Formules CNF et Graphes

Said Jabbour

CRIL
Université d'Artois, Lens

03 juillet 2007

Le problème **SAT**

- Problème central en informatique et intelligence artificielle
- Tester si un ensemble de clauses booléennes est satisfiable ou non est un problème NP-Complet (Cook-71)

Résolution du problème **SAT**

Nombreuses approches pour résoudre des instances difficiles du problème **SAT**:

- Méthodes complètes (e.g. DPLL-62)
- Méthodes incomplètes basées sur la recherche locale (e.g. Selman,Levesque,Mitchell-92)
- Les meilleurs algorithmes actuels sont basés sur Zchaff (Moskewicz,Madigan,Zhao,Zhang,Malik-01,Een,Biere-05)

Représentation sous forme de graphe

- Plusieurs graphes ont été proposés
- Utilisés pour diffèrents buts
  - Graphe de Visualisation [Sinz 05]
  - Pré-traitement [Brafman 01]
  - Schémas d'apprentissage
  - Graphe d'Implication 2-SAT [Tarjan 79]

Graphe d'implication 2-SAT

- Soit Φ une formule 2-SAT
- Le graphe associé à ϕ est $G_\phi = (S, E)$
  - $S = \{x, \neg x | x \in \mathcal{L}(\phi)\}$
  - $E = \{(\neg x, y), (\neg y, x) | (x \vee y) \in \phi\}$
- Algorithme polynomiale pour résoudre ϕ consiste en
  - Calculer la fermeture transitive de G_ϕ, $G_\phi^c = (S, E')$
  - Vérifier si il existe un litéral $x \in S$ tel que $(x, \neg x) \in E'$ et $(\neg x, x) \in E'$

Représentation sous forme de graphe

Définition (contexte)

Soit c une clause tel que $|c| \geq 2$. Un contexte η_c associé à c est une conjonction de litéraux tel que $\neg\eta_c \subset c$ et $|c - \{\neg\eta_c\}| = 2$ i.e quand η_c est vrai, la clause c devient une clause binaire.

Exemple

Soit $c = (x_1 \vee x_2 \vee x_3 \vee x_4)$.

un contexte possible est $\eta_c = \neg x_3 \wedge \neg x_4$.

c peut être réecrite sous comme : $c = (x_1 \vee \neg\eta_c \vee x_2)$.

- Pour une clause de taille n, on a $n(n-1)/2$ contextes possibles.

Représentation sous forme de graphe

Définition (graphe associé à une formule)

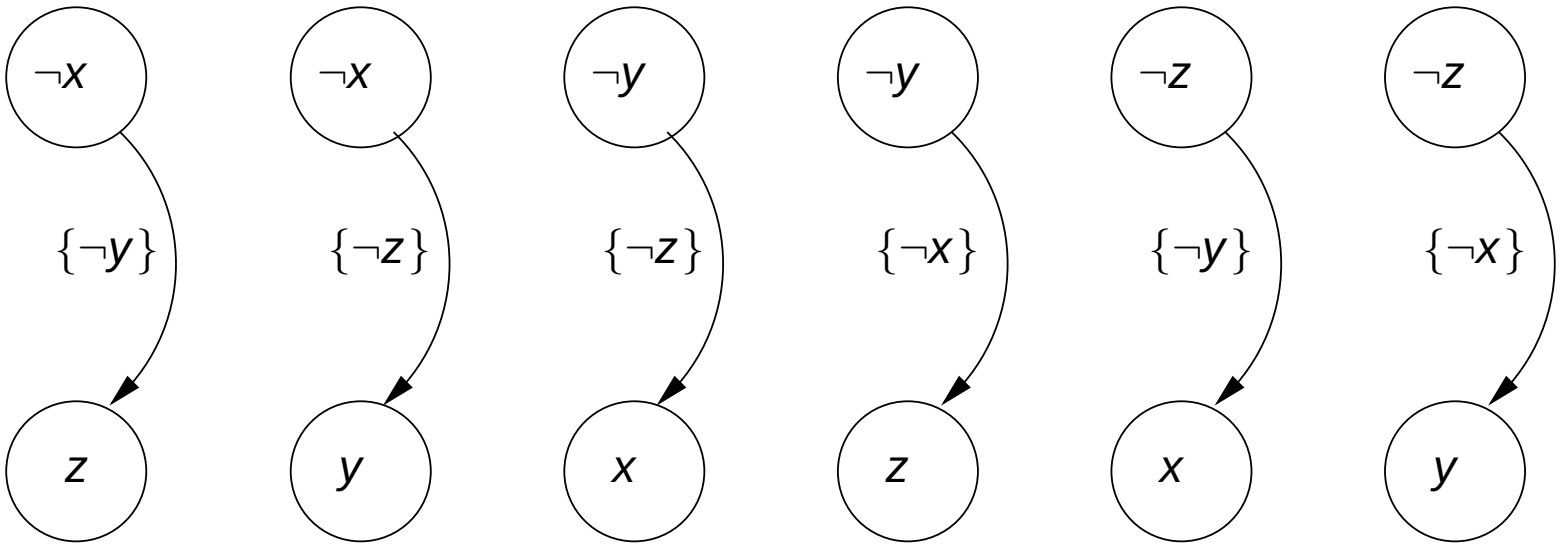
Soit ϕ une formule CNF. On définit $G_\phi = (S, E, v)$ le graphe associé à Φ comme le graphe dirigé étiqueté définit comme suit

- $S = \{x, \neg x | x \in \mathcal{L}(\phi)\}$
- $E = \{a = (\neg x, y)$ *tel que* $\exists c \in \Phi, c = (x \vee \neg \eta_c \vee y)$ *et* $v(a) = \eta_c\}$

Représentation sous forme de graphe

Exemple

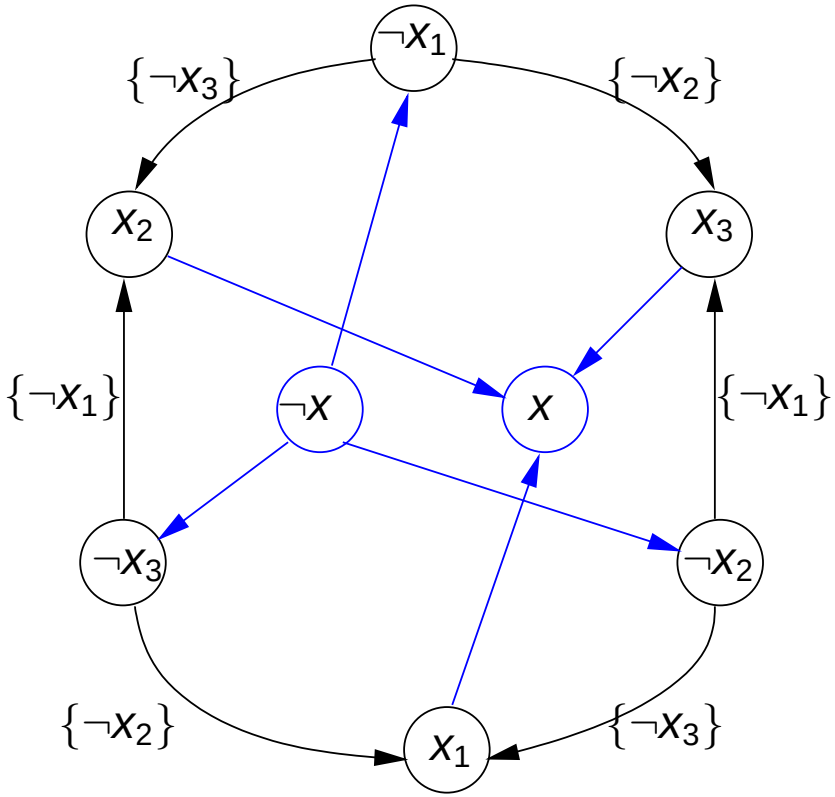
Soit $c = (x \vee y \vee z)$, *il existe 6 façons diffèrentes de représenter c:*



Représentation sous forme de graphe

Exemple

$(x_1 \vee x_2 \vee x_3) \wedge$
$(\neg x_1 \vee x) \wedge (\neg x_2 \vee x) \wedge (\neg x_3 \vee x)$



Graphe Ordonné

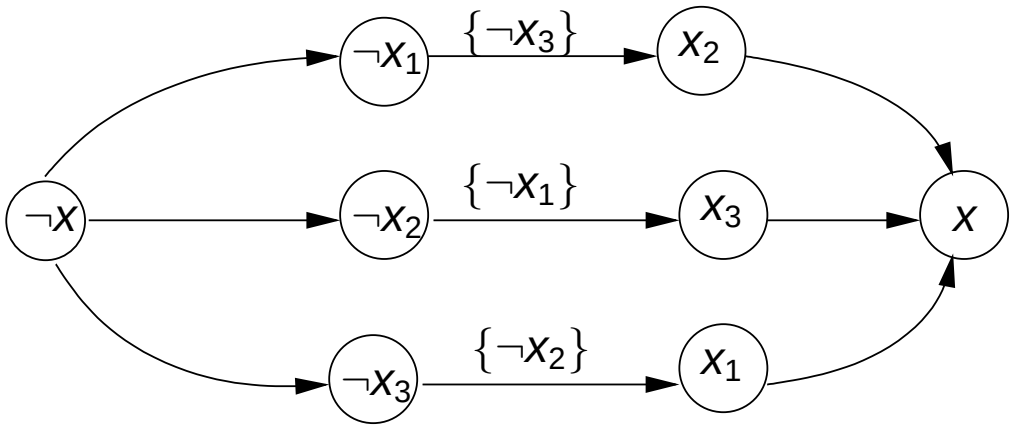
Définition (Graphe ordonné)

Soit ϕ une formule CNF. On définit le graphe ordonné G^{or}_{ϕ} associé à ϕ le graphe restreint définit comme suit :

- $S = \{x, \neg x | x \in \mathcal{L}(\phi)\}$
- $E = \bigcup_{c \in \phi} arc(s)$, *tel que* $arc(s)$ *pour* $c = (x_1 \vee x_2 \vee .. \vee x_n)$ *est définit comme :*
  - *Pour* $(1 \leq i < n)$, $a_i = (x_i, x_{i+1}, v(a_i))$ *et* $v(a_i) = \{x_j | 1 \leq j \leq n \text{ et } j \neq i \text{ et } j \neq i+1\}$, *et*
  - *Pour* $(i = n)$, $a_n = (x_n, x_1, v(a_n))$ *et* $v(a_n) = \{x_2, ..., x_{n-1}\}$

Graphe Ordonné

Exemple



$$(x_1 \vee x_2 \vee x_3)$$
$$(\neg x_1 \vee x)$$
$$(\neg x_2 \vee x)$$
$$(\neg x_3 \vee x)$$

Résolution / fermeture transitive

Définition

Soit $G_\phi = (S, A, v)$ *le graphe associé à* ϕ*,*

- $a_1 = (x, y, v(a_1)) \in A$
- *et* $a_2 = (\neg y, z, v(a_2)) \in A$*.*

On définit $tr(a_1, a_2) = a_3$ *tel que*

$a_3 = (x, z, v(a_1) \cup v(a_2) \backslash \{x, \neg z\})$

Propriété

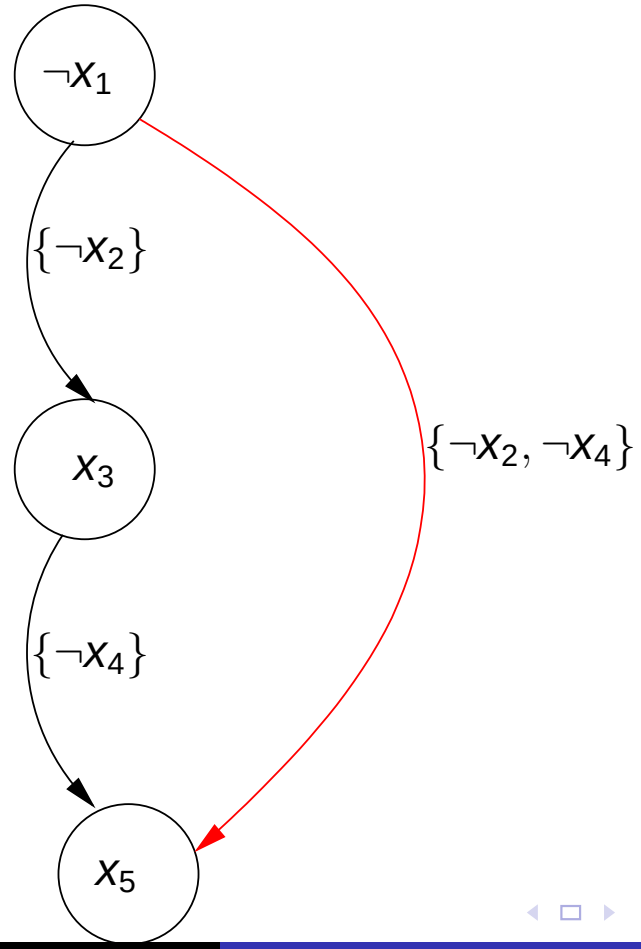
Soit $c_1 = (x \vee \eta_{c_1} \vee y) \in \phi$, $c_2 = (\neg y \vee \eta_{c_2} \vee z) \in \phi$,
$a_1 = (x, y, \eta_{c_1}) \in A$ *et* $a_2 = (\neg y, z, \eta_{c_2}) \in A$*. On a*

$res(c_1, c_2, y) = cla(tr(a_1, a_2))$

Résolution / fermeture transitive

Exemple

Soit ϕ une formule CNF.
$c_1 = (x_1 \vee x_2 \vee x_3)$ et $c_2 = (\neg x_3 \vee x_4 \vee x_5)$ deux clauses de ϕ.



Résolution / fermeture transitive

Définition (Chemin fondamentale)

Soit $p(x,y) = [x = x_{i_1}, x_{i_2}, \ldots, x_{i_k} = y]$ *Un chemin entre* x *et* y*.* $p(x,y)$ *est dit fondamentale s'il satisfait les conditions suivantes :*

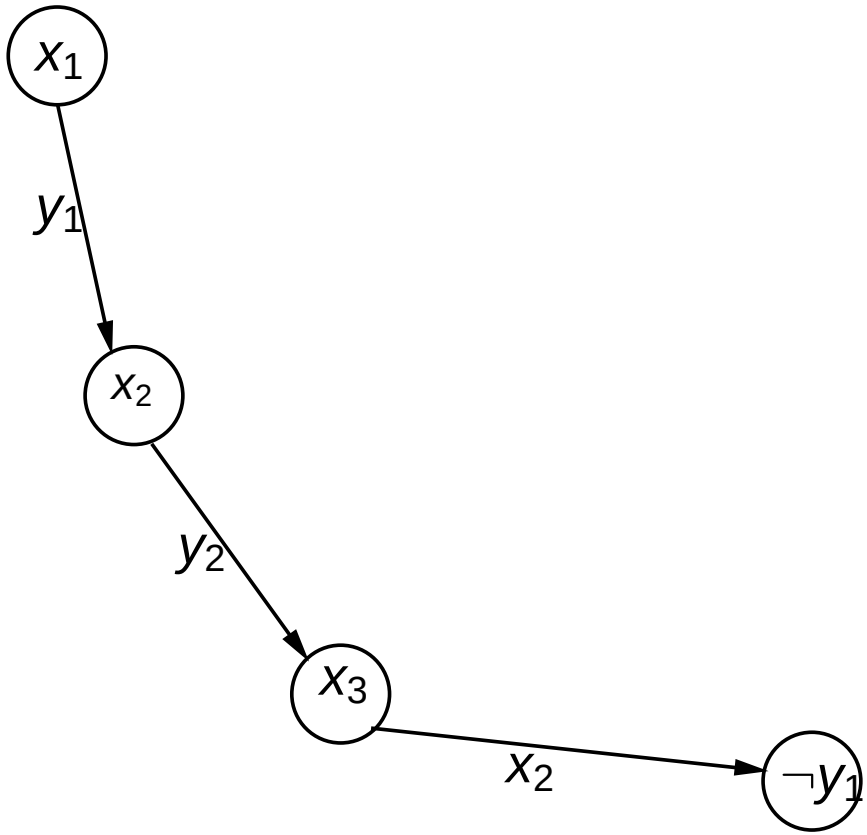
- η_p *ne contient pas un litéral et son opposé.*
- $\neg x \notin \eta_p$ *et* $y \notin \eta_p$

Propriété

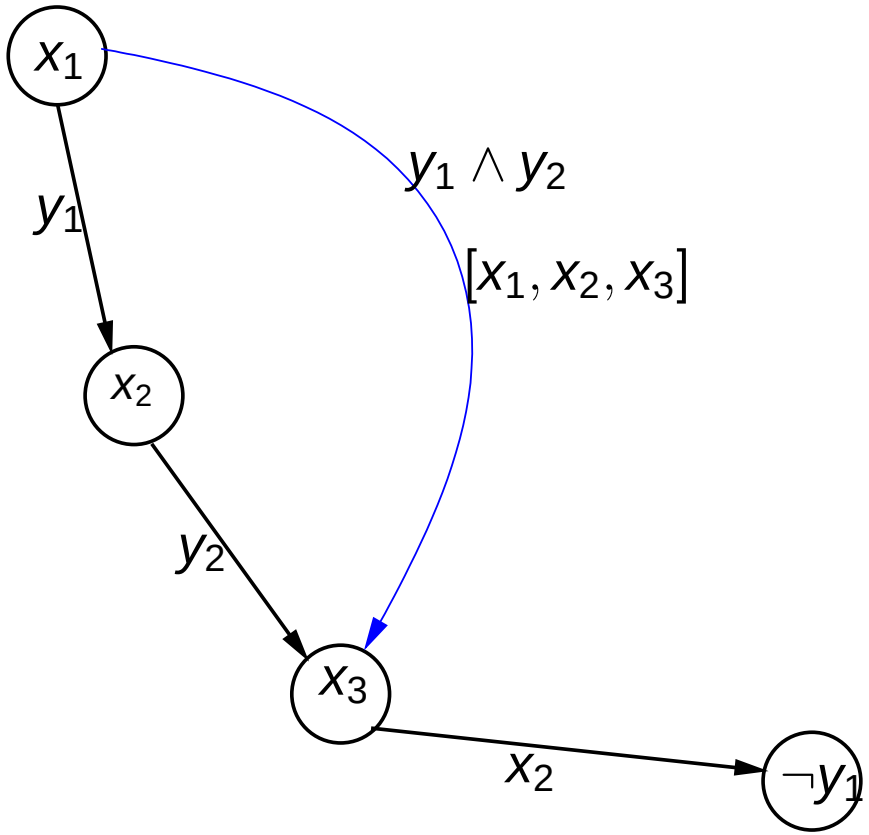
Soit $p(x,y)$ *un chemin.*

$p(x,y)$ est fondamentale ssi $cla(tr(p(x,y)))$ est une clause fondamentale (non tautologique)

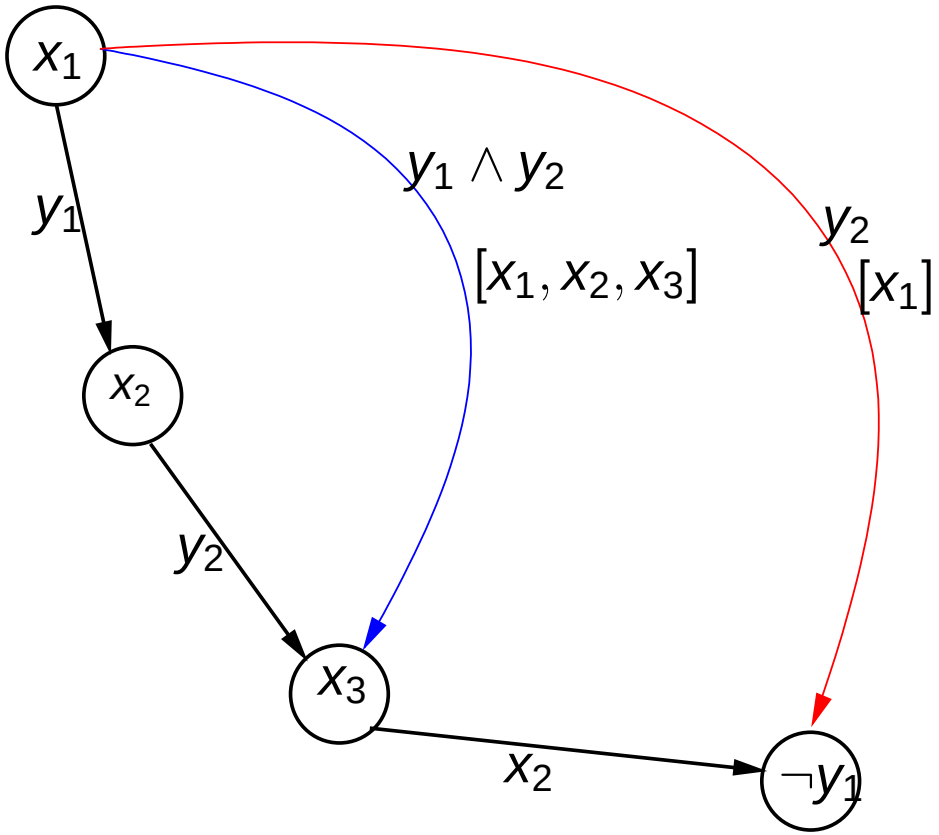
Résolution / fermeture transitive



Résolution / fermeture transitive



Résolution / fermeture transitive



Résolution / fermeture transitive

Définition

Soit $G_\phi = (S, A, v)$ le graphe associé à ϕ.
On définit $tr(G_\phi) = (S, A', v')$ tel que:

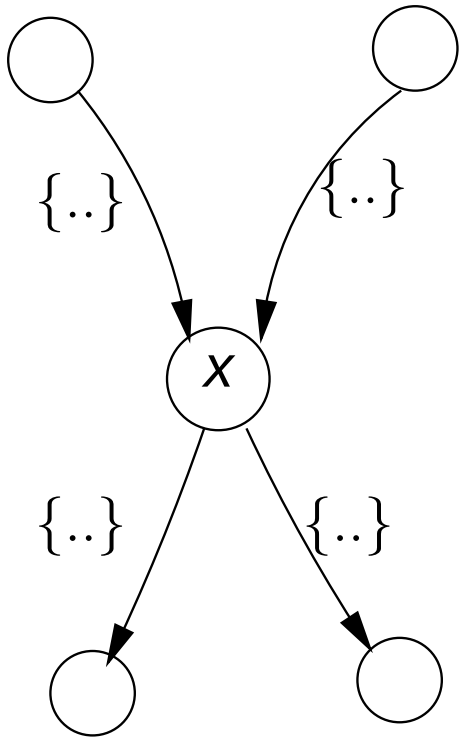
$\forall x \in S$, $\forall y \in S$ *tel que* $\exists p(x, y)$ *est un chemin fondamentale entre x et y,*

$A' = A \cup \{tr(p(x, y))\}$.

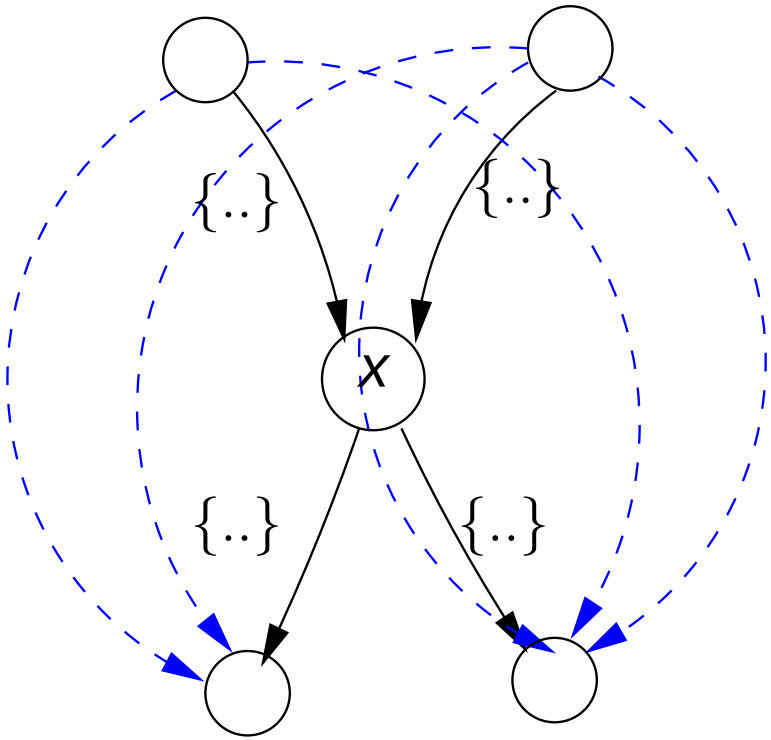
Propriété

Soit ϕ une formule CNF.
ϕ est unsatisfiable ssi $\exists k$ tel que $tr^k(G_\phi) = (S, A', v')$ et $\exists x \in S$ avec $(x, \neg x, \emptyset) \in A'$ et $(\neg x, x, \emptyset) \in A'$

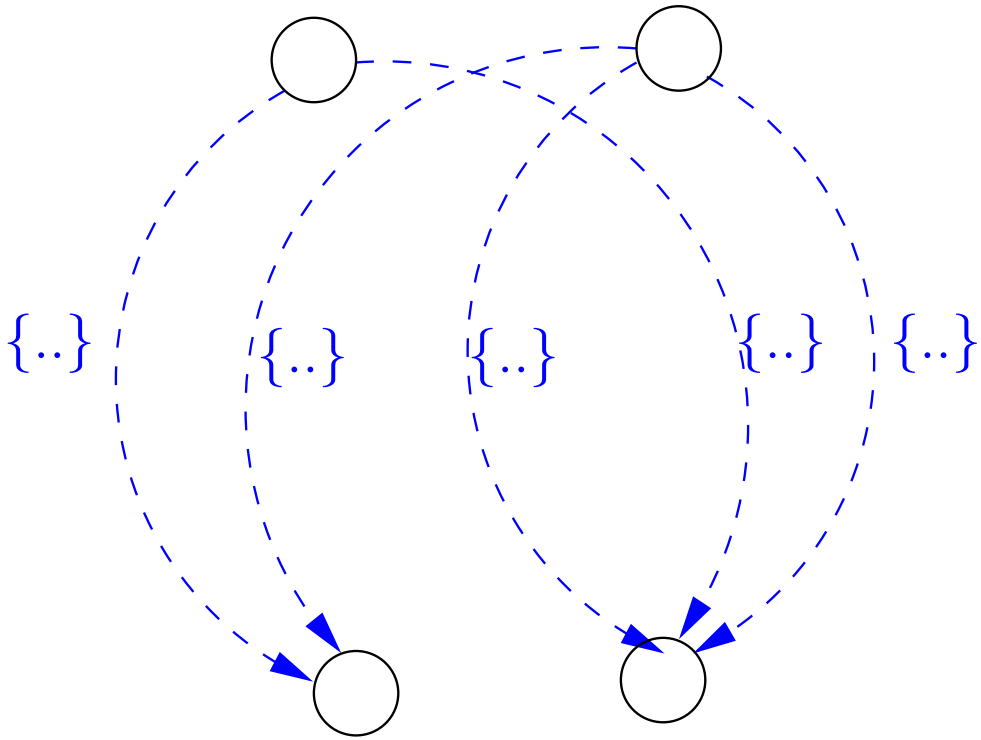
Elimination de litéraux



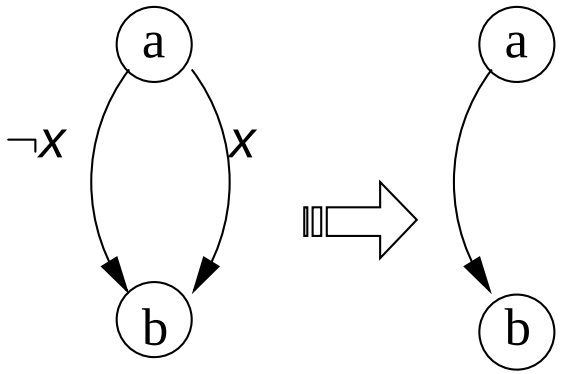
Elimination de litéraux

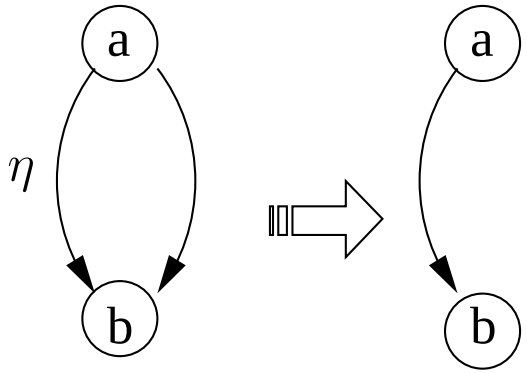


Elimination de litéraux



Réductions





Graphe et Backdoor

Détection des informations structurelles dans **SAT**

Nombreuses approches pour détecter des informations structurelles cachées:

- backbones (Dubois,Dequen-01)
- backdoors (Williams,Gomes,Selman-03, Paris05)
- équivalences (Liberatore-02)
- dépendances fonctionnelles (Gregoire,Ostrowski,Mazure,Sais-04)

Strong Backdoor

Définition

Soit ϕ une formule CNF et B un sous ensemble des variables de ϕ.

B est dit strong backdoor ssi pour toute affectation des variables de B la sous formule ϕ_B est traitable.

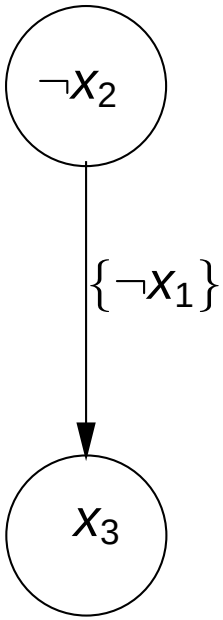
Strong Backdoor 2-SAT

Calcul de l'ensemble strong backdoor 2-SAT

- Chaque clause est représenté par un seul arc.
- Préviligier les littéraux qui apparaissent le plus comme contextes.
- La construction est faite de façon récursive

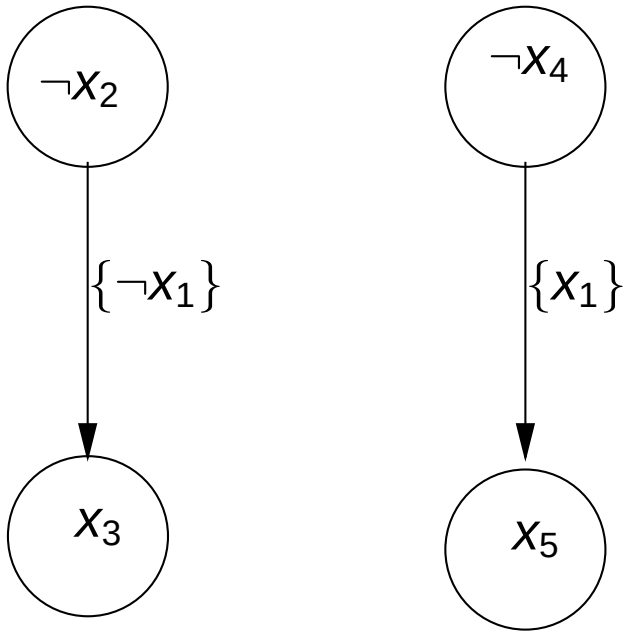
Strong Backdoor 2-SAT

$$\phi = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee x_4 \vee x_5) \wedge (\neg x_2 \vee \neg x_4 \vee \neg x_3) \wedge (x_1 \vee \neg x_2 \vee x_5)$$



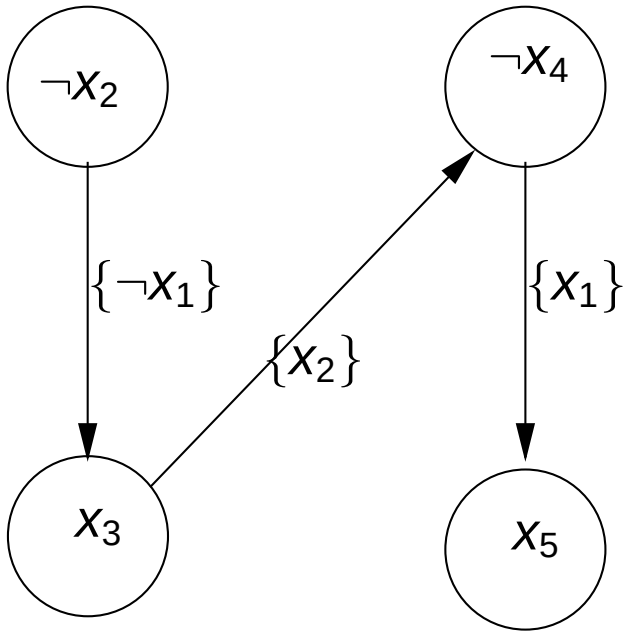
Strong Backdoor 2-SAT

$$\phi = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee x_4 \vee x_5) \wedge$$
$$(\neg x_2 \vee \neg x_4 \vee \neg x_3) \wedge (x_1 \vee \neg x_2 \vee x_5)$$



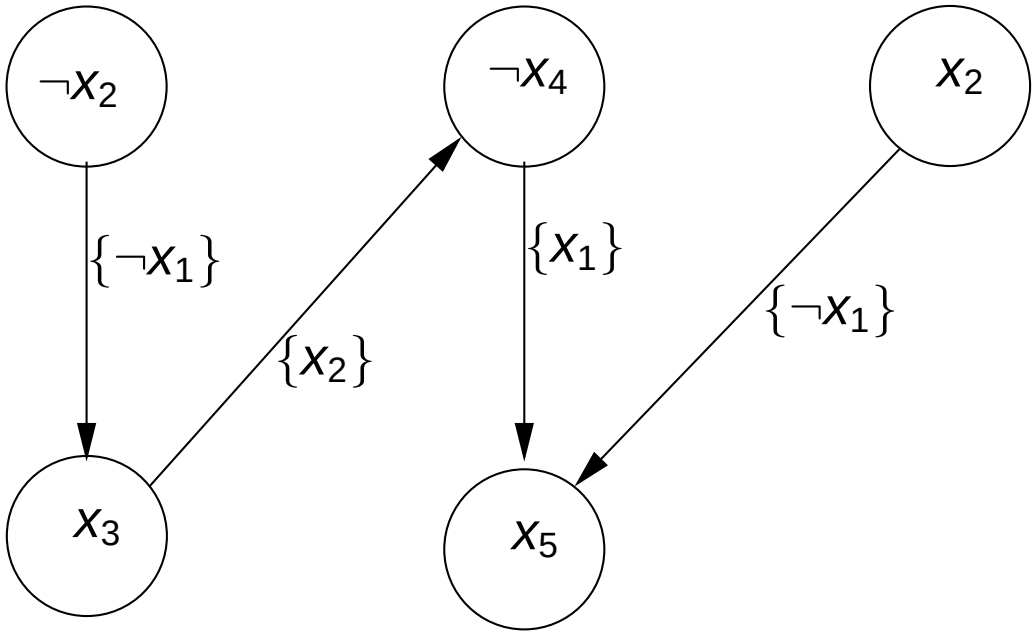
Strong Backdoor 2-SAT

$$\phi = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee x_4 \vee x_5) \wedge$$
$$(\neg x_2 \vee \neg x_4 \vee \neg x_3) \wedge (x_1 \vee \neg x_2 \vee x_5)$$



Strong Backdoor 2-SAT

$$\phi = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee x_4 \vee x_5) \wedge (\neg x_2 \vee \neg x_4 \vee \neg x_3) \wedge (x_1 \vee \neg x_2 \vee x_5)$$



$$B = \{x_1, x_2\}$$

Strong Backdoor 2-SAT

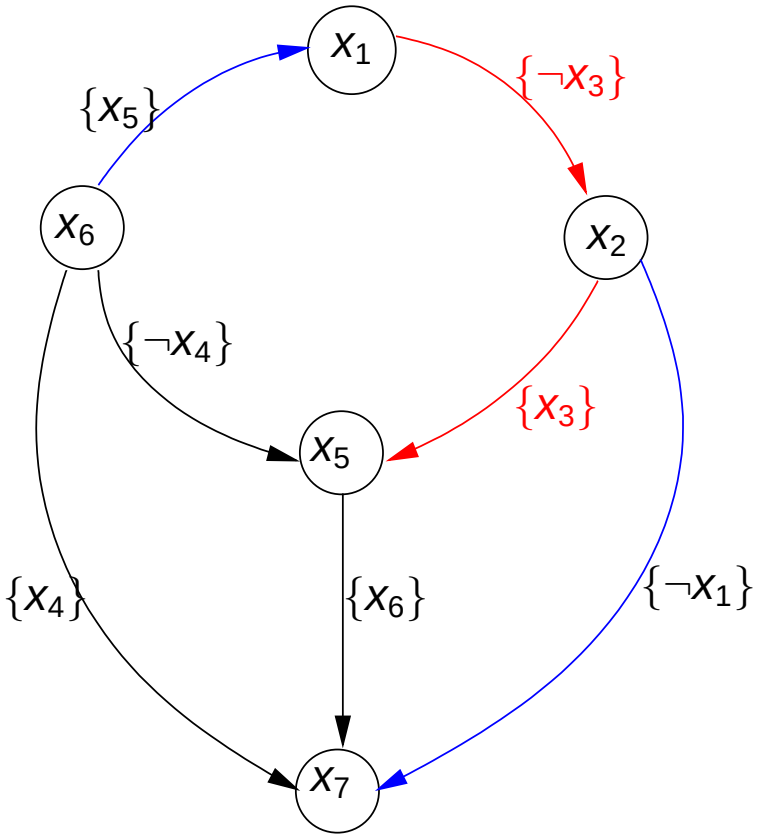
Propriété

Soit B un ensemble 2-SAT strong backdoor d'une formule CNF ϕ, et $b \in B$.

si $\forall c \in \phi$ tel que $b \in c$ or $\neg b \in c$, $|\mathcal{V}(c) \cap B| > |c| - 2$, alors la variable b peut être retirée de B.

Strong Backdoor 2-SAT

Exemple



$$B = \{x_1, x_5, x_6\}$$

Algorithm 1: Approximation d'un ensemble 2-SAT strong backdoor

Input: une formule CNF ϕ
Output: un ensemble 2-SAT strong backdoor B

begin
  Supprimer les clauses binaires de ϕ

  Créer le graphe $G^u_\phi = (S, E)$
  $B = \bigcup_{a \in E} \mathcal{V}(v(a))$
  foreach $c \in \phi$ **do** $nb(c) = |\{x \notin c \cap B\}|$
  foreach $u \in B$ **do**
    bool remove=true

    foreach $(c \in \phi | u \in c \text{ or } \neg u \in c)$ **do**
      if $(nb(c) = 2)$ **then** remove = false

    if *remove* **then** $B = B - \{u\}$
end

Strong Backdoor 2-SAT : Epérimentations

instance	#V	#C	Horn BD	2SAT BD
clauses-2-1966	75528	226124	29357 (38%)	17945 (23%)
equilarge_l2-519	4487	18555	2582 (57%)	367 (8%)
equilarge_l3-520	4496	18591	2595 (57%)	373 (8%)
linvrinv7-566	1274	4166	631 (49%)	467 (36%)
linvrinv5-564	450	1426	221 (49%)	167 (37%)
iso-ukn006-3387	1906	4188	407 (21%)	453 (23%)
iso-icl008-3242	2136	5938	461 (21%)	545 (25%)
mm-2x3-8-8-s.1-476	1148	8088	621 (54%)	132 (11%)
mm-2x3-8-8-sb.1-475	2326	80891	1661 (71%)	926 (39%)
par32-3-c	1325	5294	698 (52%)	199 (15%)
par32-4-c	1333	5326	703 (52%)	202 (15%)
pmg-12-UNSAT-3940	190	632	136 (71%)	77 (40%)
pmg-14-UNSAT-3942	577	1922	408 (70%)	232 (40%)
series18	6410	32421	1665 (25%)	1122 (17%)
strips-gripper-10t19-1143	3390	53008	1475 (43%)	1400 (41%)

Résolution par clause en utilisant le graphe

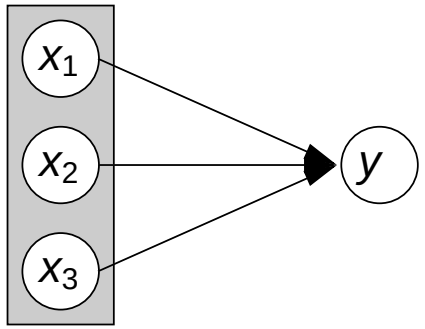
Plusieurs techniques de simplification ont été proposées

- l'élimination de variables **EEn05**
- **Elimination de litéraux purs**
- **Sous formules indépendantes**
- **l'hyper binaire résolution Bacchus03**
- **...**

Hyper Binaire résolution

Soit $\phi = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee y) \wedge (\neg x_2 \vee y) \wedge (\neg x_3 \vee y)$

sous forme de graphe:



On a $\phi \models y$

Résolution par clause en utilisant le graphe

Exemple

Soit $\phi = (x_1 \vee y_1) \wedge (x_2 \vee y_2) \wedge (x_3 \vee y_3) \wedge (\neg y_1 \vee \neg y_2 \vee \neg y_3)$

Ona $\Phi \models (x_1 \vee x_2 \vee x_3)$

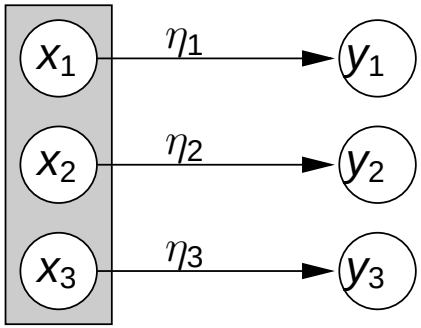
$$x_1 \vee \boxed{y_1}$$

$$x_2 \vee \boxed{y_2}$$

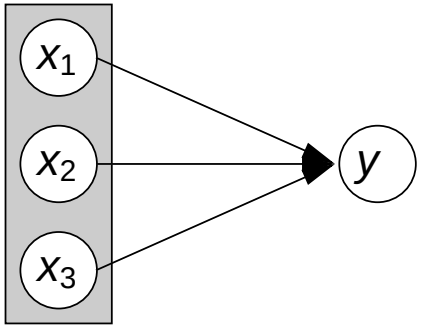
$$x_3 \vee \boxed{y_3}$$

Résolution par clause

Résolution par clause en utilisant le graphe



Résolution par clause : $\phi \models (\neg\eta_1 \vee \neg\eta_2 \vee \neg\eta_3 \vee y_1 \vee y_2 \vee y_3)$



Hyper binaire résolution : $\phi \models y$

Résolution par clause en utilisant le graphe

Comment appliquer la résolution par clause :

- Choisir une clause
- Appliquer récursivement la résolution par clause sur la nouvelle résolvante
- utiliser à chaque fois un arc qui n'a pas été utilisé
- s'arrêter en cas de tautologie.

Algorithm 2: Une étape du pré-traitement

Input: $G^{or}(V, E)$ la représentation en graphe de ϕ
$c \in \phi$
Output: c_{res} une clause résolvante
begin
    $\eta = \emptyset$
    $noeuds = \emptyset$
    foreach $x_i \in c$ **do**
        if $\exists a = (x_i, x_j, v(a)) \in E$ *tel que*
        $x_j \notin \eta$ *et* $v(a) \cap noeuds = \emptyset$ *et* $\neg x_j \notin c$ *et* $\neg v(a) \cap c = \emptyset$
        then $\eta = \eta \cup v(a)$, $noeuds = noeuds \cup \{x_j\}$
        else if $(x_i \notin \eta$ *et* $\neg x_i \notin noeuds)$
        then $noeuds = noeuds \cup \{x_i\}$
        else
        return *null*
        $c_{res} = noeuds \cup \neg\eta$
    return c_{res}
end

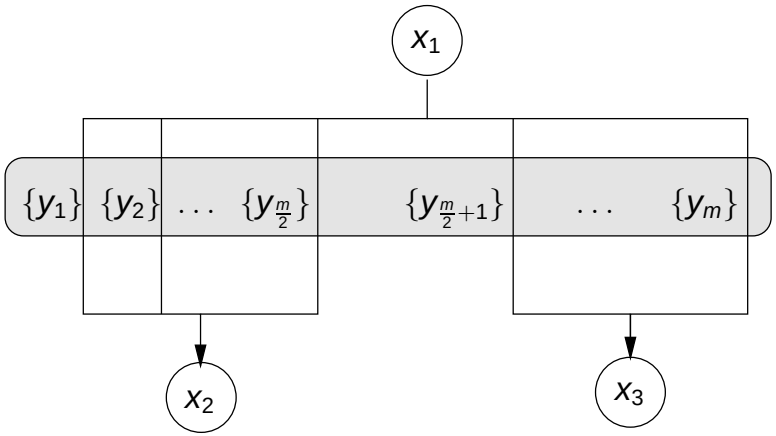
Expérimentations

instance	SAT	preproc	Minisat +preproc	Minisat
mod2-r3b.240-1-2203	Y	0.73	250.35	–
mod2-r3b.270-1-2248	Y	0.76	510.86	–
mod2c-r3b.210-2-2474	Y	1.14	488.36	–
mod2-r3b.250-1.05-2218	Y	0.72	699.16	753.9
fclqcolor-08-05-06-1273	N	0.77	36.84	66.88
fphp-012-011.05-1228	N	0.78	889.1	651.32
gensys-icl009-3135	N	6.63	759.11	–
gensys-ukn006.05-3349	N	5.56	410.73	365.82
gt-020-1305	N	2.33	821.44	–
grid-pbl-1342.05-1342	N	7.06	64.31	153.51
php-012-012-1158	Y	0.63	66.3	210.19
strips-gripper-14t27-1145	Y	48.0	199.36	–
strips-gripper-12t23-1144	Y	26.49	167.44	–
vmpc_34-1958	Y	53.63	60.86	–

Conclusion

- Définir la notion de distance sur le graphe
- Trouver des conditions minimales pour l'implication d'un litéral

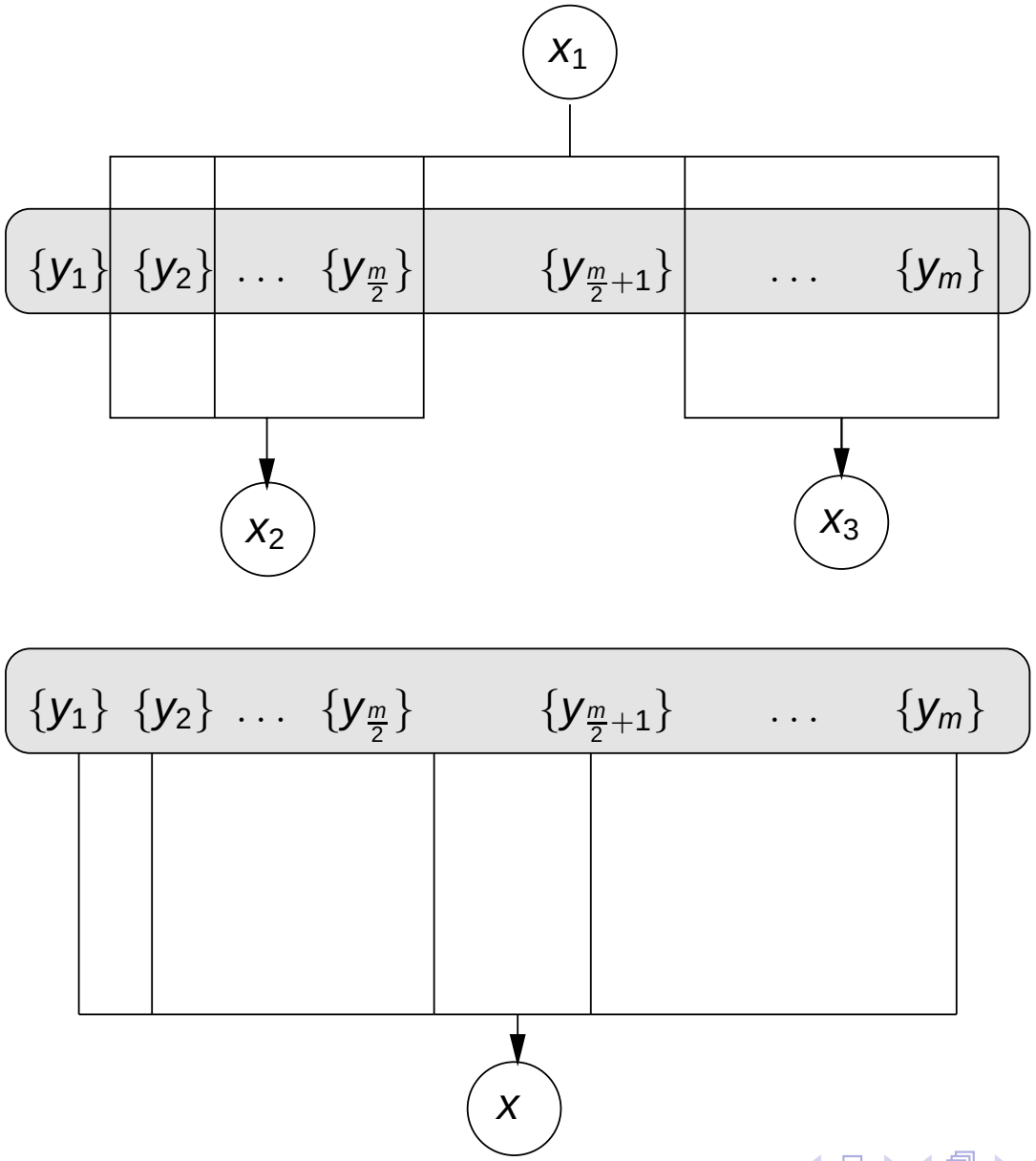
Block Résolution



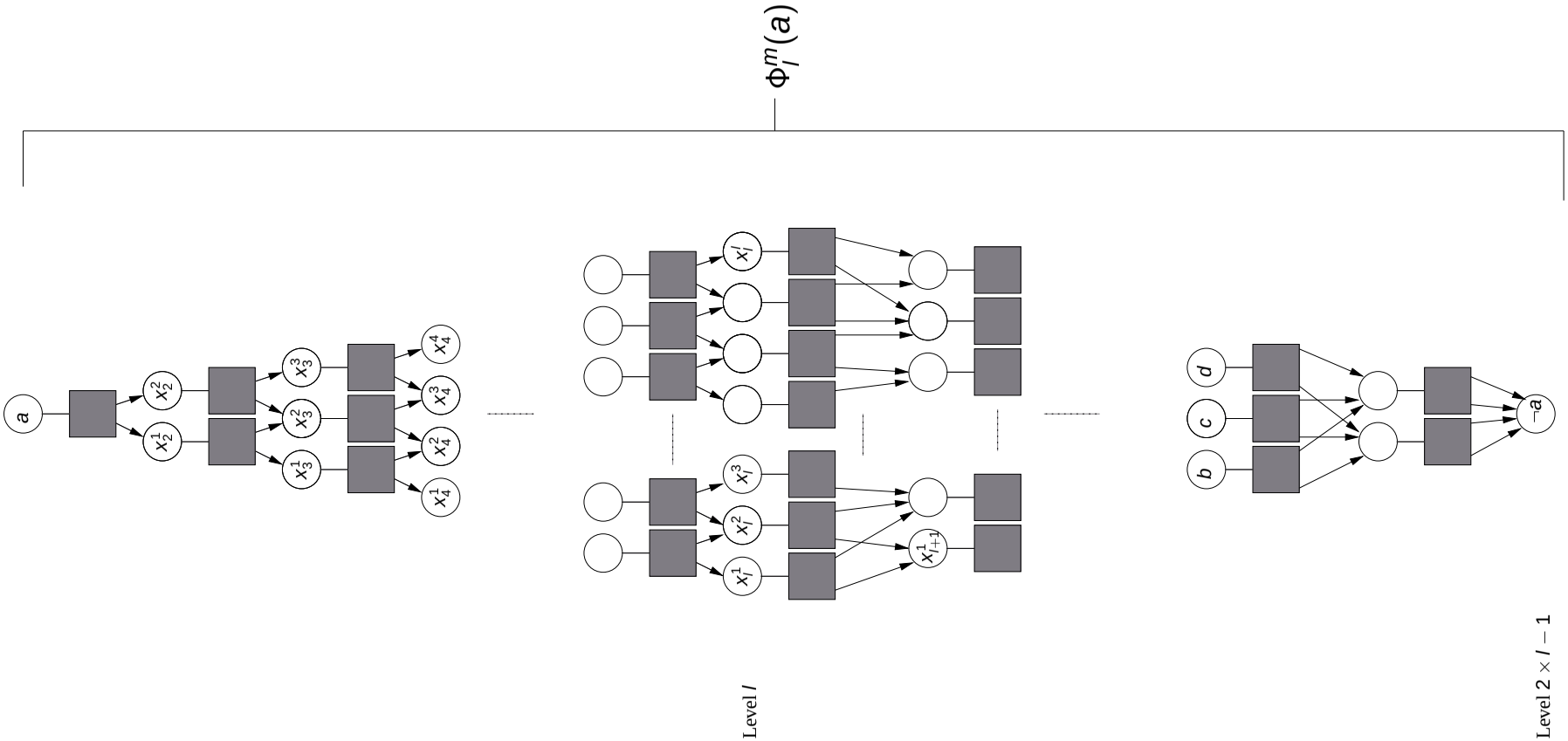
Let $\Phi =$
$(y_1 \vee y_2 \ldots \vee y_{\frac{m}{2}} \vee y_{\frac{m}{2}+1} \vee \ldots \vee y_m)$
$(\neg x1 \vee \neg y_1 \vee x_2)$
$(\neg x1 \vee \neg y_2 \vee x_2)$
...
$(\neg x1 \vee \neg y_{\frac{m}{2}} \vee x_2)$
$(\neg x1 \vee \neg y_{\frac{m}{2}+1} \vee x_3)$
...
$(\neg x1 \vee \neg y_m \vee x_3)$

Ona $\Phi \models (\neg x_1 \vee x_2 \vee x_3)$

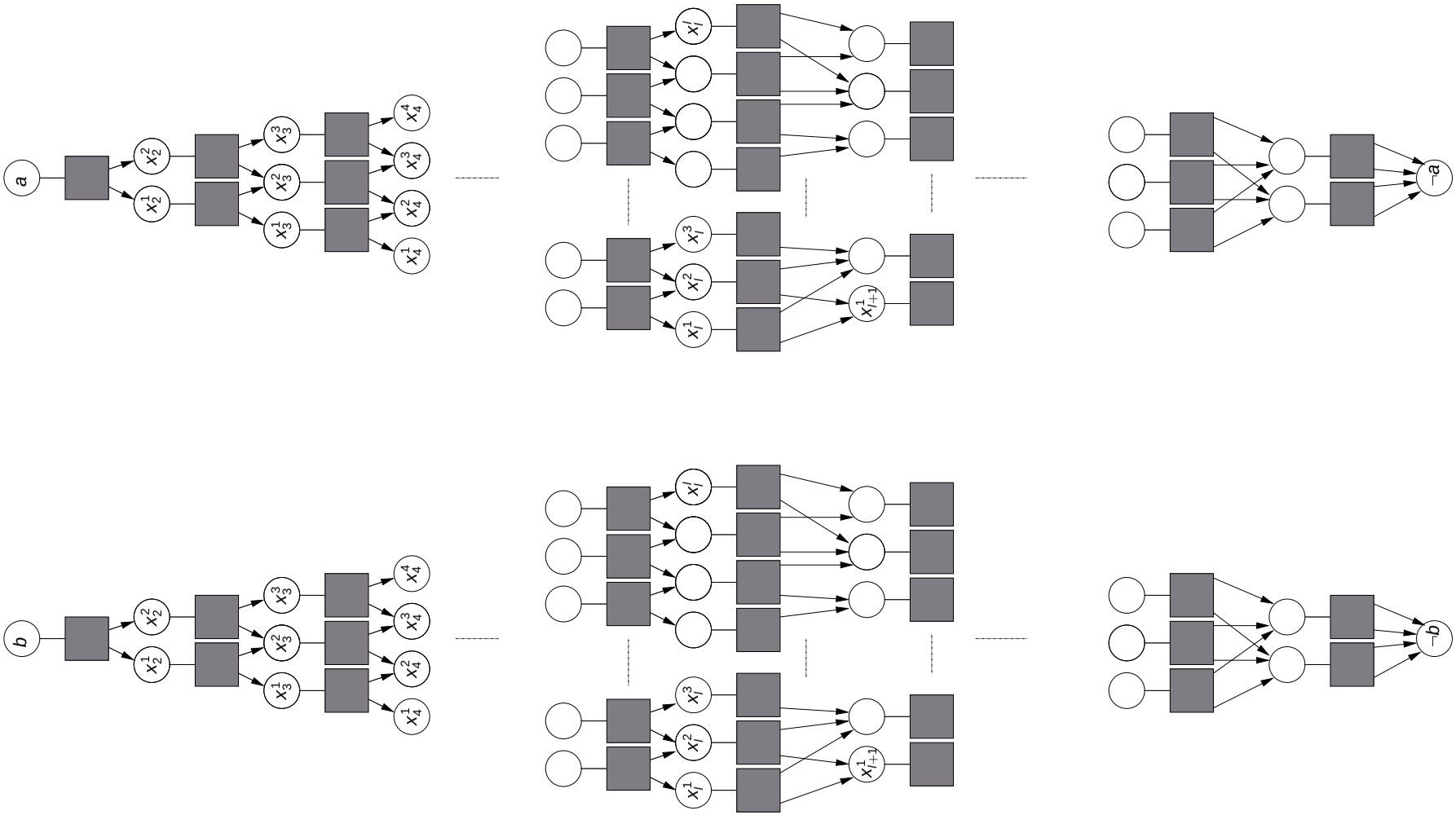
Block Résolution vs Hyper Binary Résolution



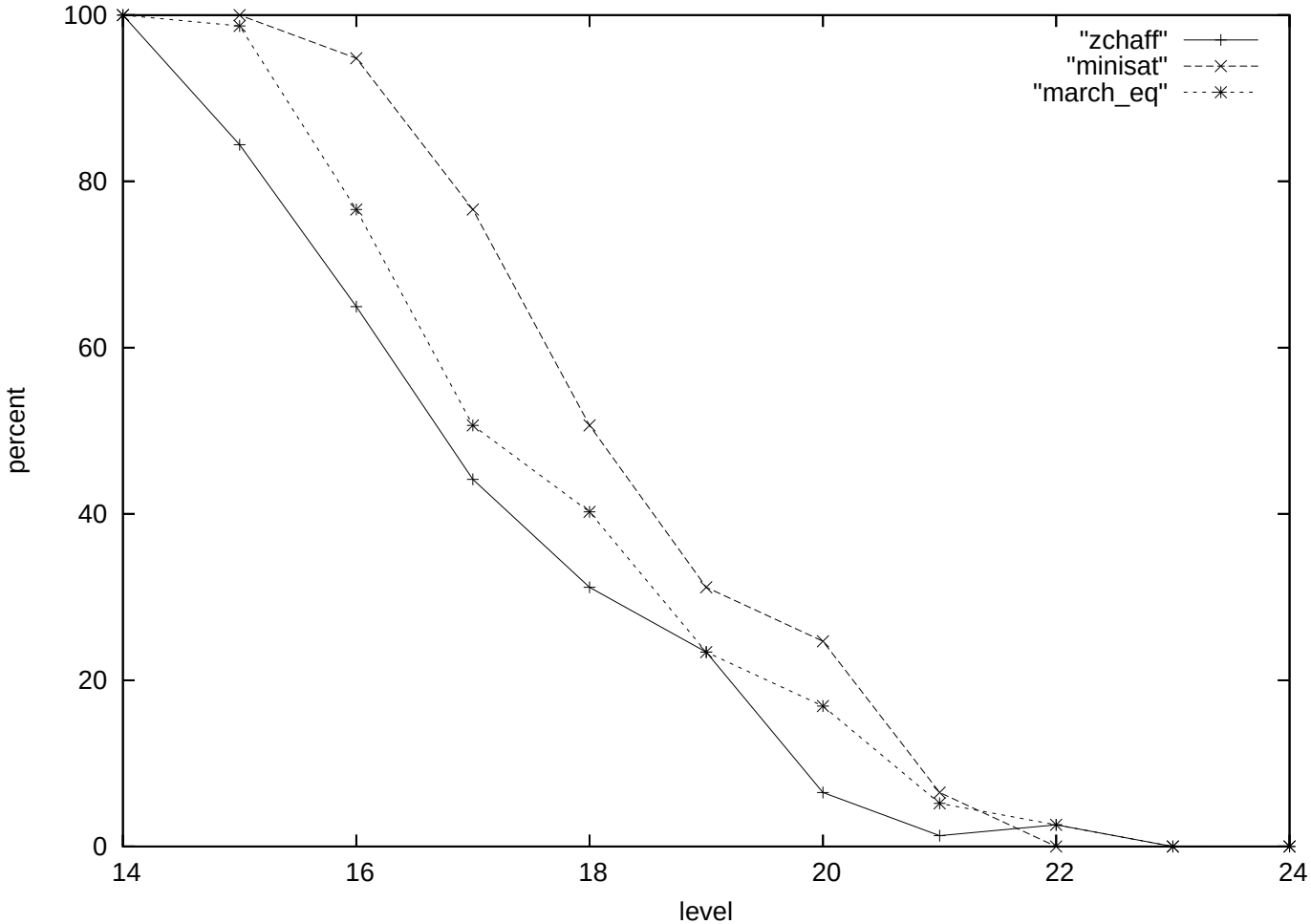
Graphe d'implication d'un litéral



Génération des instances



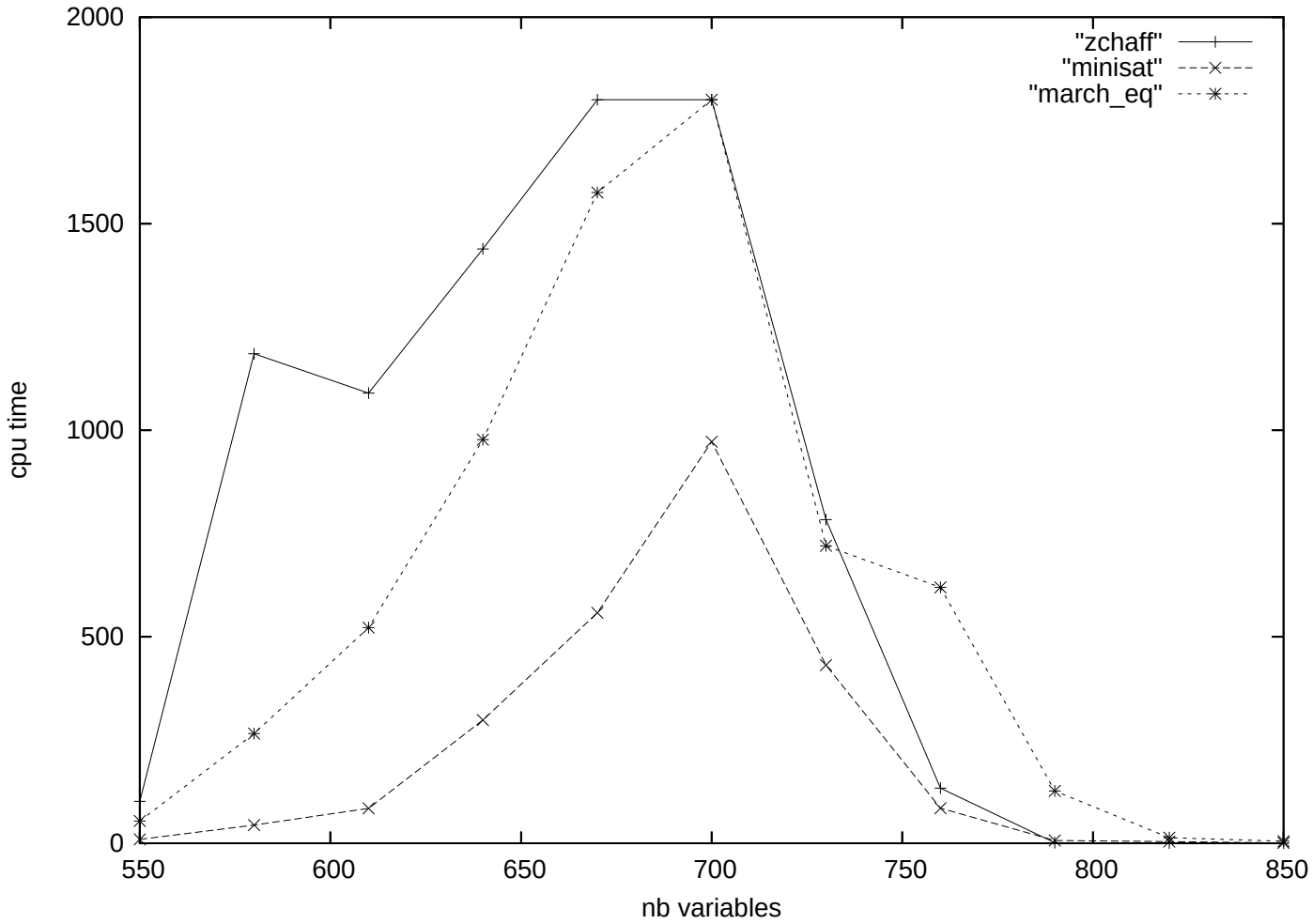
Expérimentations



$m = 4$

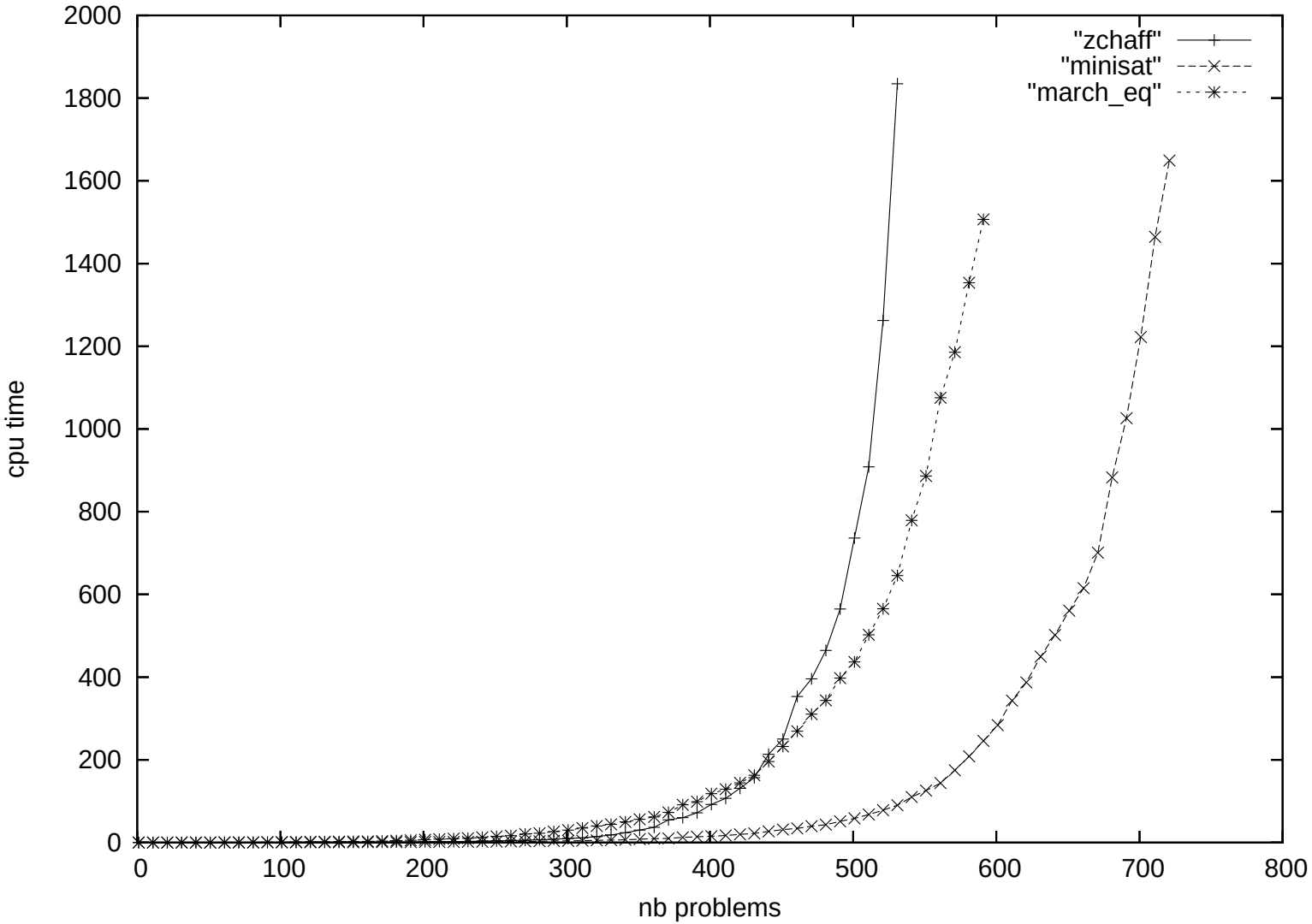
Expérimentations

Pour les blocks B_3^4



$l = 16$

Expérimentations



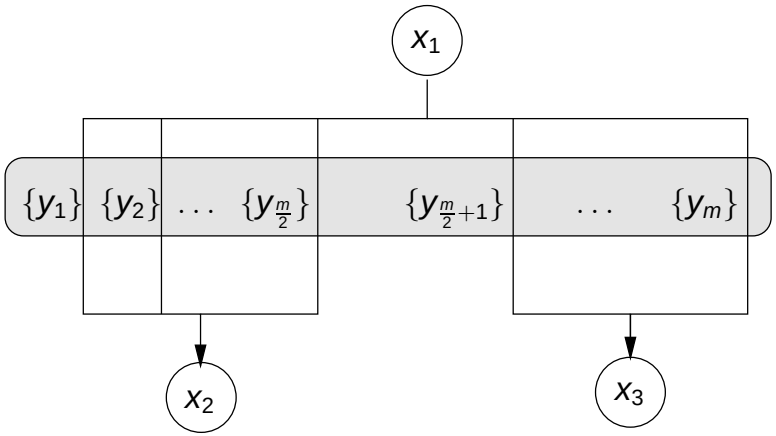
Expérimentations

		zchaff		minisat		march_eq	
level *l*	inst.	solved	avg time	solved	avg time	solved	avg time
15	63	58	68	**63**	37	60	118
16	78	65	92	**78**	113	62	218
17	84	67	96	**84**	139	67	169
18	68	45	137	**68**	199	59	327
19	52	37	240	**52**	241	35	247
20	30	1	326	**30**	421	18	359
21	8	1	872	**8**	440	7	289
22	3	-	-	-	-	**3**	1035

Expérimentations

		zchaff		minisat		march_eq	
level /	inst.	solved	time	solved	time	solved	time
14	106	106	67	**106**	5	106	40
15	91	70	155	**91**	55	85	214
16	67	23	592	**67**	254	44	523
17	26	0	-	**26**	834	3	1203
18	4	0	-	**4**	1383	-	-

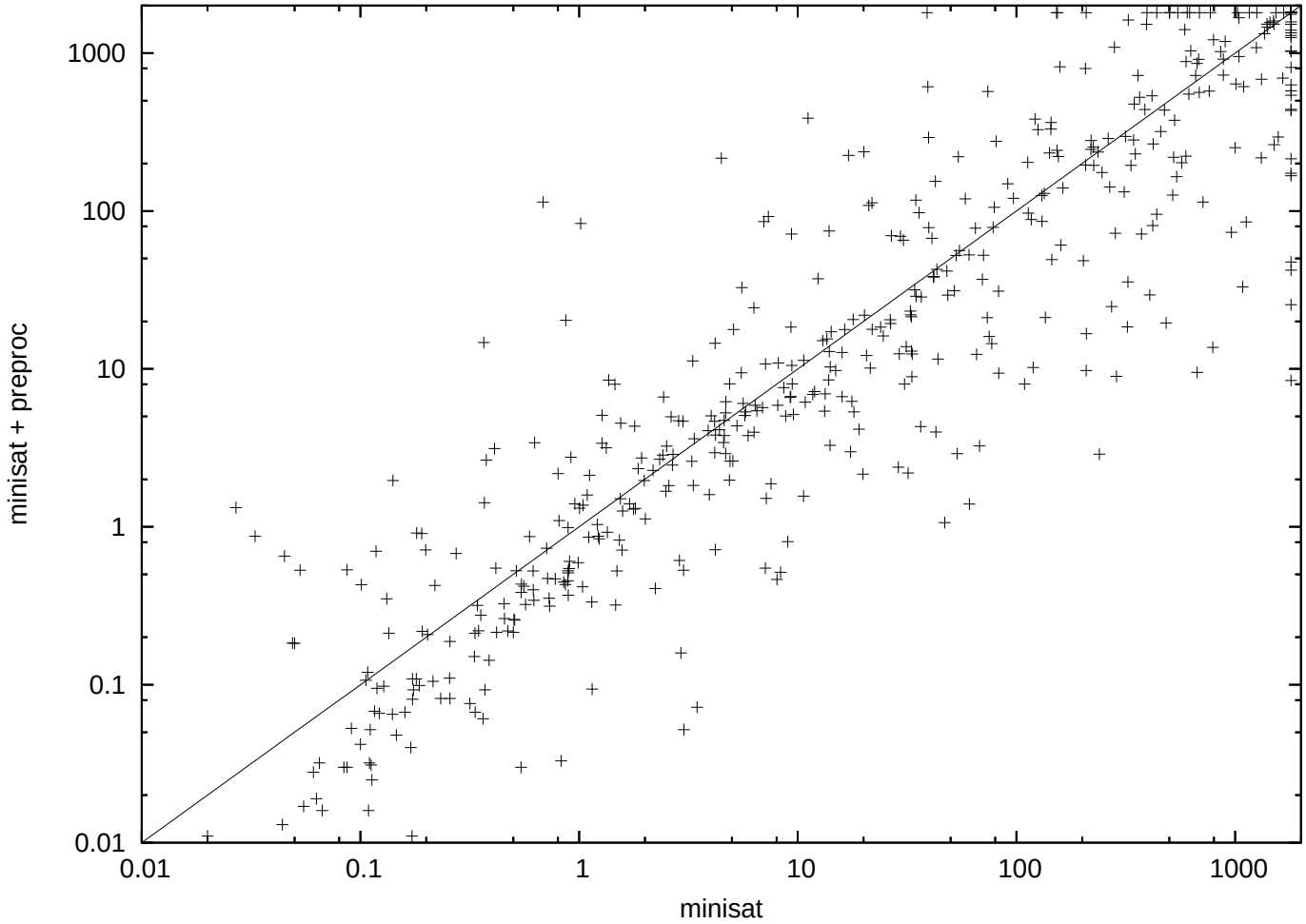
Block Résolution



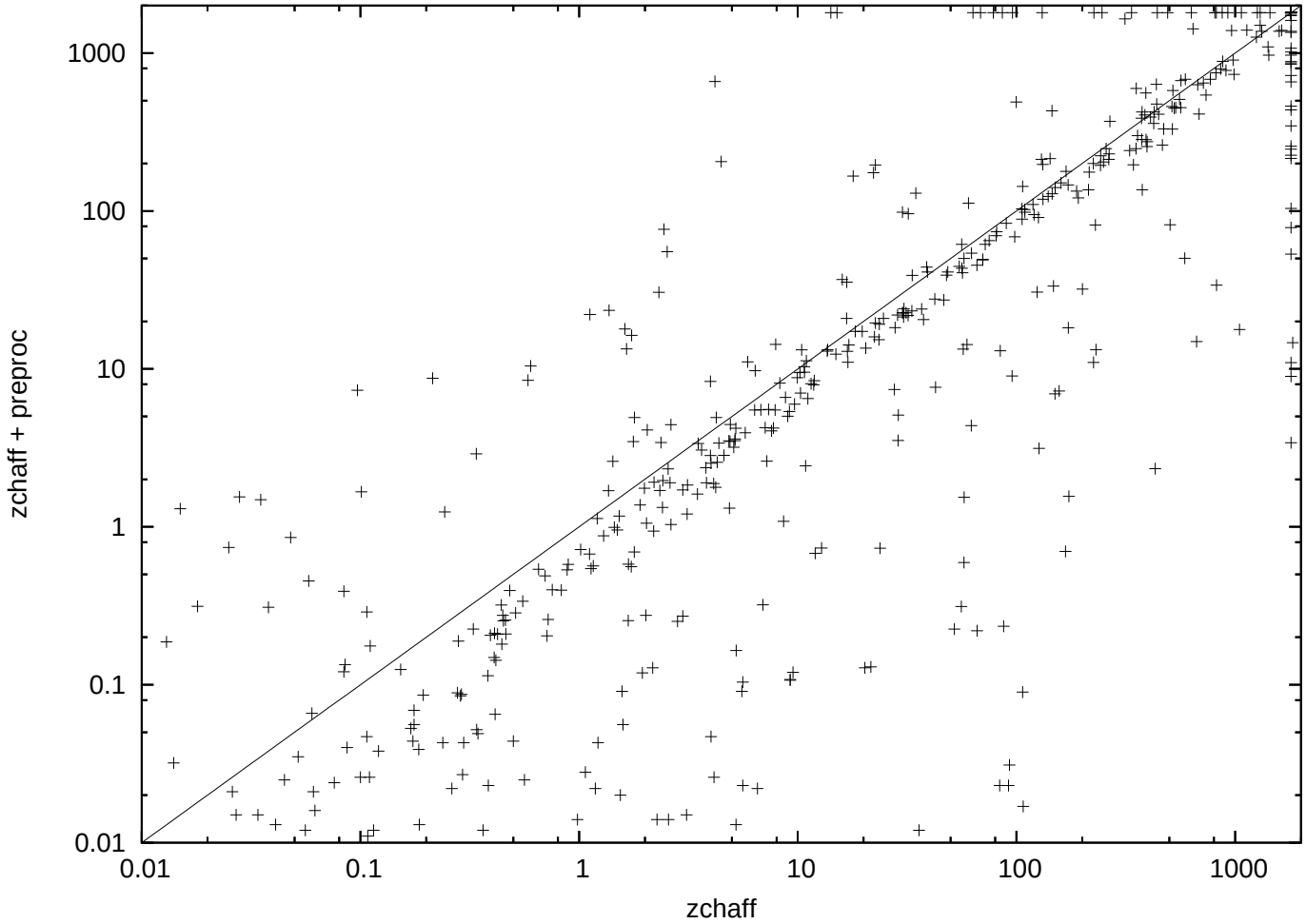
Let $\Phi =$
$(y_1 \vee y_2 \ldots \vee y_{\frac{m}{2}} \vee y_{\frac{m}{2}+1} \vee \ldots \vee y_m)$
$(\neg x1 \vee \neg y_1 \vee x_2)$
$(\neg x1 \vee \neg y_2 \vee x_2)$
...
$(\neg x1 \vee \neg y_{\frac{m}{2}} \vee x_2)$
$(\neg x1 \vee \neg y_{\frac{m}{2}+1} \vee x_3)$
...
$(\neg x1 \vee \neg y_m \vee x_3)$

On a $\Phi \models (\neg x_1 \vee x_2 \vee x_3)$

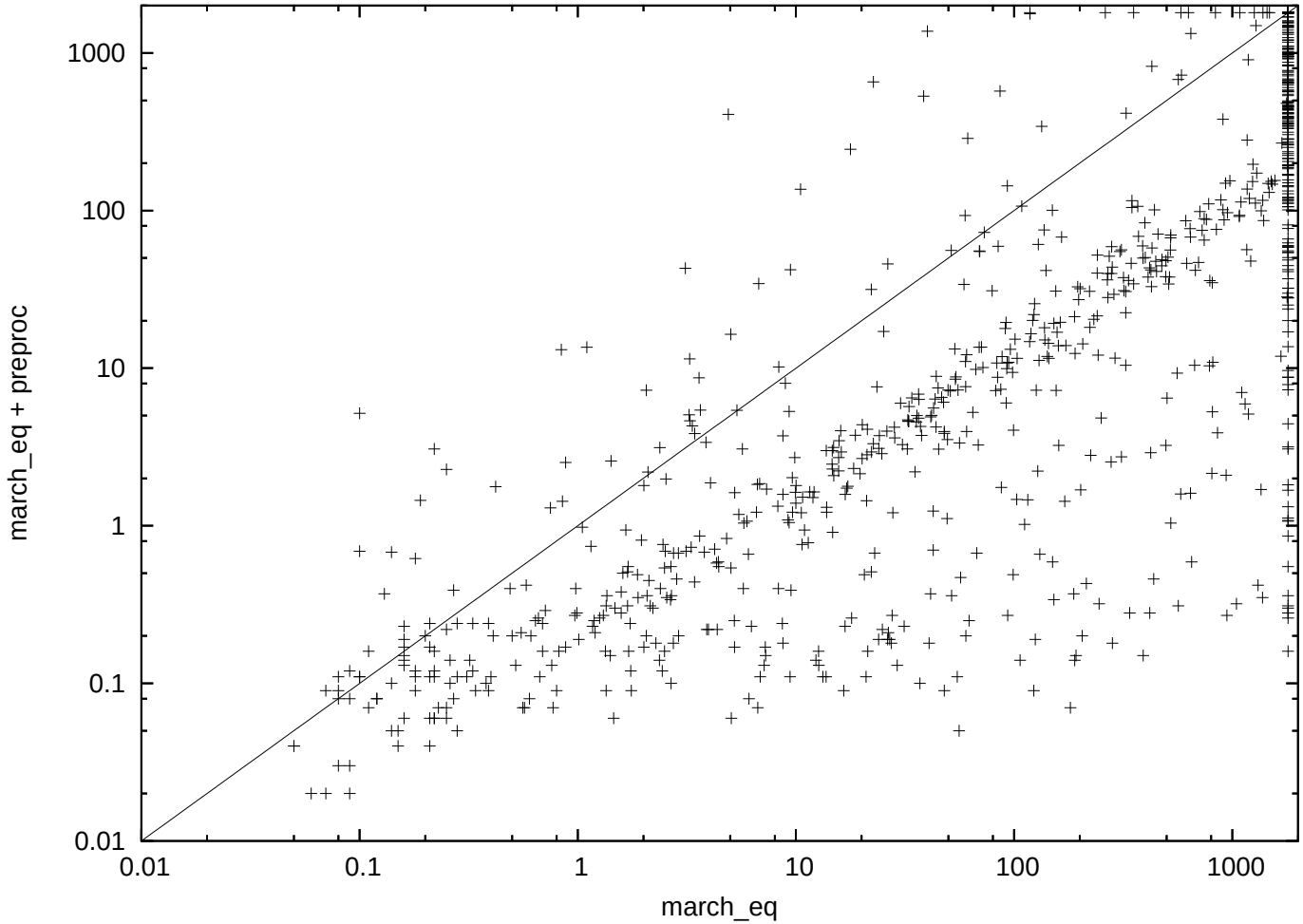
Pré-traitement des instances générées



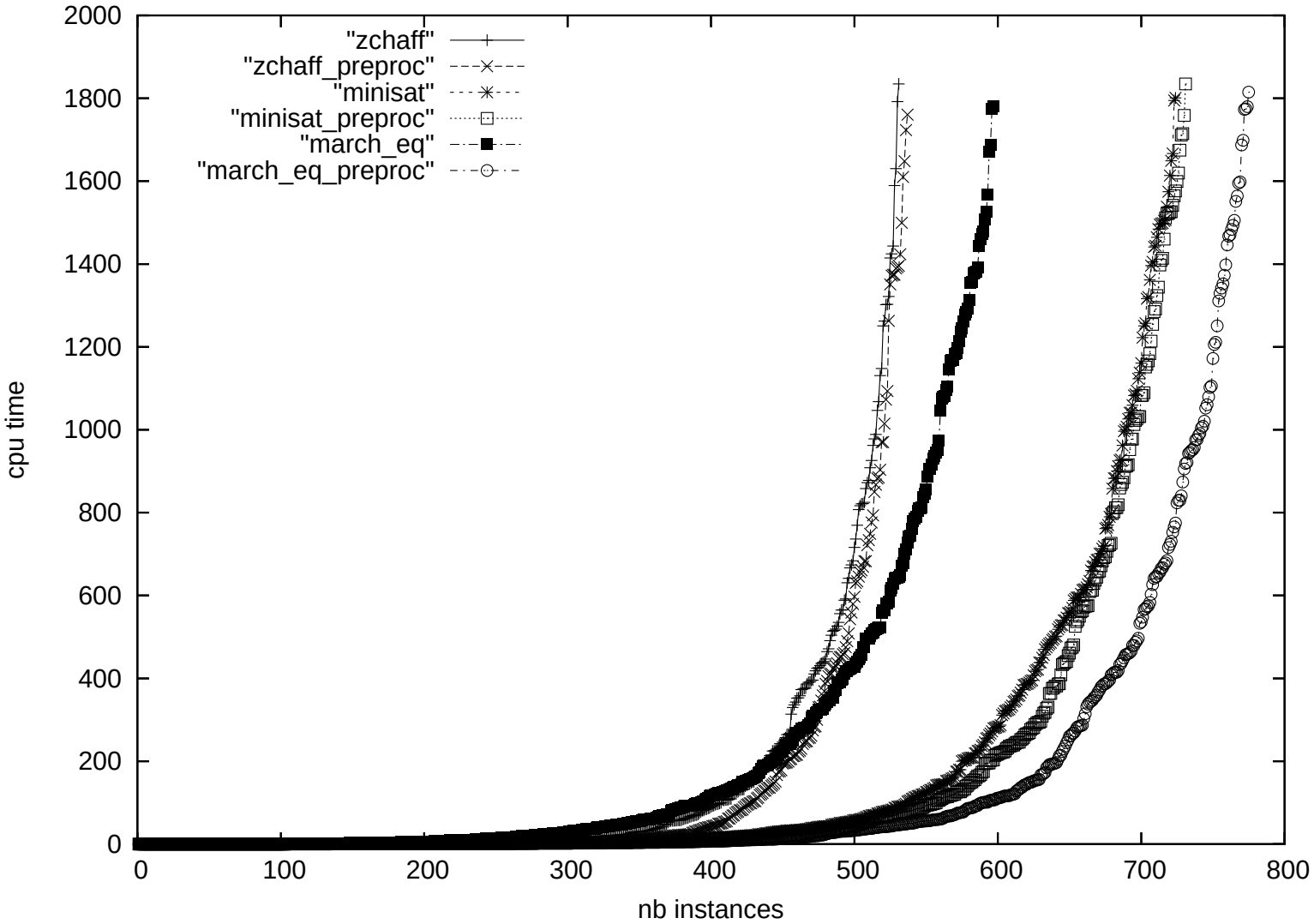
Pré-traitement des instances générées



Pré-traitement des instances générées



Pré-traitement des instances générées



Résolution par clause en utilisant le graphe

Soit $\phi = c_1 \wedge c_2 \ldots \wedge c_n$ and $c_1 = (x_1 \vee x_2 \vee x_3)$
Si :
$(\phi \wedge x_1) \models c_1^1 \wedge c_1^2 \ldots$
$(\phi \wedge x_2) \models c_2^1 \wedge c_2^2 \ldots$
$(\phi \wedge x_3) \models c_3^1 \wedge c_3^2 \ldots$
Quel résolvantes peut-on ajouter à la formule initiale à partir de ce sous ensemble de clauses?

Exemple

$(\phi \wedge x_1) \models y_1$
$(\phi \wedge x_2) \models y_1$
$(\phi \wedge x_3) \models y_2$
Alors dans ce cas on a $\phi \models (y_1 \vee y_2)$