

# Accélération des forêts aléatoires sur des environnements à mémoire contrainte via la réorganisation des données

Camélia Slimani<sup>1</sup>, Chun-Feng Wu<sup>2</sup>, Yuan-Hao Chang<sup>2</sup>, Stéphane Rubini<sup>1</sup>, Jalil Boukhobza<sup>3</sup>

<sup>1</sup>: Univ. Bretagne Occidentale, Lab-STICC (UMR6285), France

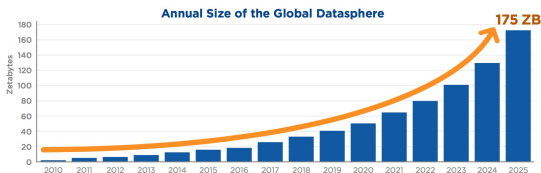
<sup>2</sup>: Institute of Information Science, Academia Sinica, Taiwan

<sup>3</sup>: ENSTA Bretagne, Lab-STICC(UMR6285), France

*Conférence francophone d'informatique en Parallélisme, Architecture et Système (COMPAS 2021)*

- 1 Contexte
- 2 Motivation
- 3 Solution proposée
  - Hypothèse
  - Vérification de l'hypothèse
  - Solution proposée
- 4 Évaluation
- 5 Conclusion et perspectives

# Contexte: Augmentation Exponentielle du volume de données



- 50% des données sont produites sur les plateformes embarquées [1];
- La tendance actuelle consiste à traiter les données collectées directement sur les ces plateformes [2][3] pour répondre aux :
  - Contraintes de sécurité;
  - Coût de communication.
- Néanmoins, ces plateformes sont:
  - Limitées en terme d'espace de travail ;
  - Contraintes en énergie.

# Forêts aléatoires (RF)

**Algorithm** Méthode de création d'un arbre de décision [4]

- 1: Création d'un bootstrap
- 2: **while** il existe un nœud impure *n* **do**
- 3:   Création aléatoire d'un sous-ensemble de propriétés *F*
- 4:   **for**  $f = 1$  to  $|F| - 1$  **do**
- 5:     Tentative de division du nœud *n* en deux nœuds enfants selon la propriété *f*
- 6:   **end for**
- 7:   Choix de la meilleure propriété  $f^*$  et création effective des nœuds enfants
- 8: **end while**

(1) Création du bootstrap ○ A C E C H A B F

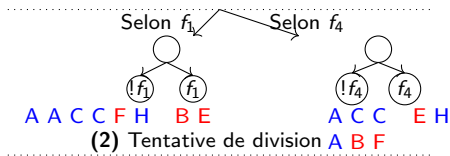
Bloc E/S	Label	$f_1$	$f_2$	$f_3$	$f_4$	Classe
(1)	A	0	0	0	0	0
	B	1	0	1	0	1
(2)	C	0	1	0	1	0
	D	0	0	0	0	0
(3)	E	1	1	1	1	1
	F	0	1	1	0	1
(4)	G	1	0	0	1	0
	H	0	1	0	1	0

# Forêts aléatoires (RF)

**Algorithm** Méthode de création d'un arbre de décision [4]

- 1: Création d'un bootstrap
- 2: **while** il existe un nœud impure *n* **do**
- 3:   Création aléatoire d'un sous-ensemble de propriétés *F*
- 4:   **for**  $f = 1$  to  $|F| - 1$  **do**
- 5:     Tentative de division du nœud *n* en deux nœuds enfants selon la propriété *f*
- 6:   **end for**
- 7:   Choix de la meilleure propriété  $f^*$  et création effective des nœuds enfants
- 8: **end while**

(1) Création du bootstrap ○ A C E C H A B F



Bloc E/S	Label	$f_1$	$f_2$	$f_3$	$f_4$	Classe
(1)	A	0	0	0	0	0
	B	1	0	1	0	1
(2)	C	0	1	0	1	0
	D	0	0	0	0	0
(3)	E	1	1	1	1	1
	F	0	1	1	0	1
(4)	G	1	0	0	1	0
	H	0	1	0	1	0

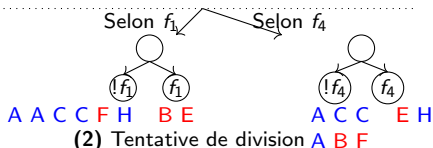
# Forêts aléatoires (RF)

**Algorithm** Méthode de création d'un arbre de décision [4]

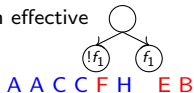
- 1: Création d'un bootstrap
- 2: **while** il existe un nœud impure *n* **do**
- 3:   Création aléatoire d'un sous-ensemble de propriétés *F*
- 4:   **for** *f* = 1 to  $|F| - 1$  **do**
- 5:     Tentative de division du nœud *n* en deux nœuds enfants selon la propriété *f*
- 6:   **end for**
- 7:   Choix de la meilleure propriété *f\** et création effective des nœuds enfants
- 8: **end while**

Bloc E/S	Label	$f_1$	$f_2$	$f_3$	$f_4$	Classe
(1)	A	0	0	0	0	0
	B	1	0	1	0	1
(2)	C	0	1	0	1	0
	D	0	0	0	0	0
(3)	E	1	1	1	1	1
	F	0	1	1	0	1
(4)	G	1	0	0	1	0
	H	0	0	1	0	1

(1) Création du bootstrap ○ A C E C H A B F



(3) Division effective



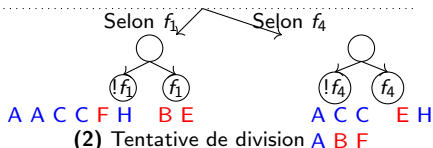
# Forêts aléatoires (RF)

**Algorithm** Méthode de création d'un arbre de décision [4]

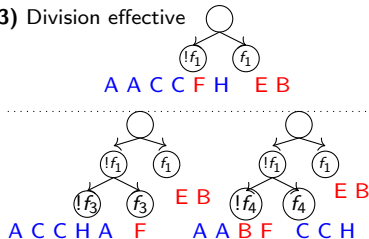
- 1: Création d'un bootstrap
- 2: **while** il existe un nœud impure *n* **do**
- 3: Création aléatoire d'un sous-ensemble de propriétés *F*
- 4: **for**  $f = 1$  to  $|F| - 1$  **do**
- 5: Tentative de division du nœud *n* en deux nœuds enfants selon la propriété *f*
- 6: **end for**
- 7: Choix de la meilleure propriété  $f^*$  et création effective des nœuds enfants
- 8: **end while**

Bloc E/S	Label	$f_1$	$f_2$	$f_3$	$f_4$	Classe
(1)	A	0	0	0	0	0
	B	1	0	1	0	1
(2)	C	0	1	0	1	0
	D	0	0	0	0	0
(3)	E	1	1	1	1	1
	F	0	1	1	0	1
(4)	G	1	0	0	1	0
	H	0	1	0	1	0

(1) Création du bootstrap ○ A C E C H A B F



(3) Division effective



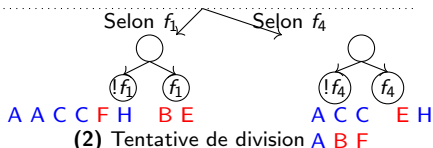
# Forêts aléatoires (RF)

**Algorithm** Méthode de création d'un arbre de décision [4]

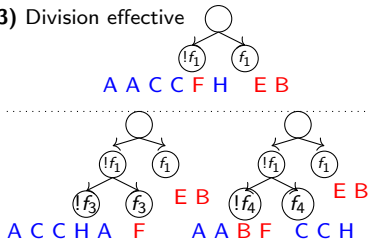
- 1: Création d'un bootstrap
- 2: **while** il existe un nœud impure *n* **do**
- 3: Création aléatoire d'un sous-ensemble de propriétés *F*
- 4: **for** *f* = 1 to  $|F| - 1$  **do**
- 5: Tentative de division du nœud *n* en deux nœuds enfants selon la propriété *f*
- 6: **end for**
- 7: Choix de la meilleure propriété *f\** et création effective des nœuds enfants
- 8: **end while**

Bloc E/S	Label	$f_1$	$f_2$	$f_3$	$f_4$	Classe
(1)	A	0	0	0	0	0
	B	1	0	1	0	1
(2)	C	0	1	0	1	0
	D	0	0	0	0	0
(3)	E	1	1	1	1	1
	F	0	1	1	0	1
(4)	G	1	0	0	1	0
	H	0	1	0	1	0

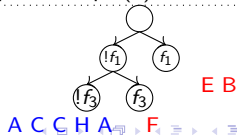
(1) Création du bootstrap ○ A C E C H A B F



(3) Division effective



(4) Répéter l'étape (2) sur le nœud gauche



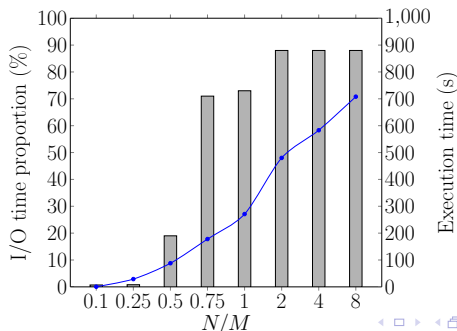


- 1 Contexte
- 2 Motivation
- 3 Solution proposée
  - Hypothèse
  - Vérification de l'hypothèse
  - Solution proposée
- 4 Évaluation
- 5 Conclusion et perspectives

# Motivation: Mesure de la proportion de temps passé à faire des E/S

## Méthode de mesure

- Restriction de l'espace mémoire disponible ( $M$ ) par rapport au volume de données à traiter ( $N$ );
- Mesure du temps de construction d'un arbre de décision et la proportion de temps d'E/S.



# Motivation: Comportement de l'algorithme d'un point de vue E/S dans le cas d'un environnement contraint en espace de travail

On suppose un espace de travail qui peut contenir 4 éléments et un bloc d'E/S de 2 éléments.

Bloc E/S	Label	$f_1$	$f_2$	$f_3$	$f_4$	Classe
(1)	A	0	0	0	0	0
	B	1	0	1	0	1
(2)	C	0	1	0	1	0
	D	0	0	0	0	0
(3)	E	1	1	1	1	1
	F	0	1	1	0	1
(4)	G	1	0	0	1	0
	H	0	1	0	1	0

Donnée utile

Donnée inutile

# Motivation: Comportement de l'algorithme d'un point de vue E/S dans le cas d'un environnement contraint en espace de travail

On suppose un espace de travail qui peut contenir 4 éléments et un bloc d'E/S de 2 éléments.

Nœud	Éléments	Blocs accédés	Pourcentage de données utilisées par bloc
$N_0$	{A, A, B, C, C, E, F, H}	(1), (2), (3), (4)	100%, 50%, 100%, 50%

Bloc E/S	Label	$f_1$	$f_2$	$f_3$	$f_4$	Classe
(1)	A	0	0	0	0	0
	B	1	0	1	0	1
(2)	C	0	1	0	1	0
	D	0	0	0	0	0
(3)	E	1	1	1	1	1
	F	0	1	1	0	1
(4)	G	1	0	0	1	0
	H	0	1	0	1	0

Donnée utile

Donnée inutile

# Motivation: Comportement de l'algorithme d'un point de vue E/S dans le cas d'un environnement contraint en espace de travail

On suppose un espace de travail qui peut contenir 4 éléments et un bloc d'E/S de 2 éléments.

Nœud	Éléments	Blocs accédés	Pourcentage de données utilisées par bloc
$N_0$	{A, A, B, C, C, E, F, H}	(1), (2), (3), (4)	100%, 50%, 100%, 50%
$N_1$	{A, A, C, C, F, H}	(1), (2), (3), (4)	50%, 50%, 50%, 50%

Bloc E/S	Label	$f_1$	$f_2$	$f_3$	$f_4$	Classe
(1)	A	0	0	0	0	0
	B	1	0	1	0	1
(2)	C	0	1	0	1	0
	D	0	0	0	0	0
(3)	E	1	1	1	1	1
	F	0	1	1	0	1
(4)	G	1	0	0	1	0
	H	0	1	0	1	0

Donnée utile

Donnée inutile

# Motivation: Comportement de l'algorithme d'un point de vue E/S dans le cas d'un environnement contraint en espace de travail

On suppose un espace de travail qui peut contenir 4 éléments et un bloc d'E/S de 2 éléments.

Nœud	Éléments	Blocs accédés	Pourcentage de données utilisées par bloc
$N_0$	{A, A, B, C, C, E, F, H}	(1), (2), (3), (4)	100%, 50%, 100%, 50%
$N_1$	{A, A, C, C, F, H}	(1), (2), (3), (4)	50%, 50%, 50%, 50%
$N_2$	{B, E}	(1), (3)	50%, 50%

Bloc E/S	Label	$f_1$	$f_2$	$f_3$	$f_4$	Classe
(1)	A	0	0	0	0	0
	B	1	0	1	0	1
(2)	C	0	1	0	1	0
	D	0	0	0	0	0
(3)	E	1	1	1	1	1
	F	0	1	1	0	1
(4)	G	1	0	0	1	0
	H	0	1	0	1	0

Donnée utile

Donnée inutile

# Motivation: Comportement de l'algorithme d'un point de vue E/S dans le cas d'un environnement contraint en espace de travail

On suppose un espace de travail qui peut contenir 4 éléments et un bloc d'E/S de 2 éléments.

Nœud	Éléments	Blocs accédés	Pourcentage de données utilisées par bloc
$N_0$	{A, A, B, C, C, E, F, H}	(1), (2), (3), (4)	100%, 50%, 100%, 50%
$N_1$	{A, A, C, C, F, H}	(1), (2), (3), (4)	50%, 50%, 50%, 50%
$N_2$	{B, E}	(1), (3)	50%, 50%
$N_3$	{A, A, C, C, H}	(1), (2), (4)	50%, 50%, 50%

Bloc E/S	Label	$f_1$	$f_2$	$f_3$	$f_4$	Classe
(1)	A	0	0	0	0	0
	B	1	0	1	0	1
(2)	C	0	1	0	1	0
	D	0	0	0	0	0
(3)	E	1	1	1	1	1
	F	0	1	1	0	1
(4)	G	1	0	0	1	0
	H	0	1	0	1	0

Donnée utile

Donnée inutile

# Motivation: Comportement de l'algorithme d'un point de vue E/S dans le cas d'un environnement contraint en espace de travail

On suppose un espace de travail qui peut contenir 4 éléments et un bloc d'E/S de 2 éléments.

Nœud	Éléments	Blocs accédés	Pourcentage de données utilisées par bloc
$N_0$	{A, A, B, C, C, E, F, H}	(1), (2), (3), (4)	100%, 50%, 100%, 50%
$N_1$	{A, A, C, C, F, H}	(1), (2), (3), (4)	50%, 50%, 50%, 50%
$N_2$	{B, E}	(1), (3)	50%, 50%
$N_3$	{A, A, C, C, H}	(1), (2), (4)	50%, 50%, 50%
$N_4$	{F}	(3)	50%

Bloc E/S	Label	$f_1$	$f_2$	$f_3$	$f_4$	Classe
(1)	A	0	0	0	0	0
	B	1	0	1	0	1
(2)	C	0	1	0	1	0
	D	0	0	0	0	0
(3)	E	1	1	1	1	1
	F	0	1	1	0	1
(4)	G	1	0	0	1	0
	H	0	1	0	1	0

Donnée utile

Donnée inutile



- 1 Contexte
- 2 Motivation
- 3 **Solution proposée**
  - Hypothèse
  - Vérification de l'hypothèse
  - Solution proposée
- 4 Évaluation
- 5 Conclusion et perspectives

# Hypothèse de similarité entre les arbres de décision

## Hypothèse

Pour une paire d'éléments donnée du bootstrap:

## Hypothèse

Pour une paire d'éléments donnée du bootstrap:

- Si les deux éléments sont classés dans un même nœud au niveau d'un arbre de décision  $T_1$ , alors la probabilité qu'ils soient dans un même nœud au niveau d'un autre arbre  $T_2$  est grande.

# Hypothèse de similarité entre les arbres de décision

## Hypothèse

Pour une paire d'éléments donnée du bootstrap:

- Si les deux éléments sont classés dans un même nœud au niveau d'un arbre de décision  $T_1$ , alors la probabilité qu'ils soient dans un même nœud au niveau d'un autre arbre  $T_2$  est grande.

## Ce qui nous intéresse dans cette propriété

# Hypothèse de similarité entre les arbres de décision

## Hypothèse

Pour une paire d'éléments donnée du bootstrap:

- Si les deux éléments sont classés dans un même nœud au niveau d'un arbre de décision  $T_1$ , alors la probabilité qu'ils soient dans un même nœud au niveau d'un autre arbre  $T_2$  est grande.

## Ce qui nous intéresse dans cette propriété

- Si deux éléments sont dans un même nœud, alors ils doivent être accédés dans une même fenêtre temporelle.

# Hypothèse de similarité entre les arbres de décision

## Hypothèse

Pour une paire d'éléments donnée du bootstrap:

- Si les deux éléments sont classés dans un même nœud au niveau d'un arbre de décision  $T_1$ , alors la probabilité qu'ils soient dans un même nœud au niveau d'un autre arbre  $T_2$  est grande.

## Ce qui nous intéresse dans cette propriété

- Si deux éléments sont dans un même nœud, alors ils doivent être accédés dans une même fenêtre temporelle.
- S'ils sont dans deux blocs de stockage séparés, les deux blocs doivent être chargés en mémoire.

# Hypothèse de similarité entre les arbres de décision

## Hypothèse

Pour une paire d'éléments donnée du bootstrap:

- Si les deux éléments sont classés dans un même nœud au niveau d'un arbre de décision  $T_1$ , alors la probabilité qu'ils soient dans un même nœud au niveau d'un autre arbre  $T_2$  est grande.

## Ce qui nous intéresse dans cette propriété

- Si deux éléments sont dans un même nœud, alors ils doivent être accédés dans une même fenêtre temporelle.
- S'ils sont dans deux blocs de stockage séparés, les deux blocs doivent être chargés en mémoire.

Si l'hypothèse de similarité est vraie, alors nous pouvons déduire à partir d'un arbre les éléments susceptibles d'être accédés pour un nœud

# Hypothèse de similarité entre les arbres de décision

## Hypothèse

Pour une paire d'éléments donnée du bootstrap:

- Si les deux éléments sont classés dans un même nœud au niveau d'un arbre de décision  $T_1$ , alors la probabilité qu'ils soient dans un même nœud au niveau d'un autre arbre  $T_2$  est grande.

## Ce qui nous intéresse dans cette propriété

- Si deux éléments sont dans un même nœud, alors ils doivent être accédés dans une même fenêtre temporelle.
- S'ils sont dans deux blocs de stockage séparés, les deux blocs doivent être chargés en mémoire.

Si l'hypothèse de similarité est vraie, alors nous pouvons déduire à partir d'un arbre les éléments susceptibles d'être accédés pour un nœud  $\Rightarrow$  possibilité de réorganiser le data-set de manière à réduire les E/S pour les arbres suivants.



## Formalisation

- Soient  $T_1$  et  $T_2$  deux arbres de décision d'une même forêt. L'arbre  $T_1$  (resp.  $T_2$ ) donne un partitionnement (clustering)  $P_1$  (resp.  $P_2$ ) du data-set tel que le nombre de partitions est égal au nombre de nœuds feuilles dans l'arbre.
- Comparer la similarité des arbres revient à comparer les partitionnements  $P_1$  et  $P_2$ .
- Métriques de comparaison de partitionnements : Rand Index, Adjusted Rand Index, etc [5].

## Méthodologie de comparaison

- Métrique choisie : Adjusted Rand Index (ARI)
- Pour chacun des data-sets que l'on test, on construit deux arbres de décision et on mesure l'ARI entre les deux partitionnements obtenus.
- On compare les ARI obtenus aux ARI des méthodes de clustering fiables et utilisées (K-means, GMM, Birch) dans l'état de l'art[6] (Les valeurs varient entre 0.07 et 0.8).

## Résultats :

Data-set	Wearable	Adult	Coverttype	Ecoli	Wine	Heart Failure
ARI	0.27	0.26	0.12	0.37	0.42	0.33

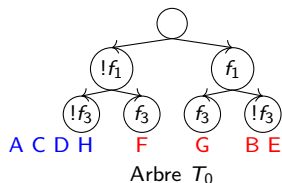
- 1 **Construction d'un premier arbre de décision  $T_0$**  : cet arbre est construit sur tout le data-set et non sur un bootstrap afin de réorganiser toutes les données;

- 1 **Construction d'un premier arbre de décision  $T_0$**  : cet arbre est construit sur tout le data-set et non sur un bootstrap afin de réorganiser toutes les données;
- 2 **Réécriture du data-set** : le data-set est réécrit de manière à ce que les données qui sont dans les mêmes nœuds feuilles sont écrits sur les mêmes blocs;

- 1 **Construction d'un premier arbre de décision  $T_0$**  : cet arbre est construit sur tout le data-set et non sur un bootstrap afin de réorganiser toutes les données;
- 2 **Réécriture du data-set** : le data-set est réécrit de manière à ce que les données qui sont dans les mêmes nœuds feuilles sont écrits sur les mêmes blocs;
- 3 **Construction des arbres de la forêt** : les arbres sont construits en utilisant le data-set réorganisé.

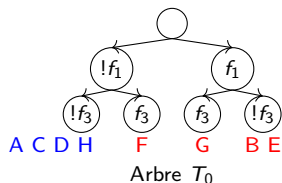
# Solution proposée

- 1 Construction d'un premier arbre de décision  $T_0$
- 2 Réécriture du data-set
- 3 Construction des arbres de la forêt



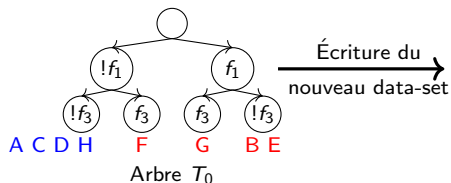
# Solution proposée

- 1 Construction d'un premier arbre de décision  $T_0$
- 2 Réécriture du data-set
- 3 Construction des arbres de la forêt



# Solution proposée

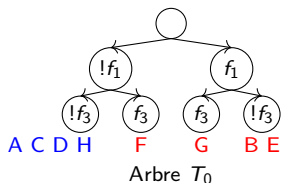
- 1 Construction d'un premier arbre de décision  $T_0$
- 2 Réécriture du data-set
- 3 Construction des arbres de la forêt



Bloc E/S	Label	$f_1$	$f_2$	$f_3$	$f_4$	Classe
(1)	A	0	0	0	0	0
	C	0	1	0	1	0
(2)	D	0	0	0	0	0
	H	0	1	0	1	0
(3)	F	0	1	1	0	1
	G	1	0	0	1	0
(4)	B	1	0	1	0	1
	E	1	1	1	1	1



- 1 Construction d'un premier arbre de décision  $T_0$
- 2 Réécriture du data-set
- 3 Construction des arbres de la forêt

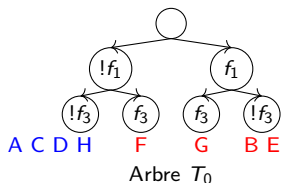


Bloc E/S	Label	$f_1$	$f_2$	$f_3$	$f_4$	Classe
(1)	A	0	0	0	0	0
	C	0	1	0	1	0
(2)	D	0	0	0	0	0
	H	0	1	0	1	0
(3)	F	0	1	1	0	1
	G	1	0	0	1	0
(4)	B	1	0	1	0	1
	E	1	1	1	1	1

Nœud	Éléments	Blocs accédés	Pourcentage de données utilisées par bloc
$N_0$	{A, A, B, C, C, E, F, H}	(1), (2), (3), (4)	100%, 50%, 50%, 100%

# Solution proposée

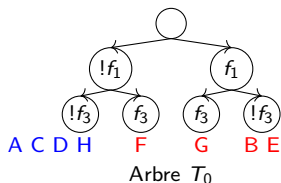
- 1 Construction d'un premier arbre de décision  $T_0$
- 2 Réécriture du data-set
- 3 Construction des arbres de la forêt



Bloc E/S	Label	$f_1$	$f_2$	$f_3$	$f_4$	Classe
(1)	A	0	0	0	0	0
	C	0	1	0	1	0
(2)	D	0	0	0	0	0
	H	0	1	0	1	0
(3)	F	0	1	1	0	1
	G	1	0	0	1	0
(4)	B	1	0	1	0	1
	E	1	1	1	1	1

Noeud	Éléments	Blocs accédés	Pourcentage de données utilisées par bloc
$N_0$	{A, A, B, C, C, E, F, H}	(1), (2), (3), (4)	100%, 50%, 50%, 100%
$N_1$	{A, A, C, C, F, H}	(1), (2), (3)	100%, 50%, 50%

- 1 Construction d'un premier arbre de décision  $T_0$
- 2 Réécriture du data-set
- 3 Construction des arbres de la forêt

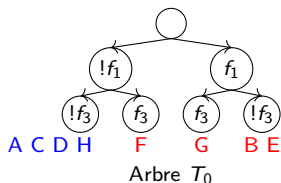


Bloc E/S	Label	$f_1$	$f_2$	$f_3$	$f_4$	Classe
(1)	A	0	0	0	0	0
	C	0	1	0	1	0
(2)	D	0	0	0	0	0
	H	0	1	0	1	0
(3)	F	0	1	1	0	1
	G	1	0	0	1	0
(4)	B	1	0	1	0	1
	E	1	1	1	1	1

Noeud	Éléments	Blocs accédés	Pourcentage de données utilisées par bloc
$N_0$	{A, A, B, C, C, E, F, H}	(1), (2), (3), (4)	100%, 50%, 50%, 100%
$N_1$	{A, A, C, C, F, H}	(1), (2), (3)	100%, 50%, 50%
$N_2$	{B, E}	(4)	100%

# Solution proposée

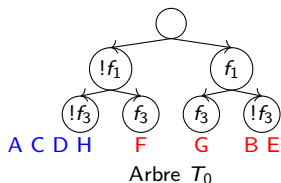
- 1 Construction d'un premier arbre de décision  $T_0$
- 2 Réécriture du data-set
- 3 Construction des arbres de la forêt



Bloc E/S	Label	$f_1$	$f_2$	$f_3$	$f_4$	Classe
(1)	A	0	0	0	0	0
	C	0	1	0	1	0
(2)	D	0	0	0	0	0
	H	0	1	0	1	0
(3)	F	0	1	1	0	1
	G	1	0	0	1	0
(4)	B	1	0	1	0	1
	E	1	1	1	1	1

Noeud	Éléments	Blocs accédés	Pourcentage de données utilisées par bloc
$N_0$	{A, A, B, C, C, E, F, H}	(1), (2), (3), (4)	100%, 50%, 50%, 100%
$N_1$	{A, A, C, C, F, H}	(1), (2), (3)	100%, 50%, 50%
$N_2$	{B, E}	(4)	100%
$N_3$	{A, A, C, C, H}	(1), (2)	100%, 50%

- 1 Construction d'un premier arbre de décision  $T_0$
- 2 Réécriture du data-set
- 3 Construction des arbres de la forêt



Bloc E/S	Label	$f_1$	$f_2$	$f_3$	$f_4$	Classe
(1)	A	0	0	0	0	0
	C	0	1	0	1	0
(2)	D	0	0	0	0	0
	H	0	1	0	1	0
(3)	F	0	1	1	0	1
	G	1	0	0	1	0
(4)	B	1	0	1	0	1
	E	1	1	1	1	1

Noeud	Éléments	Blocs accédés	Pourcentage de données utilisées par bloc
$N_0$	{A, A, B, C, C, E, F, H}	(1), (2), (3), (4)	100%, 50%, 50%, 100%
$N_1$	{A, A, C, C, F, H}	(1), (2), (3)	100%, 50%, 50%
$N_2$	{B, E}	(4)	100%
$N_3$	{A, A, C, C, H}	(1), (2)	100%, 50%
$N_4$	{F}	(3)	50%

# Agenda

- 1 Contexte
- 2 Motivation
- 3 Solution proposée
  - Hypothèse
  - Vérification de l'hypothèse
  - Solution proposée
- 4 Évaluation
- 5 Conclusion et perspectives

## Configuration mémoire

- $N$ : Volume de données à traiter;
- $M$ : Espace mémoire disponible.

$$N/M = \{1, 2, 4, 8\}.$$

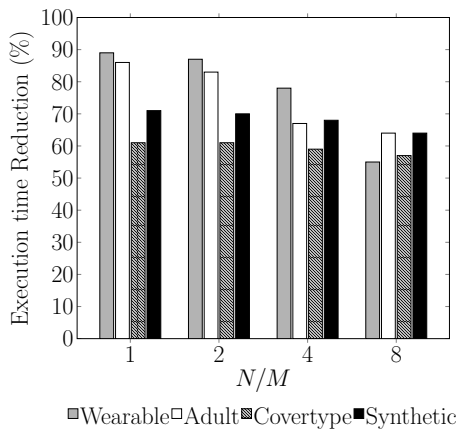
## Datasets utilisés

Dataset	Nombre de propriétés	Nombre d'éléments	Taille du dataset (Mo)
Coverttype	54	581012	239.36
Wearable	8	75128	4.58
Adult	14	48842	5.21
Synthetic	64	100000	48.82

## Méthodes de référence

- Ranger [7]

# Temps d'exécution de la méthode proposée par rapport à ranger





# Agenda





- 1 Contexte
- 2 Motivation
- 3 Solution proposée
  - Hypothèse
  - Vérification de l'hypothèse
  - Solution proposée
- 4 Évaluation
- 5 Conclusion et perspectives

## Conclusion

- Lors de la construction d'un arbre de décision dans un environnement contraint en mémoire, la proportion de temps d'E/S est importante;
- La méthode proposée vise à augmenter la localité spatiale des blocs de données pour réduire le volume des E/S lors de la construction;
- Elle repose sur la propriété de similarité entre les arbres de décision entre les arbres d'une même forêt, pour déduire les données susceptibles d'être accédés au même temps et réorganiser le data-set
- Selon nos expériences, cette méthode réduit le temps de construction d'une forêt de 55-89%.

## Perspective

- Étudier l'applicabilité de la méthode de réorganisation des data-sets pour les méthode d'apprentissage ensemblistes.

-  J. Gantz D. Reinsel and J. Rydning.  
The Digitization of the World from Edge to Core.  
page 28, 2018.
-  J. Qiu, Q. Wu, G. Ding, Y. Xu, and S. Feng.  
A survey of machine learning for big data processing.  
*EURASIP Journal on Advances in Signal Processing*, 2016.
-  O. Mutlu, S. Ghose, J. Gómez-Luna, and R. Ausavarungnirun.  
Processing data where it makes sense: Enabling in-memory computation.  
*Microprocessors and Microsystems*, 67, 2019.
-  L. Breiman.  
Random Forests.  
*Machine Learning*, 45(1), 2001.



Christophe Guyeux, Stéphane Chrétien, Gaby Bou Tayeh, Jacques Demerjian, and Jacques Bahi.

Introducing and comparing recent clustering methods for massive data management in the internet of things.

*Journal of Sensor and Actuator Networks*, 8(4), 2019.



Mayra Rodriguez, Cesar Comin, Dalcimar Casanova, Odemir Bruno, Diego Amancio, Francisco Rodrigues, and Luciano da F. Costa.

Clustering algorithms: A comparative approach.

*PLOS ONE*, 14, 12 2016.



M. Wright and A. Ziegler.

ranger: A fast implementation of random forests for high dimensional data in c++ and r.

*Journal of Statistical Software, Articles*, 77, 2017.



A. Anghel, N. Ioannou, T. P. Parnell, N. Papandreou,  
C. Mendler-Dünner, and H. Pozidis.

Breadth-first, depth-next training of random forests.

*ArXiv*, [abs/1910.06853](https://arxiv.org/abs/1910.06853), 2019.