thank you Marouane, but...**

  ☞ $\sum_{i=0}^{d} \binom{d}{i} = 2^d$, How can we deal with that amount of models ?

- **Temporal complexity of the allocations formulas** ☞

$$\begin{cases} \textit{Shapley} : O(d2^d) \\ \textit{Harsanyi} : O(3^d) \\ \textit{Weber} : O(d!) \end{cases}$$

It's always harder in practice

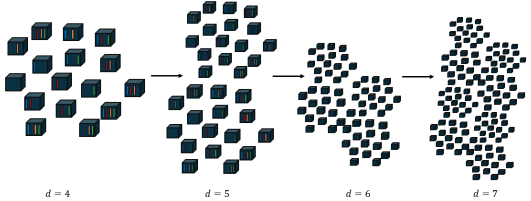
- **Thank you Marouane, but...**

  ☞ $\sum_{i=0}^{d} \binom{d}{i} = 2^d$, How can we deal with that amount of models ?

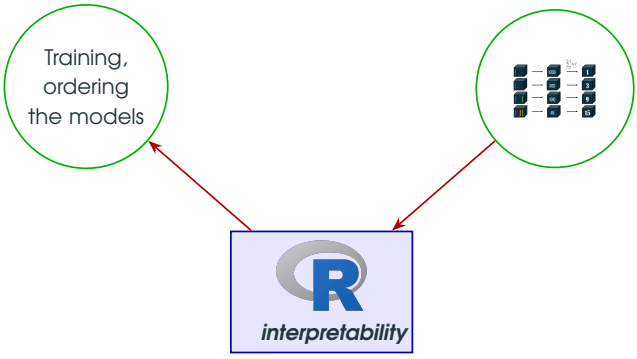
- **Temporal complexity of the allocations formulas** ☞ $\begin{cases} \textit{Shapley} : O(d2^d) \\ \textit{Harsanyi} : O(3^d) \\ \textit{Weber} : O(d!) \end{cases}$

- **Space complexity of the models**

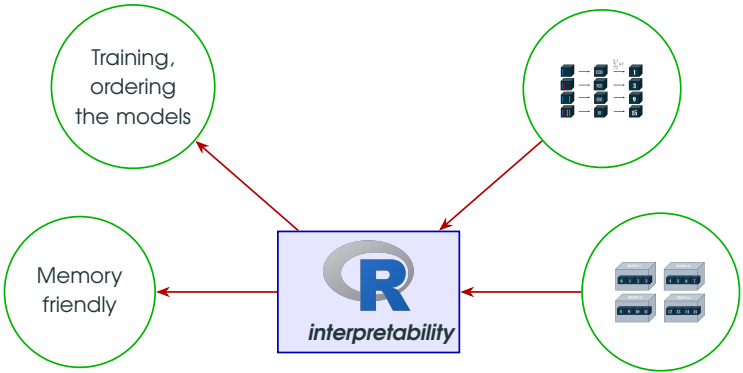
  ☞ For each feature added in the model, we need to double the amount of storage



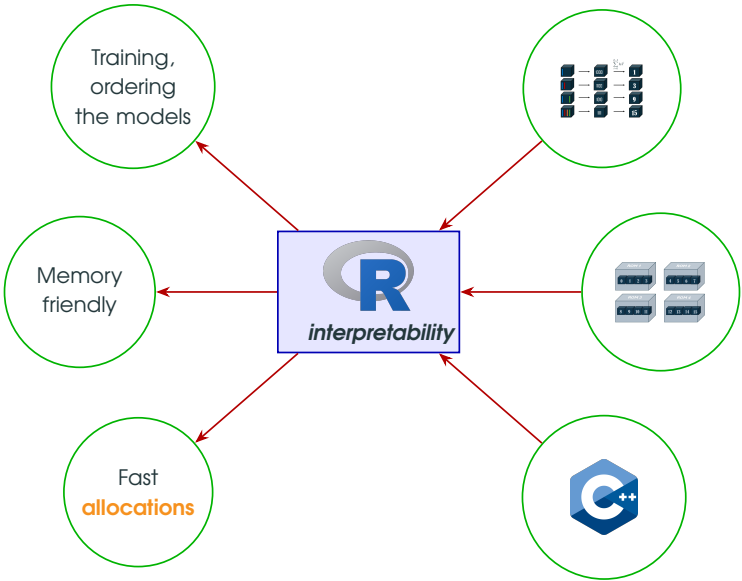
Our summer vacation project : interpretability



Our summer vacation project : interpretability



Our summer vacation project : interpretability



The art of organizing 2^d models

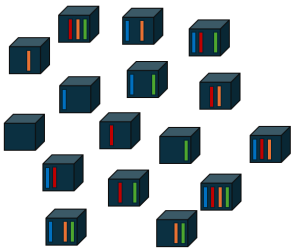
- **Other supervisors in other internships :** : "Tidy your desk"
  ☞ **Easy**

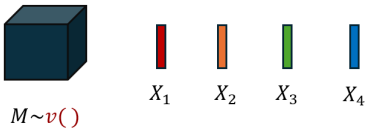
The art of organizing 2^d models

- **Other supervisors in other internships :** : "Tidy your desk"
  ☞ **Easy**
- **Marouane this summer :** : "Tidy your models"
  ☞ **Less easy**

The art of organizing 2^d models

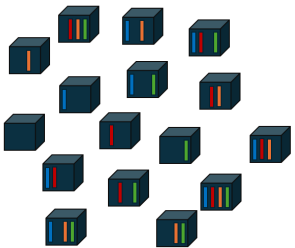
- **Other supervisors in other internships :** : "Tidy your desk"
  ☞ **Easy**
- **Marouane this summer :** : "Tidy your models"
  ☞ **Less easy**





The art of organizing 2^d models

- **Other supervisors in other internships :** : "Tidy your desk"
  ☞ **Easy**
- **Marouane this summer :** : "Tidy your models"
  ☞ **Less easy**



$M \sim v(\,)$

☞ **First question : how to organize that mess ?**

Coalition indexing : bit representation and lexicographical ordering

- **Idea**: map each coalition to an index $i \in \mathbb{N}$

1. **Remark:** It starts from 0 with the null coalition. We invert the order for better comprehension.

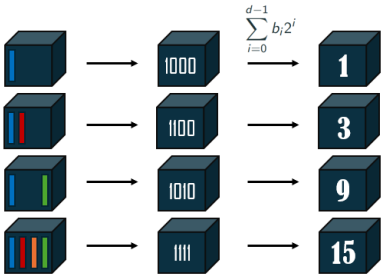
Coalition indexing : bit representation and lexicographical ordering

- **Idea**: map each coalition to an index $i \in \mathbb{N}$
  - ☞ Transform the coalition into its binary number
  - ☞ Transform the binary number into a base 2 integer[1]

1. **Remark:** It starts from 0 with the null coalition. We invert the order for better comprehension.

Coalition indexing : bit representation and lexicographical ordering

- **Idea**: map each coalition to an index $i \in \mathbb{N}$
  - ☞ Transform the coalition into its binary number
  - ☞ Transform the binary number into a base 2 integer[1]

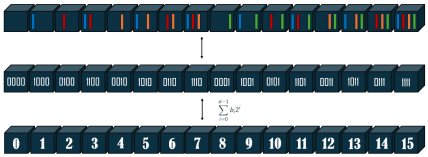


1. **Remark:** It starts from 0 with the null coalition. We invert the order for better comprehension.

Memory and parallelization issues

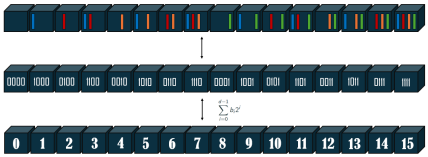
- Ok, we found a way to order all the models

  ☞ **Now, how to store it ?**



Memory and parallelization issues

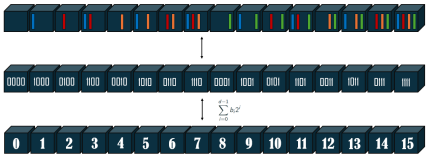
- Ok, we found a way to order all the models
  ☞ **Now, how to store it ?**



- We can't store on the memory ➢ We switch to disk storage
  ☞ **We only train the 2^d models once and for all**

Memory and parallelization issues

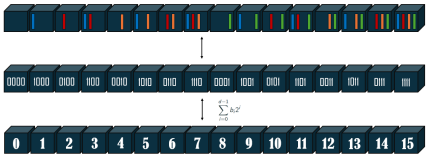
- Ok, we found a way to order all the models
  ☞ **Now, how to store it ?**



- We can't store on the memory ➢ We switch to disk storage
  ☞ **We only train the 2^d models once and for all**
- We want to parallelize so we need to batch
  ☞ **Idea : model blocks :**

Memory and parallelization issues

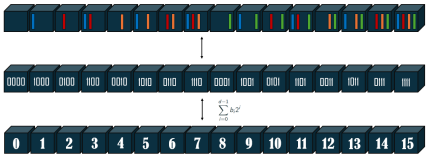
- Ok, we found a way to order all the models
  ☞ **Now, how to store it ?**



- We can't store on the memory ➢ We switch to disk storage
  ☞ **We only train the 2^d models once and for all**
- We want to parallelize so we need to batch
  ☞ **Idea : model blocks :**
  - Sequential Blocks

Memory and parallelization issues

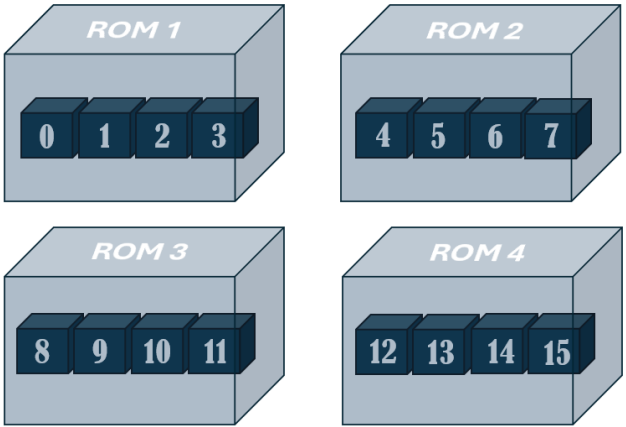
- Ok, we found a way to order all the models
  ☞ **Now, how to store it ?**



- We can't store on the memory ➢ We switch to disk storage
  ☞ **We only train the 2^d models once and for all**
- We want to parallelize so we need to batch
  ☞ **Idea : model blocks :**
  - Sequential Blocks
  - Coalitional Properties Blocks

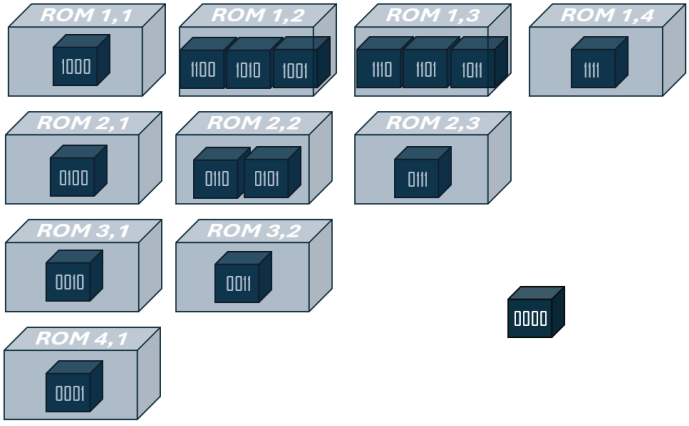
Sequential Blocks

- ☞ Natural batching by sequential blocks
- ☞ 2^b blocks for a good balance (but flexible)



Coalitional Properties Blocks

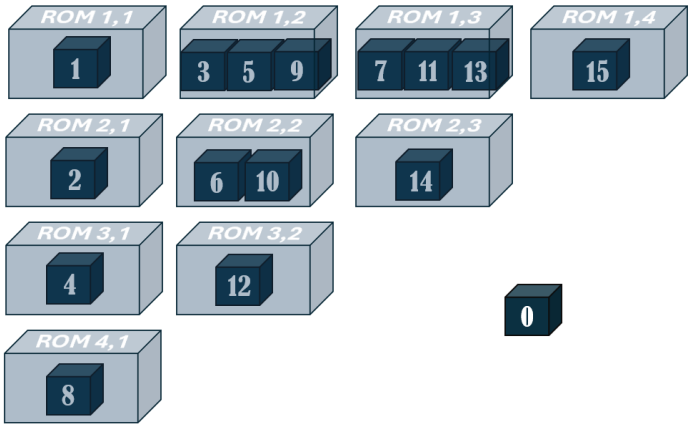
☞ Coalitions are stored depending on their first player and their number of players (coalitional properties).



Coalitional Properties Blocks

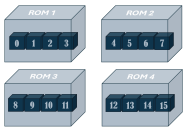
☞ Coalitions are stored depending on their first player and their number of players (coalitional properties).

☞ Inside the blocks, they are sorted by lexicographical order.



Two efficient ideas

Sequential Blocks



Advantages.

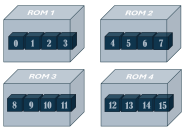
- Flexible size of blocks and memory managment
- Parallelization is easy to implement

Drawbacks.

- No particular properties inside blocks

Two efficient ideas

Sequential Blocks



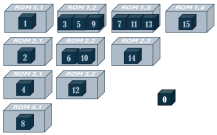
Advantages.

- Flexible size of blocks and memory managment
- Parallelization is easy to implement

Drawbacks.

- No particular properties inside blocks

Coalitional Properties Blocks



Advantages.

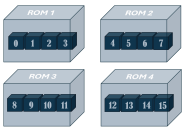
- No need to check the i first players (internal properties)

Drawbacks.

- Fixed number of blocks : $\frac{d(d+1)}{2}$
- Unbalanced sizes inside blocks
  ☞ Smart distributed parallelism

Two efficient ideas

Sequential Blocks



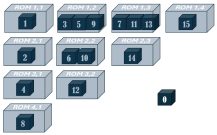
Advantages.

- Flexible size of blocks and memory managment
- Parallelization is easy to implement

Drawbacks.

- No particular properties inside blocks

Coalitional Properties Blocks



Advantages.

- No need to check the i first players (internal properties)

Drawbacks.

- Fixed number of blocks : $\frac{d(d+1)}{2}$
- Unbalanced sizes inside blocks
  ☞ Smart distributed parallelism

☞ Now, we need to adapt the formulas so they can work with parallelism

A common formula for Harsanyi, Weber and Shapley

How to parallelize that :

$$\begin{cases} \phi^H(i) = \displaystyle\sum_{A \in \mathcal{P}_D \,:\, i \in A} \lambda_i(A) \sum_{B \in \mathcal{P}_A} (-1)^{|A|-|B|} v(B) \\ \phi^W(i) = \displaystyle\sum_{\pi \in \mathcal{S}_D} p(\pi) \left[v\left(\{\pi_1, \ldots, \pi_{\pi(i)}\}\right) - v\left(\{\pi_1, \ldots, \pi_{\pi(i)-1}\}\right) \right] \end{cases}$$

A common formula for Harsanyi, Weber and Shapley

How to parallelize that :

$$\begin{cases} \phi^H(i) = \displaystyle\sum_{A \in \mathcal{P}_D \,:\, i \in A} \lambda_i(A) \sum_{B \in \mathcal{P}_A} (-1)^{|A|-|B|} v(B) \\ \phi^W(i) = \displaystyle\sum_{\pi \in \mathcal{S}_D} p(\pi) \left[v\left(\{\pi_1, \ldots, \pi_{\pi(i)}\}\right) - v\left(\{\pi_1, \ldots, \pi_{\pi(i)-1}\}\right) \right] \end{cases}$$

☞ **We can express the 3 allocations as a weighted sum of marginal contributions :**

A common formula for Harsanyi, Weber and Shapley

How to parallelize that :

$$\begin{cases} \phi^H(i) = \displaystyle\sum_{A \in \mathcal{P}_D \,:\, i \in A} \lambda_i(A) \sum_{B \in \mathcal{P}_A} (-1)^{|A|-|B|} v(B) \\ \phi^W(i) = \displaystyle\sum_{\pi \in \mathcal{S}_D} p(\pi) \left[v\left(\{\pi_1, \ldots, \pi_{\pi(i)}\}\right) - v\left(\{\pi_1, \ldots, \pi_{\pi(i)-1}\}\right) \right] \end{cases}$$

☞ **We can express the 3 allocations as a weighted sum of marginal contributions :**

$$\phi(i) = \sum_{A \in \mathcal{P}_D \,:\, i \in A} \beta(i, A) \left[v(A) - v(A \setminus i) \right], \ \forall i \in D$$

A common formula for Harsanyi, Weber and Shapley

How to parallelize that :

$$\begin{cases} \phi^H(i) = \displaystyle\sum_{A \in \mathcal{P}_D \,:\, i \in A} \lambda_i(A) \sum_{B \in \mathcal{P}_A} (-1)^{|A|-|B|} v(B) \\ \phi^W(i) = \displaystyle\sum_{\pi \in \mathcal{S}_D} p(\pi) \left[v\left(\{\pi_1, \ldots, \pi_{\pi(i)}\}\right) - v\left(\{\pi_1, \ldots, \pi_{\pi(i)-1}\}\right) \right] \end{cases}$$

☞ **We can express the 3 allocations as a weighted sum of marginal contributions :**

$$\phi(i) = \sum_{A \in \mathcal{P}_D \,:\, i \in A} \beta(i, A) \left[v(A) - v(A \setminus i) \right], \ \forall i \in D$$

☞ Shapley : $\beta(i, A) = \frac{(|A|-1)!(|D|-|A|)!}{|D|!}$

A common formula for Harsanyi, Weber and Shapley

How to parallelize that :

$$\begin{cases} \phi^H(i) = \displaystyle\sum_{A \in \mathcal{P}_D \,:\, i \in A} \lambda_i(A) \sum_{B \in \mathcal{P}_A} (-1)^{|A|-|B|} v(B) \\ \phi^W(i) = \displaystyle\sum_{\pi \in \mathcal{S}_D} p(\pi) \left[v\left(\{\pi_1, \ldots, \pi_{\pi(i)}\}\right) - v\left(\{\pi_1, \ldots, \pi_{\pi(i)-1}\}\right) \right] \end{cases}$$

☞ **We can express the 3 allocations as a weighted sum of marginal contributions :**

$$\phi(i) = \sum_{A \in \mathcal{P}_D \,:\, i \in A} \beta(i, A) \left[v(A) - v(A \setminus i) \right], \ \forall i \in D$$

☞ Shapley : $\beta(i, A) = \frac{(|A|-1)!(|D|-|A|)!}{|D|!}$

☞ Harsanyi : $\beta(i, A) = \sum_{B \supseteq A} (-1)^{|A|-|B|} \lambda_i(B)$

A common formula for Harsanyi, Weber and Shapley

How to parallelize that :

$$\begin{cases} \phi^H(i) = \sum_{A \in \mathcal{P}_D \,:\, i \in A} \lambda_i(A) \sum_{B \in \mathcal{P}_A} (-1)^{|A|-|B|} v(B) \\ \phi^W(i) = \sum_{\pi \in \mathcal{S}_D} p(\pi) \left[v\left(\{\pi_1, \ldots, \pi_{\pi(i)}\}\right) - v\left(\{\pi_1, \ldots, \pi_{\pi(i)-1}\}\right) \right] \end{cases}$$

☞ **We can express the 3 allocations as a weighted sum of marginal contributions :**

$$\phi(i) = \sum_{A \in \mathcal{P}_D \,:\, i \in A} \beta(i, A) \left[v(A) - v(A \setminus i) \right], \ \forall i \in D$$

- ☞ Shapley : $\beta(i, A) = \frac{(|A|-1)!(|D|-|A|)!}{|D|!}$
- ☞ Harsanyi : $\beta(i, A) = \sum_{B \supseteq A} (-1)^{|A|-|B|} \lambda_i(B)$
- ☞ Weber : $\beta(i, A) = \mathbb{P}_p \left\{ \{\pi_1, \ldots, \pi_{\pi(i)}\} = A \right\}$

What did we win ?

- Shapley : $\beta(i, A) = \frac{(|A|-1)!(|D|-|A|)!}{|D|!}$
  ☞ Same formulation than Shapley in 1951

What did we win ?

- Shapley : $\beta(i, A) = \frac{(|A|-1)!(|D|-|A|)!}{|D|!}$
  ☞ Same formulation than Shapley in 1951
- Harsanyi : $\beta(i, A) = \sum_{B \supseteq A} (-1)^{|A|-|B|} \lambda_i(B)$
  ☞ $d2^d$ values to compute/store, but if λ is independent of v, only d Möbius transforms are enough for the whole sample.
  ☞ We will see after a solution if λ depends on v

What did we win ?

- Shapley : $\beta(i, A) = \frac{(|A|-1)!(|D|-|A|)!}{|D|!}$
  ☞ Same formulation than Shapley in 1951
- Harsanyi : $\beta(i, A) = \sum_{B \supseteq A} (-1)^{|A|-|B|} \lambda_i(B)$
  ☞ $d2^d$ values to compute/store, but if λ is independent of v, only d Möbius transforms are enough for the whole sample.
  ☞ We will see after a solution if λ depends on v
- Weber : $\beta(i, A) = \mathbb{P}_p \left\{ \left\{ \pi_1, \ldots, \pi_{\pi(i)} \right\} = A \right\}$
  ☞ We went from $d!$ weights to $d2^d$ coefficients

What did we win ?

- Shapley : $\beta(i, A) = \frac{(|A|-1)!(|D|-|A|)!}{|D|!}$
  ☞ Same formulation than Shapley in 1951
- Harsanyi : $\beta(i, A) = \sum_{B \supseteq A} (-1)^{|A|-|B|} \lambda_i(B)$
  ☞ $d2^d$ values to compute/store, but if λ is independent of v, only d Möbius transforms are enough for the whole sample.
  ☞ We will see after a solution if λ depends on v
- Weber : $\beta(i, A) = \mathbb{P}_p \left\{ \left\{ \pi_1, \ldots, \pi_{\pi(i)} \right\} = A \right\}$
  ☞ We went from $d!$ weights to $d2^d$ coefficients
- $\phi(i) = \sum_{A \in \mathcal{P}_D \,:\, i \in A} \beta(i, A) \left[v(A) - v(A \setminus i) \right], \ \forall i \in D$
  ☞ Above all, this formulation allows us parallelize the computations

☞ Yes, but what if A and $A \setminus i$ are not in the same data blocks ?

$$\phi(i) = \sum_{A \in \mathcal{P}_D \,:\, i \in A} \beta(i, A) \left[v(A) - v(A \setminus i) \right], \ \forall i \in D$$

Split Marginal Contribution Algorithm

☞ Yes, but what if A and $A \setminus i$ are not in the same data blocks ?

$$\phi(i) = \sum_{A \in \mathcal{P}_D \,:\, i \in A} \beta(i, A) \left[v(A) - v(A \setminus i) \right], \ \forall i \in D$$

☞ **Split Marginal Contribution Algorithm**

$$\phi(i) = \sum_{A \in \mathcal{P}_D} \mathbb{1}_{i \in A} \beta(i, A) v(A) - \mathbb{1}_{i \notin A} \beta(i, A \cup i) v(A), \quad \forall i \in D$$

Split Marginal Contribution Algorithm

☞ Yes, but what if A and $A \setminus i$ are not in the same data blocks ?

$$\phi(i) = \sum_{A \in \mathcal{P}_D \,:\, i \in A} \beta(i, A) \left[v(A) - v(A \setminus i) \right], \ \forall i \in D$$

☞ **Split Marginal Contribution Algorithm**

$$\phi(i) = \sum_{A \in \mathcal{P}_D} \mathbb{1}_{i \in A} \beta(i, A) v(A) - \mathbb{1}_{i \notin A} \beta(i, A \cup i) v(A), \quad \forall i \in D$$

☞ No longer requires searching for a specific value $v(A)$ within another block.

Split Marginal Contribution Algorithm

☞ Yes, but what if A and $A \setminus i$ are not in the same data blocks ?

$$\phi(i) = \sum_{A \in \mathcal{P}_D \,:\, i \in A} \beta(i, A) \left[v(A) - v(A \setminus i) \right], \ \forall i \in D$$

☞ **Split Marginal Contribution Algorithm**

$$\phi(i) = \sum_{A \in \mathcal{P}_D} \mathbb{1}_{i \in A} \beta(i, A) v(A) - \mathbb{1}_{i \notin A} \beta(i, A \cup i) v(A), \quad \forall i \in D$$

☞ No longer requires searching for a specific value $v(A)$ within another block.

SMC Algorithm.

➢ **For** A **in** $\mathcal{P}_D$

➢ **If** $i \in A : \phi(i) \leftarrow \phi(i) + \beta(i, A) v(A)$

➢ **If** $i \notin A : \phi(i) \leftarrow \phi(i) - \beta(i, A \cup i) v(A)$

Fast Möbius Transform

- "We will see after a solution if λ depends on v"
  - ☞ We need to make $n.d$ Möbius Transforms for the whole sample
    - ☞ We consider back the initial formulation and try to find a "fast" way to compute the Möbius formula : $\mathcal{D}(v) = \sum_{B \in \mathcal{P}_A} (-1)^{|A|-|B|} v(B)$

- "We will see after a solution if λ depends on v"
  - ☞ We need to make $n.d$ Möbius Transforms for the whole sample
    - ☞ We consider back the initial formulation and try to find a "fast" way to compute the Möbius formula : $\mathcal{D}(v) = \sum_{B \in \mathcal{P}_A} (-1)^{|A|-|B|} v(B)$

The solution : Fast Möbius Transform

- "We will see after a solution if λ depends on v"
  ☞ We need to make $n.d$ Möbius Transforms for the whole sample
  ☞ We consider back the initial formulation and try to find a "fast" way to compute the Möbius formula : $\mathcal{D}(v) = \sum_{B \in \mathcal{P}_A} (-1)^{|A|-|B|} v(B)$

The solution : Fast Möbius Transform

- Algorithm "In-place" ➢ Manipulation of only one vector of length 2^d

Fast Möbius Transform

- "We will see after a solution if λ depends on v"
  ☞ We need to make $n.d$ Möbius Transforms for the whole sample
  ☞ We consider back the initial formulation and try to find a "fast" way to compute the Möbius formula : $\mathcal{D}(v) = \sum_{B \in \mathcal{P}_A} (-1)^{|A|-|B|} v(B)$

The solution : Fast Möbius Transform

- Algorithm "In-place" ➢ Manipulation of only one vector of length 2^d
- From $O(3^d)$ to $O(d2^d)$

Fast Möbius Transform

- "We will see after a solution if λ depends on v"
  ☞ We need to make $n.d$ Möbius Transforms for the whole sample
  ☞ We consider back the initial formulation and try to find a "fast" way to compute the Möbius formula : $\mathcal{D}(v) = \sum_{B \in \mathcal{P}_A} (-1)^{|A|-|B|} v(B)$

The solution : Fast Möbius Transform

- Algorithm "In-place" ➢ Manipulation of only one vector of length 2^d
- From $O(3^d)$ to $O(d2^d)$

FMT Algorithm.

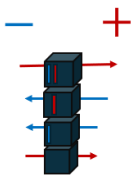
For i **in** D

➢ **For** A **in** $\mathcal{P}_D$

➢ **If** $i \in A$: $v(A) \leftarrow v(A) - v(A \setminus \{i\})$

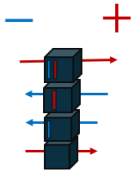
Fast Möbius Transform : Example

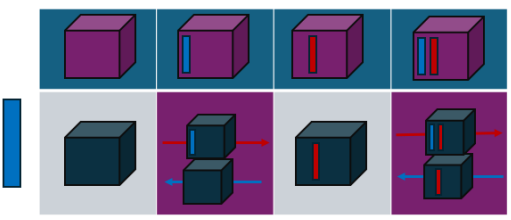
$\mathcal{D}($  $) = v(\{1,2\}) - v(\{2\}) - v(\{1\}) + v(\emptyset) =$



Fast Möbius Transform : Example

$\mathcal{D}($  $) = v(\{1,2\}) - v(\{2\}) - v(\{1\}) + v(\emptyset) =$

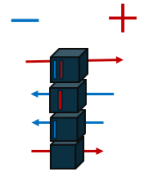


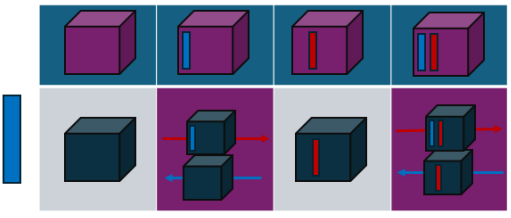


Step 1 : player

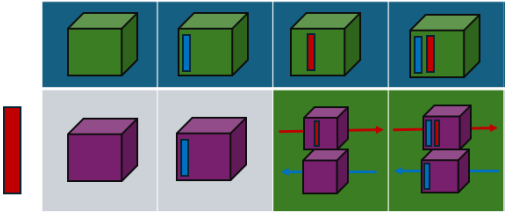
Fast Möbius Transform : Example

$\mathcal{D}($  $) = v(\{1,2\}) - v(\{2\}) - v(\{1\}) + v(\emptyset) =$





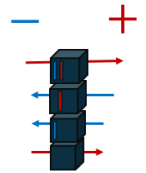
Step 1 : player

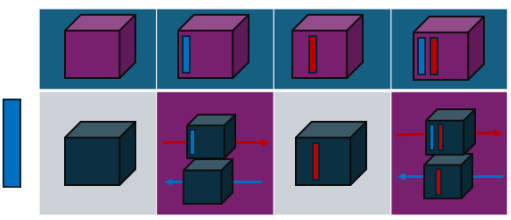


Step 2 : player

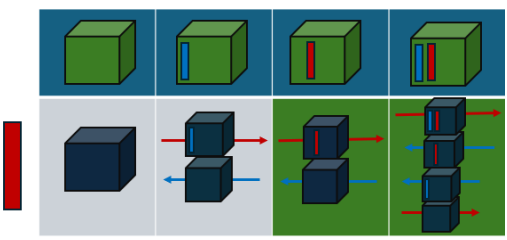
Fast Möbius Transform : Example

$\mathcal{D}($  $) = v(\{1,2\}) - v(\{2\}) - v(\{1\}) + v(\emptyset) =$





Step 1 : player



Final Möbius Transform

- We can't flat all the 2^d values

  ☞ **How to adapt this algorithm to our blocks structure ?**

Distributed Fast Möbius Transform

- We can't flat all the 2^d values
  ☞ **How to adapt this algorithm to our blocks structure ?**

The Solution : Distributed Fast Möbius Transform

- Adapted to the Sequential Blocks structure 
- Allow parallelism
- Memory-safe designed
- The number of blocks B must be a power of 2 ($B = 2^b$)

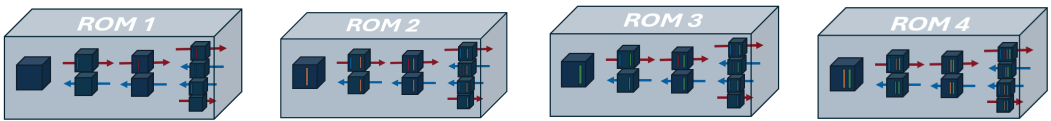
Distributed Fast Möbius Transform

Distributed FMT Algorithm.

Phase 1 : Intra-Blocks

For *bl* **in Blocks** (parallelism)

➢ **DO (scalar)** $\mathbb{R}$**-FMT inside** *bl* (considering lexicographical order inside the block)



Distributed Fast Möbius Transform

Distributed FMT Algorithm.

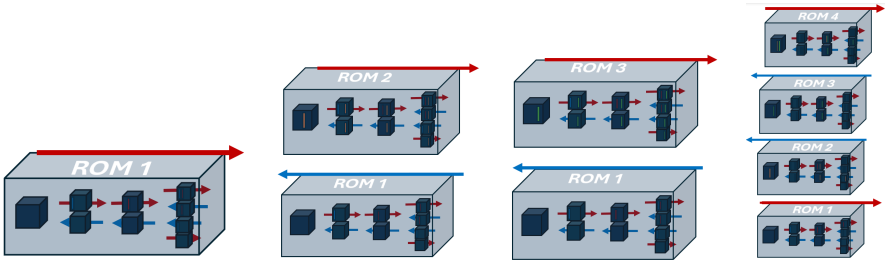
Phase 1 : Intra-Blocks

For *bl* **in Blocks** (parallelism)

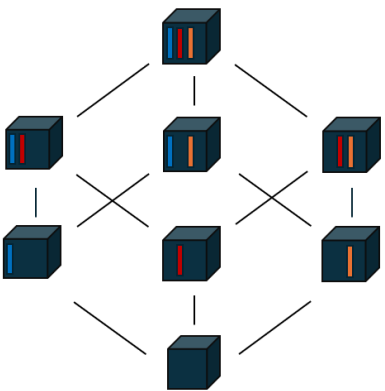
➢ **DO (scalar)** $\mathbb{R}$**-FMT inside** *bl* (considering lexicographical order inside the block)

Phase 2 : Inter-Blocks

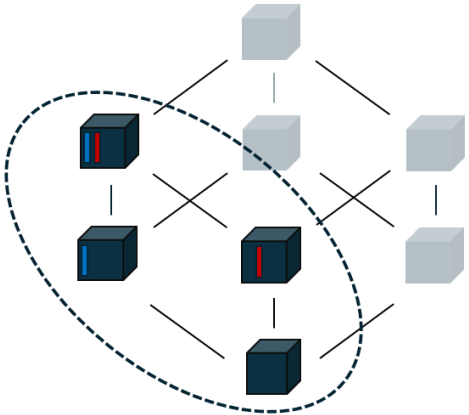
➢ **DO (vectorial)** $\mathbb{R}^{2^{d-b}}$**-FMT along Blocks** (considering lexicographical order of the blocks)



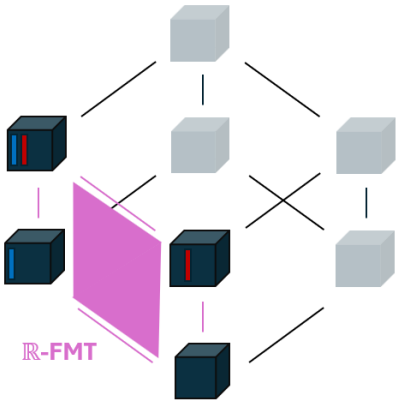
☞ **Phase 1 :** Möbius Transforms on sub-lattices



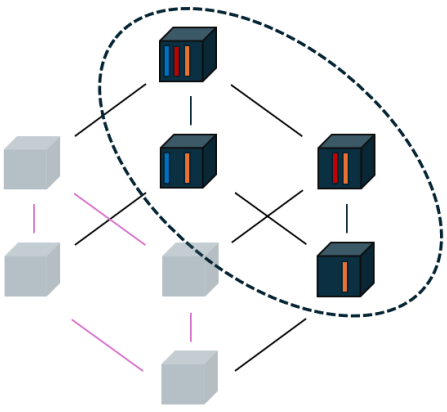
☞ **Phase 1 :** Möbius Transforms on sub-lattices



☞ **Phase 1 :** Möbius Transforms on sub-lattices

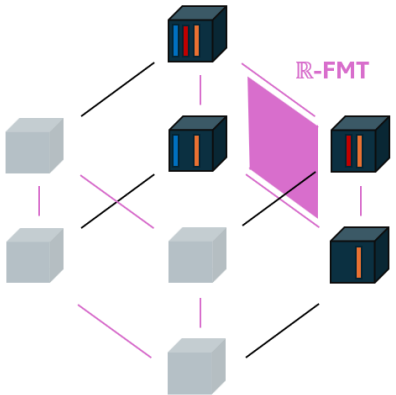


☞ **Phase 1 :** Möbius Transforms on sub-lattices



Distributed Fast Möbius Transform as an inversion on the boolean lattice

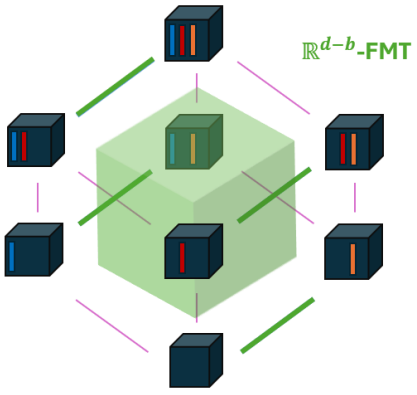
☞ **Phase 1 :** Möbius Transforms on sub-lattices



Distributed Fast Möbius Transform as an inversion on the boolean lattice

☞ **Phase 1 :** Möbius Transforms on sub-lattices

☞ **Phase 2 :** Möbius Transform



Distributed Fast Möbius Transform as a mixed Kronecker matrix-vector product

☞ Mazo and Tournier 2025 :

$$\mathcal{D}(v) = \otimes^d M.v \text{ , with : } M = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}, \ v \in \mathbb{R}^{2^d}$$

Distributed Fast Möbius Transform as a mixed Kronecker matrix-vector product

☞ Mazo and Tournier 2025 :

$$\mathcal{D}(v) = \otimes^d M.v \text{ , with : } M = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}, \ v \in \mathbb{R}^{2^d}$$

☞ Mixed Kronecker matrix-vector product :

$$(B \otimes A)\,\mathrm{vec}(V) = \mathrm{vec}(A\,V\,B^\top) \text{ with : } V \in \mathbb{R}^{2^b \times 2^{d-b}}$$

$$\mathcal{D}(v) = \otimes^d M.v = \mathrm{vec}\Big(\underbrace{(\otimes^b M)}_{\text{Phase 1, intra-blocks}} \quad V \quad \underbrace{(\otimes^{d-b} M)^\top}_{\text{Phase 2, inter-blocks}} \Big)$$

What did you learn today ?

- **Two data structures**
  ☞ Sequential Blocks, Coalional properties Blocks
- **A new common formula** for Harsanyi, Weber and Shapley
  ☞ Allows independent, parallelized computations
- **New algorithms** adapted to the Sequential Blocks structure
  ☞ Split Marginal Contribution, Fast Möbius Transform, Distributed Fast Möbius Transform

What's next ?

- ☞ Finish and publish ***interpretability***
- ☞ **Theorethically :** Find **closed formulas** for β for specific choices of weight systems λ and permutation distributions p
- ☞ **Computationally :** Find a way to **"compress"** each model (i.e. store the minimum requirement)

References i

Harsanyi, J. C. 1963. "A Simplified Bargaining Model for the n-Person Cooperative Game." Publisher: [Economics Department of the University of Pennsylvania, Wiley, Institute of Social and Economic Research, Osaka University], *International Economic Review* 4 (2): 194–220. ISSN: 0020-6598. https://doi.org/10.2307/2525487. https://www.jstor.org/stable/2525487.

I., M., Arthur Charpentier, and Agathe Fernandes Machado. 2025. "Beyond Shapley Values: Cooperative Games for the Interpretation of Machine Learning Models." In *International Joint Conference on Artificial Intelligence (IJCAI) - Workshop on Explainable Artificial Intelligence (XAI)*. Montréal, Québec, Canada: Hendrik Baier and Tobias Huber and Mor Vered and Sarath Sreedharan and Katharina Weitz and Stylianos Loukas Vasileiou, August. https://hal.science/hal-05106257.

Kung, J. P. S., G. C. Rota, and C. Hung Yan. 2012. *Combinatorics: the Rota way*. OCLC: 1226672593. New York: Cambridge University Press. ISBN: 978-0-511-80389-5.

Lindeman, R. H., P. F. Merenda, and R. Z. Gold. 1980. *Introduction to Bivariate and Multivariate Analysis* [in English]. Scott, Foresman. ISBN: 978-0-673-15099-8. https://books.google.cz/books?id=-hfvAAAAMAAJ.

Mazo, Gildas, and Laurent Tournier. 2025. "An Inference Method for Global Sensitivity Analysis." *Technometrics* 67 (2): 270–282. https://doi.org/10.1080/00401706.2024.2431113.

Rota, G. C. 1964. "On the foundations of combinatorial theory I. Theory of Möbius Functions." *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete* 2 (4): 340–368. ISSN: 1432-2064. https://doi.org/10.1007/BF00531932.

Weber, R. J. 1988. "Probabilistic values for games." Chap. 7 in *The Shapley value: essays in honor of Lloyd S. Shapley*, edited by A. E. Roth, 101–120. New York, NY: Cambridge University Press.

Thank you for your attention!

Any questions?

MAROUANEILIDRISSI.COM

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC).
Nous remercions le Conseil de Recherches en Sciences Naturelles et en Génie du Canada (CRSNG) de son soutien.