Contributions à la résolution générique des problèmes de satisfaction de contraintes

Julien Vion

Thèse encadrée par Christophe Lecoutre et Lakhdar Saïs

30 novembre 2007

Les problèmes combinatoires

- De très nombreux problèmes quotidiens ou industriels sont des problèmes fortement combinatoires

Les problèmes combinatoires

- De très nombreux problèmes quotidiens ou industriels sont des problèmes fortement combinatoires, par exemple :
  - Planification
  - Ordonnancement
  - Optimisation
  - Puzzles logiques

Les problèmes combinatoires

- De très nombreux problèmes quotidiens ou industriels sont des problèmes fortement combinatoires, par exemple :
  - Planification
  - Ordonnancement
  - Optimisation
  - Puzzles logiques
- Les algorithmes actuels permettant de décider ces problèmes sont exponentiels : $O(d^n)$

Les problèmes combinatoires

- De très nombreux problèmes quotidiens ou industriels sont des problèmes fortement combinatoires, par exemple :
  - Planification
  - Ordonnancement
  - Optimisation
  - Puzzles logiques

- Les algorithmes actuels permettant de décider ces problèmes sont exponentiels : $O(d^n)$

  Si on résoud un problème avec $n = 10$ *et* $d = 10$ *en 1 seconde, le même problème avec* $n = 20$ *et* $d = 20$ *pourra nécessiter plus de* 10^{16} *secondes, soit plus de 300 millions d'années, pour être résolu*

Les problèmes combinatoires

- De très nombreux problèmes quotidiens ou industriels sont des problèmes fortement combinatoires, par exemple :
  - Planification
  - Ordonnancement
  - Optimisation
  - Puzzles logiques

- Les algorithmes actuels permettant de décider ces problèmes sont exponentiels : $O(d^n)$

  Si on résoud un problème avec $n = 10$ *et* $d = 10$ *en 1 seconde, le même problème avec* $n = 20$ *et* $d = 20$ *pourra nécessiter plus de* 10^{16} *secondes, soit plus de 300 millions d'années, pour être résolu*

- Diverses approches RO/IA pour résoudre ces problèmes en pratique : programmation linéaire, dynamique ou par contraintes, réduction à un problème SAT ou CSP

Réseaux de Contraintes

Définitions

- **Variable** *: représente une valeur quelconque appartenant à l'ensemble fini* $\mathrm{dom}(X)$.
- **Instanciation** *: ensemble de couples* (X, v)*, notés* X_v*, avec* $v \in \mathrm{dom}(X)$
- **Contrainte** *: impliquant les variables* $\mathrm{vars}(C)$*, elle définit l'ensemble* $\mathrm{rel}(C)$ *des instanciations autorisées de ces variables*

Réseaux de Contraintes

Définitions

- **Variable** : *représente une valeur quelconque appartenant à l'ensemble fini* $\mathrm{dom}(X)$.
- **Instanciation** : *ensemble de couples* (X, v), *notés* X_v, *avec* $v \in \mathrm{dom}(X)$
- **Contrainte** : *impliquant les variables* $\mathrm{vars}(C)$, *elle définit l'ensemble* $\mathrm{rel}(C)$ *des instanciations autorisées de ces variables*

Définition

Un réseau de contraintes (CN) P *est un couple* $(\mathscr{X}, \mathscr{C})$, *avec :*

- $\mathscr{X}$: *un ensemble fini de variables.*
- $\mathscr{C}$: *un ensemble fini de contraintes. Chaque contrainte implique un sous-ensemble de* $\mathscr{X}$.

Problème de Satisfaction de Contraintes

Définition

Le problème de satisfaction de contraintes (CSP) consiste à déterminer (décider) si il existe une instanciation de toutes les variables d'un CN telles que toutes les contraintes soient satisfaites.

Problème de Satisfaction de Contraintes

Définition

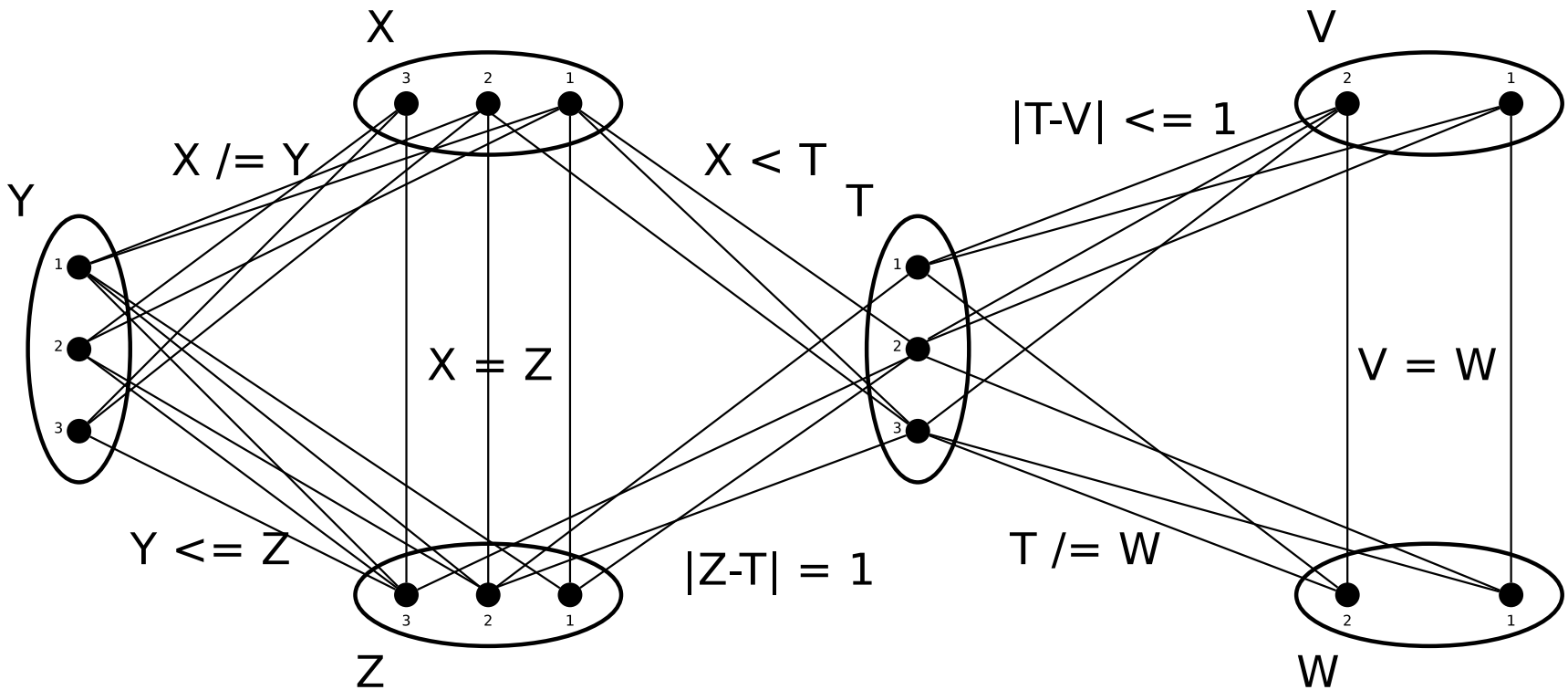
Le problème de satisfaction de contraintes (CSP) consiste à déterminer (décider) si il existe une instanciation de toutes les variables d'un CN telles que toutes les contraintes soient satisfaites.

Il s'agit d'un problème *NP*-complet : beaucoup de problèmes combinatoires (dans *NP*) peuvent être transformés en CSP

Notations

- n : nombre de variables
- d : taille du plus grand domaine
- e : nombre de contraintes
- k : arité maximale des contraintes
- K : nombre de 3-cliques (trois variables reliées deux à deux par trois contraintes)
- λ : nombre d'instanciations autorisées par l'ensemble des contraintes $\left(\lambda = \sum_{C \in \mathscr{C}} |\text{rel}(C)|\right)$

Exemple



$$n = 6,\ d = 3,\ e = 8,\ \lambda = 33$$

Résoudre un CSP : *inférence* et *recherche*

- Inférence : identifier des valeurs / instanciations inconsistantes, par exploitation des consistances.

Résoudre un CSP : *inférence* et *recherche*

- Inférence : identifier des valeurs / instanciations inconsistantes, par exploitation des consistances.

Exemple

Propriété	→ Algorithme(s)
Consistance d'arc généralisée	→ GAC-3, GAC-2001, GAC-3^{rm}...
Singleton consistance d'arc	→ SAC-1, SAC-3, SAC-SDS...
Consistance de chemin	→ PC-2001, PC-8...

Résoudre un CSP : *inférence* et *recherche*

- Inférence : identifier des valeurs / instanciations inconsistantes, par exploitation des consistances.

Exemple

Propriété	→ Algorithme(s)
Consistance d'arc généralisée	→ GAC-3, GAC-2001, GAC-3^{rm}...
Singleton consistance d'arc	→ SAC-1, SAC-3, SAC-SDS...
Consistance de chemin	→ PC-2001, PC-8...

- Recherche systématique / Recherche incomplète

Résoudre un CSP : *inférence* et *recherche*

- Inférence : identifier des valeurs / instanciations inconsistantes, par exploitation des consistances.

Exemple

Propriété	→ Algorithme(s)
Consistance d'arc généralisée	→ GAC-3, GAC-2001, GAC-3^{rm}...
Singleton consistance d'arc	→ SAC-1, SAC-3, SAC-SDS...
Consistance de chemin	→ PC-2001, PC-8...

- Recherche systématique / Recherche incomplète

Exemple

- FC, MGAC
- Hill-climbing, recherche Tabu

Plan

Plan

Consistance d'Arc

- Consistance d'Arc (Généralisée) : propriété de consistance de domaine, centrale pour la résolution des CSP
- Utilisation dans l'algorithme M(G)AC : on maintient la consistance d'arc à chaque nœud de la recherche

Consistance d'Arc

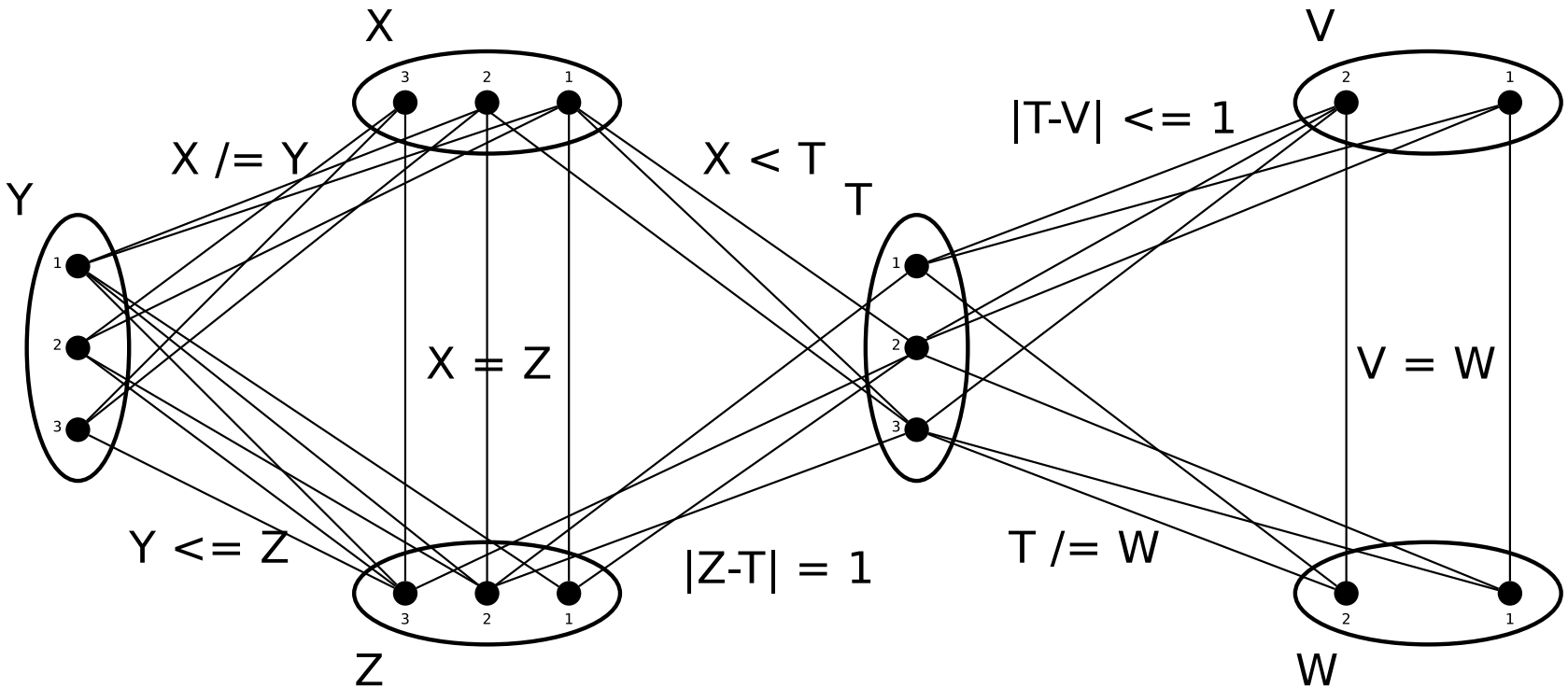
- Consistance d'Arc (Généralisée) : propriété de consistance de domaine, centrale pour la résolution des CSP
- Utilisation dans l'algorithme M(G)AC : on maintient la consistance d'arc à chaque nœud de la recherche

Definition (Consistance d'Arc (généralisée))

Soit $P = (\mathscr{X}, \mathscr{C})$ un CN

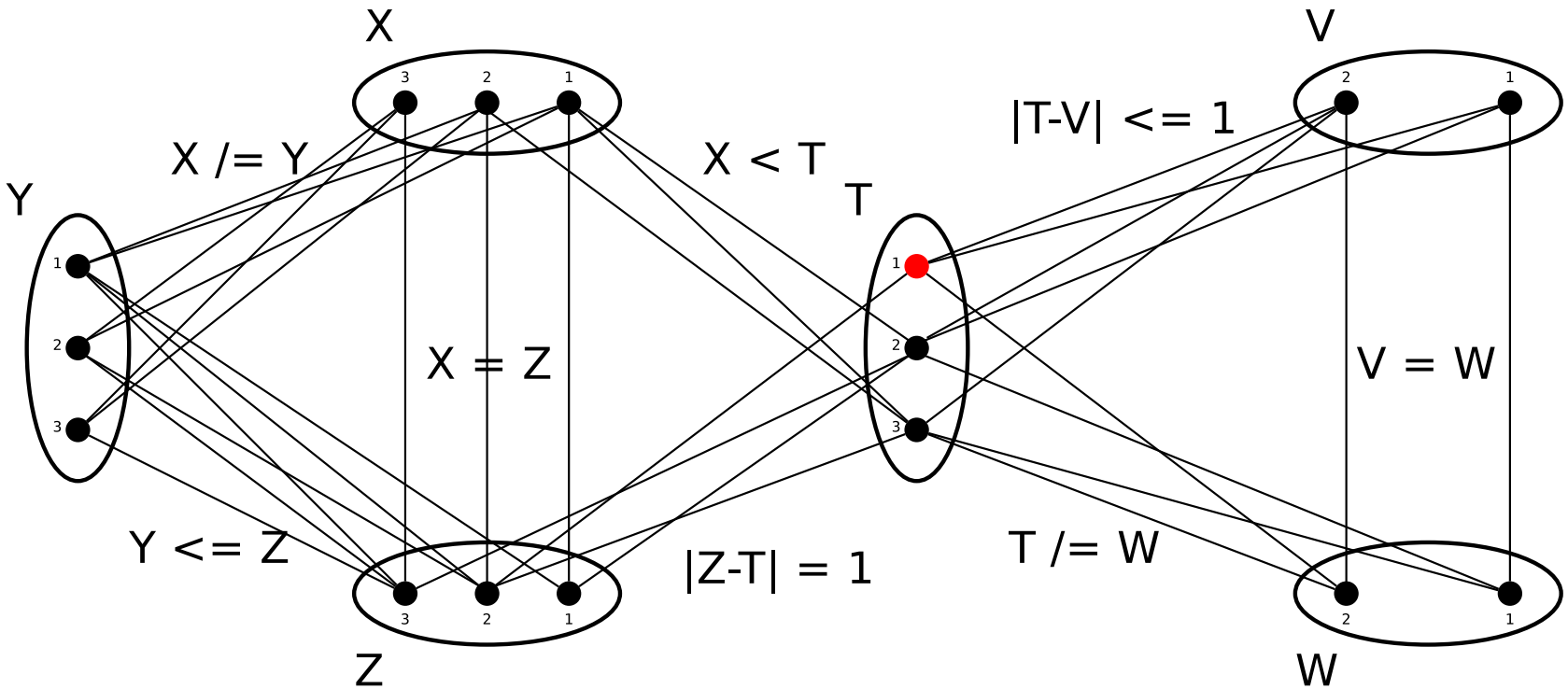
- X_a est GAC ssi $\forall C \in \mathscr{C} \mid X \in \text{vars}(C)$, $\exists I \in \text{rel}(C) \mid I$ valide et $X_a \in I$
- P est GAC ssi $\forall X \in \mathscr{X}$, $\forall a \in \text{dom}(X)$, X_a est GAC.

Exemple de filtrage par AC

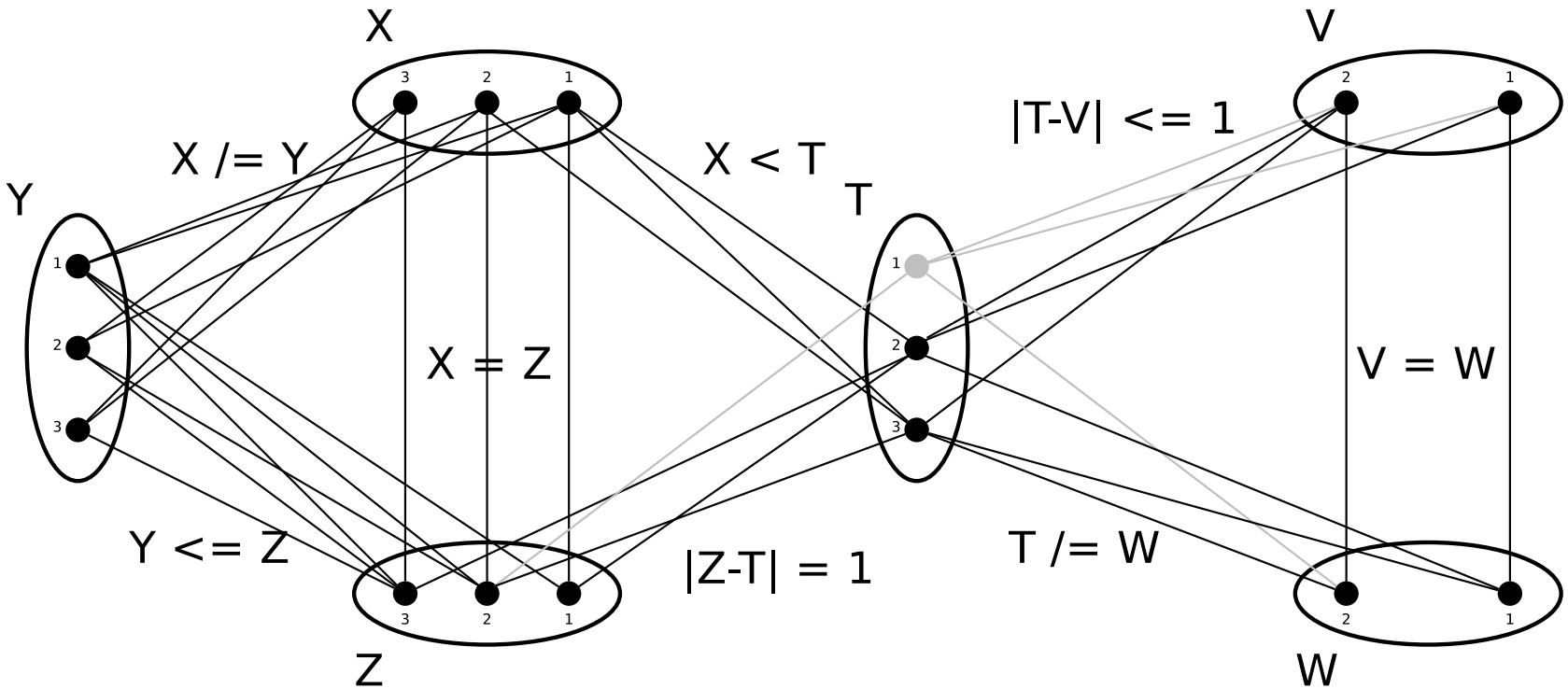


$$n = 6,\ d = 3,\ e = 8,\ \lambda = 33$$

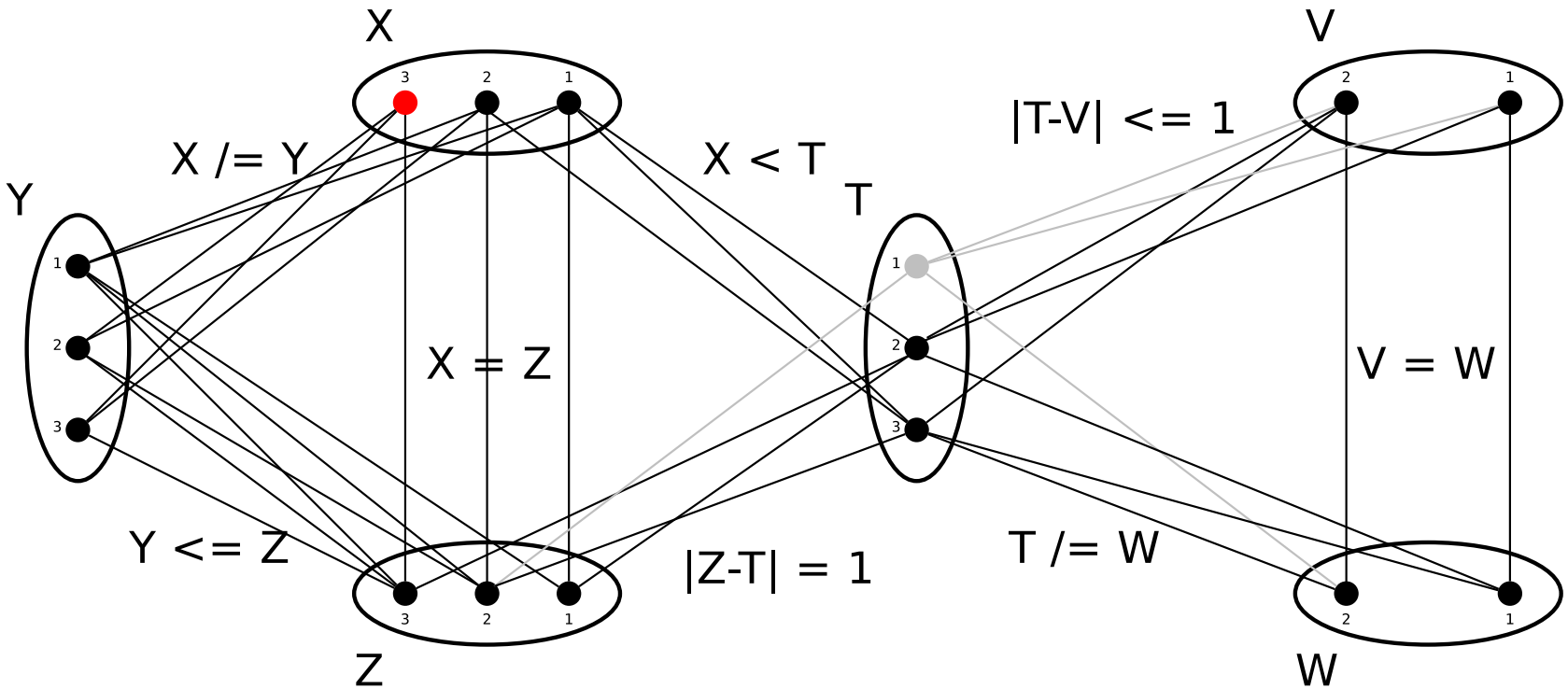
Exemple de filtrage par AC



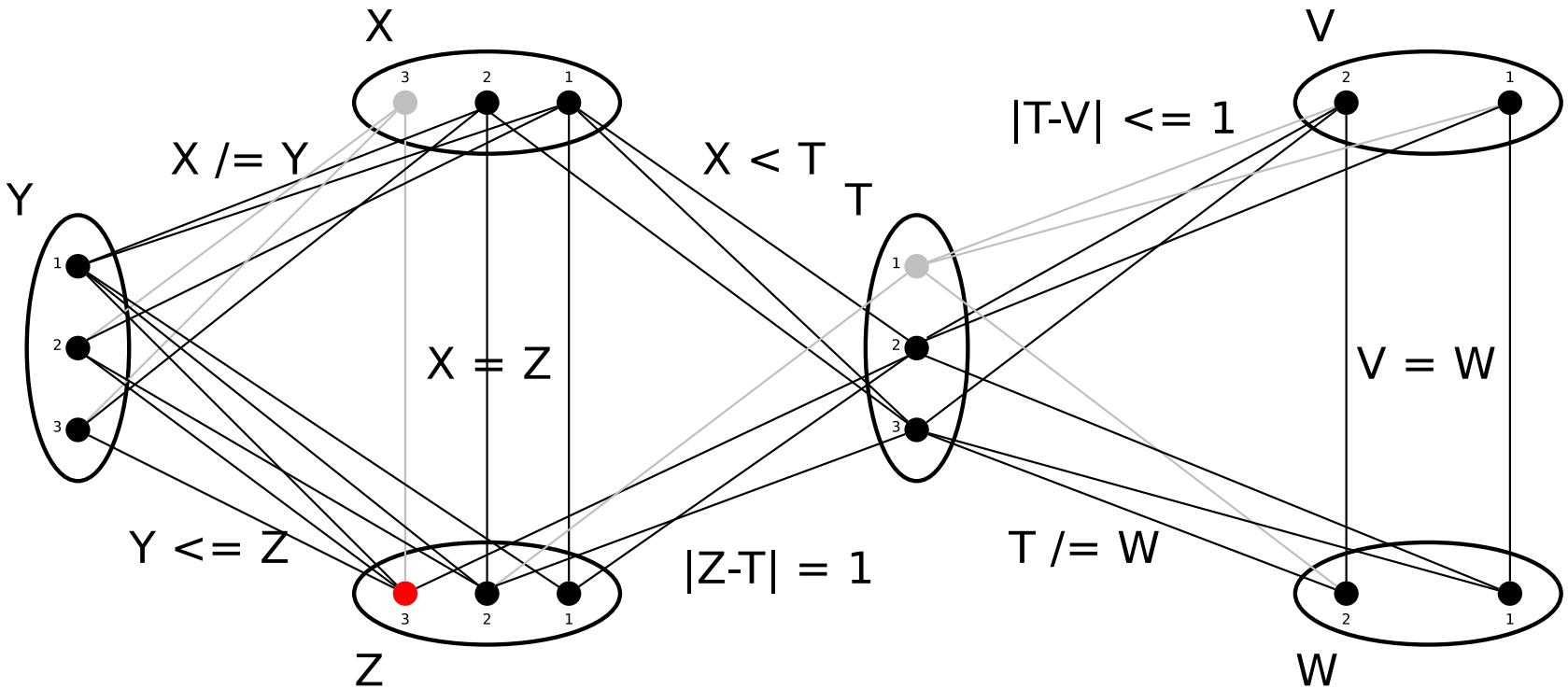
Exemple de filtrage par AC



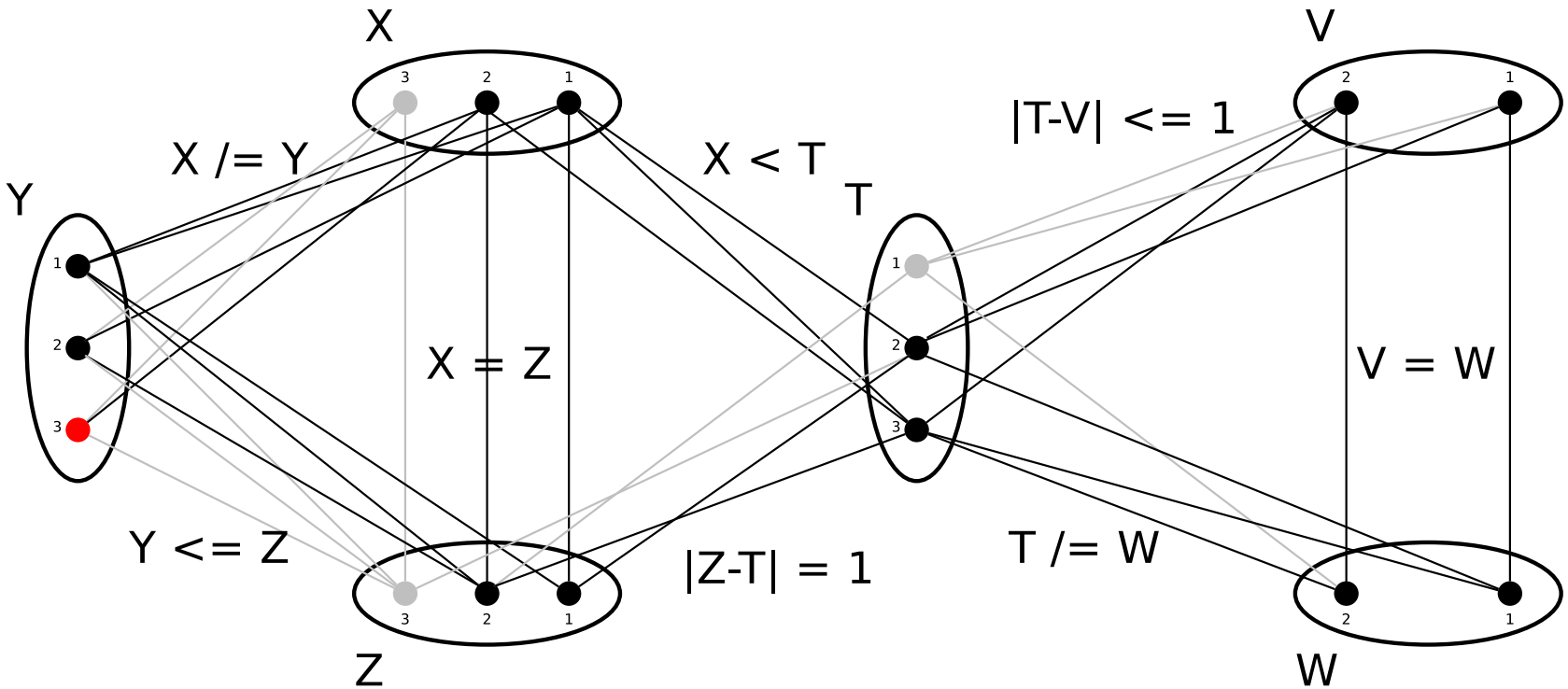
Exemple de filtrage par AC



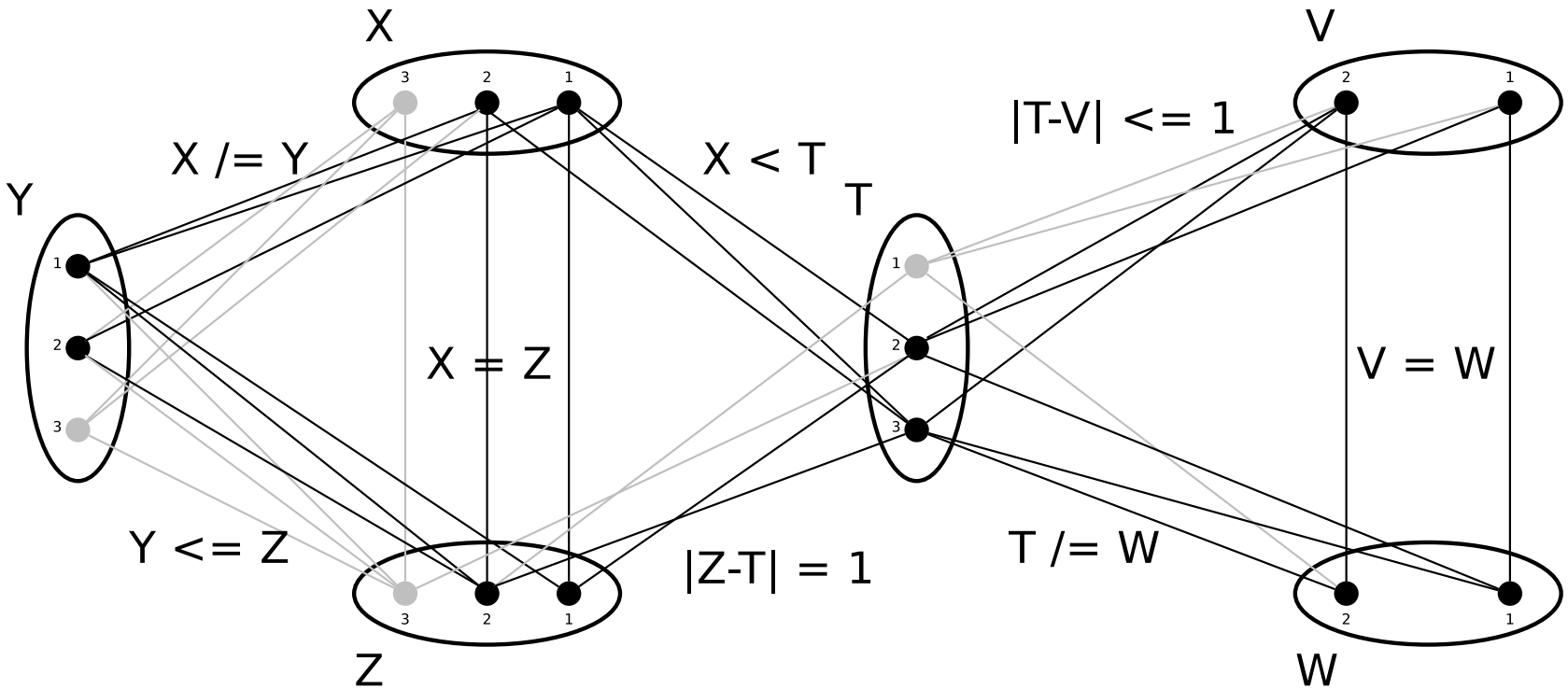
Exemple de filtrage par AC



Exemple de filtrage par AC



Exemple de filtrage par AC



$$n = 6,\ d = 2,\ e = 8,\ \lambda = 21$$

Plan

Exploiter les opérations bit-à-bit

- Principe : exploiter les opérations logiques ET et OU sur des *mots CPU* afin d'effectuer un grand nombre d'opérations logiques simultanément.

Exploiter les opérations bit-à-bit

- Principe : exploiter les opérations logiques ET et OU sur des *mots CPU* afin d'effectuer un grand nombre d'opérations logiques simultanément.
- Représentation binaire des domaines et des relations

Exploiter les opérations bit-à-bit

- Principe : exploiter les opérations logiques ET et OU sur des *mots CPU* afin d'effectuer un grand nombre d'opérations logiques simultanément.
- Représentation binaire des domaines et des relations
- Algorithmes AC-3^{bit} et AC-3^{bit+rm} (en combinaison avec l'exploitation des résidus)

Exploiter les opérations bit-à-bit

- Principe : exploiter les opérations logiques ET et OU sur des *mots CPU* afin d'effectuer un grand nombre d'opérations logiques simultanément.
- Représentation binaire des domaines et des relations
- Algorithmes AC-3^{bit} et AC-3^{bit+rm} (en combinaison avec l'exploitation des résidus)
- Autres applications possibles de la représentation binaire et des opérations bit-à-bit :
  - Détection de sous-domaines (cassage de symétries par détection de dominance)
  - Détection de valeurs substituables

Expérimentations

		MAC avec			
		AC-2001	AC-3	AC-3^{rm}	AC-3^{bit}
$\langle 40; 11; 414; 0{,}2\rangle$ (100 instances)	*cpu*	19,6	15,0	14,5	10,0
	mem	8,8	8,0	8,4	8,0
$\langle 40; 25; 180; 0{,}5\rangle$ (100 instances)	*cpu*	28,9	27,8	21,2	11,5
	mem	8,4	7,9	8,2	7,9
$\langle 40; 180; 84; 0{,}9\rangle$ (100 instances)	*cpu*	24,3	36,6	18,4	6,7
	mem	15	14	14	14
jobshop enddr (16 instances)	*cpu*	1 660,3	2 115,5	1 320,4	496,1
	mem	14	13	14	13
qwh-20 (10 instances)	*cpu*	266,0	183,0	242,0	153,0
	mem	33	21	44	21
domino-1000-1000	*cpu*	89,5	5 911,0	62,4	25,1
	mem	66	42	54	42

cpu en secondes, *mem* en Mio

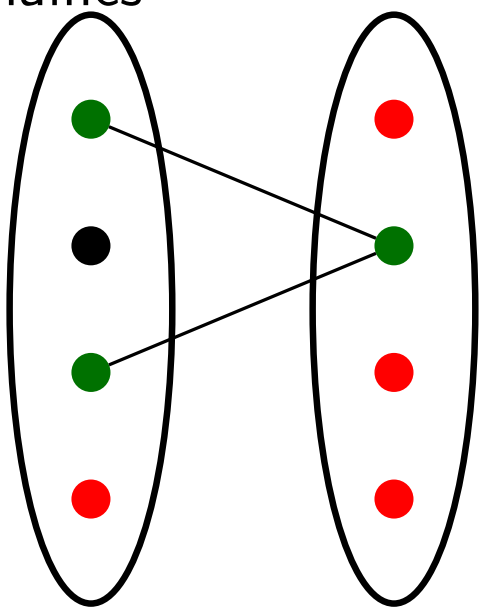
Plan

CSP continus

- Représenter les domaines de manière compacte sous forme d'intervalles ($\text{dom}(X) = \{1..1000\} = \{\min(X)..\max(X)\}$)
- Algorithmes issus des CSP continus ($\text{dom}(X) = \mathbb{R}$): 2B-consistance (Hull-consistance), 3B-consistance

CSP continus

- Représenter les domaines de manière compacte sous forme d'intervalles ($\text{dom}(X) = \{1..1000\} = \{\min(X)..\max(X)\}$)
- Algorithmes issus des CSP continus ($\text{dom}(X) = \mathbb{R}$): 2B-consistance (Hull-consistance), 3B-consistance
- 2B-consistance = consistance d'arc dont l'action est limitée aux bornes des domaines



CSP continus

- Représenter les domaines de manière compacte sous forme d'intervalles ($\text{dom}(X) = \{1..1000\} = \{\min(X)..\max(X)\}$)
- Algorithmes issus des CSP continus ($\text{dom}(X) = \mathbb{R}$): 2B-consistance (Hull-consistance), 3B-consistance
- 2B-consistance = consistance d'arc dont l'action est limitée aux bornes des domaines

Definition (2B-consistance)

$P = (\mathscr{X}, \mathscr{C})$ est 2B-consistant ssi $\forall X \in \mathscr{X}$,
$\min(X)$ et $\max(X)$ sont AC

CSP continus

- Représenter les domaines de manière compacte sous forme d'intervalles ($\text{dom}(X) = \{1..1000\} = \{\min(X)..\max(X)\}$)
- Algorithmes issus des CSP continus ($\text{dom}(X) = \mathbb{R}$): 2B-consistance (Hull-consistance), 3B-consistance
- 2B-consistance = consistance d'arc dont l'action est limitée aux bornes des domaines

Definition (2B-consistance)

$P = (\mathscr{X}, \mathscr{C})$ est 2B-consistant ssi $\forall X \in \mathscr{X}$,
$\min(X)$ et $\max(X)$ sont AC

- 3B-consistance = singleton 2B-consistance aux bornes

Definition (3B-consistance)

$P = (\mathscr{X}, \mathscr{C})$ est 3B-consistant ssi $\forall X \in \mathscr{X}$,
$2B(P|_{X=\min(X)}) \neq \bot$ et $2B(P|_{X=\max(X)}) \neq \bot$

2B-consistance : complexités

- La 2B consistance s'établit en $O(ed^2)$ sur un réseau binaire (avec ou sans l'enregistrement des supports "à la" AC-2001)

2B-consistance : complexités

- La 2B consistance s'établit en $O(ed^2)$ sur un réseau binaire (avec ou sans l'enregistrement des supports "à la" AC-2001)
- Appliqué à un réseau déjà 2B consistant, appliquer un algorithme de 2B consistance se fait en $O(ed)$

2B-consistance : complexités

- La 2B consistance s'établit en $O(ed^2)$ sur un réseau binaire (avec ou sans l'enregistrement des supports "à la" AC-2001)
- Appliqué à un réseau déjà 2B consistant, appliquer un algorithme de 2B consistance se fait en $O(ed)$
- Algorithme 2B+ : contrôles "opportunistes" supplémentaires, inspirés de Max-RPC (relaxation de PC)

3B-consistance : complexités

- Trois algorithmes proposés :
  - 3B-1, en $O(en^2d^3)$
  - 3B-1d, variante de 3B-1 effectuant des réductions de domaine dichotomiques, en $O(en^2d^3\log(d))$
  - 3B-2, en $O(end^3)$

3B-consistance : complexités

- Trois algorithmes proposés :
  - 3B-1, en $O(en^2d^3)$
  - 3B-1d, variante de 3B-1 effectuant des réductions de domaine dichotomiques, en $O(en^2d^3\log(d))$
  - 3B-2, en $O(end^3)$
- Tous ces algorithmes sont en $O(end^2)$ quand ils sont appliqués sur un réseau déjà 3B-consistant

Application des 2B/3B consistances sur des problèmes d'ordonnancement

Maintenir la consistance sur des instances d'Open Shop 7×7

	AC	2B	3B-1	3B-1d	3B-2
Distance moyenne à T_{OPT}	4.79%	28.6%	4.28%	3.00%	3.32%

Timeout = 300s

Application des 2B/3B consistances sur des problèmes d'ordonnancement

Maintenir la consistance sur des instances d'Open Shop 7×7

	AC	2B	3B-1	3B-1d	3B-2
Distance moyenne à T_{OPT}	4.79%	28.6%	4.28%	3.00%	3.32%

Timeout = 300s

Maintenir la consistance sur des instances de Jop Shop 8×8

	AC	2B	2B+	3B - 1	3B - 1d	3B - 2
CPU ($T = T_{OPT}$)	212	216	282	88	73	117
% UNSAT détecté ($T = T_{OPT} - 1$)	45%	55%	30%	85%	85%	85%

Timeout = 300s

Application des 2B/3B consistances sur des problèmes d'ordonnancement

Maintenir la consistance sur des instances d'Open Shop 7×7

	AC	2B	3B-1	3B-1d	3B-2
Distance moyenne à T_{OPT}	4.79%	28.6%	4.28%	3.00%	3.32%

Timeout = 300s

Maintenir la consistance sur des instances de Jop Shop 8×8

	AC	2B	2B+	3B - 1	3B - 1d	3B - 2
CPU ($T = T_{OPT}$)	212	216	282	88	73	117
% UNSAT détecté ($T = T_{OPT} - 1$)	45%	55%	30%	85%	85%	85%

Timeout = 300s

Travail aux bornes moins adapté pour d'autres problèmes réels (RLFAP...)

Plan

Consistance Duale

Définition

- *Une instanciation de longueur 2* $\{X_a, Y_b\}$ *est Dual-consistante (DC) ssi*
  - $Y_b \in \text{AC}(P|_{X=a})$ *et*
  - $X_a \in \text{AC}(P|_{Y=b})$.
- (X, Y) *est Dual-consistant ssi toutes les instanciations* I *telles que* $\text{vars}(I) = \{X, Y\}$ *qui sont consistantes sont aussi DC*
- P *est Dual-consistant ssi tous les couples de variables distinctes de* P *dont DC*

Consistance Duale

Définition

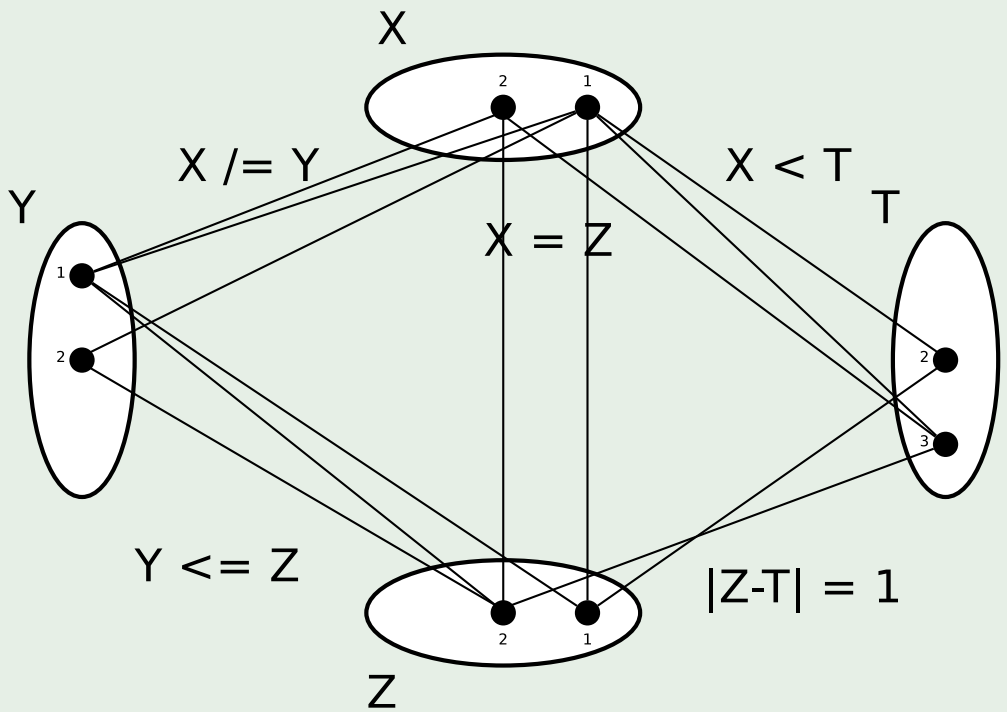
- *Une instanciation de longueur 2* $\{X_a, Y_b\}$ *est Dual-consistante (DC) ssi*
  - $Y_b \in \text{AC}(P|_{X=a})$ *et*
  - $X_a \in \text{AC}(P|_{Y=b})$.
- (X, Y) *est Dual-consistant ssi toutes les instanciations* I *telles que* $\text{vars}(I) = \{X, Y\}$ *qui sont consistantes sont aussi DC*
- P *est Dual-consistant ssi tous les couples de variables distinctes de* P *dont DC*

Propriété

$$\text{PC} = \text{DC}$$

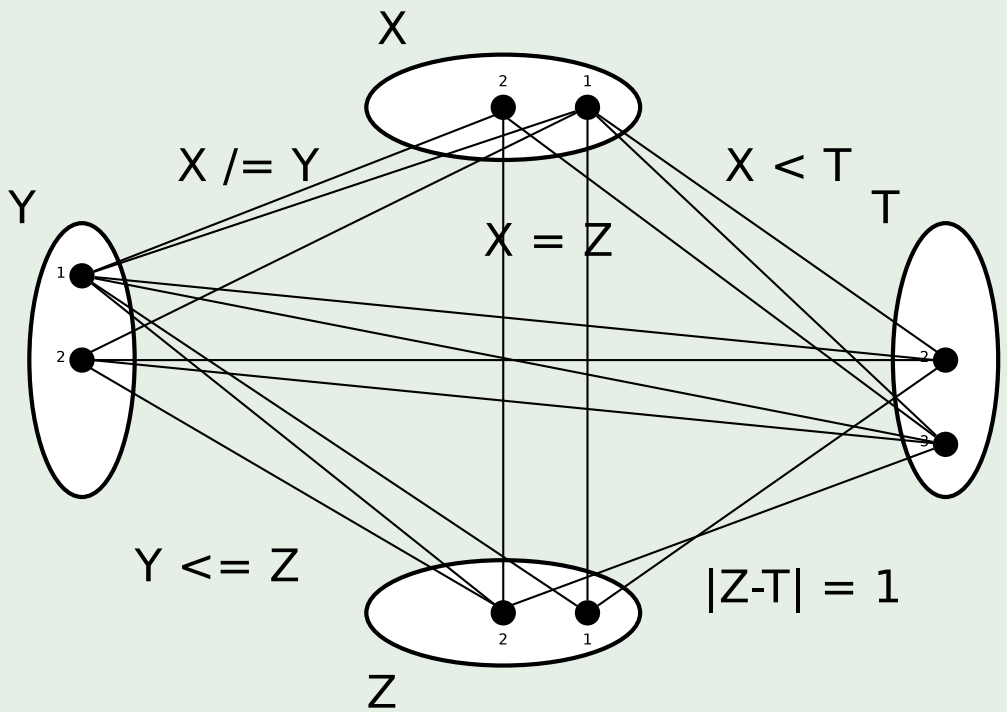
Établir DC

Exemple



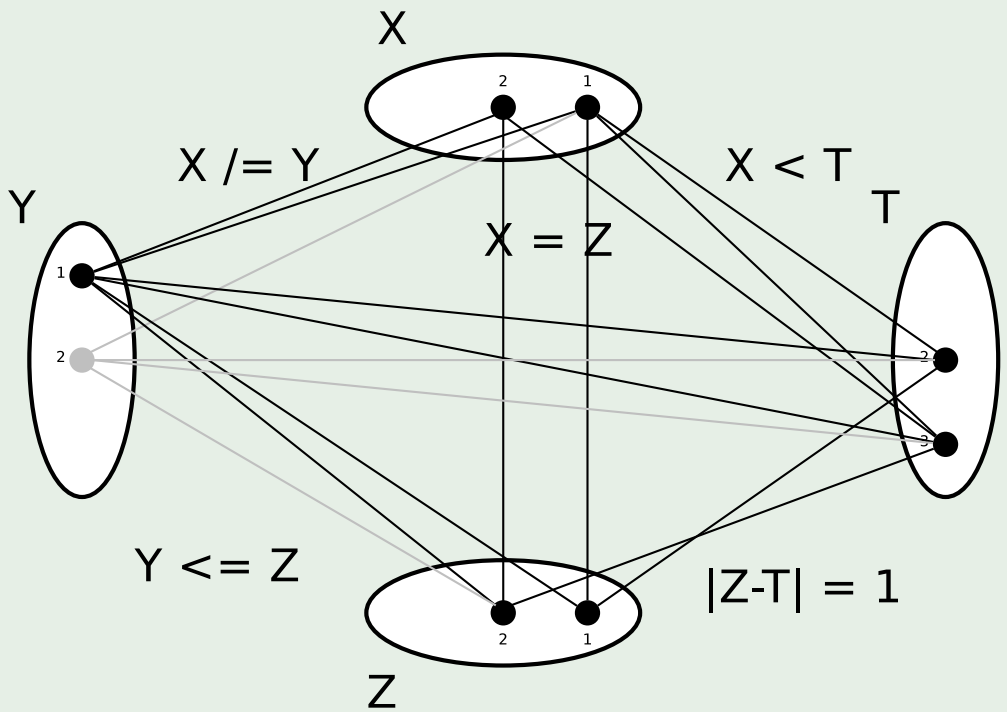
Établir DC

Exemple



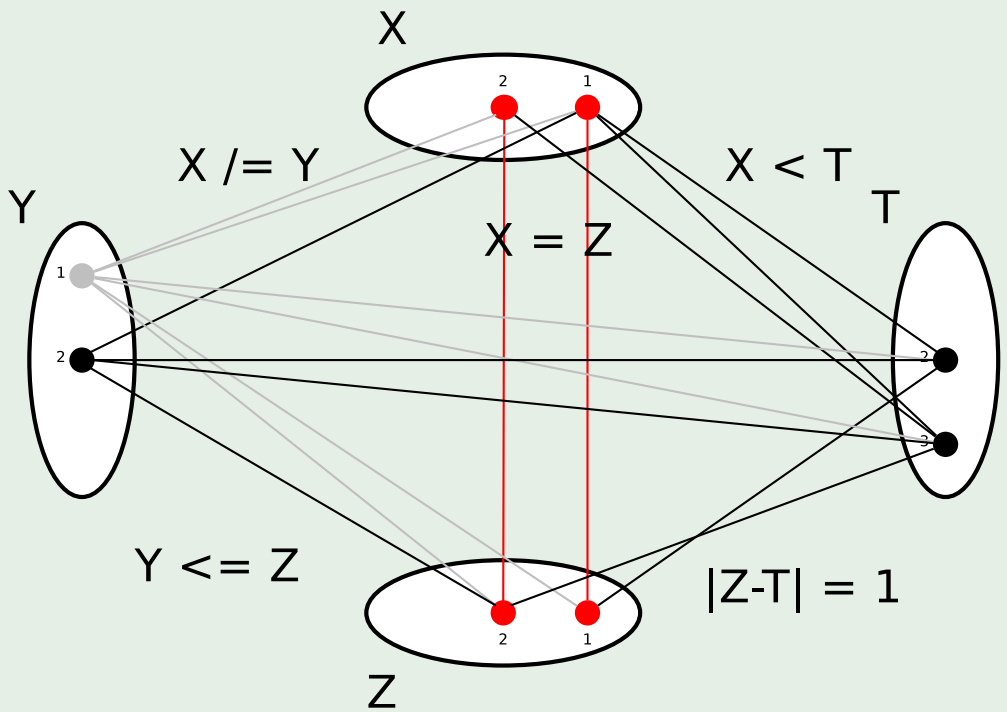
Établir DC

Exemple



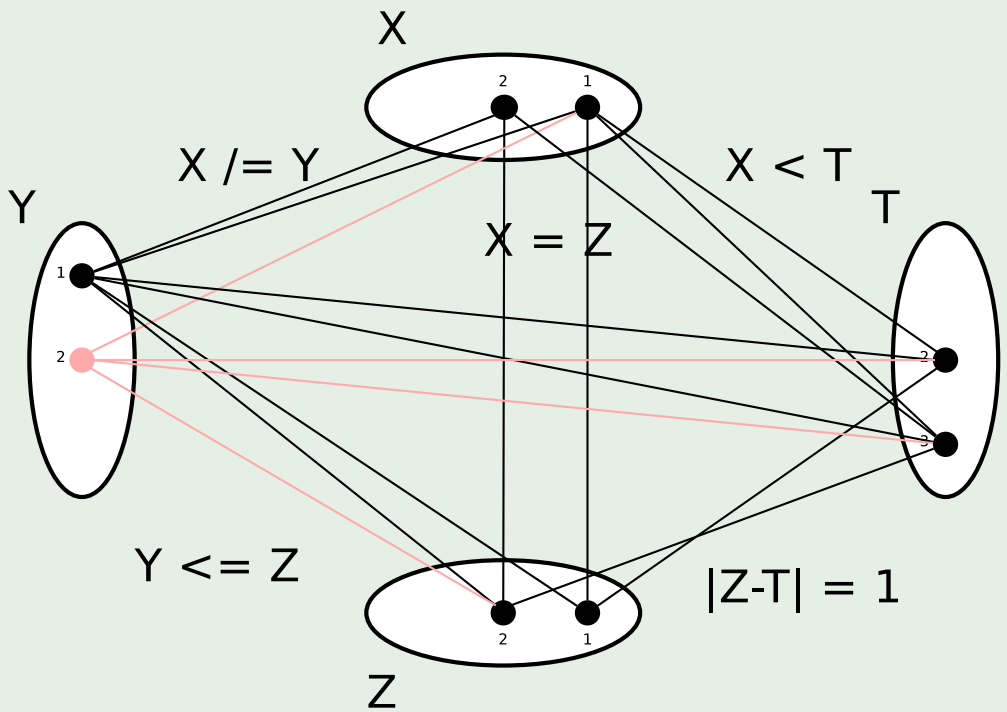
Établir DC

Exemple



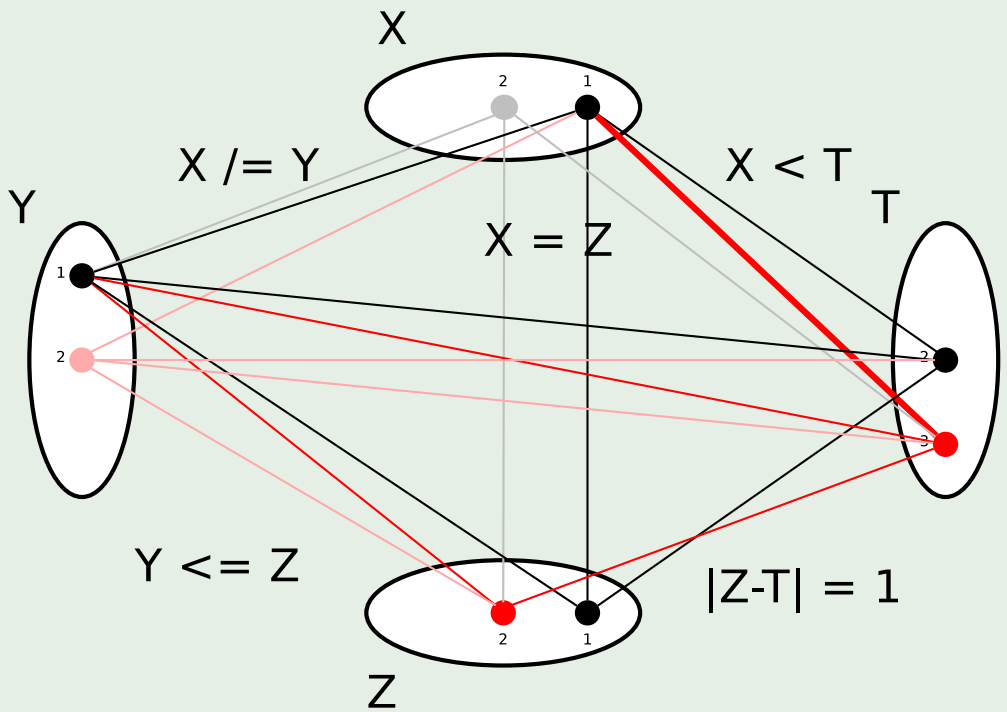
Établir DC

Exemple



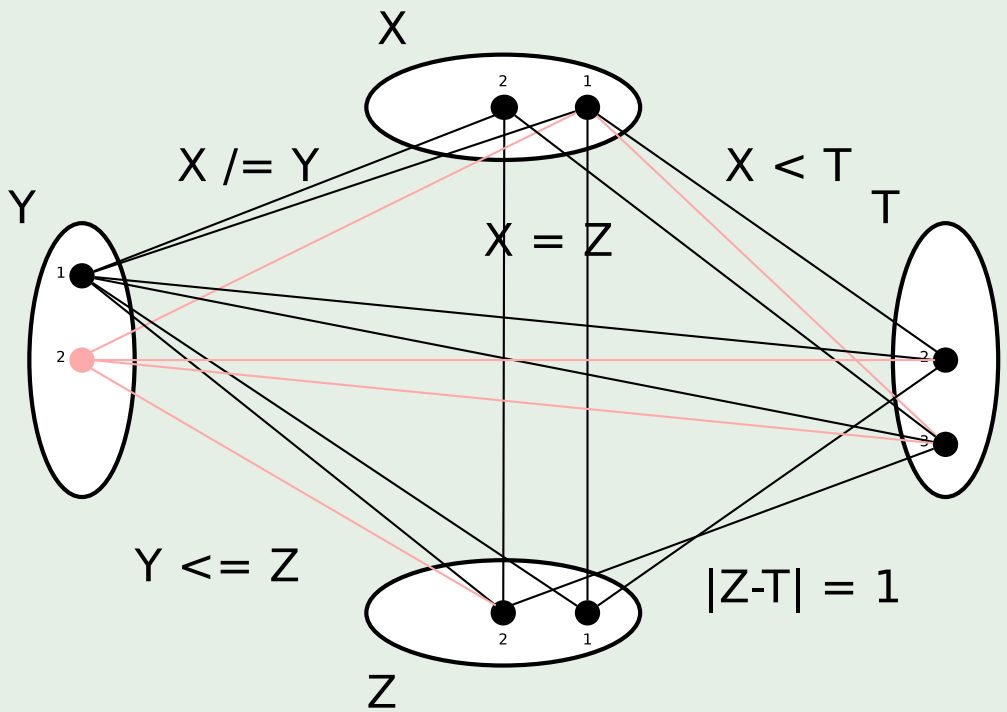
Établir DC

Exemple



Établir DC

Exemple



Établir sDC

sDC-1

Algorithme simple proche des algorithmes de SAC et de l'algorithme de [McGregor 1979] établissant PC
Maintenance systématiqueme de la consistance d'arc
Exploitation de la propriété $\mathrm{AC}(P|_{X=a}, \{X\}) = \mathrm{AC}(P|_{X=a}, \mathscr{X})$

Établir sDC

sDC-1

Algorithme simple proche des algorithmes de SAC et de l'algorithme de [McGregor 1979] établissant PC
Maintenance systématiqueme de la consistance d'arc
Exploitation de la propriété $\mathrm{AC}(P|_{X=a}, \{X\}) = \mathrm{AC}(P|_{X=a}, \mathscr{X})$

sDC-2

Amélioration pratique de sDC-1 exploitant partiellement l'incrémentalité de AC

Établir sDC

sDC-1

Algorithme simple proche des algorithmes de SAC et de l'algorithme de [McGregor 1979] établissant PC
Maintenance systématiqueme de la consistance d'arc
Exploitation de la propriété $\mathrm{AC}(P|_{X=a}, \{X\}) = \mathrm{AC}(P|_{X=a}, \mathscr{X})$

sDC-2

Amélioration pratique de sDC-1 exploitant partiellement l'incrémentalité de AC

sDC-3

Amélioration de la complexité de sDC-1 et sDC-2 en exploitant totalement l'incrémentalité de AC

sDC-1($P = (\mathscr{X}, \mathscr{C})$: CN)

AC($P, \mathscr{X}$)

$mark \leftarrow X \leftarrow first(\mathscr{X})$

répéter

  si *checkVar*-1(P, X) **alors**

    AC($P, \{X\}$)

    $mark \leftarrow X$

  $X \leftarrow next\text{-}modulo(\mathscr{X}, X)$

jusqu'à $X = mark$

checkVar-1($P = (\mathscr{X}, \mathscr{C})$: CN, X: Variable): Boolean

$modif \leftarrow$ **faux**

pour chaque $a \in \mathrm{dom}^P(X)$ **faire**

- $P' \leftarrow P$
- $\mathrm{AC}(P'|_{X=a}, \{X\})$
- **si** $P' = \bot$ **alors**
  - supprimer a de $\mathrm{dom}^P(X)$
  - $modif \leftarrow$ **vrai**
- **sinon**
  - **pour chaque** $Y \in \mathscr{X} \mid Y \neq X$ **faire**
    - soit C telle que $\mathrm{vars}(C) = \{X, Y\}$
    - **pour chaque** $b \in \mathrm{dom}^P(Y) \mid b \notin \mathrm{dom}^{P'}(Y)$ **faire**
      - retirer $\{X_a, Y_b\}$ de $\mathrm{rel}^P(C)$
      - $modif \leftarrow$ **vrai**

retourner $modif$

sDC : complexités

- Complexité spatiale est en $O(ed^2) \in O(n^2d^2)$

sDC : complexités

- Complexité spatiale est en $O(ed^2) \in O(n^2d^2)$
- Complexité temporelle de sDC-1/sDC-2 en $O(\lambda n^3 d^3) \in O(n^5 d^5)$

sDC : complexités

- Complexité spatiale est en $O(ed^2) \in O(n^2d^2)$
- Complexité temporelle de sDC-1/sDC-2 en $O(\lambda n^3 d^3) \in O(n^5 d^5)$
- Complexité temporelle de sDC-3 en $O(n^3 d^4)$ — identique à sPC-8 — proche de l'optimal $O(n^3 d^3)$

sDC : complexités

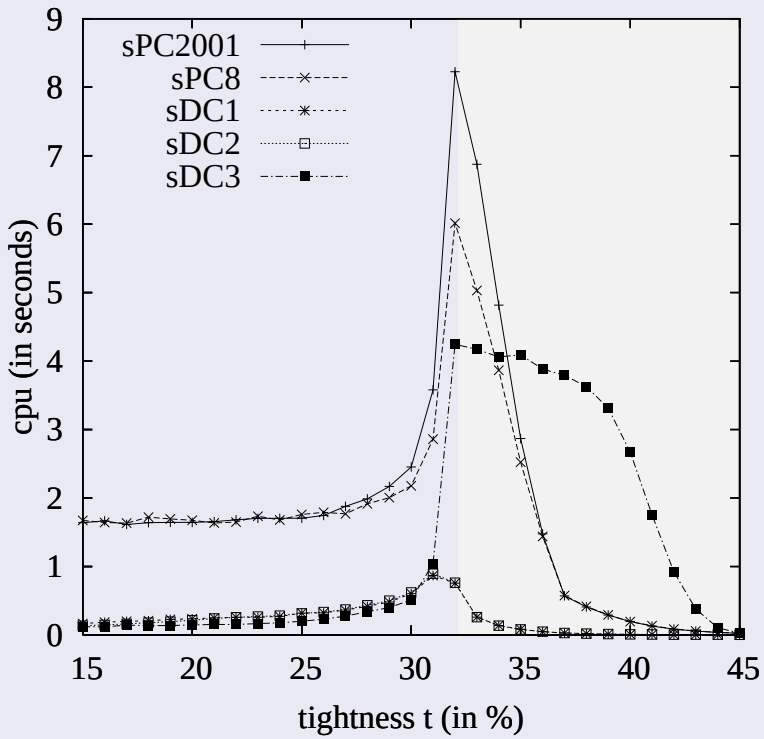
- Complexité spatiale est en $O(ed^2) \in O(n^2d^2)$
- Complexité temporelle de sDC-1/sDC-2 en $O(\lambda n^3 d^3) \in O(n^5 d^5)$
- Complexité temporelle de sDC-3 en $O(n^3 d^4)$ — identique à sPC-8 — proche de l'optimal $O(n^3 d^3)$
- Pire des cas :
  - *checkVar*-1 est appelé λ fois
  - une seule instanciation est supprimée à chaque "tour"
  - En pratique, quelques tours suffisent
  - Une grande partie du travail est effectuée dès le premier tour

sDC : complexités

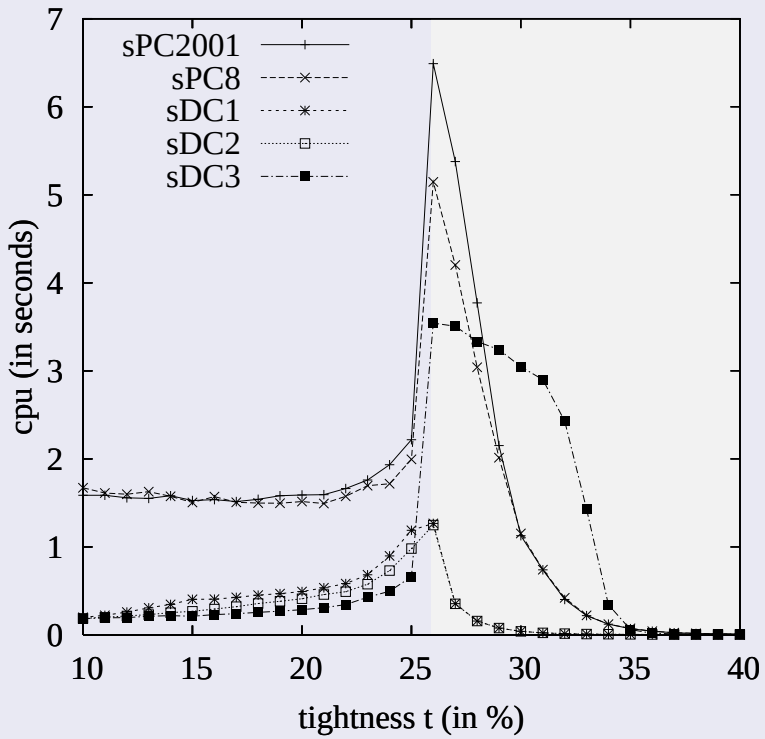
- Complexité spatiale est en $O(ed^2) \in O(n^2d^2)$
- Complexité temporelle de sDC-1/sDC-2 en $O(\lambda n^3 d^3) \in O(n^5 d^5)$
- Complexité temporelle de sDC-3 en $O(n^3 d^4)$ — identique à sPC-8 — proche de l'optimal $O(n^3 d^3)$
- Pire des cas :
  - *checkVar*-1 est appelé λ fois
  - une seule instanciation est supprimée à chaque "tour"
  - En pratique, quelques tours suffisent
  - Une grande partie du travail est effectuée dès le premier tour
- Appliqué à un réseau déjà sDC/sPC, la complexité dans le pire des cas atteint $O(n^3 d^3)$

Expérimentations sur instances aléatoires

$\langle 50, 10, e, t \rangle$



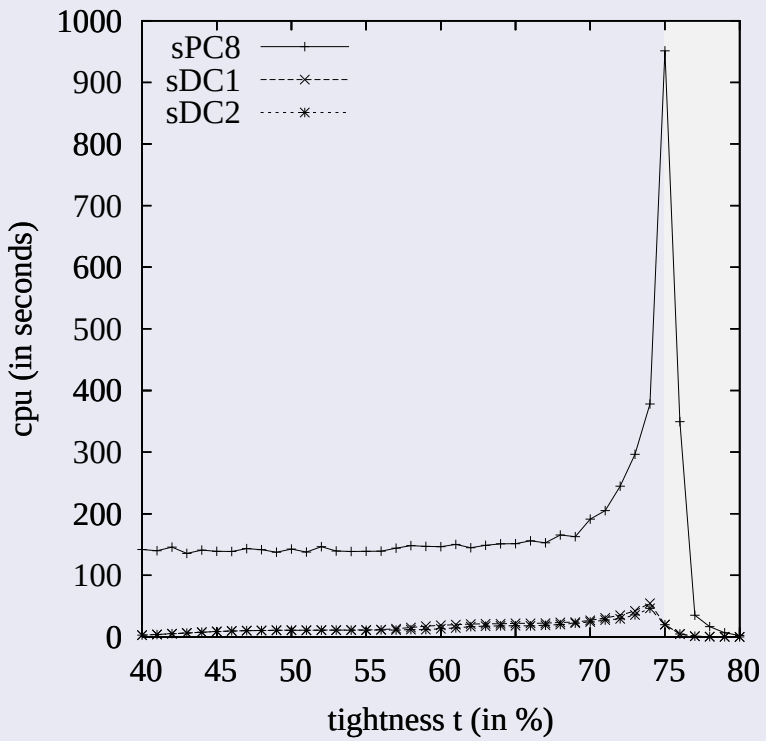
$e = 612$



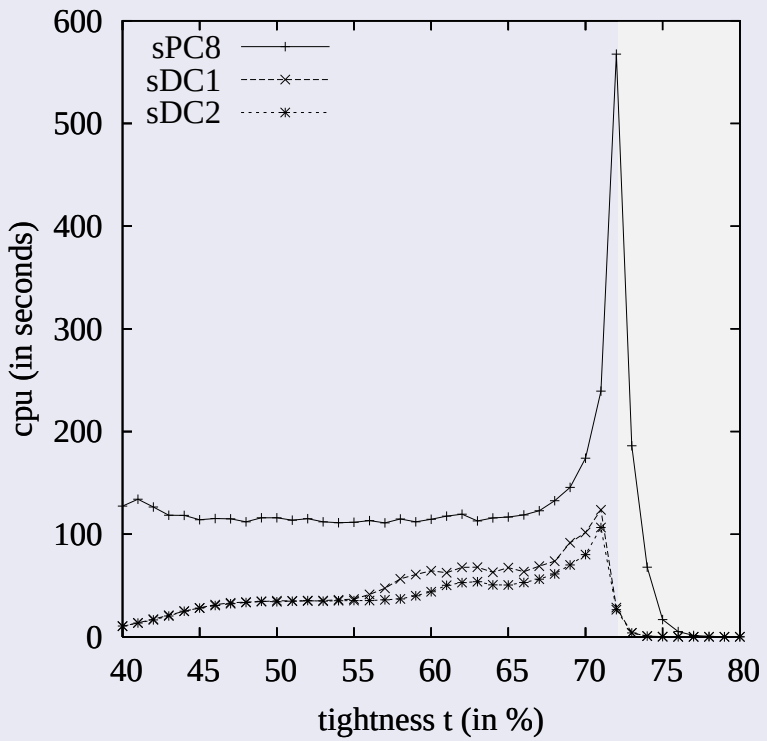
$e = 1225$

Expérimentations sur instances aléatoires

$\langle 50, 90, e, t \rangle$



$e = 612$



$e = 1225$

Instances académiques

Instances		sPC-8	sPC-2001	sDC-1	sDC-2	sDC-3
queens-30	cpu	5,1	5,4	2,2	2,3	2,6
	mem	17	76	17	17	37
queens-50	cpu	50,9		4,6	4,5	5,3
	mem	30	–	22	22	149
queens-100	cpu	1 549		62	58	
	mem	197	–	73	73	–
langford-3-16	cpu	45,5	66,7	4,9	4,4	57,8
	mem	27	612	21	21	129
langford-3-20	cpu	140		11	9,7	198
	mem	43	–	26	26	250
langford-3-30	cpu	1 247		60	50	
	mem	138	–	56	56	–

Consistances conservatives

- Objectif : éviter l'inconvénient majeur de la consistance de chemin/consistance duale

Consistances conservatives

- Objectif : éviter l'inconvénient majeur de la consistance de chemin/consistance duale
- Consistance de chemin conservative [Debruyne 1999] : les instanciations inconsistantes ne sont enregistrées que si les contraintes originales du problème peuvent les prendre en compte

Consistances conservatives

- Objectif : éviter l'inconvénient majeur de la consistance de chemin/consistance duale
- Consistance de chemin conservative [Debruyne 1999] : les instanciations inconsistantes ne sont enregistrées que si les contraintes originales du problème peuvent les prendre en compte

Definition (Consistance Duale Conservative (CDC))

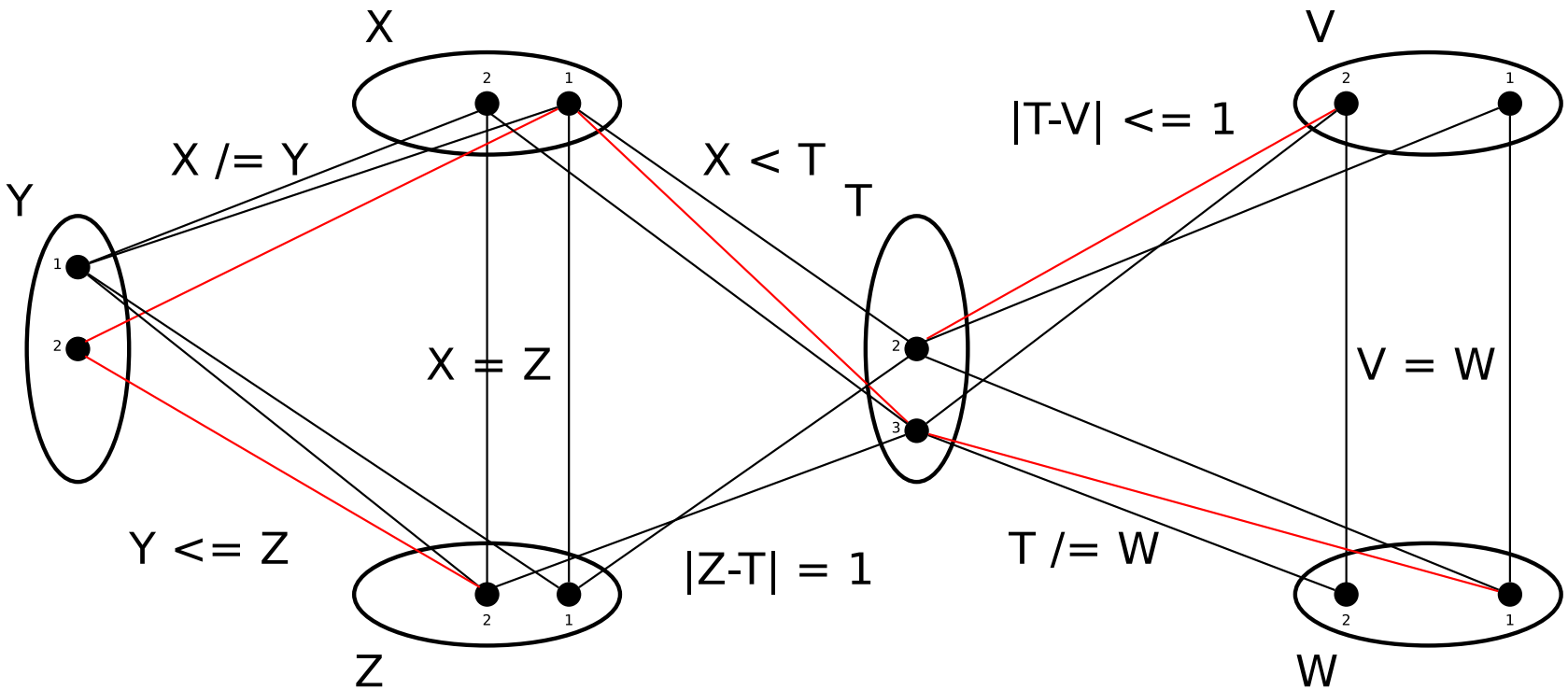
Étant donné un CN $P = (\mathscr{X}, \mathscr{C})$,

- (X, Y) est dual-consistant conservatif si et seulement si
  - soit $\nexists C \in \mathscr{C} \mid \text{vars}(C) = \{X, Y\}$,
  - soit (X, Y) est DC
- P est CDC ssi $\forall (X, Y) \in \mathscr{X}^2 \mid X \neq Y$, (X, Y) est CDC.

Propriété

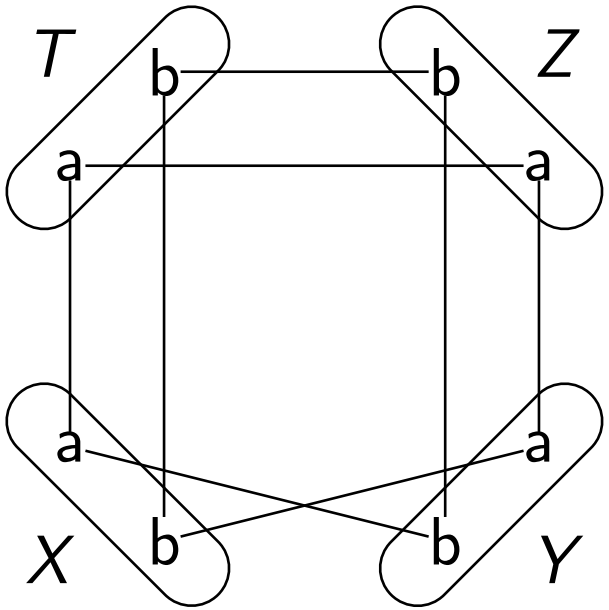
$$\text{CDC} \succ \text{CPC}$$

Établir CPC/CDC



$$\lambda = 16$$

CDC $\succ$ CPC : illustration



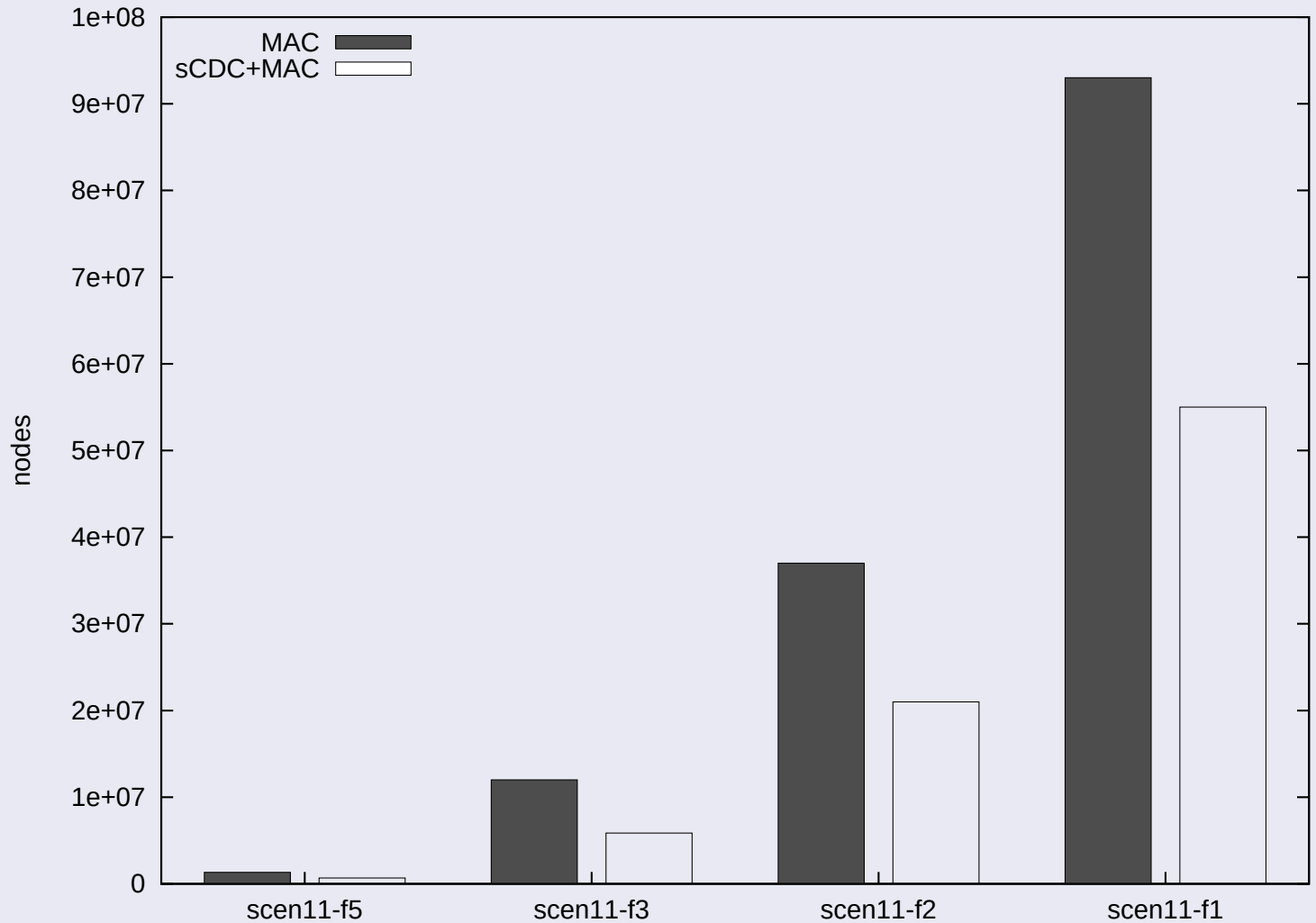
Pas de 3-cliques : ce CN est CPC
Cependant, il n'est pas CDC : $\mathrm{AC}(P|_{X=a}) = \bot$

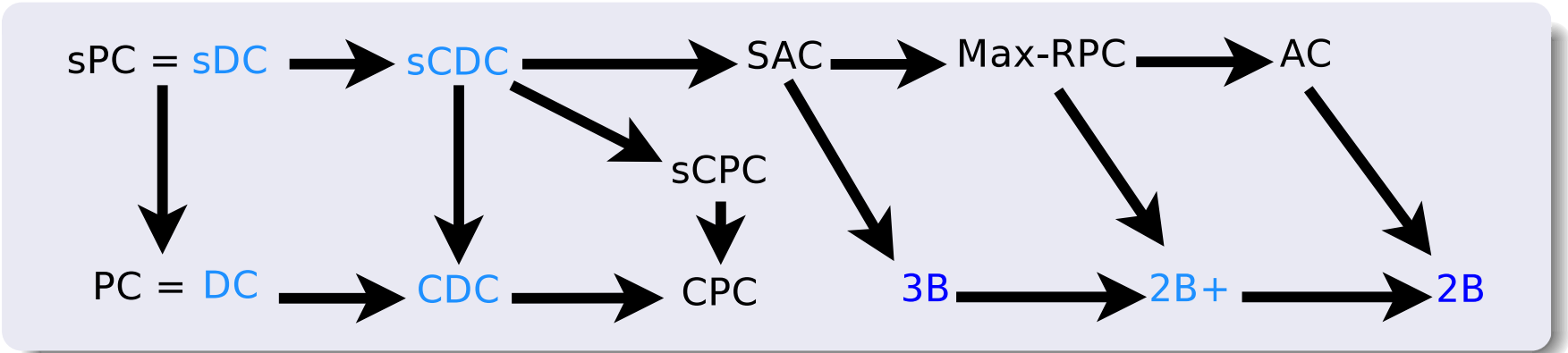
Expérimentations

	AC-3^{rm}	SAC-SDS	sCPC-8	sCPC-2001	sDC-1
knights-50-5 ($K = 10$; $D = 100\%$)					
cpu	12,38	34,43	1759		21,49
mem	5	163	29	–	19
λ	31 331 580	0	0		0
pigeons-50 ($K = 19\ 600$; $D = 100\%$)					
cpu	1,38	2,85	33,82	44,52	2,7
mem	2	12	9	636	5
λ	2 881 200	2 881 200	2 881 200	2 881 200	2 881 200
fapp01-200-4 ($K = 247$; $D = 0{,}5\%$)					
cpu	10,73		16,05	18,63	104,05
mem	15	–	22	254	17
λ	3 612 163		3 317 135	3 317 135	2 117 575
scen-11 ($K = 13\ 775$; $D = 1{,}7\%$)					
cpu	2,87		85,82	78,49	9,78
mem	5	–	22	426	16
λ	5 434 107		4 829 442	4 829 442	4 828 650

Expérimentations

Impact de sCDC en prétraitement sur MAC




sPC = sDC
sCDC
SAC
Max-RPC
AC
sCPC
PC = DC
CDC
CPC
3B
2B+
2B

Plan

Plan

MGAC(P = ($\mathscr{X}$,$\mathscr{C}$) : CN, *maxBT* : entier): Booléen

si $maxBT < 0$ **alors lever** *Expiration*

si $\mathscr{X} = \emptyset$ **alors retourner vrai**

sélectionner X_a tel que $X \in \mathscr{X}$ et $a \in dom(x)$

$P' \leftarrow \text{GAC}(P|_{X=a}, \{X\})$

si $P' \neq \bot \wedge MGAC(P' \backslash X, maxBT)$ **alors**
  retourner vrai

$P' \leftarrow \text{GAC}(P|_{X \neq a}, \{X\})$

retourner $P' \neq \bot \wedge MGAC(P', maxBT - 1)$

MGAC(P = ($\mathscr{X}$,$\mathscr{C}$) : CN, *maxBT* : entier): Booléen

si $maxBT < 0$ **alors lever** *Expiration*

si $\mathscr{X} = \emptyset$ **alors retourner vrai**

sélectionner X_a tel que $X \in \mathscr{X}$ et $a \in dom(x)$

$P' \leftarrow \text{GAC}(P|_{X=a}, \{X\})$

si $P' \neq \bot \wedge MGAC(P' \backslash X, maxBT)$ **alors**
  retourner vrai

$P' \leftarrow \text{GAC}(P|_{X \neq a}, \{X\})$

retourner $P' \neq \bot \wedge MGAC(P', maxBT - 1)$

Heuristiques

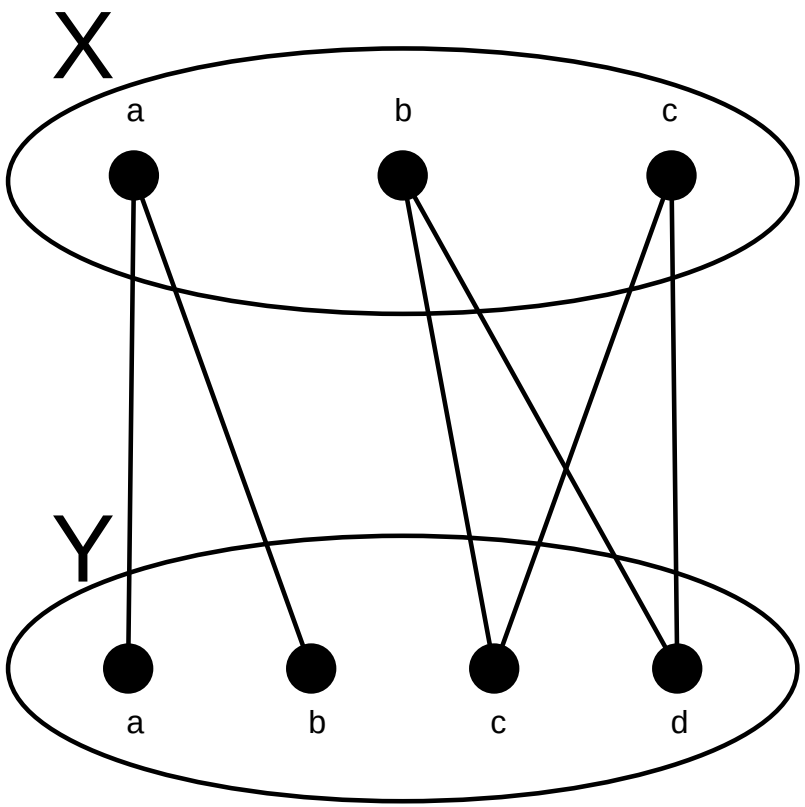
- L'impact des heuristiques de choix de variable est reconnu pour être d'une importance cruciale pour la résolution des CSP

Heuristiques

- L'impact des heuristiques de choix de variable est reconnu pour être d'une importance cruciale pour la résolution des CSP
- Qu'en est-il des heuristiques de choix de valeur, en particulier dans le cadre d'une recherche avec branchement binaire ?

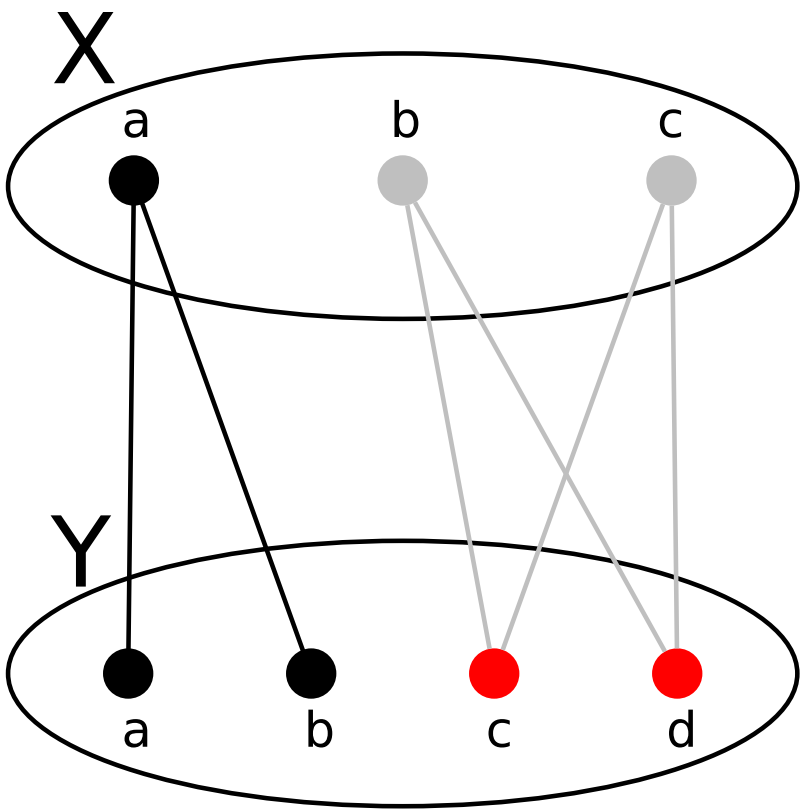
Heuristiques

- L'impact des heuristiques de choix de variable est reconnu pour être d'une importance cruciale pour la résolution des CSP
- Qu'en est-il des heuristiques de choix de valeur, en particulier dans le cadre d'une recherche avec branchement binaire ?



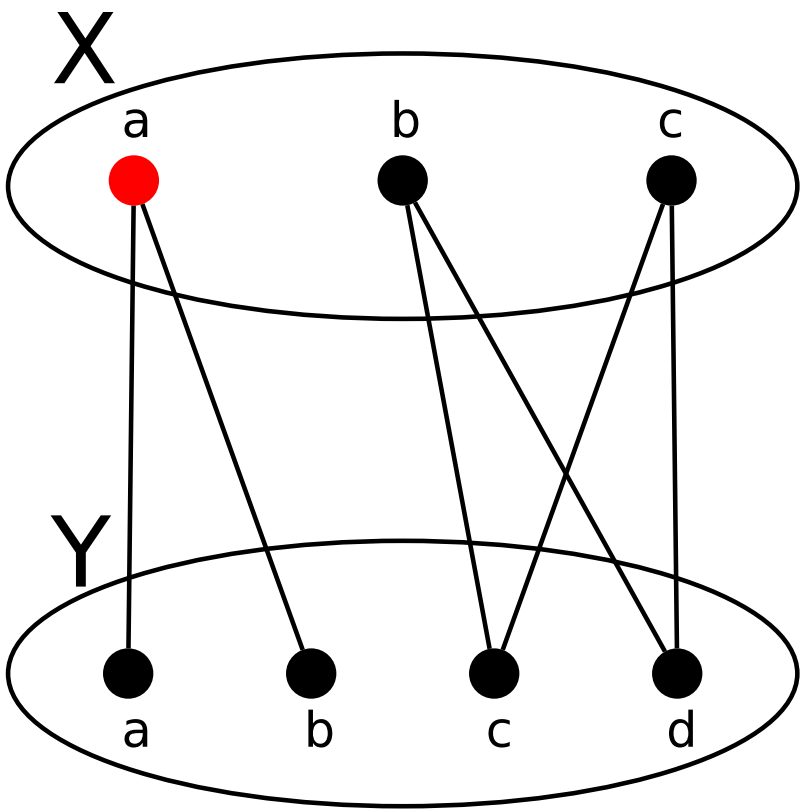
Heuristiques

- L'impact des heuristiques de choix de variable est reconnu pour être d'une importance cruciale pour la résolution des CSP
- Qu'en est-il des heuristiques de choix de valeur, en particulier dans le cadre d'une recherche avec branchement binaire ?



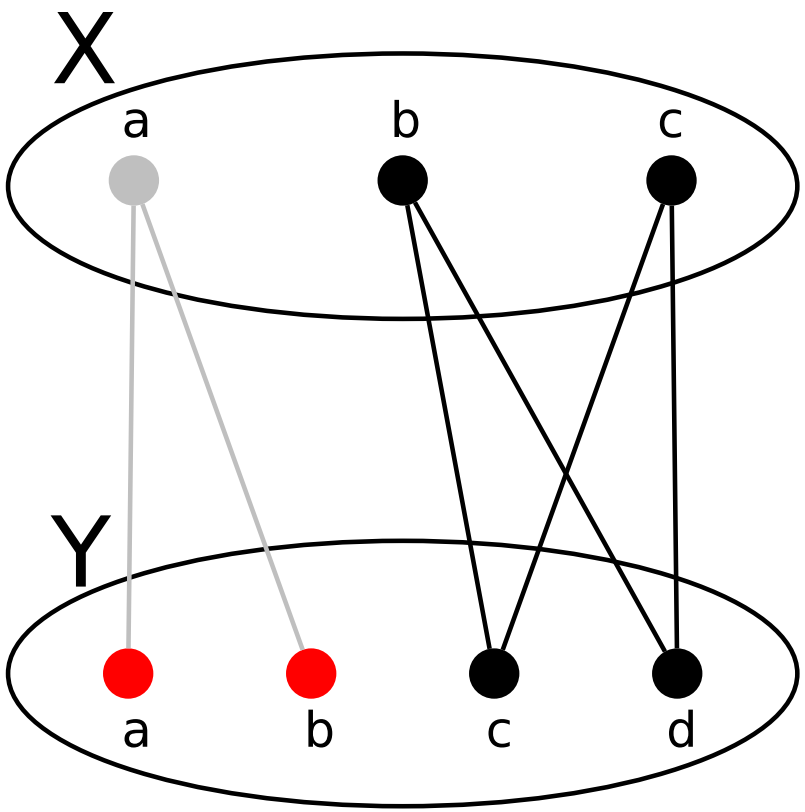
Heuristiques

- L'impact des heuristiques de choix de variable est reconnu pour être d'une importance cruciale pour la résolution des CSP
- Qu'en est-il des heuristiques de choix de valeur, en particulier dans le cadre d'une recherche avec branchement binaire ?



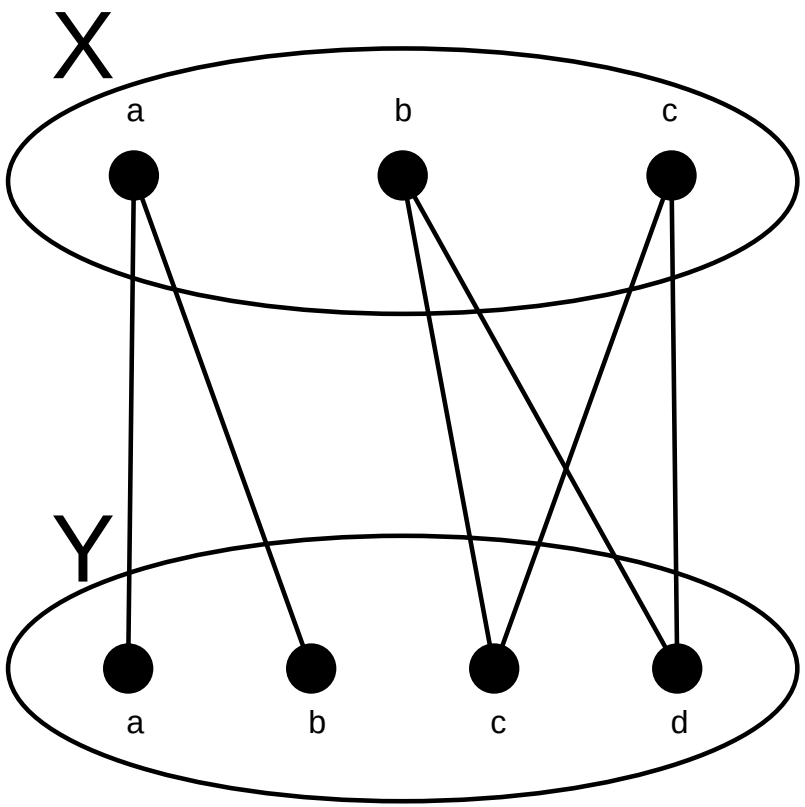
Heuristiques

- L'impact des heuristiques de choix de variable est reconnu pour être d'une importance cruciale pour la résolution des CSP
- Qu'en est-il des heuristiques de choix de valeur, en particulier dans le cadre d'une recherche avec branchement binaire ?



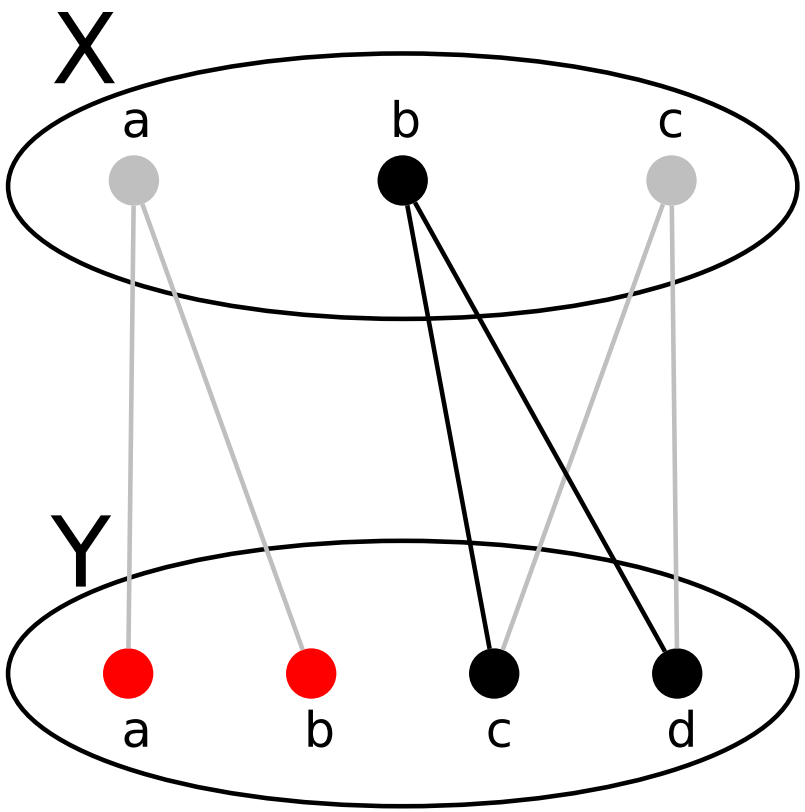
Heuristiques

- L'impact des heuristiques de choix de variable est reconnu pour être d'une importance cruciale pour la résolution des CSP
- Qu'en est-il des heuristiques de choix de valeur, en particulier dans le cadre d'une recherche avec branchement binaire ?



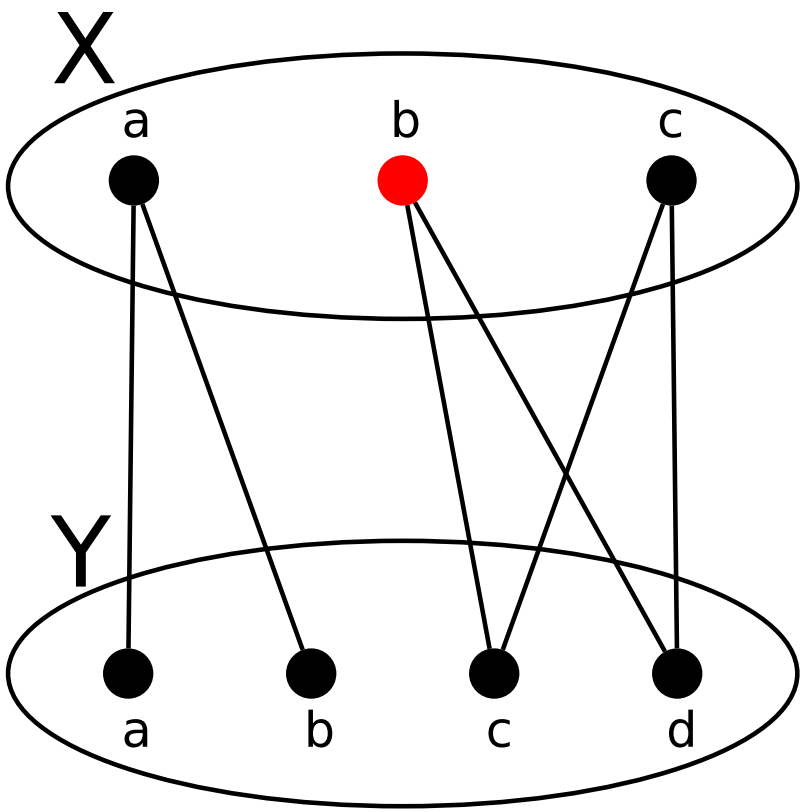
Heuristiques

- L'impact des heuristiques de choix de variable est reconnu pour être d'une importance cruciale pour la résolution des CSP
- Qu'en est-il des heuristiques de choix de valeur, en particulier dans le cadre d'une recherche avec branchement binaire ?



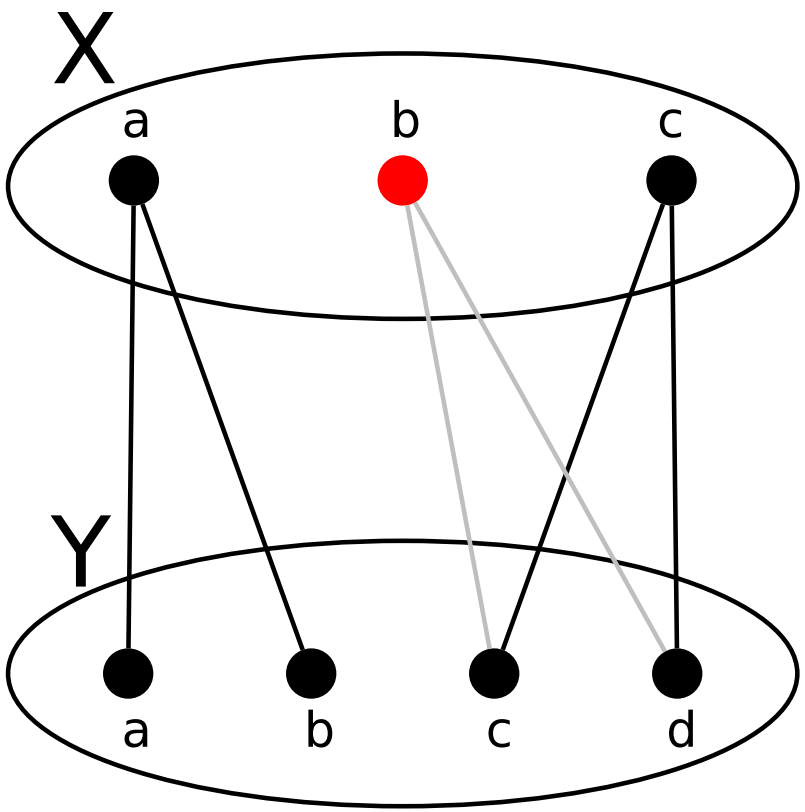
Heuristiques

- L'impact des heuristiques de choix de variable est reconnu pour être d'une importance cruciale pour la résolution des CSP
- Qu'en est-il des heuristiques de choix de valeur, en particulier dans le cadre d'une recherche avec branchement binaire ?



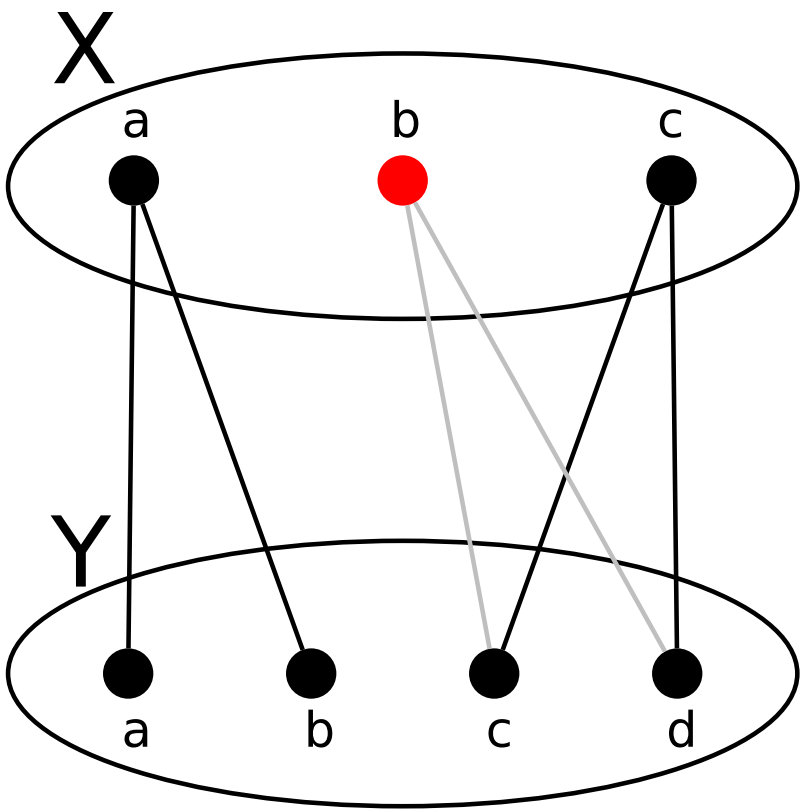
Heuristiques

- L'impact des heuristiques de choix de variable est reconnu pour être d'une importance cruciale pour la résolution des CSP
- Qu'en est-il des heuristiques de choix de valeur, en particulier dans le cadre d'une recherche avec branchement binaire ?



Heuristiques

- L'impact des heuristiques de choix de variable est reconnu pour être d'une importance cruciale pour la résolution des CSP
- Qu'en est-il des heuristiques de choix de valeur, en particulier dans le cadre d'une recherche avec branchement binaire ?



- Stratégie *promise* ou *fail-first* ?

Heuristiques SAT et heuristiques CSP

- Heuristique SAT "Jeroslow-Wang à deux faces"

Heuristiques SAT et heuristiques CSP

- Heuristique SAT "Jeroslow-Wang à deux faces"

$$H_w(\Sigma, x) = \sum_{x \in c} w(|c|) + \sum_{\neg x \in c} w(|c|)$$

Heuristiques SAT et heuristiques CSP

- Heuristique SAT "Jeroslow-Wang à deux faces"

$$H_w(\Sigma, x) = \sum_{x \in c} w(|c|) + \sum_{\neg x \in c} w(|c|)$$

$$\Sigma = (a \vee \neg b \vee c) \wedge (\neg a \vee c) \wedge (\neg a \vee b \vee d) \vee (b \vee \neg d \vee e)$$
$$H_w(\Sigma, a) = w(3) + w(2) + w(3)$$

Heuristiques SAT et heuristiques CSP

- Heuristique SAT "Jeroslow-Wang à deux faces"

$$H_w(\Sigma, x) = \sum_{x \in c} w(|c|) + \sum_{\neg x \in c} w(|c|)$$

$$\Sigma = (a \vee \neg b \vee c) \wedge (\neg a \vee c) \wedge (\neg a \vee b \vee d) \vee (b \vee \neg d \vee e)$$
$$H_w(\Sigma, a) = w(3) + w(2) + w(3)$$

Heuristique $H_w^{\otimes}$: on sélectionne le littéral x ayant le score $H_w(\Sigma, x)$ minimal ou maximal ($\otimes$)
w est une fonction de pondération quelconque

Heuristiques SAT et heuristiques CSP

- Heuristique SAT "Jeroslow-Wang à deux faces"

$$H_w(\Sigma, x) = \sum_{x \in c} w(|c|) + \sum_{\neg x \in c} w(|c|)$$

$$\Sigma = (a \vee \neg b \vee c) \wedge (\neg a \vee c) \wedge (\neg a \vee b \vee d) \vee (b \vee \neg d \vee e)$$
$$H_w(\Sigma, a) = w(3) + w(2) + w(3)$$

Heuristique $H_w^{\otimes}$: on sélectionne le littéral x ayant le score $H_w(\Sigma, x)$ minimal ou maximal ($\otimes$)
w est une fonction de pondération quelconque

- Codages des CSP binaires vers SAT :
  - Codage direct [DeKleer 1989]
  - Codage par supports [Gent 2002]

Lien direct littéral (SAT) $\leftrightarrow$ valeur (CSP)

Heuristiques obtenues de $D\text{-}H_w^{\otimes}$ et $S\text{-}H_w^{\otimes}$

Codage	$H_w^{\otimes}$		**Heuristique**		**Stratégie**
			connue ?	nom	
(D)irect	$\otimes = max$	$w(\alpha) = 1$	O	*max-conflicts*	*fail-first*
	$\otimes = min$	$w(\alpha) = 1$	O	*min-conflicts*	*promise*
(S)upport	$\otimes = max$	$w(\alpha) = 1$	O	*min-conflicts*	*promise*
	$\otimes = min$	$w(\alpha) = 1$	O	*max-conflicts*	*fail-first*
	$\otimes = max$	$w(\alpha) = \alpha$	N	*max-inverse*	*promise*
	$\otimes = min$	$w(\alpha) = \alpha$	N	*min-inverse*	*fail-first*
	$\otimes = max$	$w(\alpha) = 2^{-\alpha}$	N	*max-jw*	*fail-first*

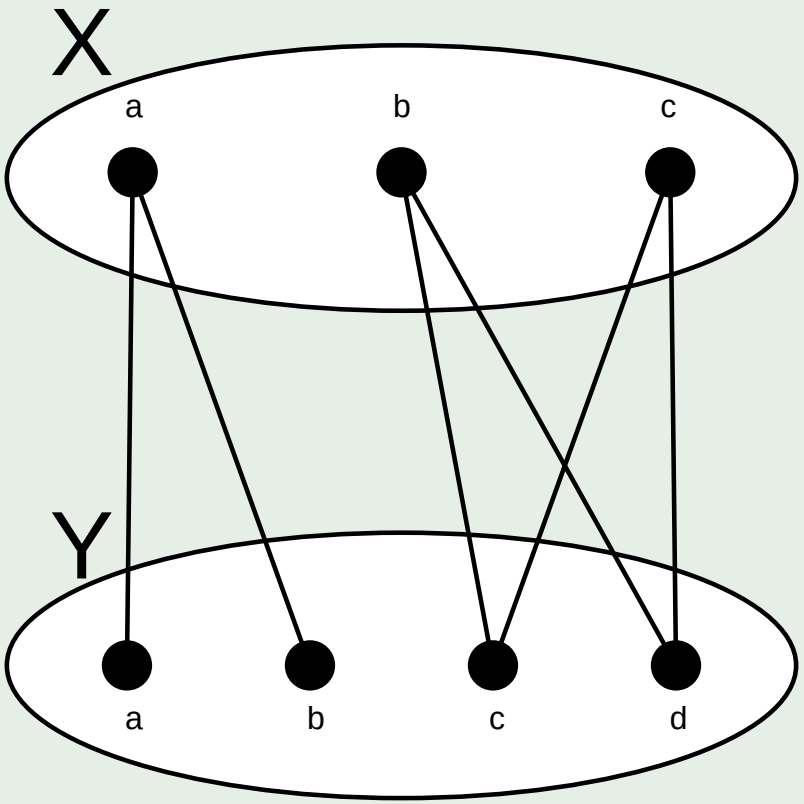
Heuristiques obtenues de $D\text{-}H_w^{\otimes}$ et $S\text{-}H_w^{\otimes}$

Codage	$H_w^{\otimes}$		**Heuristique**		**Stratégie**
			connue ?	nom	
(D)irect	$\otimes = max$	$w(\alpha) = 1$	O	*max-conflicts*	*fail-first*
	$\otimes = min$	$w(\alpha) = 1$	O	*min-conflicts*	*promise*
(S)upport	$\otimes = max$	$w(\alpha) = 1$	O	*min-conflicts*	*promise*
	$\otimes = min$	$w(\alpha) = 1$	O	*max-conflicts*	*fail-first*
	$\otimes = max$	$w(\alpha) = \alpha$	N	*max-inverse*	*promise*
	$\otimes = min$	$w(\alpha) = \alpha$	N	*min-inverse*	*fail-first*
	$\otimes = max$	$w(\alpha) = 2^{-\alpha}$	N	*max-jw*	*fail-first*

Exemple ($H_{inv} = H_{\alpha \to \alpha}$)

$$H_{inv}(P, X_a) = \sum_{C \in \mathscr{C} \mid X \in vars(C)} \left(2 \times |\mathcal{S}p(C, X_a)| + \sum_{Y_b \in \mathcal{S}pV(C, X_a)} |\mathcal{S}p(C, Y_b)| \right)$$

Exemple

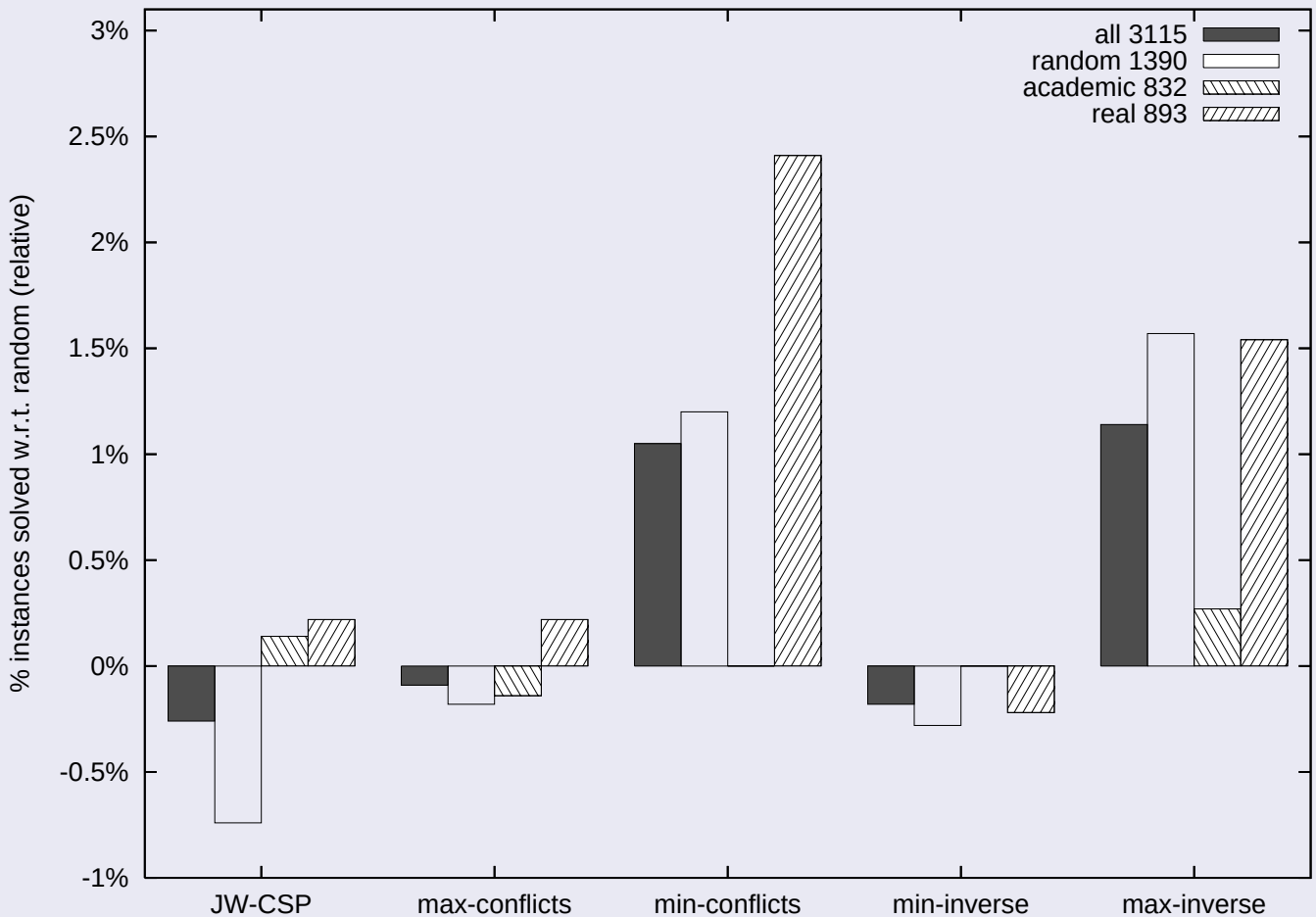


$H_{inv}(P, X_a) = 8$
$H_{inv}(P, X_b) = 10$
$H_{inv}(P, X_c) = 10$

$H_{inv}(P, Y_a) = 4$
$H_{inv}(P, Y_b) = 4$
$H_{inv}(P, Y_c) = 8$
$H_{inv}(P, Y_d) = 8$

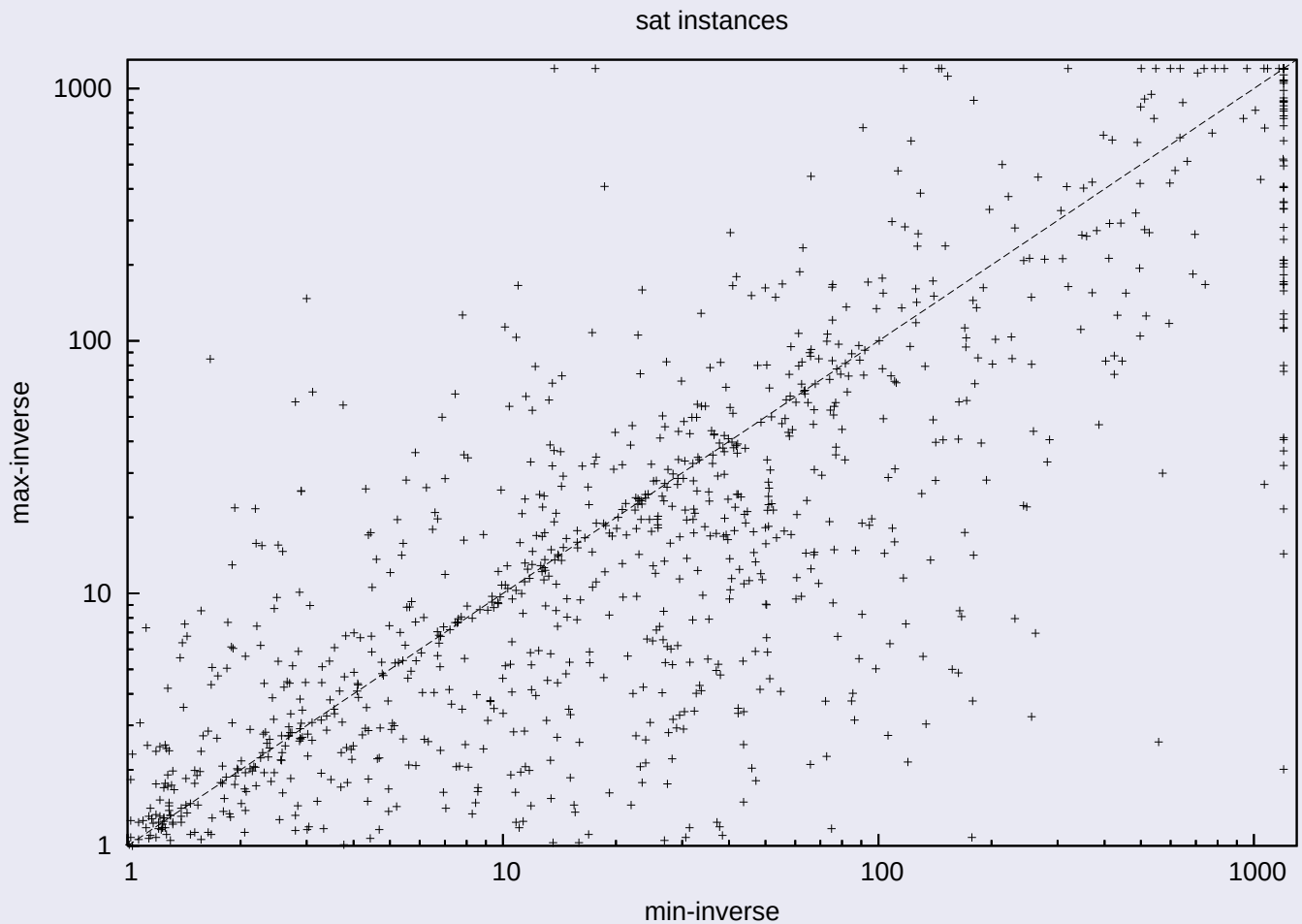
Expérimentations

Écart relatif du nombre d'instances résolues par les différentes heuristiques par rapport à l'heuristique *random* (*timeout* = 1200*s*



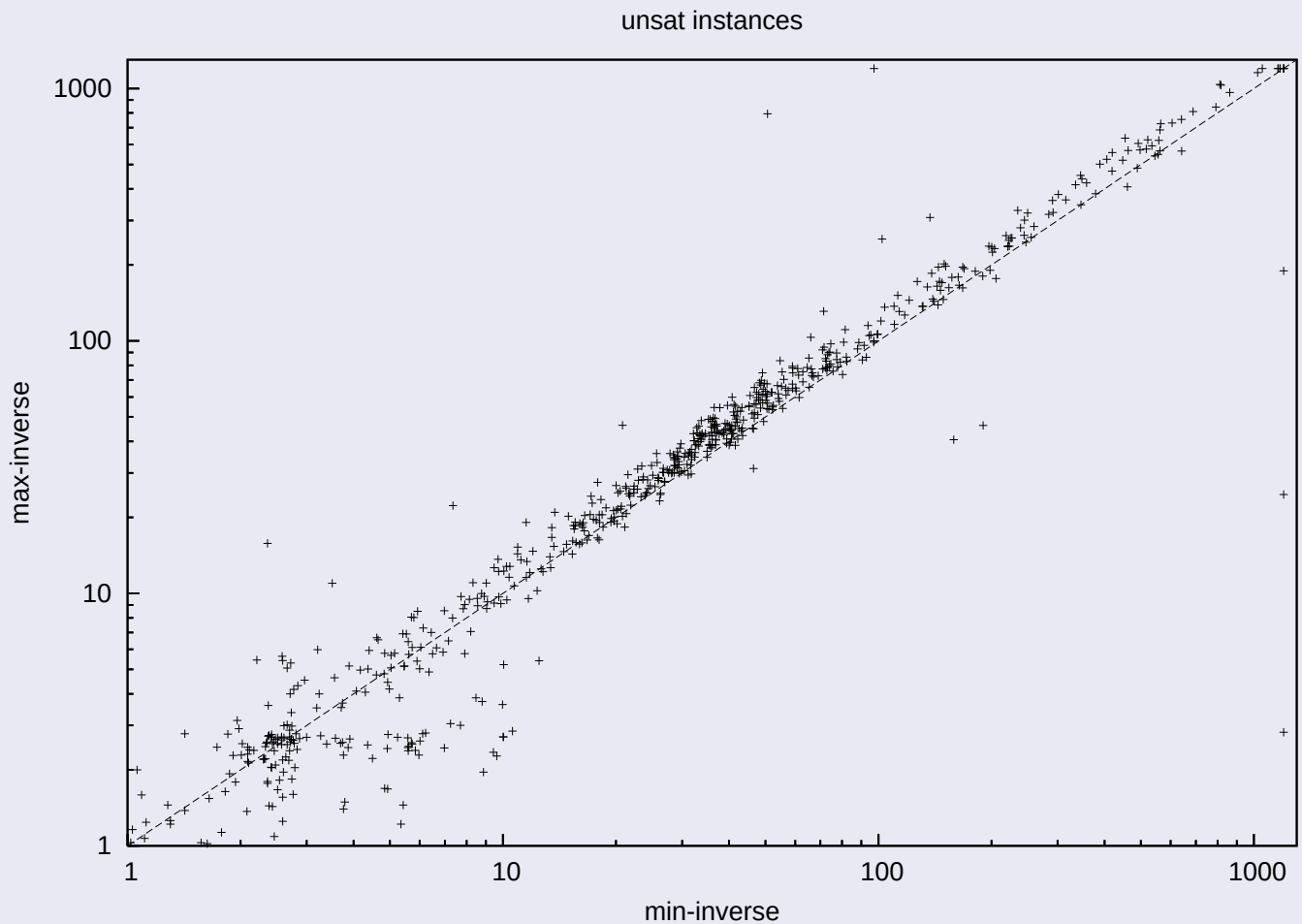
Expérimentations

Instances satisfiables



Expérimentations

Instances insatisfiables



Plan

Algorithmes hybrides ?

- La recherche systématique souffre de mauvais choix initiaux (*thrashing*) $\rightarrow$ importance de l'inférence (GAC) et des heuristiques (*dom/wdeg*)

  $\rightarrow$ *Algorithme MGAC-dom/wdeg*

- Les algorithmes de recherche locale sont incomplets mais souvent très rapides (malgré le problème des minima locaux)

  $\rightarrow$ *Algorithme WMC, basé sur la Breakout Method [Morris 1993]*

Algorithmes hybrides ?

- La recherche systématique souffre de mauvais choix initiaux (*thrashing*) $\rightarrow$ importance de l'inférence (GAC) et des heuristiques (*dom/wdeg*)

  $\rightarrow$ *Algorithme MGAC-dom/wdeg*

- Les algorithmes de recherche locale sont incomplets mais souvent très rapides (malgré le problème des minima locaux)

  $\rightarrow$ *Algorithme WMC, basé sur la Breakout Method [Morris 1993]*

- Faire coopérer les deux principes efficacement est un défi [Selman 1997]

Pondération des contraintes

- L'heuristique de choix de variables *dom/wdeg* se base sur la pondération de contraintes pour identifier les contraintes les plus difficiles à satisfaire

Pondération des contraintes

- L'heuristique de choix de variables *dom/wdeg* se base sur la pondération de contraintes pour identifier les contraintes les plus difficiles à satisfaire
- La méthode de recherche locale Breakout pondère les contraintes falsifiées aux minima locaux pour diversifier sa recherche

Pondération des contraintes

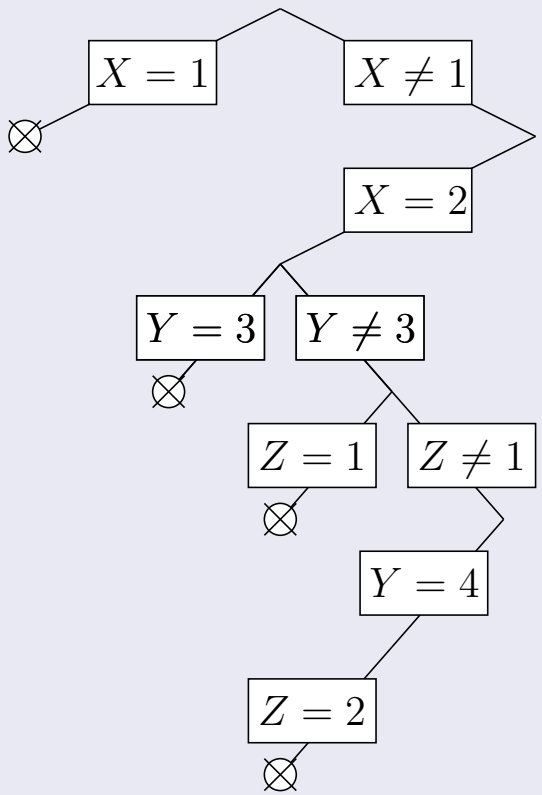
- L'heuristique de choix de variables *dom/wdeg* se base sur la pondération de contraintes pour identifier les contraintes les plus difficiles à satisfaire
- La méthode de recherche locale Breakout pondère les contraintes falsifiées aux minima locaux pour diversifier sa recherche
- La recherche locale et la méthode Breakout sont de bonnes heuristiques pour identifier les noyaux insatisfiables d'un réseau [Mazure et al. 1996, Eisenberg & Faltings 2003]

Pondération des contraintes

- L'heuristique de choix de variables *dom/wdeg* se base sur la pondération de contraintes pour identifier les contraintes les plus difficiles à satisfaire
- La méthode de recherche locale Breakout pondère les contraintes falsifiées aux minima locaux pour diversifier sa recherche
- La recherche locale et la méthode Breakout sont de bonnes heuristiques pour identifier les noyaux insatisfiables d'un réseau [Mazure et al. 1996, Eisenberg & Faltings 2003]
- Partager les poids entre Breakout et *dom/wdeg* ?

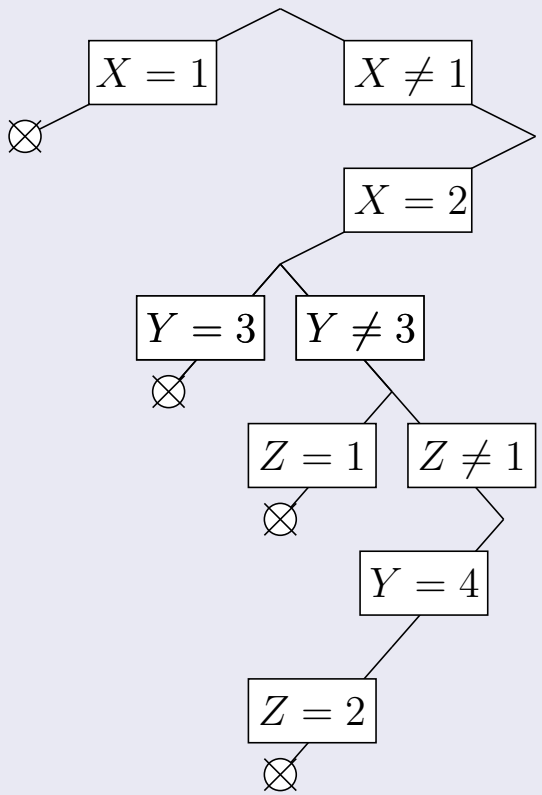
Enregistrer des sous-arbres inconsistants [Lecoutre et al. 2006]

Un arbre de recherche interrompu



Enregistrer des sous-arbres inconsistants [Lecoutre et al. 2006]

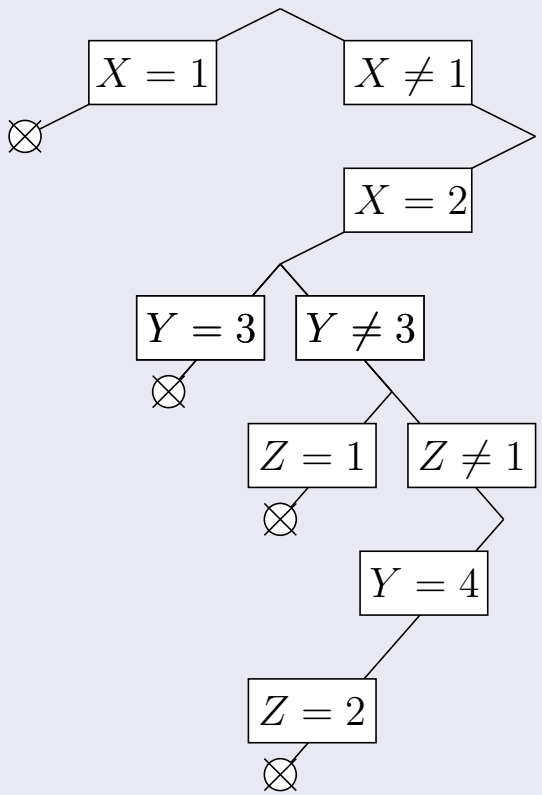
Un arbre de recherche interrompu



$\rightarrow X \neq 1$

Enregistrer des sous-arbres inconsistants [Lecoutre et al. 2006]

Un arbre de recherche interrompu

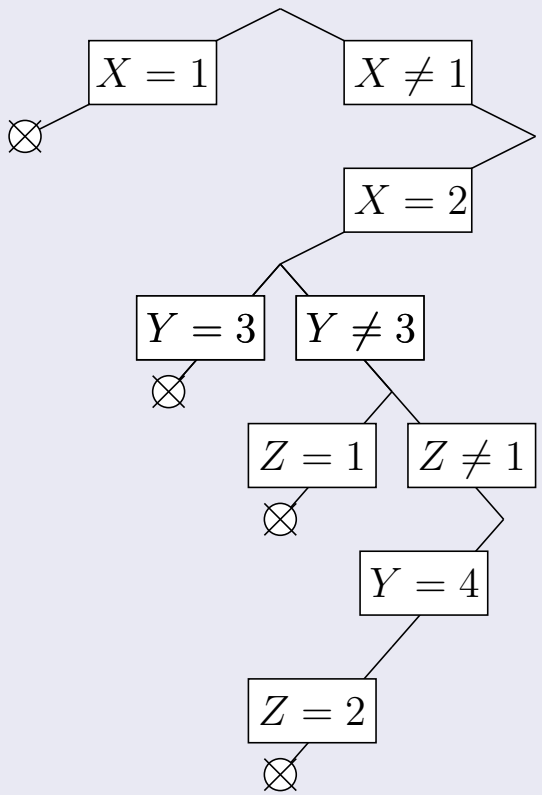


$\rightarrow X \neq 1$

$\rightarrow X \neq 2 \vee Y \neq 3$

Enregistrer des sous-arbres inconsistants [Lecoutre et al. 2006]

Un arbre de recherche interrompu



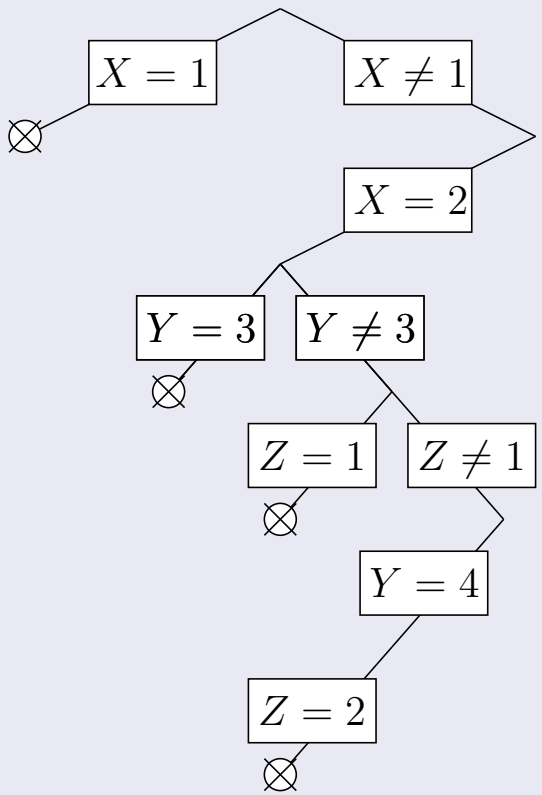
$\rightarrow X \neq 1$

$\rightarrow X \neq 2 \vee Y \neq 3$

$\rightarrow X \neq 2 \vee Z \neq 1$

Enregistrer des sous-arbres inconsistants [Lecoutre et al. 2006]

Un arbre de recherche interrompu



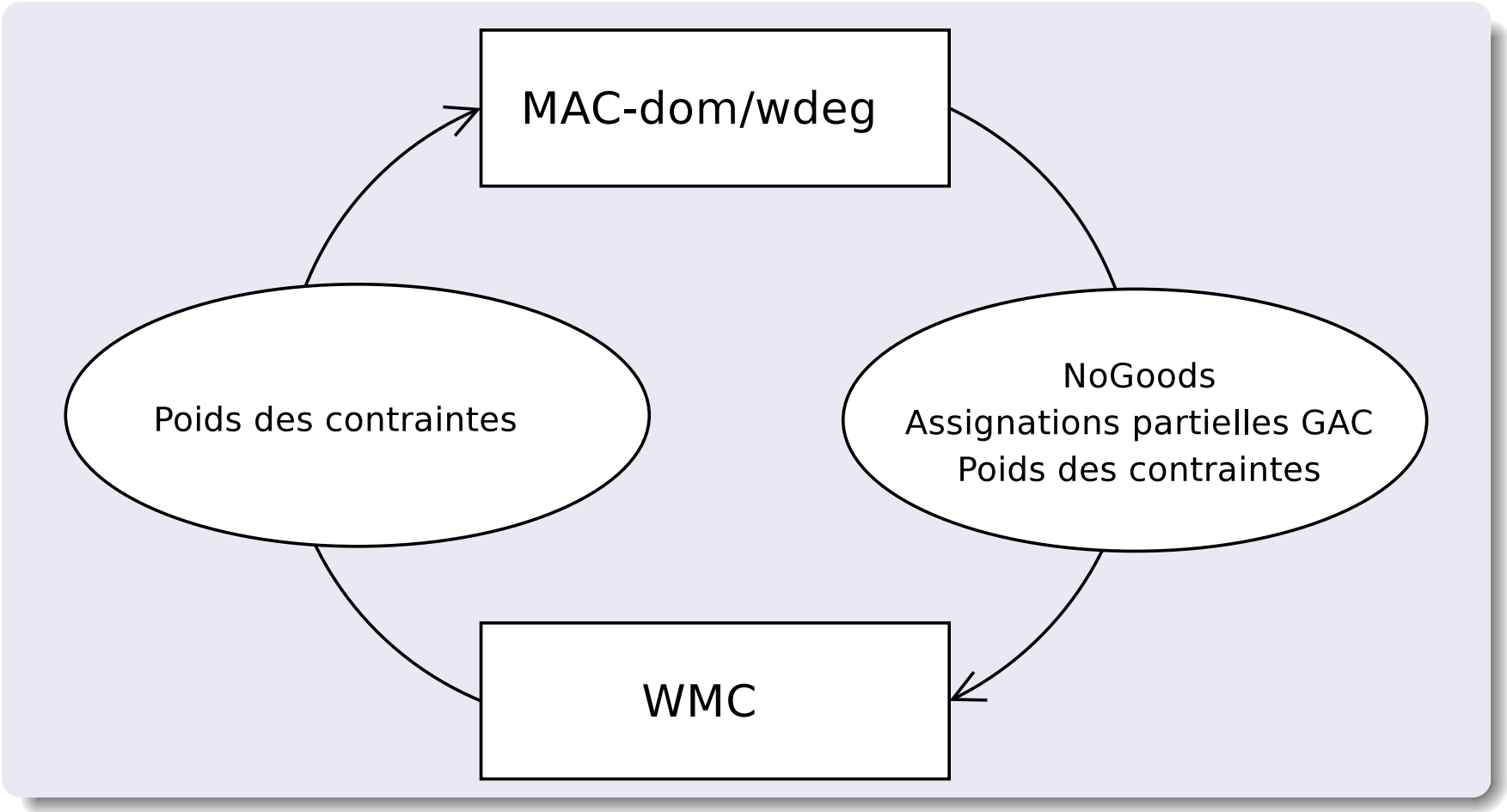
$\rightarrow X \neq 1$

$\rightarrow X \neq 2 \vee Y \neq 3$

$\rightarrow X \neq 2 \vee Z \neq 1$

$\rightarrow X \neq 2 \vee Y \neq 4 \vee Z \neq 2$

Algorithme hybride : schéma général



Expérimentations : instances satisfiables — nb d'instances résolues

instance	*nb*	MGAC	WMC	Hybrid
all interval	13	12	13	13
qcp-qwh-bqwh	253	248	246	253
aim	96	94	86	82
fapp	147	141	132	141
ruler	9	8	6	8
shop	127	103	110	104
par	18	16	10	14
queens	20	17	18	19
ramsey	11	9	11	11
rlfap	25	25	21	24
rand-d>n	94	94	33	42
rand-d<n	672	561	589	582

Problèmes insatisfiables

instance	MGAC		Hybrid			
	assgns	*CPU*	WMC *CPU*	MGAC *assgns*	MGAC *CPU*	total
scen11-f8	13 596	108	126	470	19	146
scen11-f6	40 323	294	129	6 894	62	191
scen11-f5	68 307	713	258	47 692	379	634
os-5-95-2	79 599	106	21	3 148	11	32
os-5-95-5	30 262	52	40	11 372	29	69
os-5-95-8	24 137	55	47	12 543	30	77
qK-50-5-add	12 652	328	188	2 495	79	267
qK-50-5-mul	13 482	452	398	2 495	73	525
qK-80-5-add	$>$ 38 000	$>$ 6 000	855	6 395	603	1 558
qK-80-5-mul	$>$ 25 000	$>$ 6 000	1 666	6 395	972	2 638

Plan

Boîte noire ou boîte de verre

Boîte de verre	Boîte noire
Système de contraintes	Prouveur CSP
Nombreuses fonctionnalités	Une fonctionnalité
Langage de programmation	Variables, Contraintes
Réglage fin	Pas de paramètres
Nécessite une expertise	Define & Solve [Puget 2004]
Complexité	Simplicité
Contraintes spécifiques (globales)	Généricité

Les prouveurs SAT sont des boîtes noires (zChaff, MiniSAT)
La plupart des prouveurs C(S)P sont des boîtes de verre (Gecode, ILOG Solver, Choco...)
De nombreux membres de la communauté CSP s'orientent vers les boîtes noires (compétitions de prouveurs, ILOG CP Optimizer, Minion [Gent et al. 2006], Abscon [Lecoutre et al. 2004])

CSP4J : un logiciel libre

- L'API CSP4J est un logiciel libre, disponible sous licence LGPL 2.1

CSP4J : un logiciel libre

- L'API CSP4J est un logiciel libre, disponible sous licence LGPL 2.1
- Applications de démonstration disponibles sous licence GPL 2.0 :
  - Competiteur et compilateur XCSP 2.0
  - Extracteur de noyaux insatisfiables (MUC)
  - Résolution d'Open Shop
  - Résolution de Sudoku

CSP4J : un logiciel libre

- L'API CSP4J est un logiciel libre, disponible sous licence LGPL 2.1
- Applications de démonstration disponibles sous licence GPL 2.0 :
  - Competiteur et compilateur XCSP 2.0
  - Extracteur de noyaux insatisfiables (MUC)
  - Résolution d'Open Shop
  - Résolution de Sudoku
- Résultats encourageants à la Seconde Compétition Internationale de prouveurs CSP : résultats en moyenne inférieurs de 20% au meilleur prouveur de chaque catégorie

Plan

Conclusion

- Plusieurs techniques de résolution pratique du problème *NP*-complet de satisfaction de contraintes
- Techniques totalement génériques ("boîtes noires")

Conclusion

- Plusieurs techniques de résolution pratique du problème *NP*-complet de satisfaction de contraintes
- Techniques totalement génériques ("boîtes noires")
- Inférence :
  - AC-3^{bit} et AC-3^{bit+rm}, optimisations de AC-3 et AC-3^{rm}
    $\longrightarrow$ vers d'autres optimisations des algorithmes classiques ?
    $\longrightarrow$ utilisation plus intensive des résidus / *watched literals* ?
    [Hemery et al. 2006, Gent et al. 2006]

Conclusion

- Plusieurs techniques de résolution pratique du problème *NP*-complet de satisfaction de contraintes
- Techniques totalement génériques ("boîtes noires")
- Inférence :
  - AC-3^{bit} et AC-3^{bit+rm}, optimisations de AC-3 et AC-3^{rm}
    $\longrightarrow$ vers d'autres optimisations des algorithmes classiques ?
    $\longrightarrow$ utilisation plus intensive des résidus / *watched literals* ?
    [Hemery et al. 2006, Gent et al. 2006]
  - Consistances aux bornes (2B, 2B+, 3B) [CPAI 05]
    $\longrightarrow$ intégration d'autres techniques issues des CSP continus ?
    $\longrightarrow$ vers une intégration entre CSP discrets et continus ?

Conclusion

- Plusieurs techniques de résolution pratique du problème *NP*-complet de satisfaction de contraintes
- Techniques totalement génériques ("boîtes noires")
- Inférence :
  - AC-3^{bit} et AC-3^{bit+rm}, optimisations de AC-3 et AC-3^{rm}
    → vers d'autres optimisations des algorithmes classiques ?
    → utilisation plus intensive des résidus / *watched literals* ?
    [Hemery et al. 2006, Gent et al. 2006]
  - Consistances aux bornes (2B, 2B+, 3B) [CPAI 05]
    → intégration d'autres techniques issues des CSP continus ?
    → vers une intégration entre CSP discrets et continus ?
  - Consistance duale :
    - Nouveaux algorithmes pour établir PC [CP 07]
    - Consistance duale conservative [AAAI 07, JFPC 07]

    → autres consistances fortes exploitant la consistance d'arc ?
    → application de (C)DC aux réseaux non-binaires ?
    → utilisation des nogoods pour établir des consistances ?
    → maintenir des consistances fortes pendant la recherche ?

Conclusion

- Heuristiques de recherche :
  - Nouvelles heuristiques issues de SAT [SAT/CP 06, JSAT]
    $\longrightarrow$ meilleure expoitation de Jeroslow-Wang ?
    $\longrightarrow$ autres heuristiques/techniques issues de SAT ?

Conclusion

- Heuristiques de recherche :
  - Nouvelles heuristiques issues de SAT [SAT/CP 06, JSAT]
    $\longrightarrow$ meilleure expoitation de Jeroslow-Wang ?
    $\longrightarrow$ autres heuristiques/techniques issues de SAT ?
  - Hybridation MGAC-*dom*/*wdeg*/WMC [JFPC 07, DP CP 07]
    $\longrightarrow$ recherche locale sur CSP (novelty...) ?
    $\longrightarrow$ plus de statistiques pendant la recherche (contraintes/variables critiques, réparations...) ?
    $\longrightarrow$ améliorer *dom*/*wdeg* ?
    $\longrightarrow$ extraction de MUC/MUST ?

Conclusion

- Heuristiques de recherche :
  - Nouvelles heuristiques issues de SAT [SAT/CP 06, JSAT]
    → meilleure expoitation de Jeroslow-Wang ?
    → autres heuristiques/techniques issues de SAT ?
  - Hybridation MGAC-*dom*/*wdeg*/WMC [JFPC 07, DP CP 07]
    → recherche locale sur CSP (novelty...) ?
    → plus de statistiques pendant la recherche (contraintes/variables critiques, réparations...) ?
    → améliorer *dom*/*wdeg* ?
    → extraction de MUC/MUST ?
- Développement de CSP4J, participation aux compétitions internationales de prouveurs de CSP [CPAI 08]
  → amélioration et diffusion ?
  → plus applications de CSP4J ?