EXTENSION DE COMPACT-DIAGRAM AUX SMART MVDS

JFPC19

Hélène Verhaeghe[1], Christophe Lecoutre[2], Pierre Schaus[1]

12 Juin 2019

[1] ICTEAM, UCLouvain, Place Sainte Barbe 2, 1348 Louvain-la-Neuve, Belgium, $\{firstname.lastname\}@uclouvain.be$
[2] CRIL-CNRS UMR 8188, Université d'Artois, F-62307 Lens, France, $lecoutre@cril.fr$

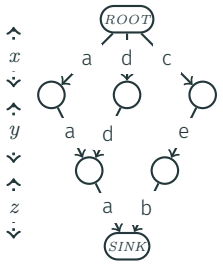
UCLouvain

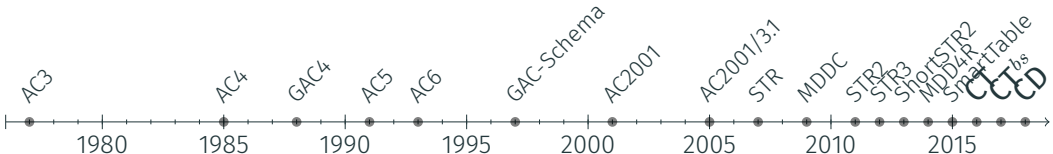
UCLouvain

	x	y	z
τ_1	a	a	a
τ_2	d	d	a
τ_3	c	e	b
$\vdots$	$\vdots$	$\vdots$	$\vdots$

Les tables sont les contraintes les plus vieilles de la PPC

Les MDDs sont équivalents aux tables



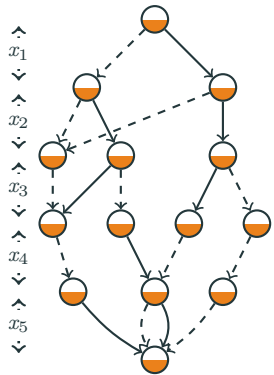


2016 : Nouvel algorithme! Compact-Table [CP2016], basé sur les opérations bit à bit, devance completement les algorithmes existants

LE MVD SIMPLEMENT INTELLIGENT

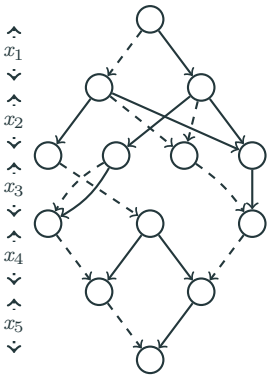
UCLouvain

Diagramme Decision Multi-valeur



Noeuds décisionnel

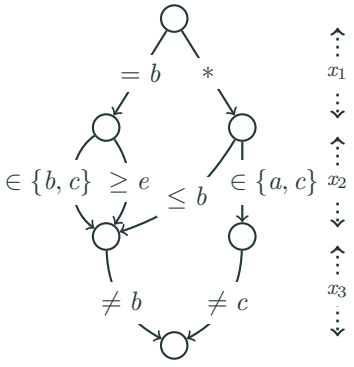
Diagramme Variable Multi-valeur



Noeuds non-décisionnel

UCLouvain

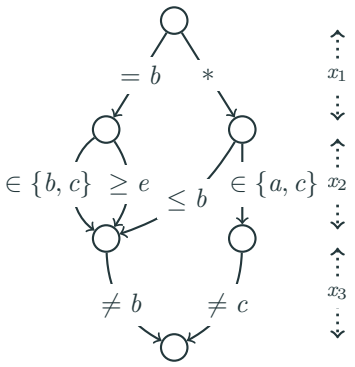
Un MVD simplement intelligent



UCLouvain

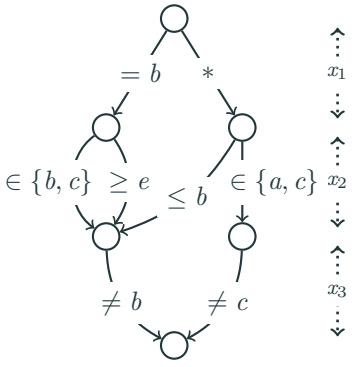
Un MVD simplement intelligent possède des étiquettes représentant plusieurs valeurs

a b c d e f



UCLouvain

Un MVD simplement intelligent possède des étiquettes représentant plusieurs valeurs

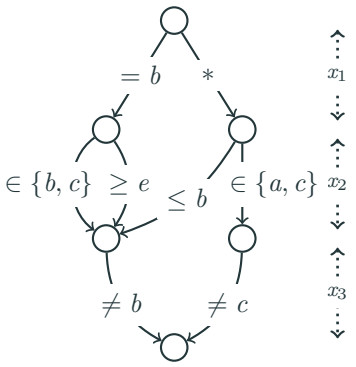


valeur simple: $= e$

a	b	c	d	e	f
✗	✗	✗	✗	✔	✗

UCLouvain

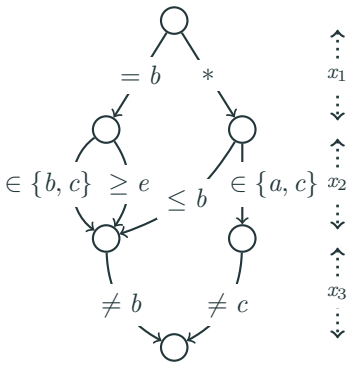
Un MVD simplement intelligent possède des étiquettes représentant plusieurs valeurs



	a	b	c	d	e	f
valeur simple: $= e$	✗	✗	✗	✗	✔	✗
valeur universelle: $*$	✔	✔	✔	✔	✔	✔

UCLouvain

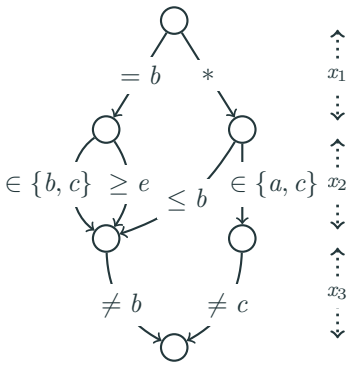
Un MVD simplement intelligent possède des étiquettes représentant plusieurs valeurs



	a	b	c	d	e	f
valeur simple: $= e$	✗	✗	✗	✗	✔	✗
valeur universelle: $*$	✔	✔	✔	✔	✔	✔
exclusion: $\neq e$	✔	✔	✔	✔	✗	✔

UCLouvain

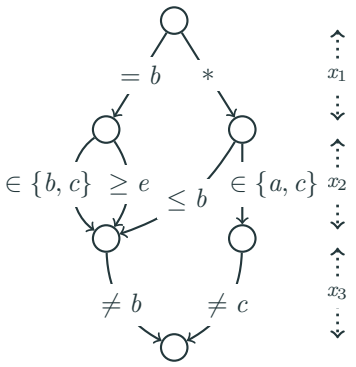
Un MVD simplement intelligent possède des étiquettes représentant plusieurs valeurs



	a	b	c	d	e	f
valeur simple: $= e$	✗	✗	✗	✗	✔	✗
valeur universelle: $*$	✔	✔	✔	✔	✔	✔
exclusion: $\neq e$	✔	✔	✔	✔	✗	✔
borne suppérieure: $\leq c$	✔	✔	✔	✗	✗	✗

UCLouvain

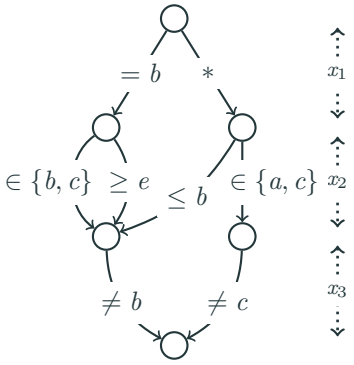
Un MVD simplement intelligent possède des étiquettes représentant plusieurs valeurs



	a	b	c	d	e	f
valeur simple: $= e$	✗	✗	✗	✗	✔	✗
valeur universelle: $*$	✔	✔	✔	✔	✔	✔
exclusion: $\neq e$	✔	✔	✔	✔	✗	✔
borne suppérieure: $\leq c$	✔	✔	✔	✗	✗	✗
borne inférieure: $\geq c$	✗	✗	✔	✔	✔	✔

UCLouvain

Un MVD simplement intelligent possède des étiquettes représentant plusieurs valeurs



	a	b	c	d	e	f
valeur simple: $= e$	✗	✗	✗	✗	✔	✗
valeur universelle: $*$	✔	✔	✔	✔	✔	✔
exclusion: $\neq e$	✔	✔	✔	✔	✗	✔
borne suppérieure: $\leq c$	✔	✔	✔	✗	✗	✗
borne inférieure: $\geq c$	✗	✗	✔	✔	✔	✔
ensemble: $\in \{a, c, d\}$	✔	✗	✔	✔	✗	✗

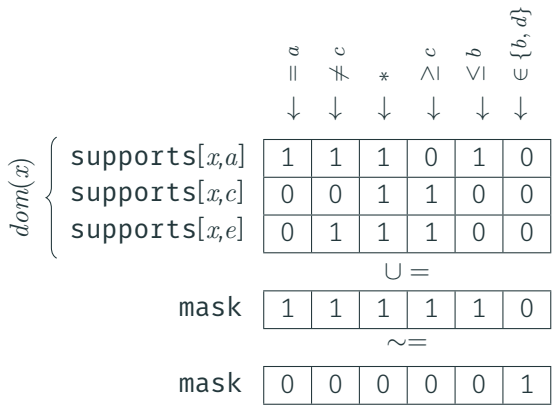
L'algorithme compact-diagram^{bs}

Algorithm: Partie retrait direct de la mise à jour

if la couche d'arête ne contient pas de $\in$ **then**
- **if** $|\Delta(x)| < |dom(x)|$ **then**
  - Mise à jour incrémentale ($=$, $\neq$, $*$);
  - Mise à jour de borne inférieure ($\leq$);
  - Mise à jour de borne suppérieure ($\geq$);
- **else**
  - Mise à jour par réinitialisation ($=$, $\neq$, $*$, $\leq$, $\geq$, $\in$);

else
- Mise à jour par réinitialisation ($=$, $\neq$, $*$, $\leq$, $\geq$, $\in$);

UCLouvain



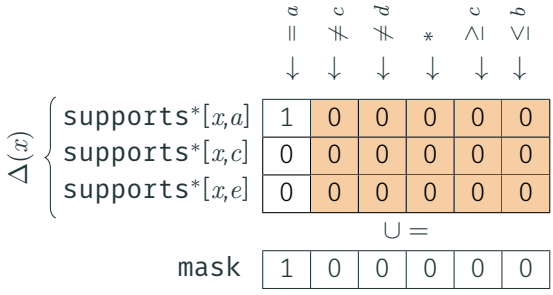
Algorithm: Mise à jour par réinitialisation

foreach valeur $a \in dom(x)$ **do**
  $\texttt{mask}[x] \leftarrow \texttt{mask}[x] \mid \texttt{supports}[x, a]$;

$\texttt{mask}[x] \leftarrow \sim \texttt{mask}[x]$;

Un bit de $\texttt{supports}[x,a]$ est mis à 1 si l'étiquette accepte a.

UCLouvain



Algorithm: Mise à jour incrémentale

foreach valeur $a \in \Delta_x$ **do**
  mask$[x] \leftarrow$ mask$[x]$ |
    supports$^*[x, a]$;

Un bit de supports$^*[x,a]$ est mis à 1 si l'étiquette accepte **seulement** a.

UCLouvain

	$= a \downarrow$	$\neq c \downarrow$	$* \downarrow$	$\geq a \downarrow$	$\leq c \downarrow$	$\leq b \downarrow$
supportsMin$[x, x.min]$	0	1	1	1	1	0
supportsMax$[x, x.max]$	1	1	1	1	1	1
			$\sim =$			
$\sim$ supportsMin$[x, c]$	1	0	0	0	0	1
$\sim$ supportsMax$[x, d]$	0	0	0	0	0	0
			$\cup =$			
mask	1	0	0	0	0	1

Algorithm: Mise à jour de borne inférieure et suppérieure

```
if dom(x).minChangé() then
    mask[x] ← mask[x] | ∼ supportsMin[x, x.min];
if dom(x).maxChangé() then
    mask[x] ← mask[x] | ∼ supportsMax[x, x.max];
```

Un bit de supportsMin$[x, a]$ est mis à 1 si l'étiquette accepte au moins une valeur $\geq a$.

Un bit de supportsMax$[x, a]$ est mis à 1 si l'étiquette accepte au moins une valeur $\leq a$.

mot 0				mot 1				mot 2			
w_0	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	-	-
$=$	$\leq$	$\geq$	$\in$	$\neq$	$>$	$\notin$	$<$	$\neq$	$*$		

UCLouvain

mot 0				mot 1				mot 2			
w_0	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	-	-
$=$	$\leq$	$\geq$	$\in$	$\neq$	$>$	$\notin$	$<$	$\neq$	$*$		

$\Downarrow$

mot 0				mot 1				mot 2				mot 3			
w_0	w_4	w_8	w_9	w_3	w_6	-	-	w_1	w_7	-	-	w_2	w_5	-	-
$=$	$\neq$	$\neq$	$*$	$\in$	$\notin$			$\leq$	$<$			$\geq$	$>$		

UCLouvain

mot 0				mot 1				mot 2				mot 3			
w_0	w_4	w_8	w_9	w_3	w_6	-	-	w_1	w_7	-	-	w_2	w_5	-	-
$=$	$\neq$	$\neq$	$*$	$\in$	$\notin$			$\leq$	$<$			$\geq$	$>$		

Mise à jour incrémentale ou par réinitialisation (dépendant si $|\Delta(x)| < |dom(x)|$)

Mise à jour par réinitialisation

Mise à jour de borne inférieure

Mise à jour de borne suppérieure

RÉSULTATS

UCLouvain

Une Table

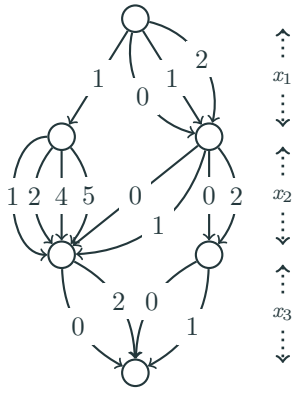
x_1	x_2	x_3
0	0	0
0	0	1
0	0	2
0	1	0
0	1	2
0	2	0
0	2	1
1	0	0
1	0	1
1	0	2
$\vdots$	$\vdots$	$\vdots$

UCLouvain

Une Table

x_1	x_2	x_3
0	0	0
0	0	1
0	0	2
0	1	0
0	1	2
0	2	0
0	2	1
1	0	0
1	0	1
1	0	2
⋮	⋮	⋮

transformée en un MVD

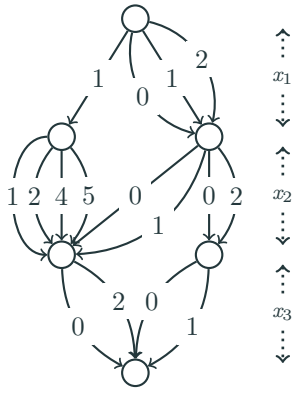


UCLouvain

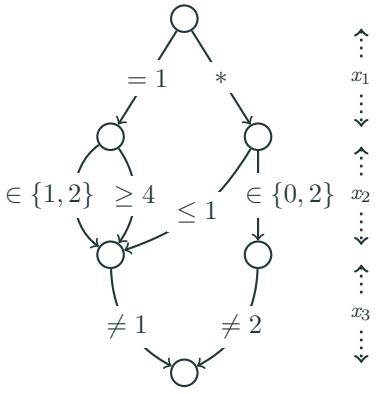
Une Table

x_1	x_2	x_3
0	0	0
0	0	1
0	0	2
0	1	0
0	1	2
0	2	0
0	2	1
1	0	0
1	0	1
1	0	2
⋮	⋮	⋮

transformée en un MVD



transformée en un bs-MVD



UCLouvain

Une Table

x_1	x_2	x_3
0	0	0
0	0	1
0	1	0
0	1	1
0	2	1
0	3	0
0	3	1
1	0	0
1	0	1
1	1	0
$\vdots$	$\vdots$	$\vdots$

UCLouvain

Une Table

x_1	x_2	x_3
0	0	0
0	0	1
0	1	0
0	1	1
0	2	1
0	3	0
0	3	1
1	0	0
1	0	1
1	1	0
$\vdots$	$\vdots$	$\vdots$

transformée en une bs-Table

x_1	x_2	x_3
$=1$	$=2$	≤ 1
$*$	$\neq 2$	≤ 1
$*$	≤ 2	$=1$

UCLouvain

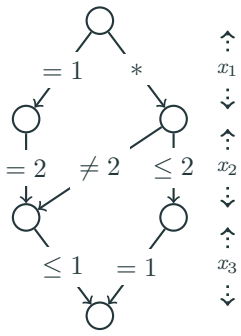
Une Table

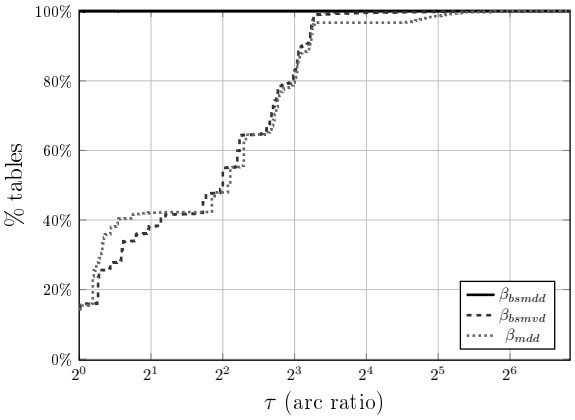
x_1	x_2	x_3
0	0	0
0	0	1
0	1	0
0	1	1
0	2	1
0	3	0
0	3	1
1	0	0
1	0	1
1	1	0
$\vdots$	$\vdots$	$\vdots$

transformée en une bs-Table

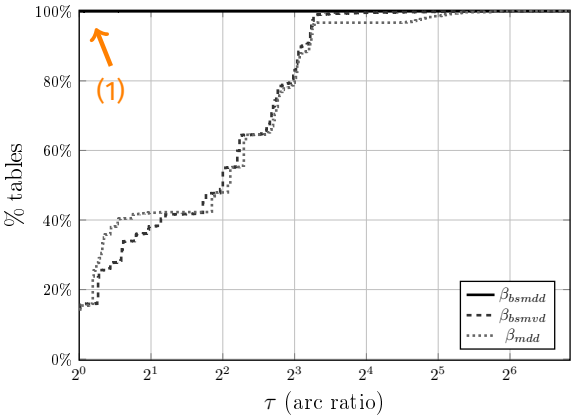
x_1	x_2	x_3
$=1$	$=2$	≤ 1
$*$	$\neq 2$	≤ 1
$*$	≤ 2	$=1$

transformée en un bs-MVD



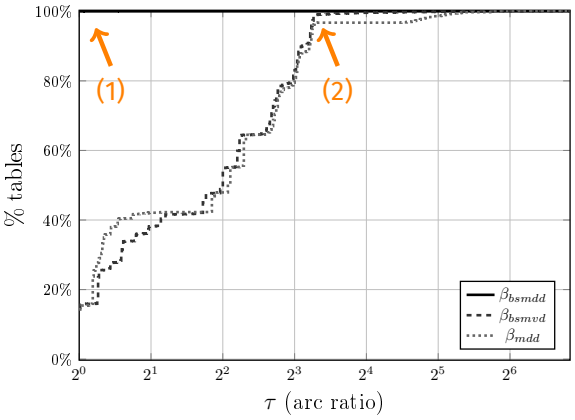

100%
80%
60%
40%
20%
0%
% tables
2^0
2^1
2^2
2^3
2^4
2^5
2^6
τ (arc ratio)
β_{bsmdd}
β_{bsmvd}
β_{mdd}

Arête:



Arête:

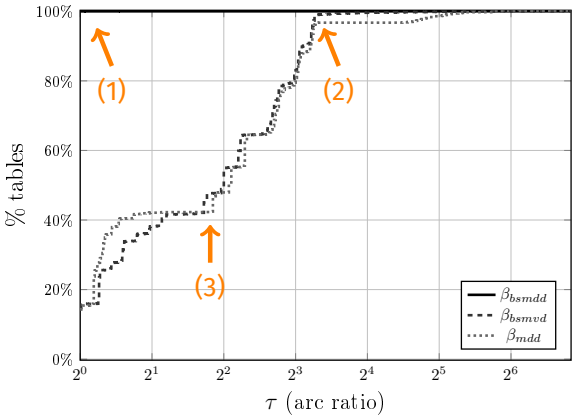
(1) bs-MDDs ont toujours moins d'arêtes



Arête:

(1) bs-MDDs ont toujours moins d'arêtes

(2) bs-MDDs ont jusqu'à 10× moins d'arêtes

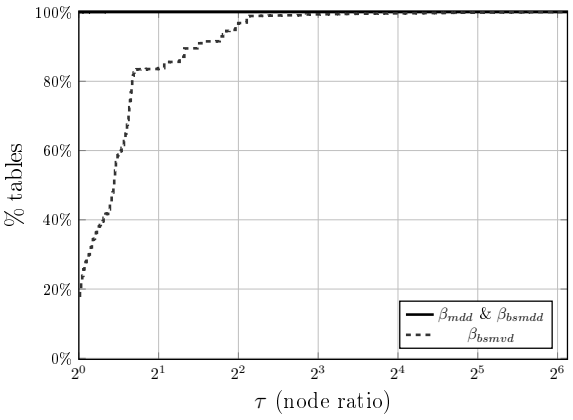


Arête:

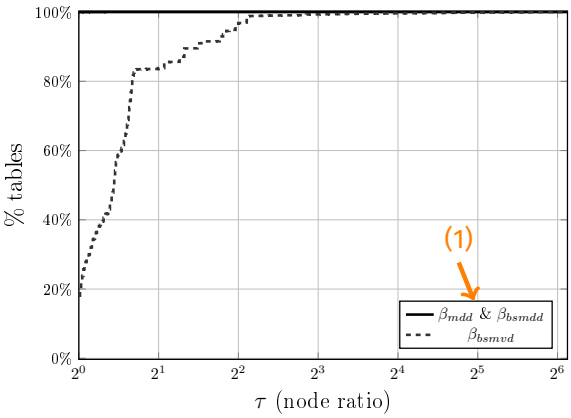
(1) bs-MDDs ont toujours moins d'arêtes

(2) bs-MDDs ont jusqu'à 10× moins d'arêtes

(3) bs-MVDs et MDDs ont un nombre similaire d'arêtes

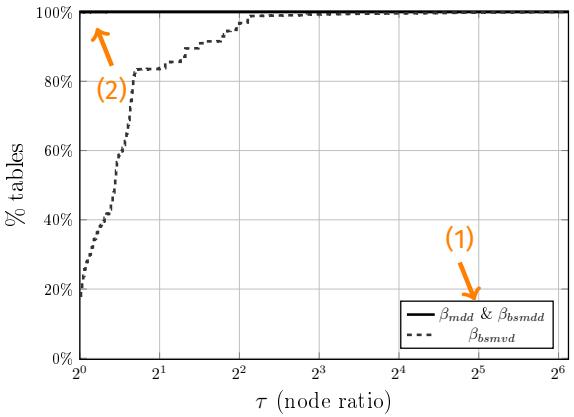

100%
80%
60%
40%
20%
0%
% tables
2^0
2^1
2^2
2^3
2^4
2^5
2^6
τ (node ratio)
β_{mdd} & β_{bsmdd}
β_{bsmvd}

Nodes:



Nodes:

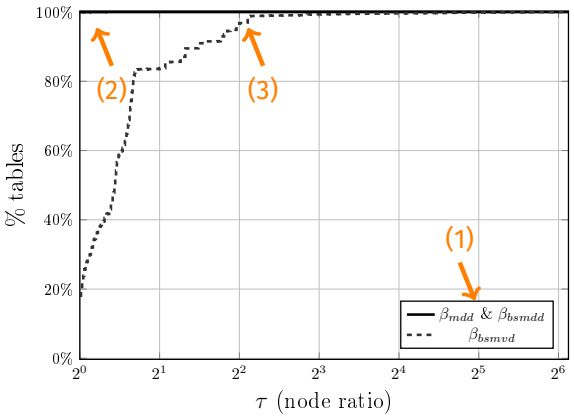
(1) bs-MDDs et MDDs ont le même nombre de noeuds (par construction)



Nodes:

(1) bs-MDDs et MDDs ont le même nombre de noeuds (par construction)

(2) bs-MDDs et MDDs ont toujours moins de noeuds



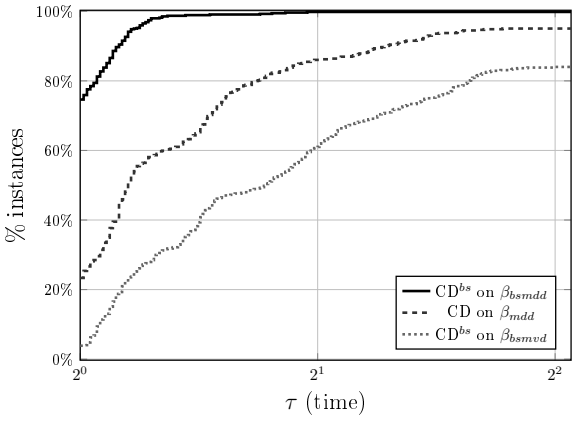
Nodes:

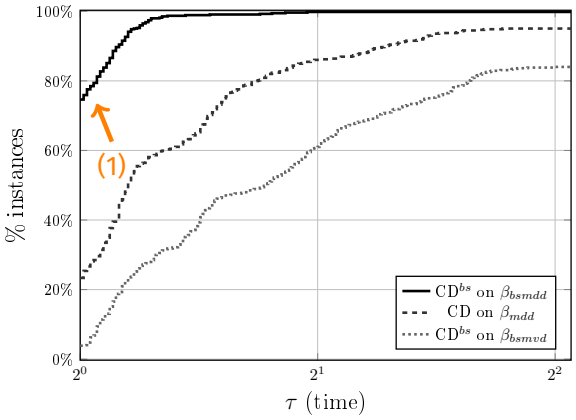
(1) bs-MDDs et MDDs ont le même nombre de noeuds (par construction)

(2) bs-MDDs et MDDs ont toujours moins de noeuds

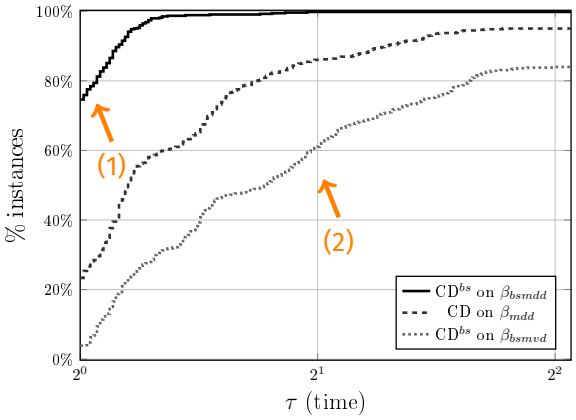
(3) bs-MVDs jusqu'à 4× plus de noeuds

RÉSULTATS (4000 INSTANCES)





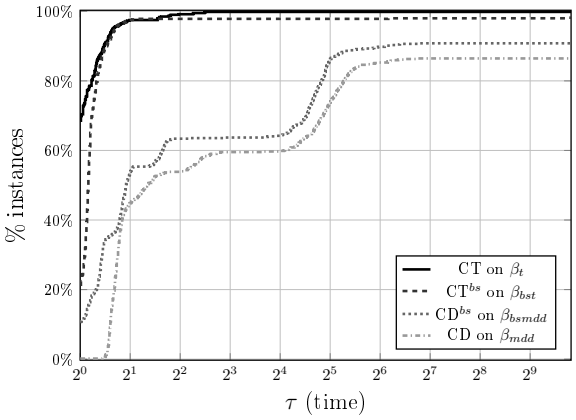
(1) CD^{bs} sur les bs-MDDs (moins d'arêtes) meilleur 80% du temps

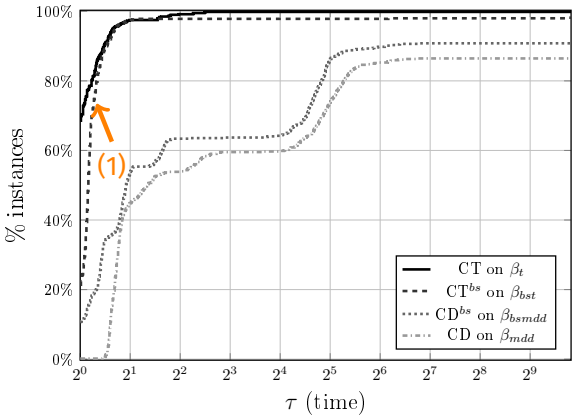


(1) CD^{bs} sur les bs-MDDs (moins d'arêtes) meilleur 80% du temps

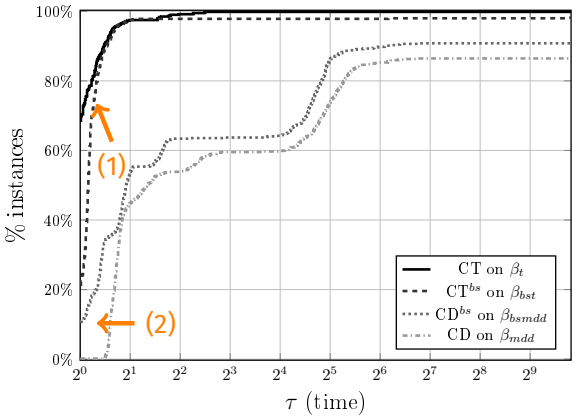
(2) CD^{bs} sur les bs-MVDs (plus de noeuds) pire

RESULTS (4000 INSTANCES)



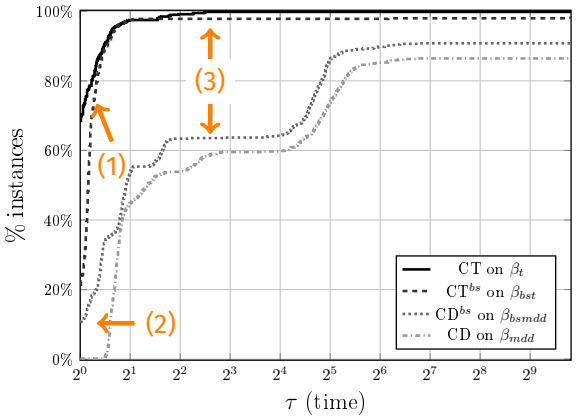


(1) CT et CT^{bs} toujours dominants



(1) CT et CT^{bs} toujours dominants

(2) CD^{bs} devient efficace quand la compression est grande

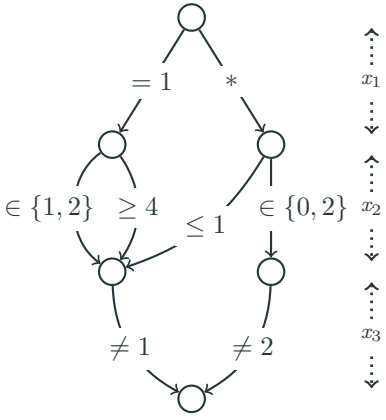


(1) CT et CT^{bs} toujours dominants

(2) CD^{bs} devient efficace quand la compression est grande

(3) réduction de l'écart

- Nouveau type de diagramme (MVD simplement intelligent) permettant **moins d'arêtes**
- Propagateur **dédié** (CD^{bs})
- **Réduction de l'écart** entre les algorithmes de propagation basé sur les tables (CT) et les diagrammes (CD^{bs})



Merci pour votre attention!

Des questions?