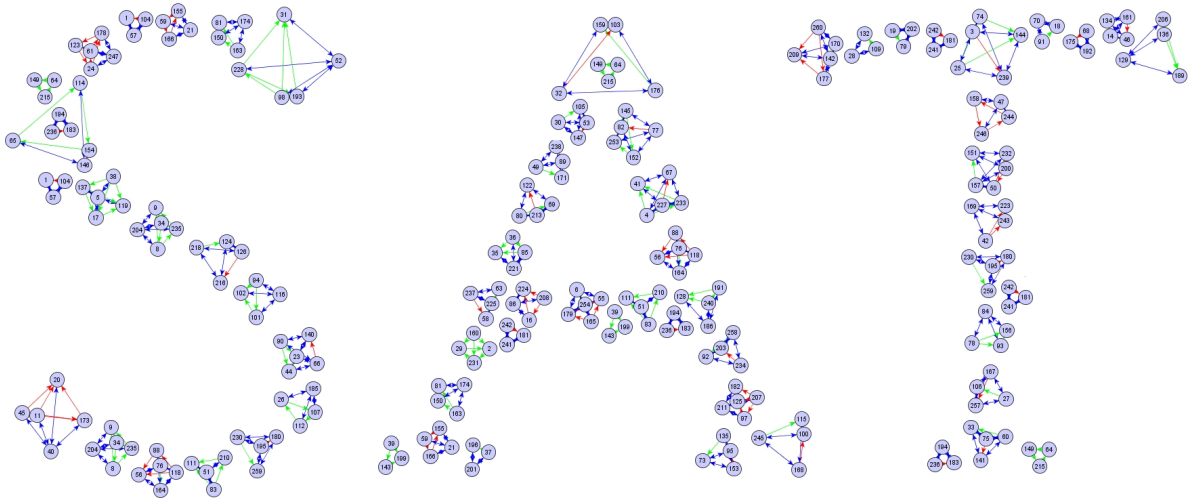
Résolution pratique de problèmes : SAT et ASP

Pascal Nicolas

Laurent Simon

Résolution pratique de problèmes



Laurent Simon

mél : simon@lri.fr web : www.lri.fr/~simon

LRI - CNRS
Université Orsay Paris 11,
91405 Orsay Cedex

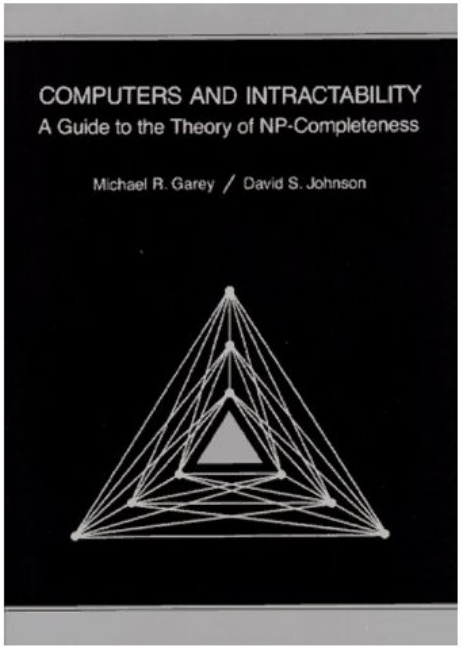
La brique SAT

Le capitaine Cook

Depuis [Cook, 1971], SAT est la brique de base de la théorie de la complexité.

- Très étudié pour ses limitations théoriques
- Très étudié pour sa résolution pratique

Il y a trente ans, il y a dix ans



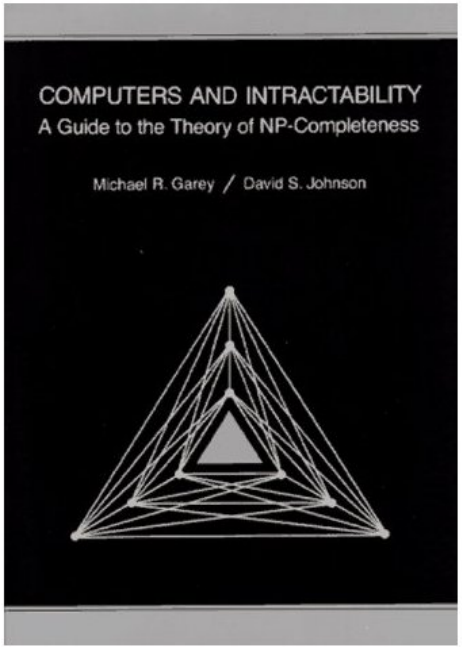
Un intérêt réduit de la réduction

Il y a 10 à 20 ans, la réduction polynomiale de tout problème à SAT était vu comme une question *seulement* d'intérêt théorique [Garey and Johnson, 1979].

Un problème semble difficile à résoudre en pratique ?
« Réduisez-le à SAT, et montrez qu'on ne peut pas le résoudre ! »

Et maintenant ?

Il y a trente ans, il y a dix ans



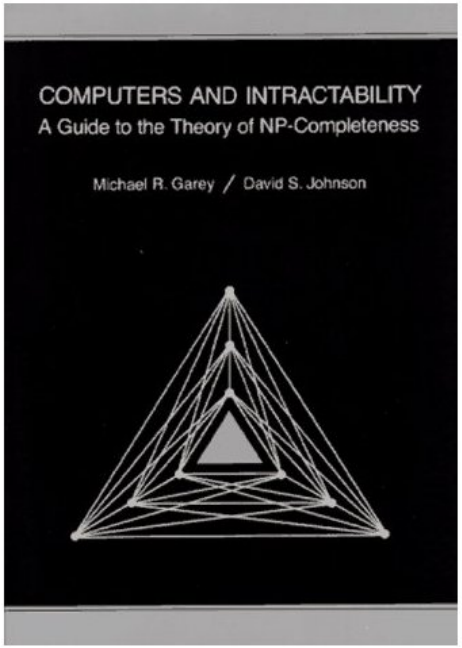
Un intérêt réduit de la réduction

Il y a 10 à 20 ans, la réduction polynomiale de tout problème à SAT était vu comme une question *seulement* d'intérêt théorique [Garey and Johnson, 1979].

Un problème semble difficile à résoudre en pratique ?
« Réduisez-le à SAT, et montrez qu'on ne peut pas le résoudre ! »

Et maintenant ?

Il y a trente ans, il y a dix ans



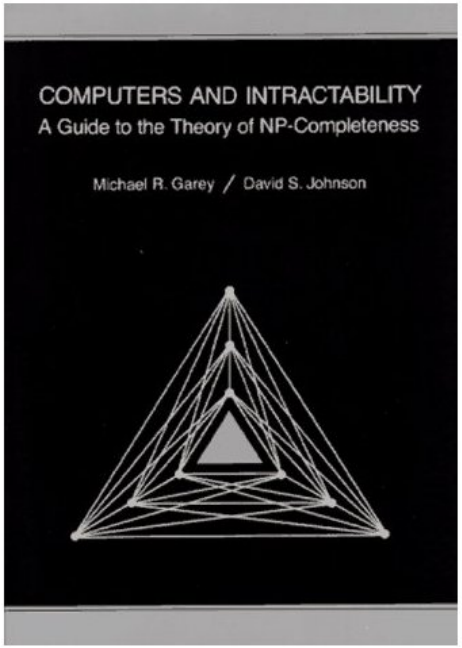
Un intérêt réduit de la réduction

Il y a 10 à 20 ans, la réduction polynomiale de tout problème à SAT était vu comme une question *seulement* d'intérêt théorique [Garey and Johnson, 1979].

Un problème semble difficile à résoudre en pratique ?
« Réduisez-le à SAT, et montrez qu'on ne peut pas le résoudre ! »

Et maintenant ?

Il y a trente ans, il y a dix ans



Un intérêt réduit de la réduction

Il y a 10 à 20 ans, la réduction polynomiale de tout problème à SAT était vu comme une question *seulement* d'intérêt théorique [Garey and Johnson, 1979].

Un problème semble difficile à résoudre en pratique ?
« Réduisez-le à SAT, et montrez qu'on ne peut pas le résoudre ! »

Et maintenant ?

D'INCROYABLES PROGRÈS PRATIQUES ET THÉORIQUES

DU POINT DE VUE PRATIQUE

Dans les années 90, les formules traitables étaient de quelques centaines de clauses sur une station de recherche...

Aujourd'hui, sur un PC *famillial*, on peut résoudre des problèmes de taille industrielle en :

- Electronic Design Automation (EDA)
  - Vérification de Microprocesseurs
  - Génération automatique de tests
- (Bounded) Model Checking (des solveurs incorporés dans des outils de model checking *industriels*)
- Planification
- Utilisé pour prouver des théorèmes de logiques plus expressives (au dessus de NP)

D'INCROYABLES PROGRÈS PRATIQUES ET THÉORIQUES

DU POINT DE VUE PRATIQUE

Dans les années 90, les formules traitables étaient de quelques centaines de clauses sur une station de recherche...
Aujourd'hui, sur un PC *famillial*, on peut résoudre des problèmes de taille industrielle en :

- Electronic Design Automation (EDA)
  - Vérification de Microprocesseurs
  - Génération automatique de tests
- (Bounded) Model Checking (des solveurs incorporés dans des outils de model checking *industriels*)
- Planification
- Utilisé pour prouver des théorèmes de logiques plus expressives (au dessus de NP)

D'INCROYABLES PROGRÈS PRATIQUES ET THÉORIQUES

DU POINT DE VUE PRATIQUE

Dans les années 90, les formules traitables étaient de quelques centaines de clauses sur une station de recherche...
Aujourd'hui, sur un PC *famillial*, on peut résoudre des problèmes de taille industrielle en :

- Electronic Design Automation (EDA)
  - Vérification de Microprocesseurs
  - Génération automatique de tests
- (Bounded) Model Checking (des solveurs incorporés dans des outils de model checking *industriels*)
- Planification
- Utilisé pour prouver des théorèmes de logiques plus expressives (au dessus de NP)

D'INCROYABLES PROGRÈS PRATIQUES ET THÉORIQUES

DU POINT DE VUE THÉORIQUE

AUJOURD'HUI, SUR DES INSTANCES ALÉATOIRES

On arrive à résoudre des problèmes UNSAT jusqu'à 600 variables au seuil, et plusieurs dizaines de milliers de variables du côté SAT.

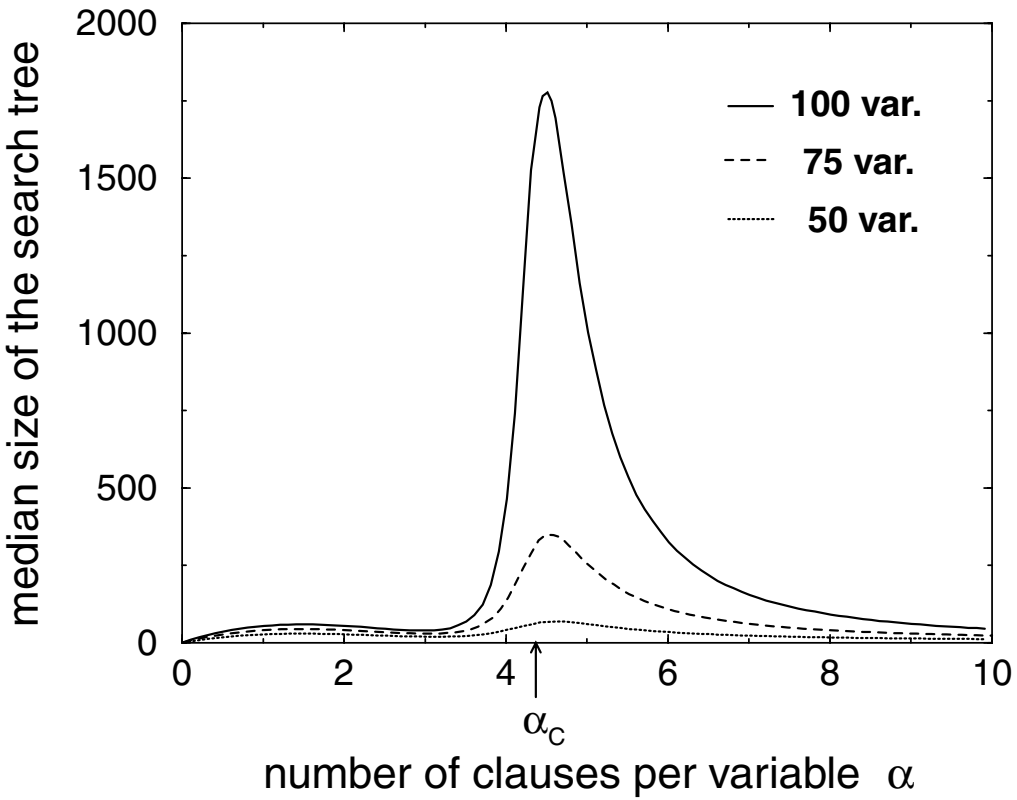


ILLUSTRATION DES PROGRÈS

DLL62 : LA PROCÉDURE DURE

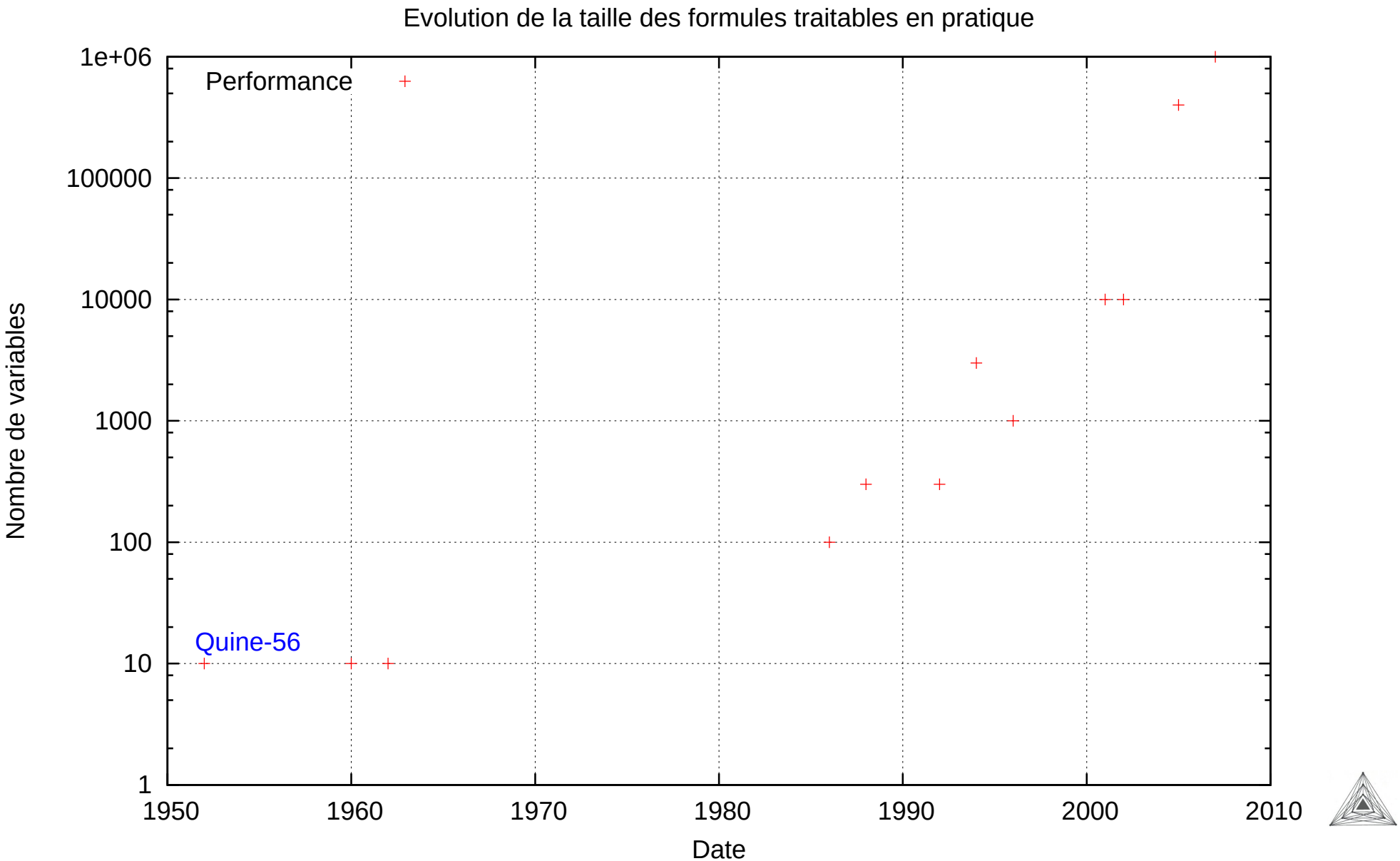


ILLUSTRATION DES PROGRÈS

DLL62 : LA PROCÉDURE DURE

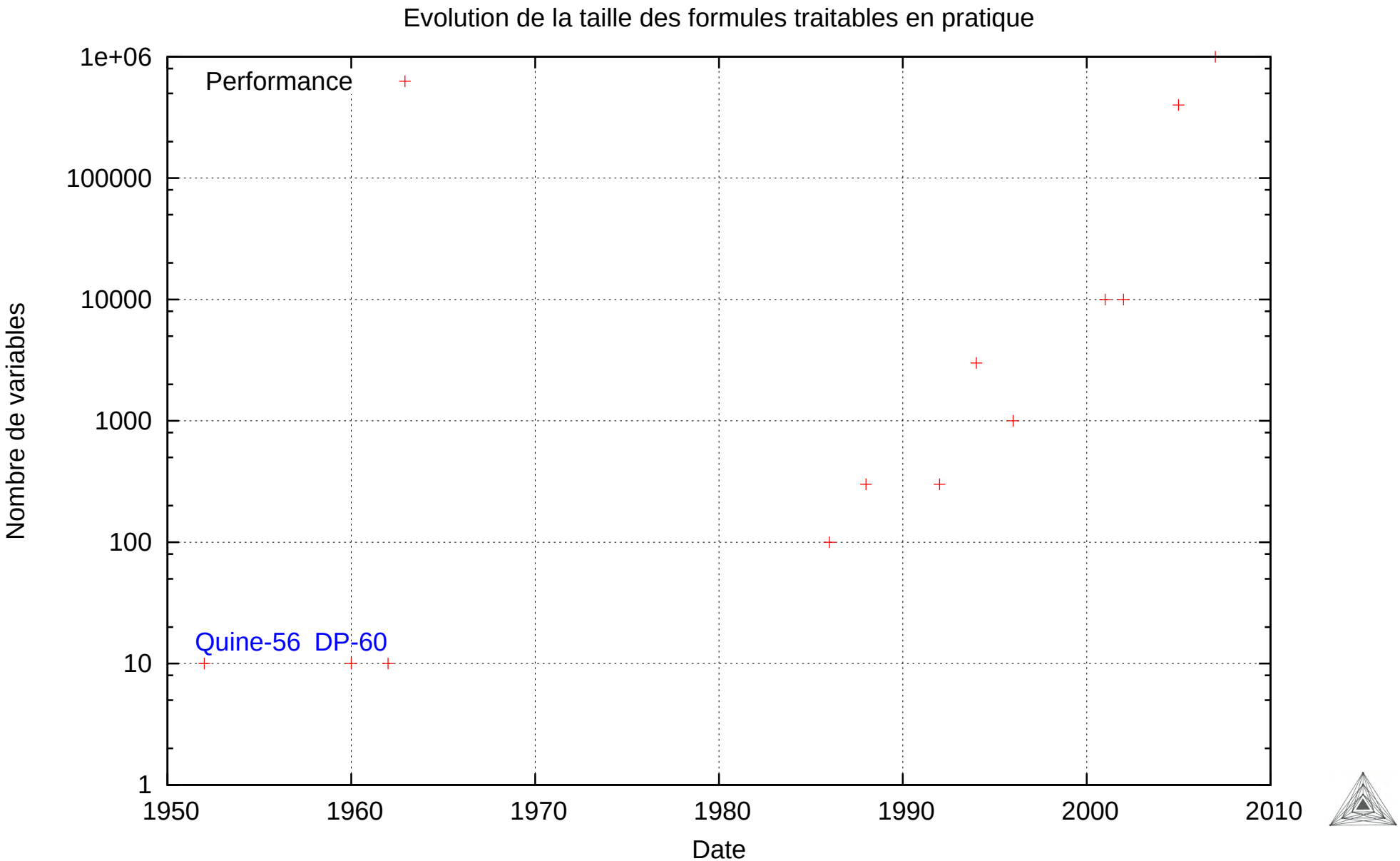


ILLUSTRATION DES PROGRÈS

DLL62 : LA PROCÉDURE DURE

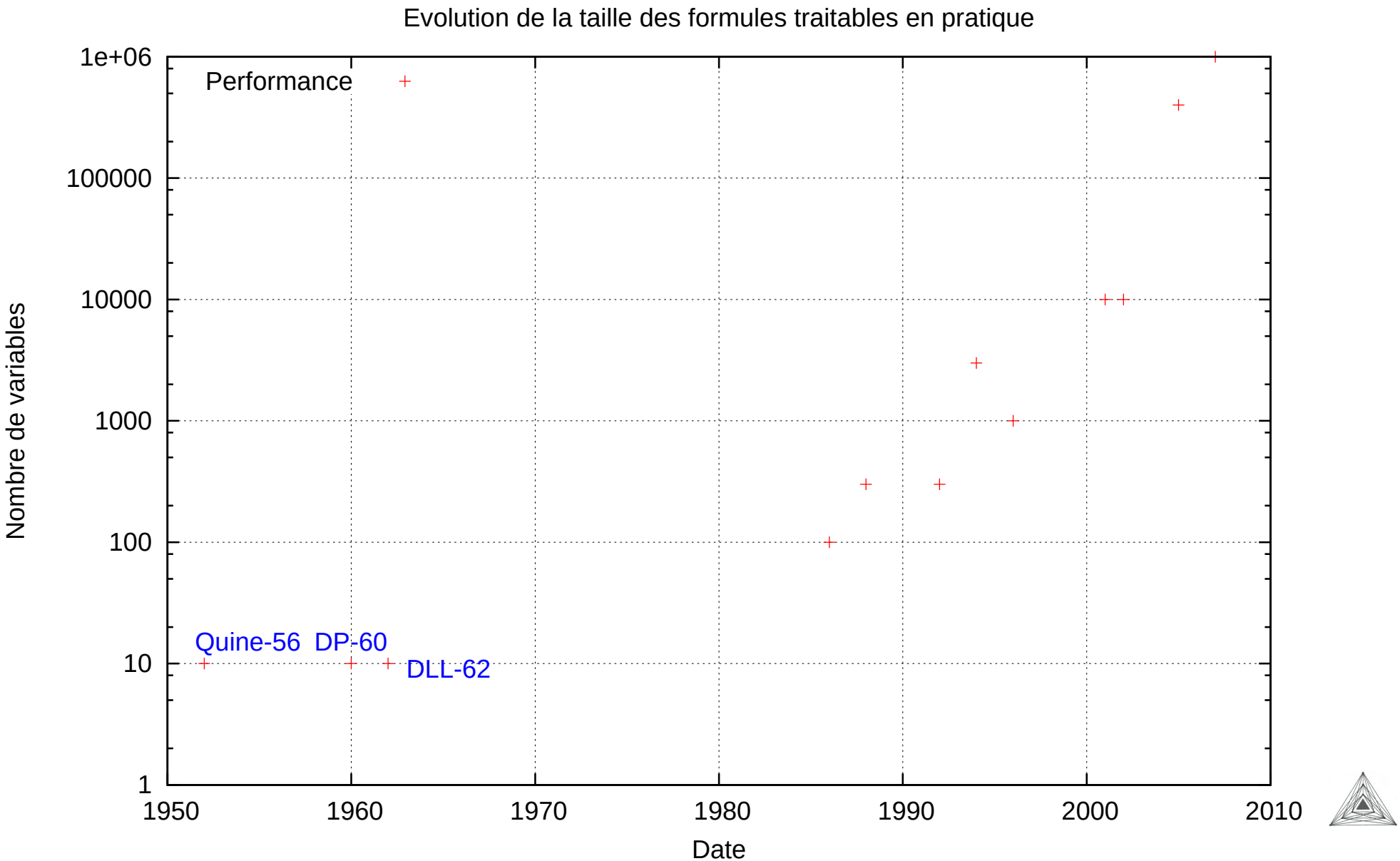


ILLUSTRATION DES PROGRÈS

DLL62 : LA PROCÉDURE DURE

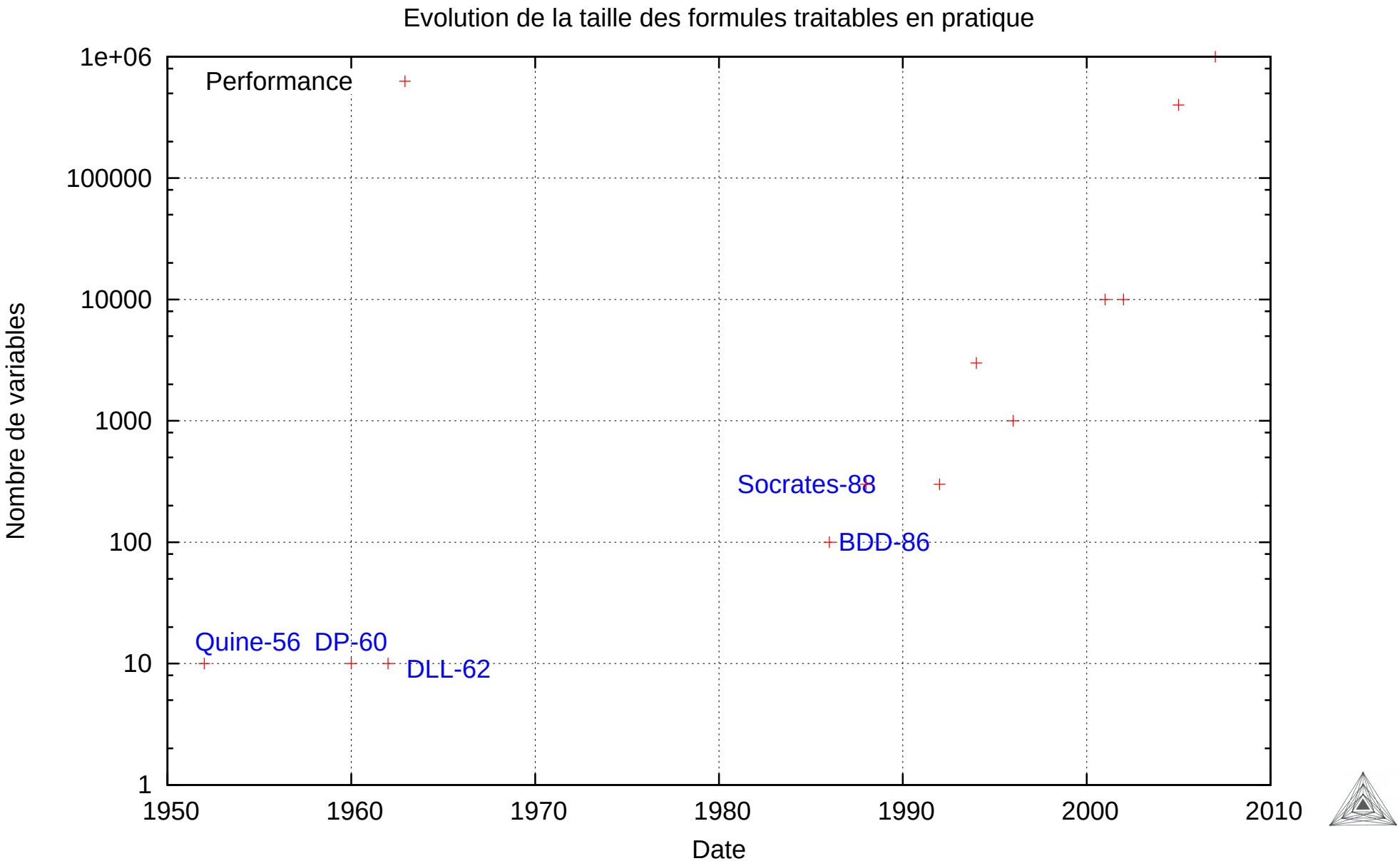


ILLUSTRATION DES PROGRÈS

DLL62 : LA PROCÉDURE DURE

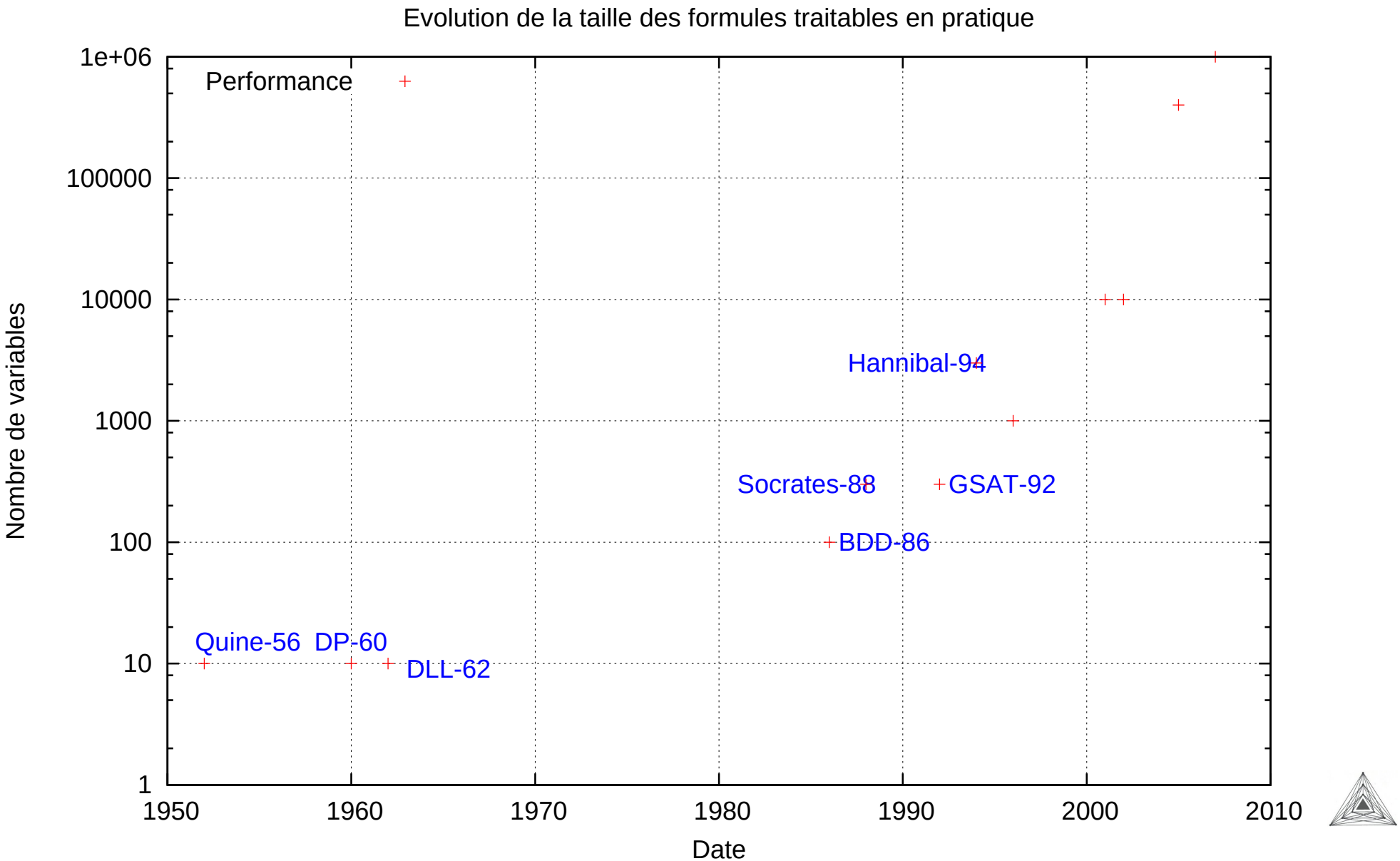


ILLUSTRATION DES PROGRÈS

DLL62 : LA PROCÉDURE DURE

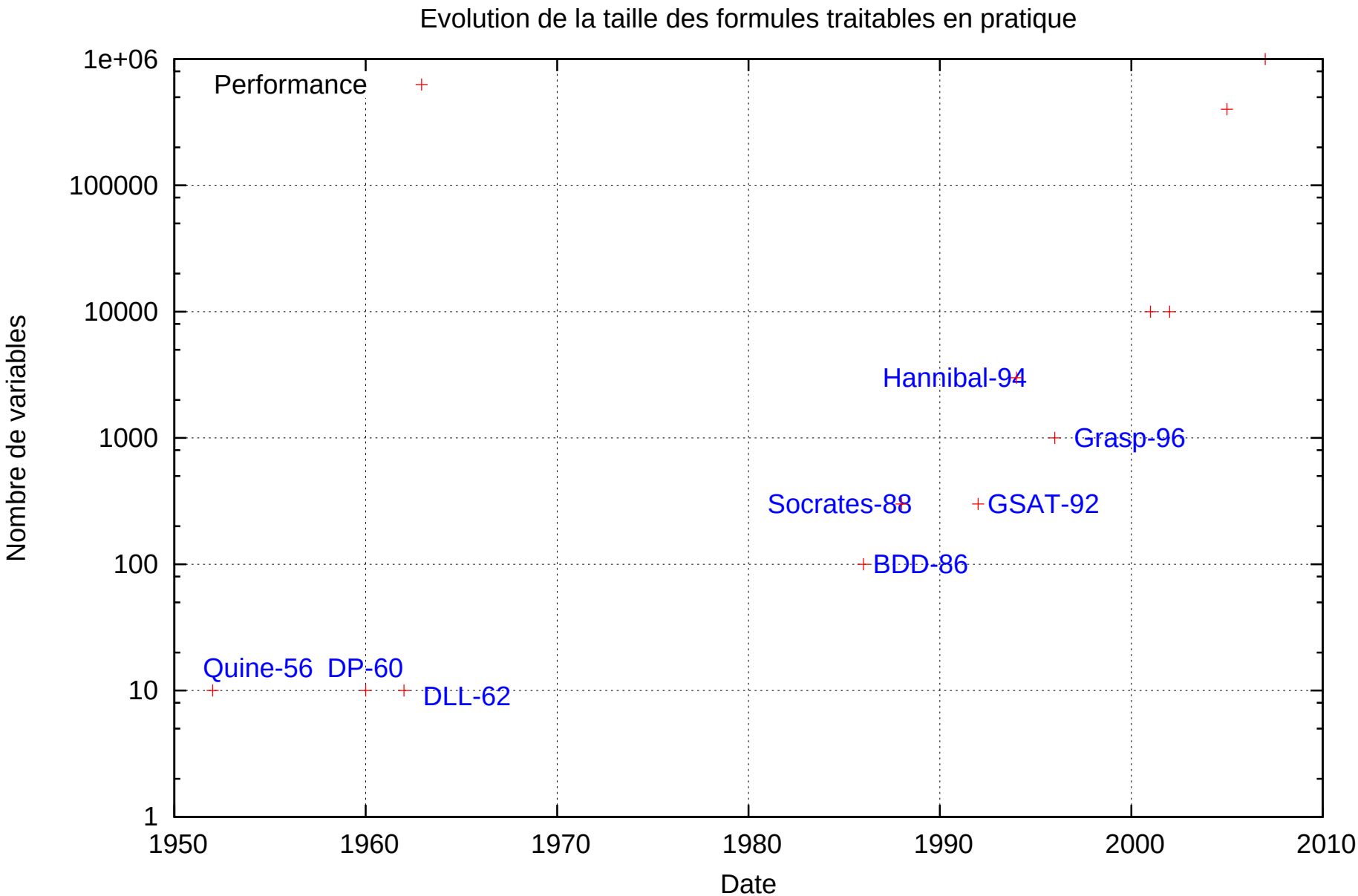


ILLUSTRATION DES PROGRÈS

DLL62 : LA PROCÉDURE DURE

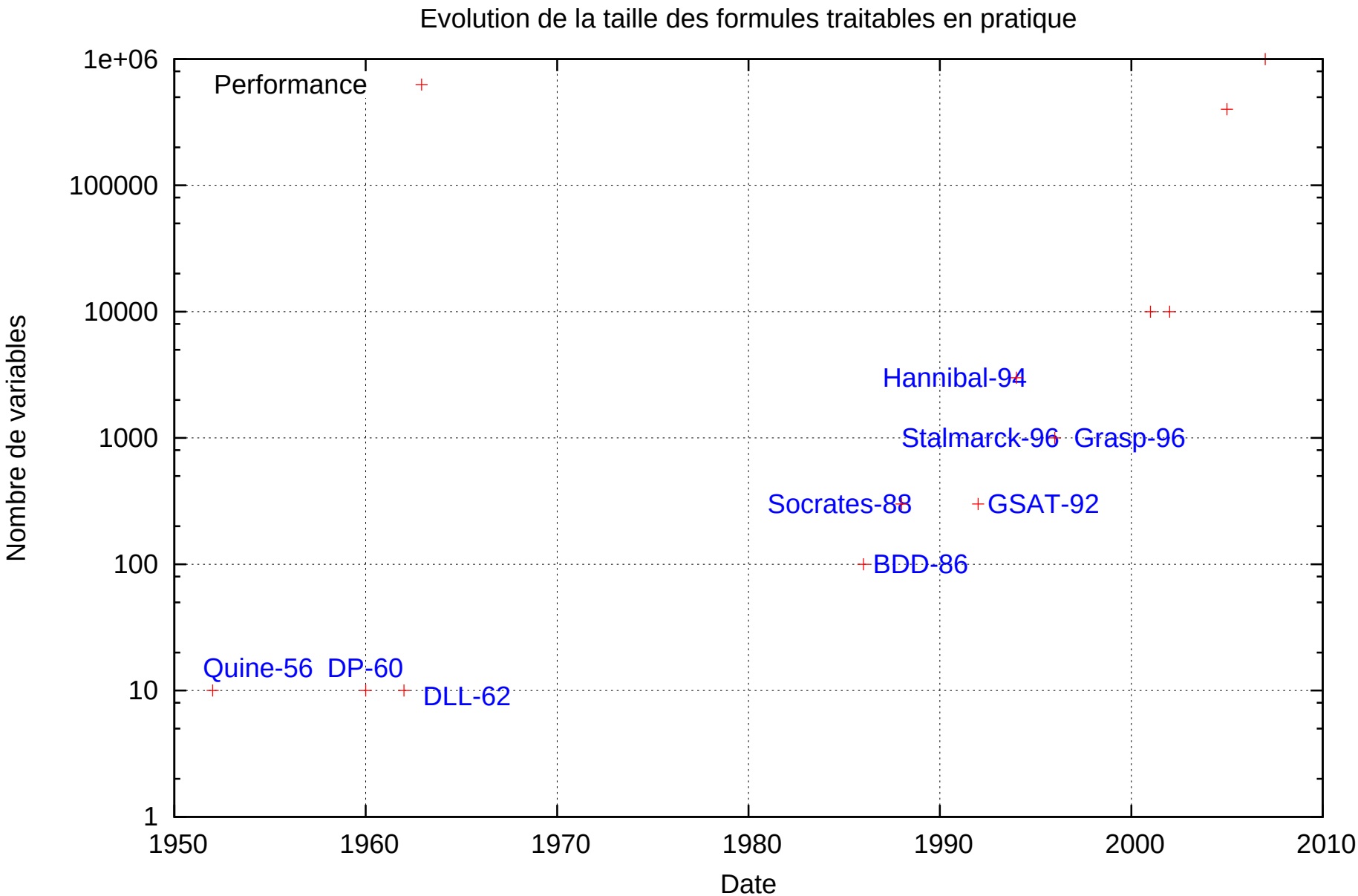


ILLUSTRATION DES PROGRÈS

DLL62 : LA PROCÉDURE DURE

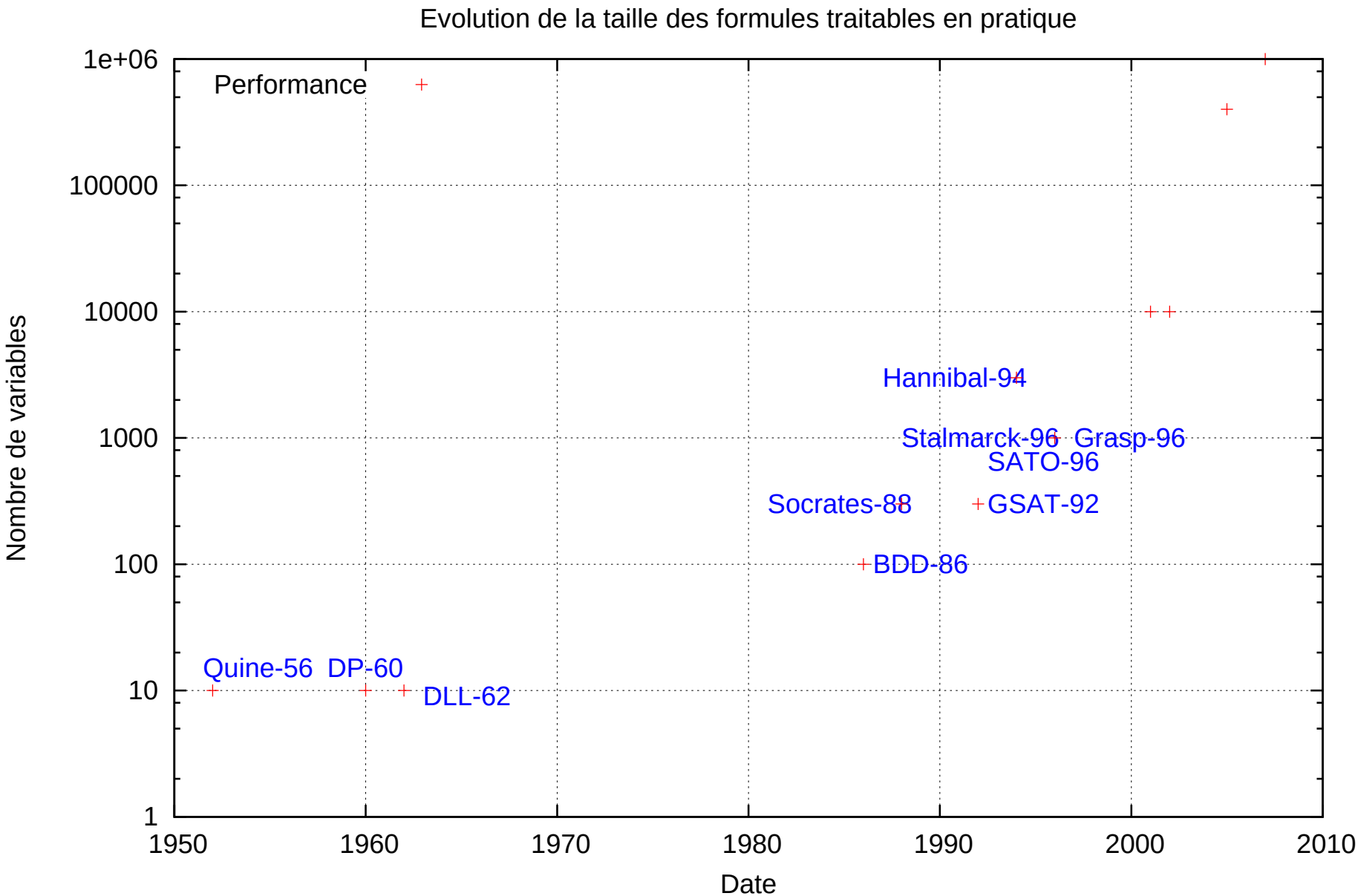


ILLUSTRATION DES PROGRÈS

DLL62 : LA PROCÉDURE DURE

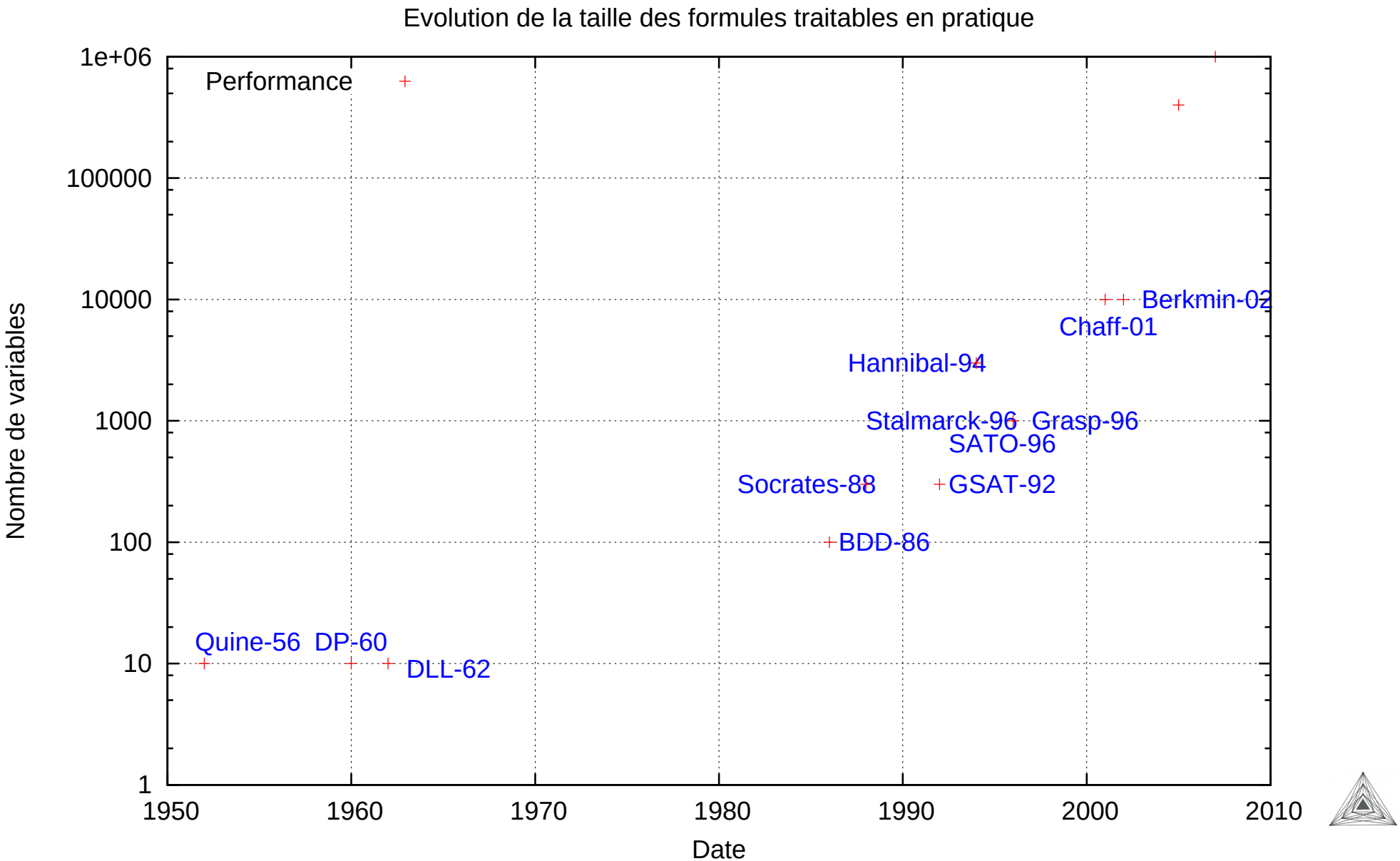


ILLUSTRATION DES PROGRÈS

DLL62 : LA PROCÉDURE DURE

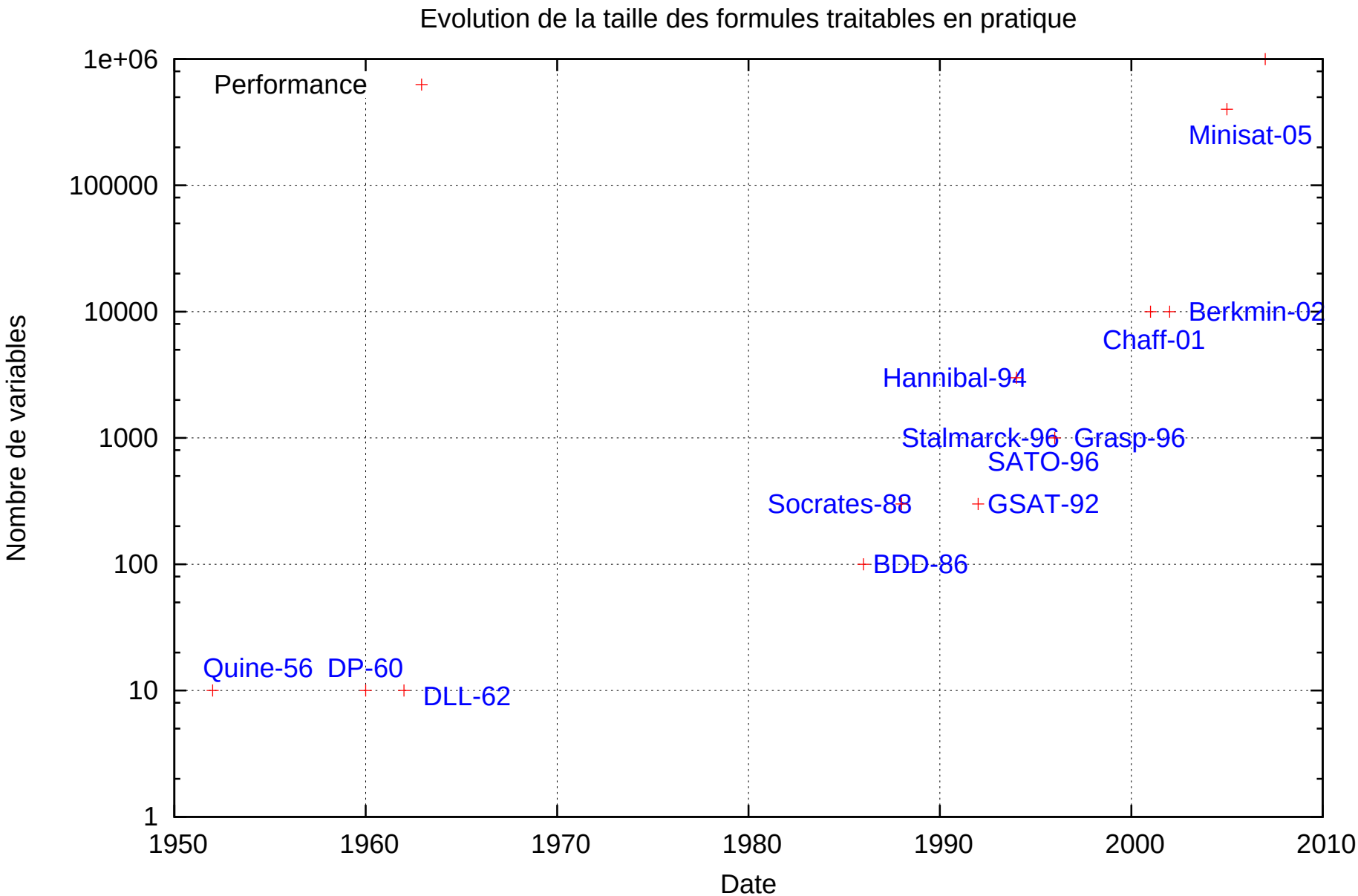


ILLUSTRATION DES PROGRÈS

DLL62 : LA PROCÉDURE DURE

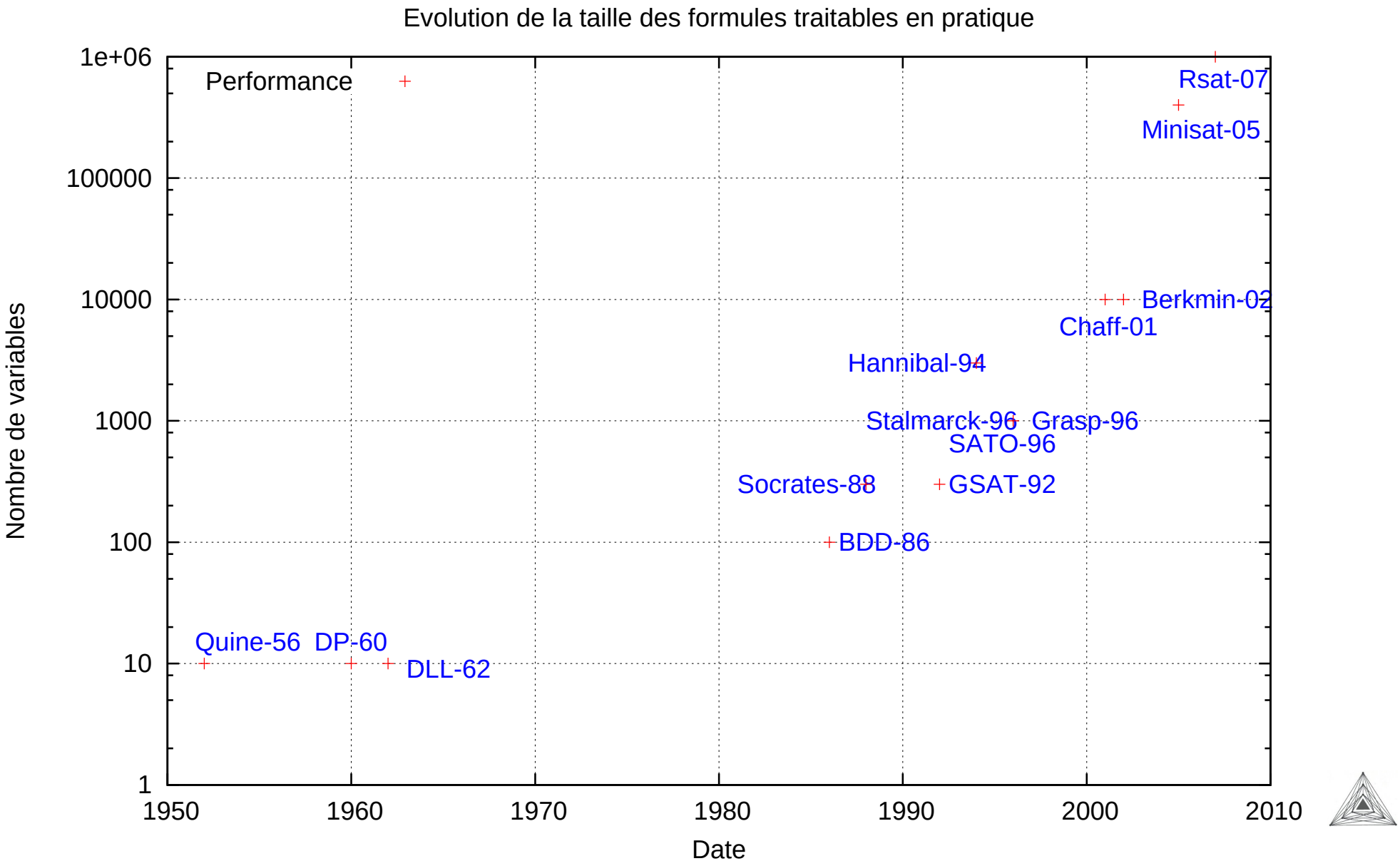
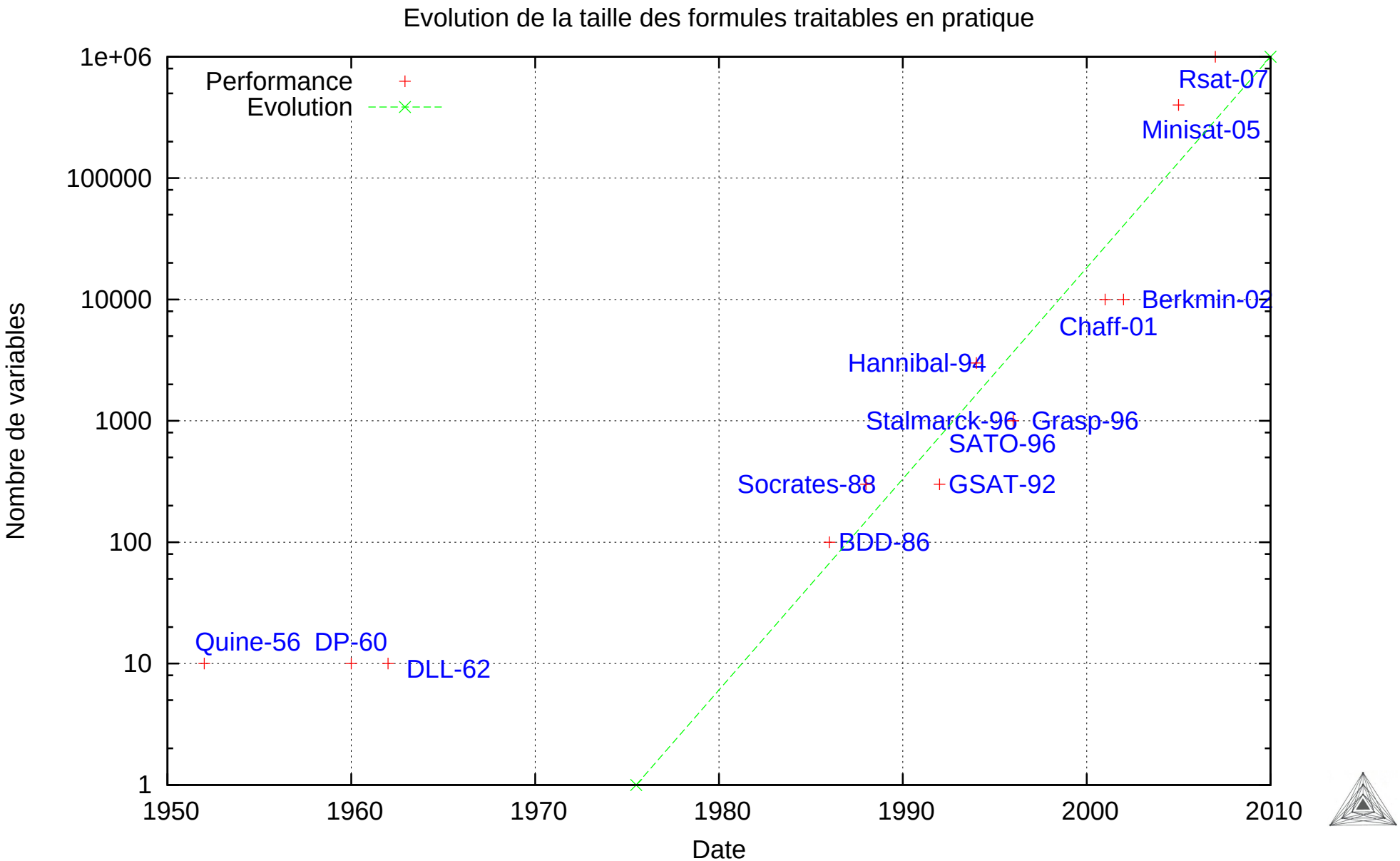


ILLUSTRATION DES PROGRÈS

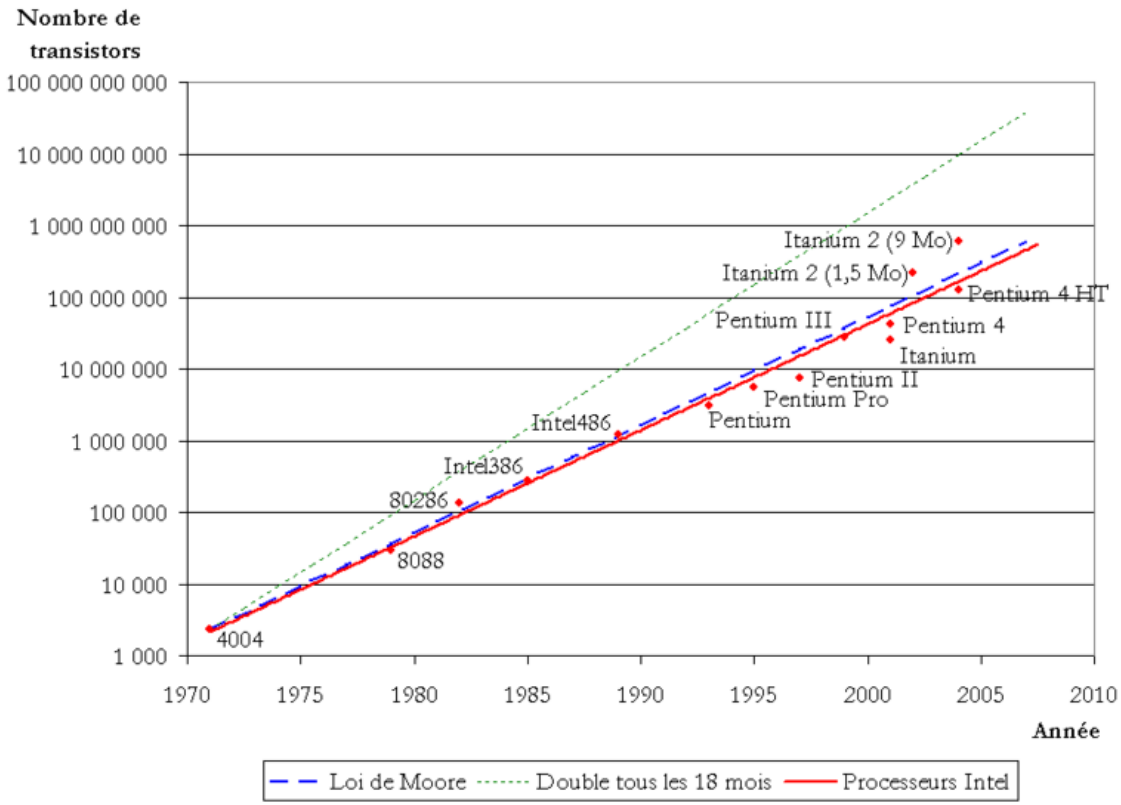
DLL62 : LA PROCÉDURE DURE



Des forces de progrès en marche

3 facteurs (et plus)

1. Algorithmes sophistiqués **et** simples
2. Implantations *ingénieuses*
3. Seuil de capacité franchit par les machines?



UNE BRIQUETTE PERSONNELLE

LES PROGRÈS DE LA COMPÉTITION SAT

UNE NOUVELLE COMPÉTITION DEPUIS 2002

- Organisé pour *mesurer* l'impact de `zchaff`
- Motive toute la communauté, fournit des benchmarks
- Offre une vitrine *officielle* et *uniforme*
- Ouvre la problématique d'expérimentation d'algorithmes

	2002	2003	2004	2005	2007
#Solvers	28	34	55	43	44
#Submit.	19	22	27	22	28
Time 1st	2400	900	600	1200	1200
Time 2nd	21600	7200	2400	12000	10000
Total	750	522	516	1328	915
Judges		J. FRANCO, H. VAN MAAREN, T. WALSH	F. BACCHUS, J. MARQUES-SILVA, H. KLEINE BUNING	A. VAN GELDER, A. BIERE, O. KULLMAN	E. SPECKENMEYER, G. SUTCLIFFE, L. ZHANG

UNE BRIQUETTE PERSONNELLE

LES PROGRÈS DE LA COMPÉTITION SAT

UNE NOUVELLE COMPÉTITION DEPUIS 2002

- Organisé pour *mesurer* l'impact de `zchaff`
- Motive toute la communauté, fournit des benchmarks
- Offre une vitrine *officielle* et *uniforme*
- Ouvre la problématique d'expérimentation d'algorithmes

	2002	2003	2004	2005	2007
#Solvers	28	34	55	43	44
#Submit.	19	22	27	22	28
Time 1st	2400	900	600	1200	1200
Time 2nd	21600	7200	2400	12000	10000
Total	750	522	516	1328	915
Judges		J. FRANCO, H. VAN MAAREN, T. WALSH	F. BACCHUS, J. MARQUES-SILVA, H. KLEINE BUNING	A. VAN GELDER, A. BIERE, O. KULLMAN	E. SPECKENMEYER, G. SUTCLIFFE, L. ZHANG

UNE BRIQUETTE PERSONNELLE

LES PROGRÈS DE LA COMPÉTITION SAT

UNE NOUVELLE COMPÉTITION DEPUIS 2002

- Organisé pour *mesurer* l'impact de `zchaff`
- Motive toute la communauté, fournit des benchmarks
- Offre une vitrine *officielle* et *uniforme*
- Ouvre la problématique d'expérimentation d'algorithmes

	2002	**2003**	**2004**	**2005**	**2007**
#Solvers	28	34	55	43	44
#Submit.	19	22	27	22	28
Time 1st	2400	900	600	1200	1200
Time 2nd	21600	7200	2400	12000	10000
Total	750	522	516	1328	915
Judges		J. FRANCO, H. VAN MAAREN, T. WALSH	F. BACCHUS, J. MARQUES-SILVA, H. KLEINE BUNING	A. VAN GELDER, A. BIERE, O. KULLMAN	E. SPECKENMEYER, G. SUTCLIFFE, L. ZHANG

UNE BRIQUETTE PERSONNELLE

LES PROGRÈS DE LA COMPÉTITION SAT

UNE NOUVELLE COMPÉTITION DEPUIS 2002

- Organisé pour *mesurer* l'impact de `zchaff`
- Motive toute la communauté, fournit des benchmarks
- Offre une vitrine *officielle* et *uniforme*
- Ouvre la problématique d'expérimentation d'algorithmes

	2002	**2003**	**2004**	**2005**	**2007**
#Solvers	28	34	55	43	44
#Submit.	19	22	27	22	28
Time 1st	2400	900	600	1200	1200
Time 2nd	21600	7200	2400	12000	10000
Total	750	522	516	1328	915
Judges		J. FRANCO, H. VAN MAAREN, T. WALSH	F. BACCHUS, J. MARQUES-SILVA, H. KLEINE BUNING	A. VAN GELDER, A. BIERE, O. KULLMAN	E. SPECKENMEYER, G. SUTCLIFFE, L. ZHANG

UNE BRIQUETTE PERSONNELLE

LES PROGRÈS DE LA COMPÉTITION SAT

UNE NOUVELLE COMPÉTITION DEPUIS 2002

- Organisé pour *mesurer* l'impact de `zchaff`
- Motive toute la communauté, fournit des benchmarks
- Offre une vitrine *officielle* et *uniforme*
- Ouvre la problématique d'expérimentation d'algorithmes

	2002	**2003**	**2004**	**2005**	**2007**
#Solvers	28	34	55	43	44
#Submit.	19	22	27	22	28
Time 1st	2400	900	600	1200	1200
Time 2nd	21600	7200	2400	12000	10000
Total	750	522	516	1328	915
Judges		J. FRANCO, H. VAN MAAREN, T. WALSH	F. BACCHUS, J. MARQUES-SILVA, H. KLEINE BUNING	A. VAN GELDER, A. BIERE, O. KULLMAN	E. SPECKENMEYER, G. SUTCLIFFE, L. ZHANG

La première (?) expérimentation SAT

L'ordinateur contre l'humain

« The superiority of the present procedure (i.e. DP) over those previously available is indicated in part by the fact that a formula on which Gilmore's routine for the IBM 704 causes the machine to compute for 21 minutes without obtaining a result was worked successfully by hand computation using the present method in 30 minutes » [Davis and Putnam, 1960]

« The well-formed formula (...) which was beyond the scope of Gilmore's program

» [Davis et al., 1962]

La première (?) expérimentation SAT

L'ordinateur contre l'humain

« The superiority of the present procedure (i.e. DP) over those previously available is indicated in part by the fact that a formula on which Gilmore's routine for the IBM 704 causes the machine to compute for 21 minutes without obtaining a result was worked successfully by hand computation using the present method in 30 minutes » [Davis and Putnam, 1960]

« The well-formed formula (...) which was beyond the scope of Gilmore's program was proved in under two minutes with the present program » [Davis et al., 1962]

La première (?) expérimentation SAT

L'ordinateur contre l'humain

« The superiority of the present procedure (i.e. DP) over those previously available is indicated in part by the fact that a formula on which Gilmore's routine for the IBM 704 causes the machine to compute for 21 minutes without obtaining a result was worked successfully by hand computation using the present method in 30 minutes » [Davis and Putnam, 1960]

« The well-formed formula (...) which was beyond the scope of Gilmore's program was proved in under two minutes with the present program » [Davis et al., 1962]

La première (?) expérimentation SAT

L'ordinateur contre l'humain

« The superiority of the present procedure (i.e. DP) over those previously available is indicated in part by the fact that a formula on which Gilmore's routine for the IBM 704 causes the machine to compute for 21 minutes without obtaining a result was worked successfully by hand computation using the present method in 30 minutes » [Davis and Putnam, 1960]

« The well-formed formula (...) which was beyond the scope of Gilmore's program was proved in under two minutes with the present program » [Davis et al., 1962]

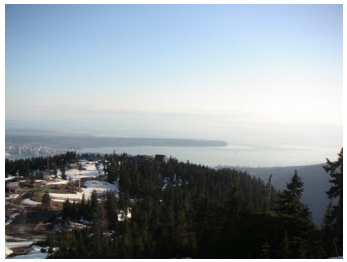
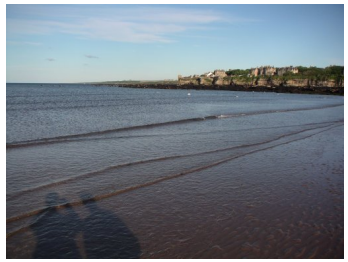
La première (?) expérimentation SAT

L'ordinateur contre l'humain

« The superiority of the present procedure (i.e. DP) over those previously available is indicated in part by the fact that a formula on which Gilmore's routine for the IBM 704 causes the machine to compute for 21 minutes without obtaining a result was worked successfully by hand computation using the present method in 30 minutes » [Davis and Putnam, 1960]

« The well-formed formula (...) which was beyond the scope of Gilmore's program was proved in under two minutes with the present program » [Davis et al., 1962]

Petite histoire de la compétition

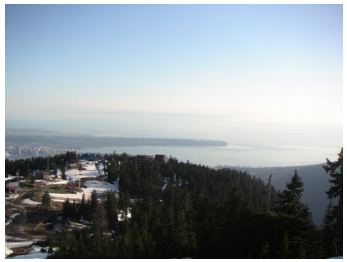
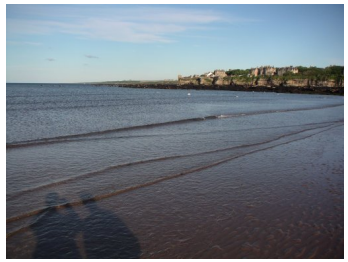

Avant 2001

- Paderborn en 1991/1992
- the Second DIMACS Challenge, en 1992/1993
- Beijing competition, en 1996.

Après 2001

Un nouveau cycle annuel de compétitions.

Petite histoire de la compétition

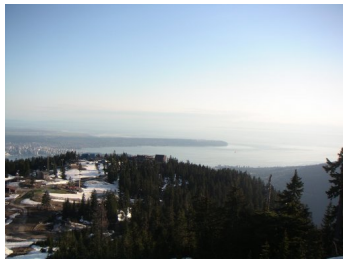
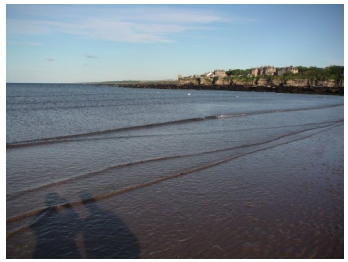

Avant 2001

- Paderborn en 1991/1992
- the Second DIMACS Challenge, en 1992/1993
- Beijing competition, en 1996.

Après 2001

Un nouveau cycle annuel de compétitions.

Petite histoire de la compétition


Avant 2001

- Paderborn en 1991/1992
- the Second DIMACS Challenge, en 1992/1993
- Beijing competition, en 1996.

Après 2001

Un nouveau cycle annuel de compétitions.

Quelques notations ou rappels

Logique propositionnelle

- $V = \{x_1, \ldots, x_n\}$ est l'ensemble des variables propositionnelles, l_i est un littéral, c-à-d une variable x_i ou sa négation $\neg x_i$.
- Une clause est une disjonction de littéraux $c_i = l_1 \vee l_2 \ldots \vee l_{n_i}$. Une clause unitaire ne contient qu'un seul littéral.
- Une formule Σ est sous forme normale conjonctive ($\mathcal{CNF}$) lorsque Σ s'écrit comme une conjonction de clauses $\Sigma = c_1 \wedge c_2 \ldots \wedge c_m$ (on peut aussi parler d'ensembles de clauses).

Règle de Résolution (Coupure) [Gentzen, 1934], [Robinson, 1965]

Soient $c_1 = (x \vee a_1 \vee a_2 \vee \ldots a_n)$ et $c_2 = (\neg x \vee b_1 \vee b_2 \vee \ldots b_m)$ $c = (a_1 \vee a_2 \vee \ldots a_n \vee b_1 \vee \ldots b_m)$ est appelée *résolvante* des clauses c_1 et c_2 (sur la variable x).

Quelques notations ou rappels

Logique propositionnelle

- $V = \{x_1, \ldots, x_n\}$ est l'ensemble des variables propositionnelles, l_i est un littéral, c-à-d une variable x_i ou sa négation $\neg x_i$.
- Une clause est une disjonction de littéraux $c_i = l_1 \vee l_2 \ldots \vee l_{n_i}$. Une clause unitaire ne contient qu'un seul littéral.
- Une formule Σ est sous forme normale conjonctive ($\mathcal{CNF}$) lorsque Σ s'écrit comme une conjonction de clauses $\Sigma = c_1 \wedge c_2 \ldots \wedge c_m$ (on peut aussi parler d'ensembles de clauses).

Règle de Résolution (Coupure) [Gentzen, 1934], [Robinson, 1965]

Soient $c_1 = (x \vee a_1 \vee a_2 \vee \ldots a_n)$ et $c_2 = (\neg x \vee b_1 \vee b_2 \vee \ldots b_m)$ $c = (a_1 \vee a_2 \vee \ldots a_n \vee b_1 \vee \ldots b_m)$ est appelée *résolvante* des clauses c_1 et c_2 (sur la variable x).

Quelques notations ou rappels

Logique propositionnelle

- $V = \{x_1, \ldots, x_n\}$ est l'ensemble des variables propositionnelles, l_i est un littéral, c-à-d une variable x_i ou sa négation $\neg x_i$.
- Une clause est une disjonction de littéraux $c_i = l_1 \vee l_2 \ldots \vee l_{n_i}$. Une clause unitaire ne contient qu'un seul littéral.
- Une formule Σ est sous forme normale conjonctive ($\mathcal{CNF}$) lorsque Σ s'écrit comme une conjonction de clauses $\Sigma = c_1 \wedge c_2 \ldots \wedge c_m$ (on peut aussi parler d'ensembles de clauses).

Règle de Résolution (Coupure) [Gentzen, 1934], [Robinson, 1965]

Soient $c_1 = (x \vee a_1 \vee a_2 \vee \ldots a_n)$ et $c_2 = (\neg x \vee b_1 \vee b_2 \vee \ldots b_m)$ $c = (a_1 \vee a_2 \vee \ldots a_n \vee b_1 \vee \ldots b_m)$ est appelée *résolvante* des clauses c_1 et c_2 (sur la variable x).

UNE PROCÉDURE SIMPLE POUR SAT

Algorithme 1 : BT($\Sigma, \mathcal{I}$)

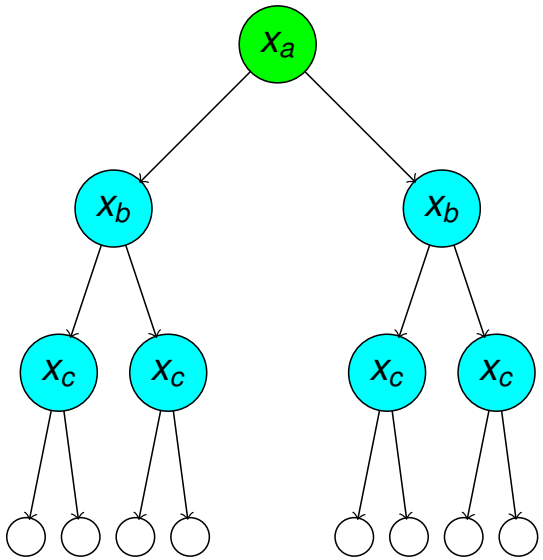
Data : Σ Une formule sous CNF
Data : $\mathcal{I}$ Une interprétation partielle
Result : SAT si un modèle est trouvé, UNSAT sinon
begin
    if $\Sigma|\mathcal{I}$ *est vide* **then** retourner SAT ;
    if $\Sigma|\mathcal{I}$ *contient* $\bot$ **then** retourner UNSAT ;
    Soit l un littéral de $\Sigma|\mathcal{I}$;
    if *BT(*$\Sigma, \mathcal{I}.l$*) retourne* SAT **then** retourner SAT ;
    retourner BT($\Sigma, \mathcal{I}.\neg l$)
end

$\Sigma|\mathcal{I}$ est la formule réduite par $\mathcal{I}$, dans laquelle toutes les clauses satisfaites par $\mathcal{I}$ sont retirées, et tous les littéraux faux par $\mathcal{I}$ sont retirés de leurs clauses.

En avant vers le retour arrière

Déroulement à l'ancienne... mais pas trop

$\neg x_a \vee x_b \vee x_c$
$x_a \vee x_c \vee x_d$
$x_a \vee x_c \vee \neg x_d$
$x_a \vee \neg x_c \vee x_d$
$x_a \vee \neg x_c \vee \neg x_d$
$\neg x_b \vee \neg x_c \vee x_d$
$\neg x_a \vee x_b \neg x_c$
$\neg x_a \vee \neg x_b \vee x_c$



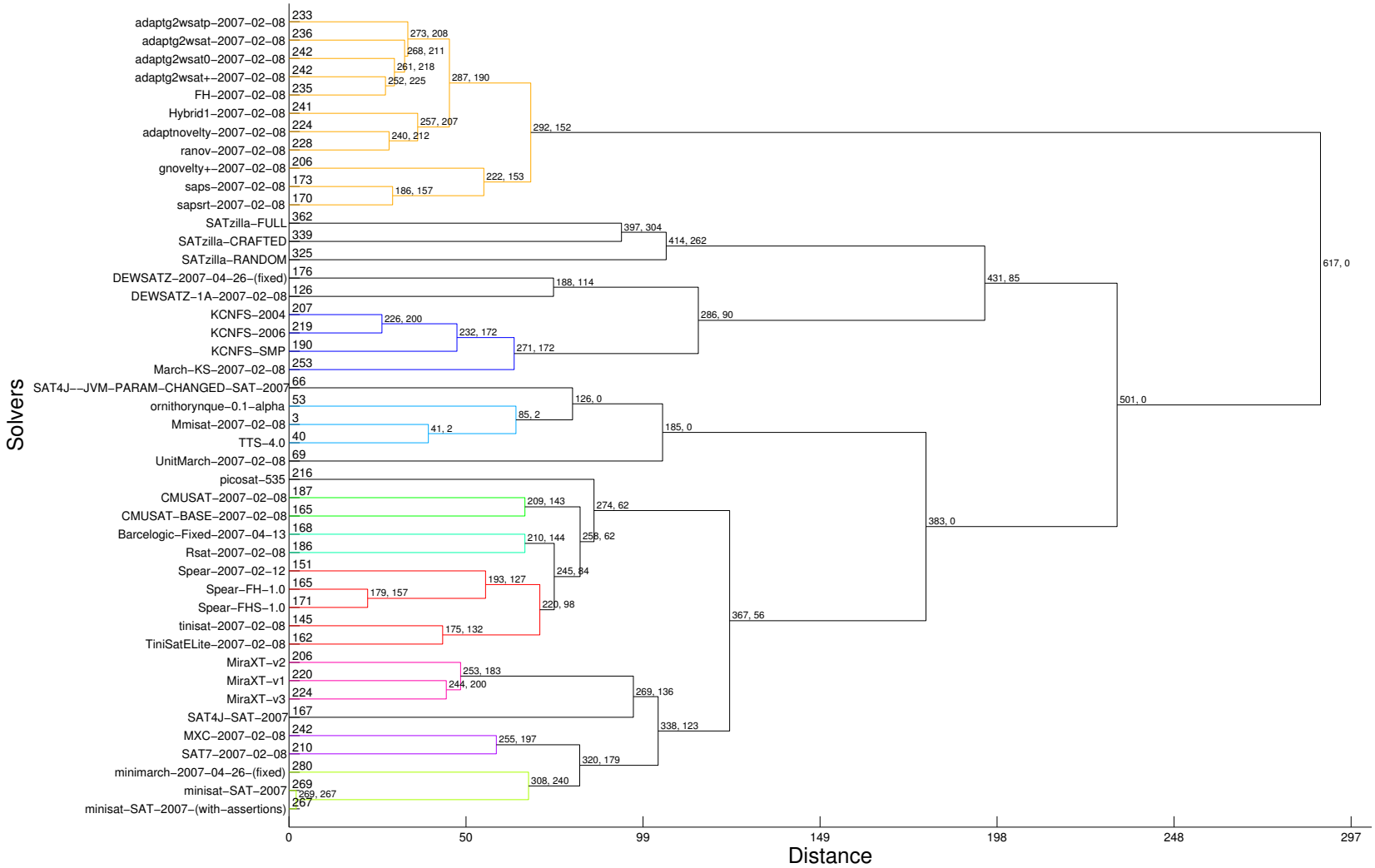
La procédure dure

Son expression récursive

Algorithme 2 : DPLL($\Sigma, \mathcal{I}$) (La procédure DPL-62)

Données : Σ une formule CNF
Données : $\mathcal{I}$ une interprétation partielle
Résultat : SAT si un modèle est trouvé, UNSAT sinon
début
  si $\Sigma|\mathcal{I}$ *est vide* **alors** retourner SAT ;
  si $\Sigma|\mathcal{I}$ *contient* $\bot$ **alors** retourner UNSAT ;
  si $\Sigma|\mathcal{I}$ *contient une clause unitaire* l **alors** retourner DPLL($\Sigma, \mathcal{I}.l$);
  si $\Sigma|\mathcal{I}$ *contient un littéral pur* l **alors** retourner DPLL($\Sigma, \mathcal{I}.l$);
  Soit l un littéral de $\Sigma|\mathcal{I}$;
  si *BT*($\Sigma, \mathcal{I}.l$) *retourne* SAT **alors** retourner SAT ;
  retourner BT($\Sigma, \mathcal{I}.\neg l$)
fin

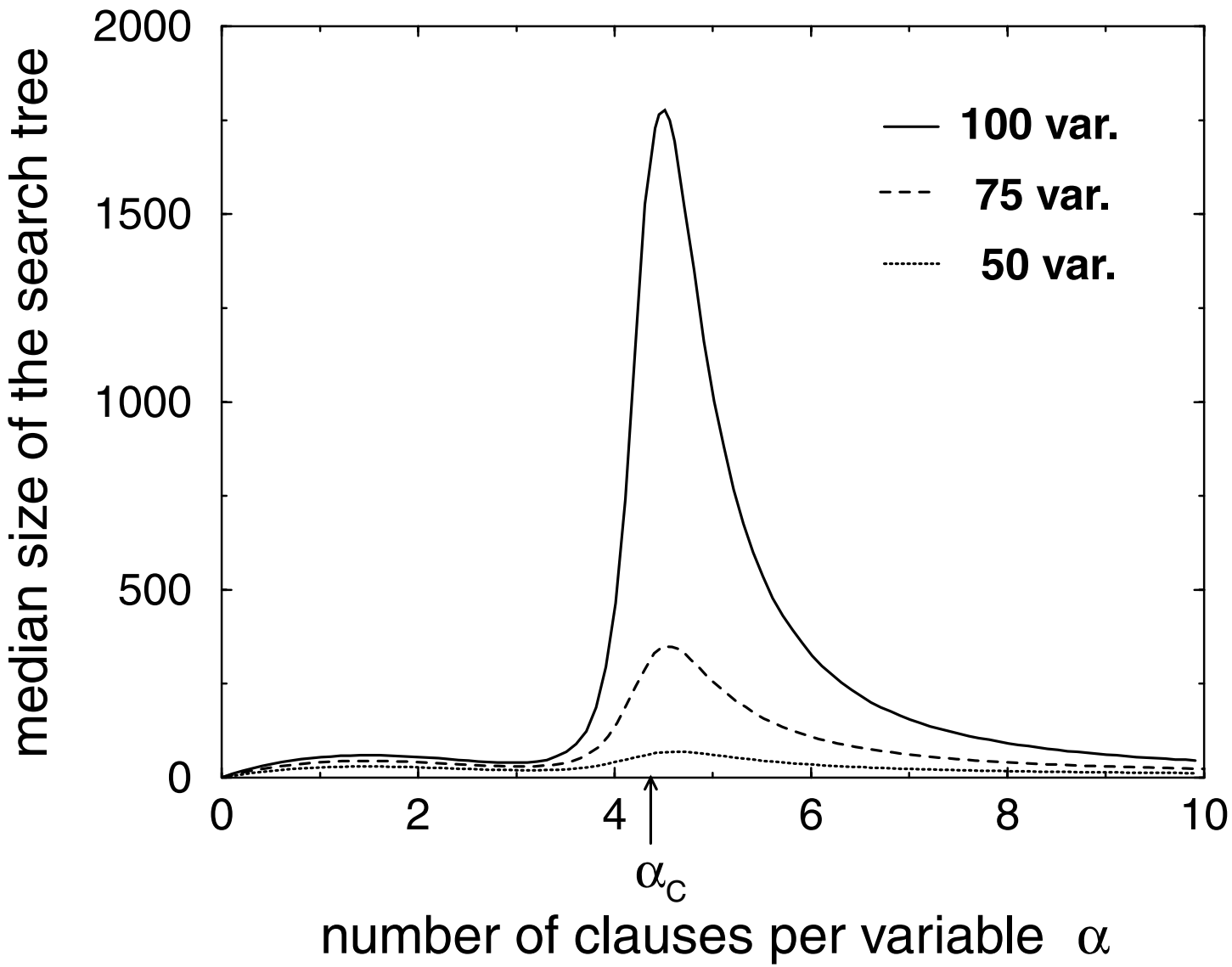
REGROUPEMENT DES SOLVEURS



Plan de l'exposé

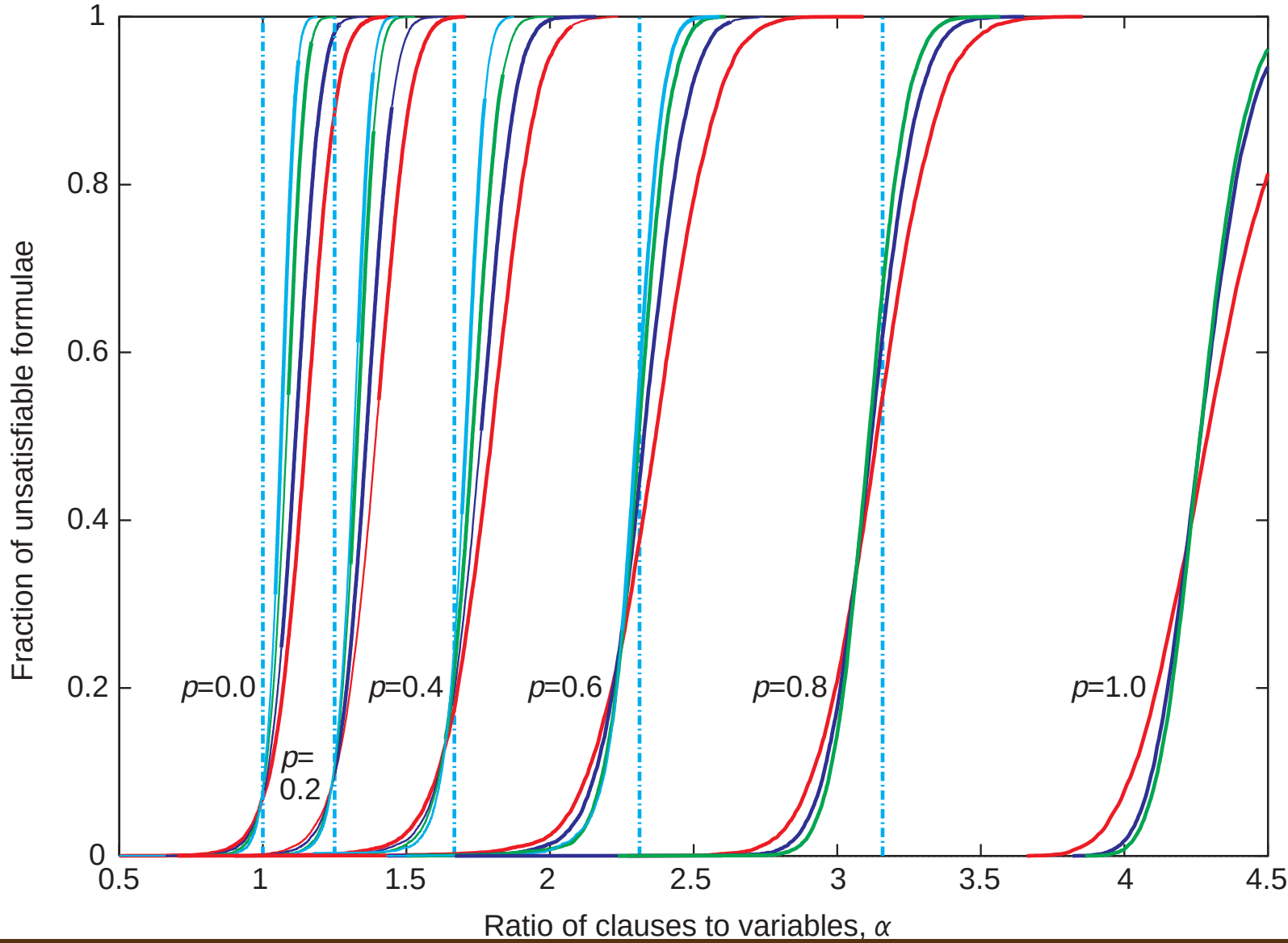
Plan de l'exposé

LE SEUIL, FAMEUX SEUIL

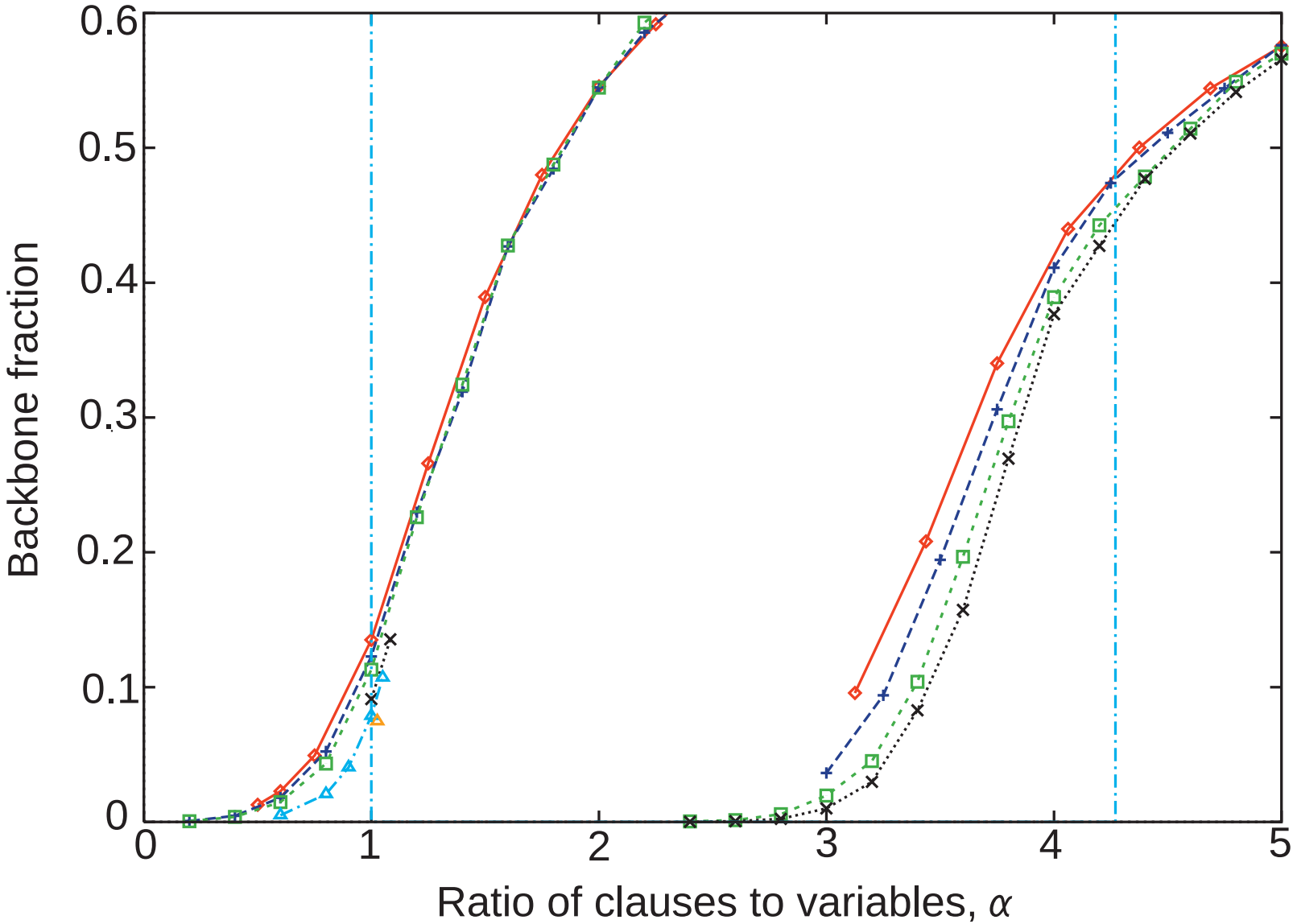


Le (superbe) seuil SAT...

De 3SAT vers (2+p)SAT, un autre seuil



Les backbones des instances aléatoires



SATZ DE A À Z

EN UN SEUL TRANSPARENT [LI AND ANBULAGAN, 1997]

MISER SUR LA PROPAGATION UNITAIRE

La propagation unitaire est la base de la performance des solveurs.

PROBLÈME Difficile de savoir *a priori* combien de propagation en cascades vont être effectivement faites.

SOLUTION Faire un lookahead pour voir vraiment ce qu'il se passe... sur certaines variables seulement.

Heuristique : choisir la variable équilibrant au mieux (mais simplifiant le plus par PU) les deux sous arbres.

UNE PREUVE EXPÉRIMENTALE

Il a été observé [Li and Gérard, 2000] que l'on ne pouvait pas améliorer ses résultats de manière significative, uniquement avec l'heuristique... En pratique, cela a été contredit pourtant par kcnfs.

SATZ DE A À Z

EN UN SEUL TRANSPARENT [LI AND ANBULAGAN, 1997]

MISER SUR LA PROPAGATION UNITAIRE

La propagation unitaire est la base de la performance des solveurs.

PROBLÈME Difficile de savoir *a priori* combien de propagation en cascades vont être effectivement faites.

SOLUTION Faire un lookahead pour voir vraiment ce qu'il se passe... sur certaines variables seulement.

Heuristique : choisir la variable équilibrant au mieux (mais simplifiant le plus par PU) les deux sous arbres.

UNE PREUVE EXPÉRIMENTALE

Il a été observé [Li and Gérard, 2000] que l'on ne pouvait pas améliorer ses résultats de manière significative, uniquement avec l'heuristique... En pratique, cela a été contredit pourtant par kcnfs.

SATZ DE A À Z

EN UN SEUL TRANSPARENT [LI AND ANBULAGAN, 1997]

MISER SUR LA PROPAGATION UNITAIRE

La propagation unitaire est la base de la performance des solveurs.

PROBLÈME Difficile de savoir *a priori* combien de propagation en cascades vont être effectivement faites.

SOLUTION Faire un lookahead pour voir vraiment ce qu'il se passe... sur certaines variables seulement.

Heuristique : choisir la variable équilibrant au mieux (mais simplifiant le plus par PU) les deux sous arbres.

UNE PREUVE EXPÉRIMENTALE

Il a été observé [Li and Gérard, 2000] que l'on ne pouvait pas améliorer ses résultats de manière significative, uniquement avec l'heuristique... En pratique, cela a été contredit pourtant par kcnfs.

KCNFS : EXPLOITATIONS DES *backbones*

MAIS QUE SONT LES BACKBONES ?

Ce sont des variables qui gardent **la même valeur** dans **tous** les modèles de la formule. Impact :

- Se tromper sur leur valeur en haut de l'arbre de recherche est catastrophique
- Peut expliquer une distribution de temps non *normale*

IDÉE : RECHERCHE DES BACKBONES [DUBOIS AND DEQUEN, 2001]

La fonction heuristique h permet de brancher sur les variables ayant les plus grandes chances d'avoir la même valeur dans tout le sous-arbre exploré (prise en compte de tout l'arbre).
Inconvénient : le calcul de l'heuristique est coûteux.

En pratique, en 2001, Kcnfs arrive à résoudre des instances de 700 variables au seuil, en 26 jours (5 jours pour 650 variables).

KCNFS : EXPLOITATIONS DES *backbones*

MAIS QUE SONT LES BACKBONES ?

Ce sont des variables qui gardent **la même valeur** dans **tous** les modèles de la formule. Impact :

- Se tromper sur leur valeur en haut de l'arbre de recherche est catastrophique
- Peut expliquer une distribution de temps non *normale*

IDÉE : RECHERCHE DES BACKBONES [DUBOIS AND DEQUEN, 2001]

La fonction heuristique *h* permet de brancher sur les variables ayant les plus grandes chances d'avoir la même valeur dans tout le sous-arbre exploré (prise en compte de tout l'arbre).
Inconvénient : le calcul de l'heuristique est coûteux.

En pratique, en 2001, Kcnfs arrive à résoudre des instances de 700 variables au seuil, en 26 jours (5 jours pour 650 variables).

KCNFS : EXPLOITATIONS DES *backbones*

MAIS QUE SONT LES BACKBONES ?

Ce sont des variables qui gardent **la même valeur** dans **tous** les modèles de la formule. Impact :

- Se tromper sur leur valeur en haut de l'arbre de recherche est catastrophique
- Peut expliquer une distribution de temps non *normale*

IDÉE : RECHERCHE DES BACKBONES [DUBOIS AND DEQUEN, 2001]

La fonction heuristique h permet de brancher sur les variables ayant les plus grandes chances d'avoir la même valeur dans tout le sous-arbre exploré (prise en compte de tout l'arbre).

Inconvénient : le calcul de l'heuristique est coûteux.

En pratique, en 2001, Kcnfs arrive à résoudre des instances de 700 variables au seuil, en 26 jours (5 jours pour 650 variables).

KCNFS : EXPLOITATIONS DES *backbones*

MAIS QUE SONT LES BACKBONES ?

Ce sont des variables qui gardent **la même valeur** dans **tous** les modèles de la formule. Impact :

- Se tromper sur leur valeur en haut de l'arbre de recherche est catastrophique
- Peut expliquer une distribution de temps non *normale*

IDÉE : RECHERCHE DES BACKBONES [DUBOIS AND DEQUEN, 2001]

La fonction heuristique h permet de brancher sur les variables ayant les plus grandes chances d'avoir la même valeur dans tout le sous-arbre exploré (prise en compte de tout l'arbre).
Inconvénient : le calcul de l'heuristique est coûteux.

En pratique, en 2001, Kcnfs arrive à résoudre des instances de 700 variables au seuil, en 26 jours (5 jours pour 650 variables).

KCNFS : EXPLOITATIONS DES *backbones*

MAIS QUE SONT LES BACKBONES ?

Ce sont des variables qui gardent **la même valeur** dans **tous** les modèles de la formule. Impact :

- Se tromper sur leur valeur en haut de l'arbre de recherche est catastrophique
- Peut expliquer une distribution de temps non *normale*

IDÉE : RECHERCHE DES BACKBONES [DUBOIS AND DEQUEN, 2001]

La fonction heuristique h permet de brancher sur les variables ayant les plus grandes chances d'avoir la même valeur dans tout le sous-arbre exploré (prise en compte de tout l'arbre).
Inconvénient : le calcul de l'heuristique est coûteux.

En pratique, en 2001, Kcnfs arrive à résoudre des instances de 700 variables au seuil, en 26 jours (5 jours pour 650 variables).

FIGURE DES RÉSULTATS COMPÉTITION SAT 2007

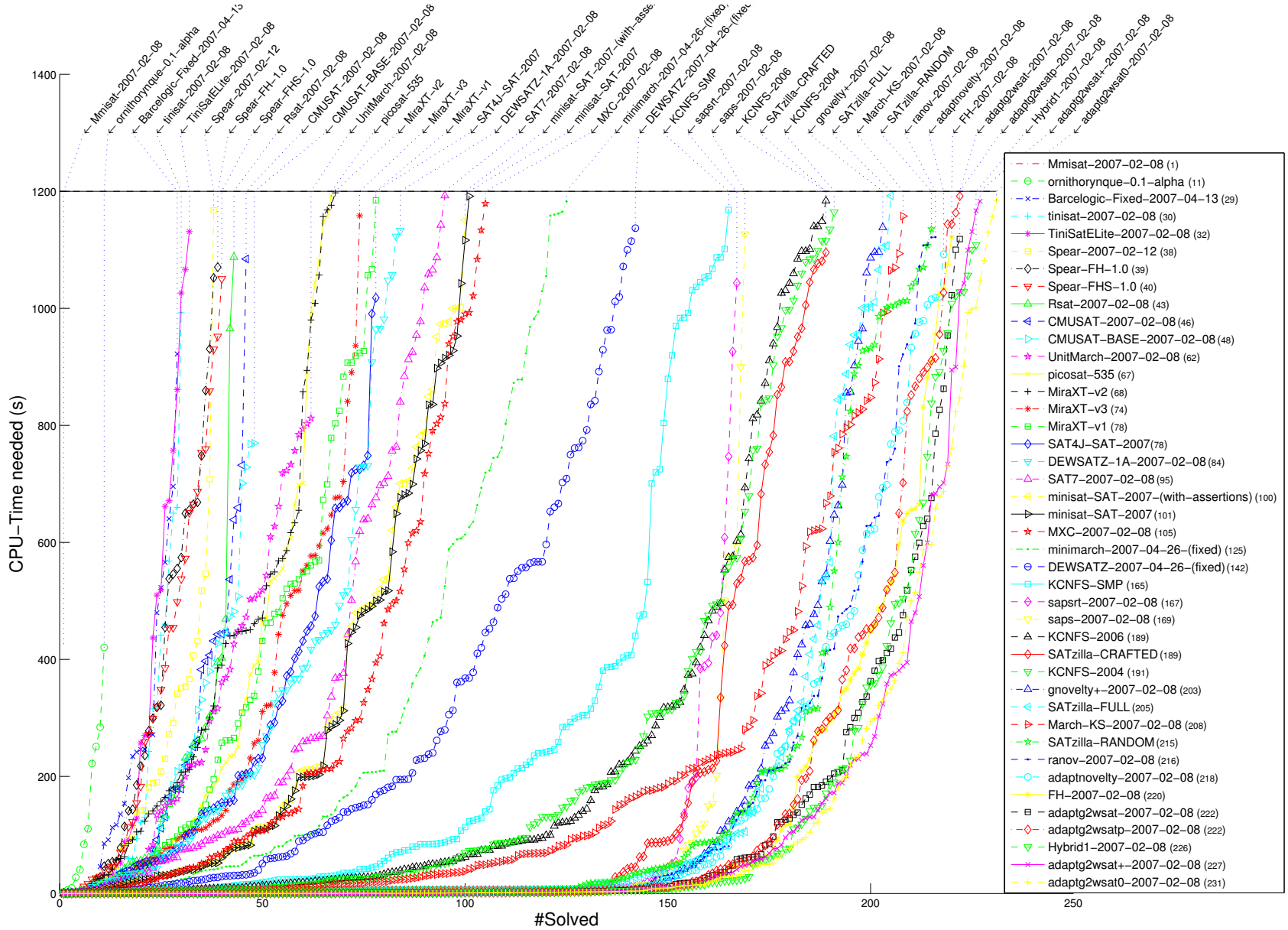
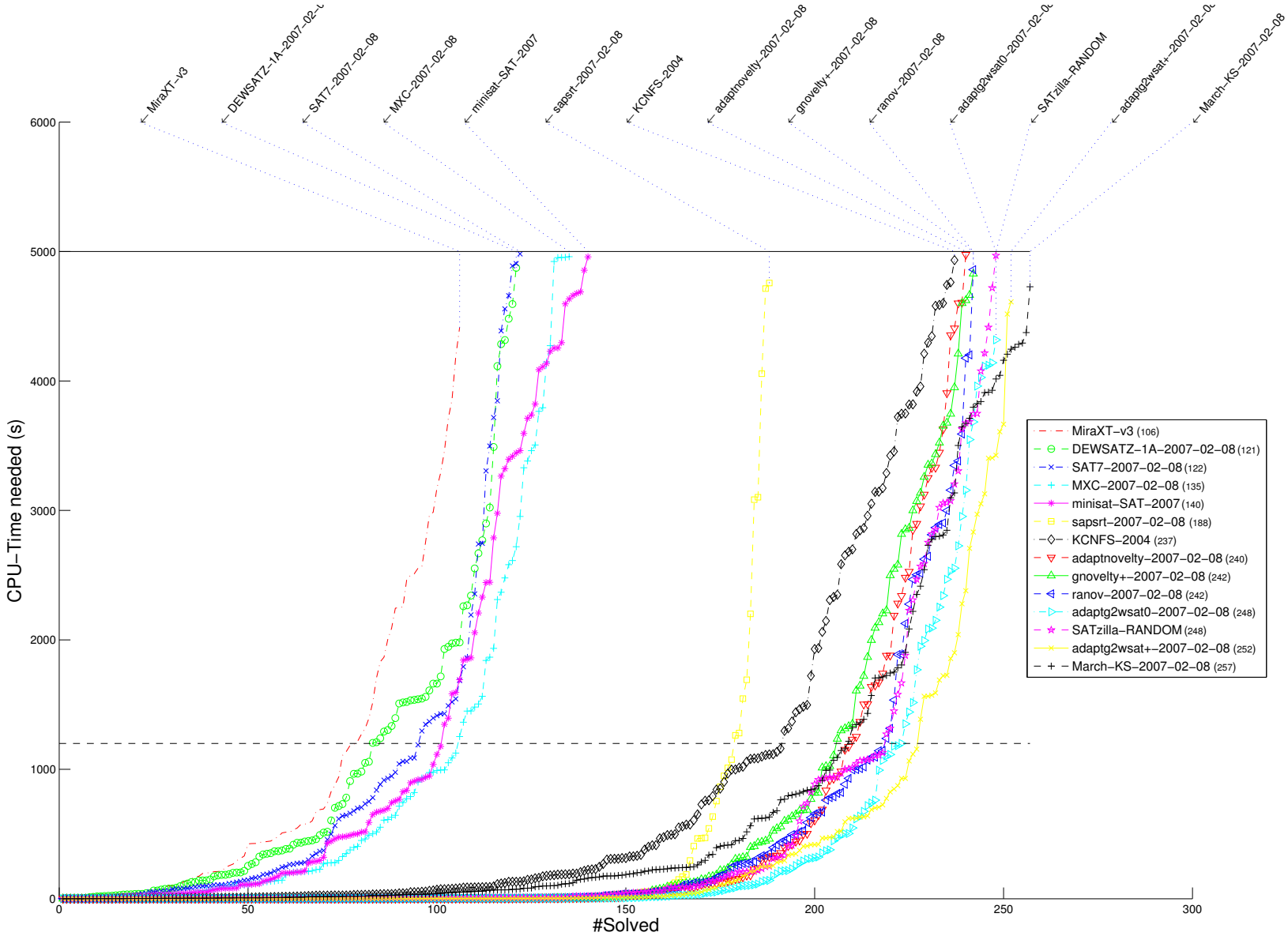


FIGURE DES RÉSULTATS COMPÉTITION SAT 2007



2007 : KCNFS EST DÉTRONÉ !

DEPUIS QUELQUES MOIS SEULEMENT....

Kcnfs est resté imbattu (et de loin) dans les compétitions SAT sur la catégorie UNSAT... Jusqu'en 2007.

DANS MARCH-KS

- Dans certains cas, un double lookahead est lancé
- Prise en compte de la dispersion particulière des solutions lors du parcours de l'arbre de recherche
- Certains raisonnements d'équivalence
- Structures de données / Caches efficaces...

2007 : KCNFS EST DÉTRONÉ !

DEPUIS QUELQUES MOIS SEULEMENT....

Kcnfs est resté imbattu (et de loin) dans les compétitions SAT sur la catégorie UNSAT... Jusqu'en 2007.

DANS MARCH-KS

- Dans certains cas, un double lookahead est lancé
- Prise en compte de la dispersion particulière des solutions lors du parcours de l'arbre de recherche
- Certains raisonnements d'équivalence
- Structures de données / Caches efficaces...

RECHERCHES LOCALES

GSAT [SELMAN ET AL., 1992], WALKSAT [SELMAN ET AL., 1996], ADAPTNOVELTY

De grands espoirs lorsque benchs aléatoires et benchs structurés n'étaient pas différenciés dans les évaluations.

IDÉE (GÉNÉRALE) DE LA RECHERCHE LOCALE

Tant que non *NombreMaxEssaisAtteints*
  Générer au hasard une solution potentielle au problème
  Tant que non *DeplacementsMaxAtteints*
    Générer un ensemble de voisins de ces solutions
    Remplacer la solution actuelle par l'une de ses solutions
    Mettre à jour la meilleure solution *Best* vue
    Si SolutionTrouve Alors Retourner *Best*
  Fin Tant que
Fin Tant que

Quelques travaux tentent d'adapter la recherche locale aux exemples structurés, en restreignant les voisinages aux *backdoors*.

Recherches locales

GSAT [Selman et al., 1992], Walksat [Selman et al., 1996], adaptnovelty

De grands espoirs lorsque benchs aléatoires et benchs structurés n'étaient pas différenciés dans les évaluations.

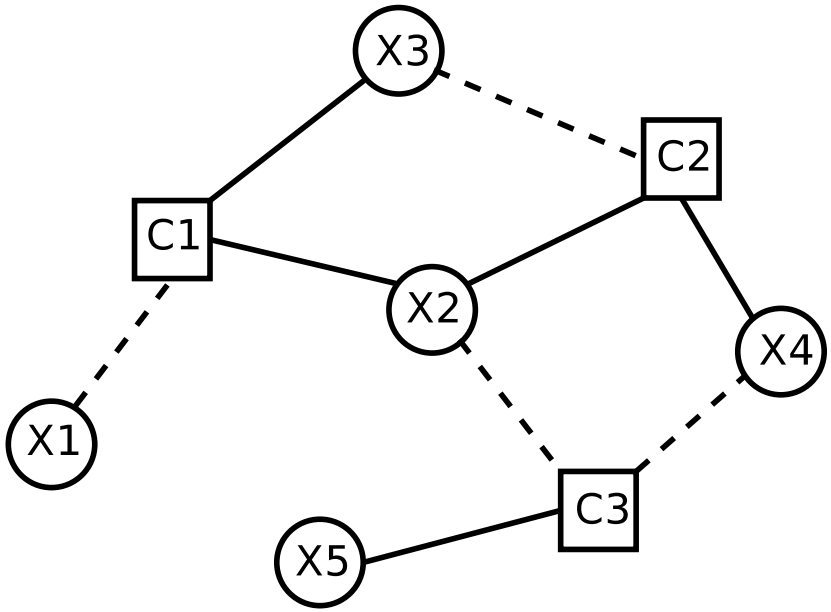
Idée (générale) de la recherche locale

Tant que non *NombreMaxEssaisAtteints*
  Générer au hasard une solution potentielle au problème
  Tant que non *DeplacementsMaxAtteints*
    Générer un ensemble de voisins de ces solutions
    Remplacer la solution actuelle par l'une de ses solutions
    Mettre à jour la meilleure solution *Best* vue
    Si SolutionTrouve Alors Retourner *Best*
  Fin Tant que
Fin Tant que

Quelques travaux tentent d'adapter la recherche locale aux exemples structurés, en restreignant les voisinages aux *backdoors*.

Un survol de la Survey Propagations

Les valeurs de certaines variables (des backbones) sont trouvées dans une première phase. Puis une recherche locale est lancée après l'affectation de ces rares variables.



L'algorithme converge si le graphe est sous forme d'arbre. Dans les très grandes formules, il y a de grandes chances que des sous parties aient ces propriétés.

ALÉATOIRE UNSAT, TOUJOURS UN CHALLENGE

AVANT LE SEUIL très facile pour la recherche locale.

JUSTE AVANT LE SEUIL très facile pour SP (sur les grandes instances)

BIEN APRÈS LE SEUIL facile pour des algos spécialisés (basés sur la recherche de clauses XOR)

Malgré de gros progrès, plus beaucoup de nouveau pour résoudre efficacement les UNSAT aléatoires au seuil.

QUESTION PROVOCATRICE

Tenter de résoudre les instances aléatoire a-t-il un sens ?

- Oui, du point de vue théorique (et comment !)
- Mais du point de vue pratique...

ALÉATOIRE UNSAT, TOUJOURS UN CHALLENGE

AVANT LE SEUIL très facile pour la recherche locale.

JUSTE AVANT LE SEUIL très facile pour SP (sur les grandes instances)

BIEN APRÈS LE SEUIL facile pour des algos spécialisés (basés sur la recherche de clauses XOR)

Malgré de gros progrès, plus beaucoup de nouveau pour résoudre efficacement les UNSAT aléatoires au seuil.

QUESTION PROVOCATRICE

Tenter de résoudre les instances aléatoire a-t-il un sens ?

- Oui, du point de vue théorique (et comment !)
- Mais du point de vue pratique...

ALÉATOIRE UNSAT, TOUJOURS UN CHALLENGE

AVANT LE SEUIL très facile pour la recherche locale.

JUSTE AVANT LE SEUIL très facile pour SP (sur les grandes instances)

BIEN APRÈS LE SEUIL facile pour des algos spécialisés (basés sur la recherche de clauses XOR)

Malgré de gros progrès, plus beaucoup de nouveau pour résoudre efficacement les UNSAT aléatoires au seuil.

QUESTION PROVOCATRICE

Tenter de résoudre les instances aléatoire a-t-il un sens ?

- Oui, du point de vue théorique (et comment !)
- Mais du point de vue pratique...

Plan de l'exposé

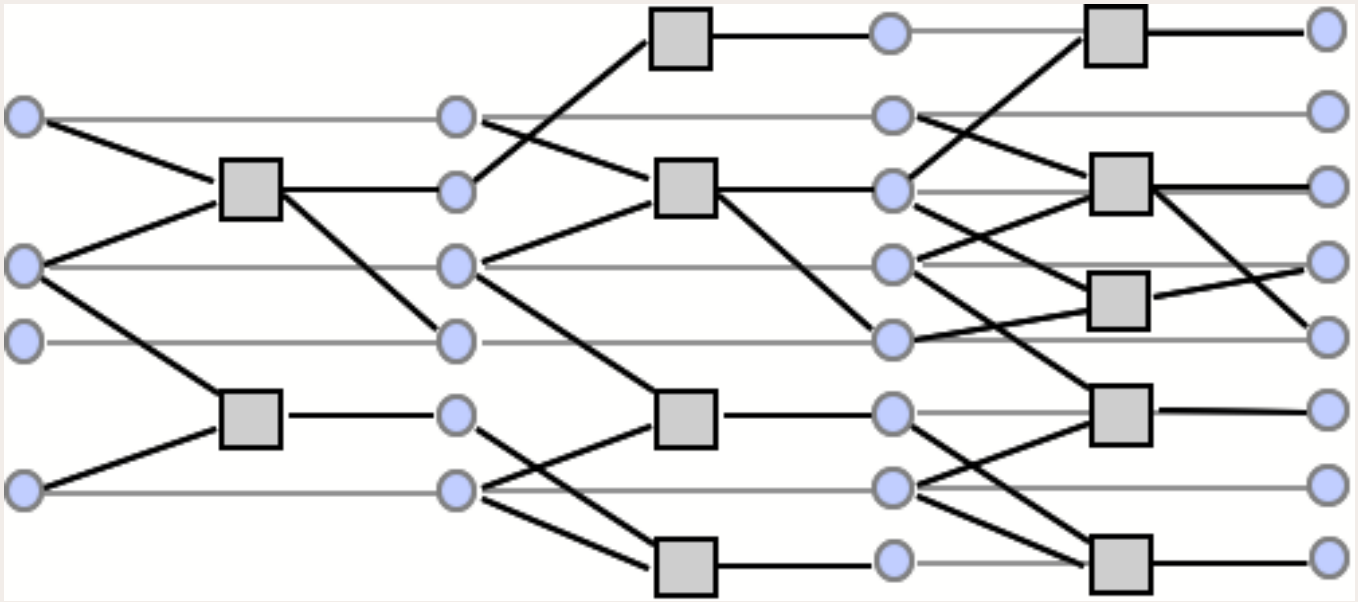
UNE PREMIÈRE RÉVOLUTION

37 ANS DE STRIPS, UNE RÉVOLUTION : GRAPHPLAN

FORMULATION STRIPS

```
MOVE(x,y,z)
PRE: CLEAR(x), ON(x,y), CLEAR(z)
ADD: CLEAR(y), ON(x,z)
DEL: CLEAR(z), ON(x,y)
```

PRINCIPE (GRAPHIQUES) DE GRAPHPLAN [95]



UNE PREMIÈRE RÉVOLUTION

37 ANS DE STRIPS, UNE RÉVOLUTION : GRAPHPLAN

1. Génère un GraphPlan de longueur k
2. Transforme les contraintes du graphe en clauses. Chaque instance d'une action ou d'un fait à chaque moment est une proposition.
3. Utilise un solveur SAT
4. Si on ne trouve pas de solution (UNSAT ou TimeOut), incrémente k et recommence
5. Si on trouve une solution, transforme la solution en solution de planificiation
6. Simplifie la solution.

UNE PREMIÈRE RÉVOLUTION

37 ANS DE STRIPS, UNE RÉVOLUTION : GRAPHPLAN

1. Génère un GraphPlan de longueur k
2. Transforme les contraintes du graphe en clauses. Chaque instance d'une action ou d'un fait à chaque moment est une proposition.
3. Utilise un solveur SAT
4. Si on ne trouve pas de solution (UNSAT ou TimeOut), incrémente k et recommence
5. Si on trouve une solution, transforme la solution en solution de planificiation
6. Simplifie la solution.

UNE PREMIÈRE RÉVOLUTION

37 ANS DE STRIPS, UNE RÉVOLUTION : GRAPHPLAN

1. Génère un GraphPlan de longueur k
2. Transforme les contraintes du graphe en clauses. Chaque instance d'une action ou d'un fait à chaque moment est une proposition.
3. Utilise un solveur SAT
4. Si on ne trouve pas de solution (UNSAT ou TimeOut), incrémente k et recommence
5. Si on trouve une solution, transforme la solution en solution de planificiation
6. Simplifie la solution.

UNE PREMIÈRE RÉVOLUTION

37 ANS DE STRIPS, UNE RÉVOLUTION : GRAPHPLAN

1. Génère un GraphPlan de longueur k
2. Transforme les contraintes du graphe en clauses. Chaque instance d'une action ou d'un fait à chaque moment est une proposition.
3. Utilise un solveur SAT
4. Si on ne trouve pas de solution (UNSAT ou TimeOut), incrémente k et recommence
5. Si on trouve une solution, transforme la solution en solution de planificiation
6. Simplifie la solution.

UNE PREMIÈRE RÉVOLUTION

37 ANS DE STRIPS, UNE RÉVOLUTION : GRAPHPLAN

1. Génère un GraphPlan de longueur k
2. Transforme les contraintes du graphe en clauses. Chaque instance d'une action ou d'un fait à chaque moment est une proposition.
3. Utilise un solveur SAT
4. Si on ne trouve pas de solution (UNSAT ou TimeOut), incrémente k et recommence
5. Si on trouve une solution, transforme la solution en solution de planificiation
6. Simplifie la solution.

UNE PREMIÈRE RÉVOLUTION

37 ANS DE STRIPS, UNE RÉVOLUTION : GRAPHPLAN

1. Génère un GraphPlan de longueur k
2. Transforme les contraintes du graphe en clauses. Chaque instance d'une action ou d'un fait à chaque moment est une proposition.
3. Utilise un solveur SAT
4. Si on ne trouve pas de solution (UNSAT ou TimeOut), incrémente k et recommence
5. Si on trouve une solution, transforme la solution en solution de planificiation
6. Simplifie la solution.

Recette de planification

Un peu d'imagination

Supposons que vous vouliez préparer un petit-déjeuner surprise à votre fiançée, accompagné d'un cadeau à emballer. Bien entendu, elle dort.

Etat initial : *poubelles*, *mainPropres*, *silence*
Etat Final : *dej*, *cadeau*, *nonpoubelles*

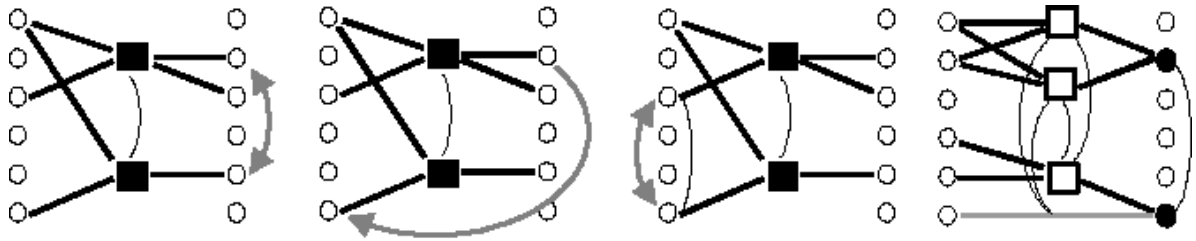
Action	Préconditions	Effets
Cuisiner()	mainPropres	dej
Envelopper()	silence	cadeau
Sortir()	aucunes	non poubelles, non mainPropres
Chariot()	aucunes	non poubelles, non silence

ENCODAGE SURPRISE

Idée	Encodage
Conditions initiales	$poubelles^0 \wedge mainPropres^0 \wedge silence^0$
Conditions finales, après k étapes	$dej^2 \wedge cadeau^2 \wedge \neg poubelles^2$
Exemple d'encodage (les faits qui changent)	$Cuisiner^0 \rightarrow mainPropres^0 \wedge dej^1$ $Cuisiner^1 \rightarrow mainPropres^1 \wedge dej^2$
Ne pas oublier ceux qui ne changent pas	$Cuisiner^0 \wedge pouvelles^0 \rightarrow poubelles^1$
Tout changement d'état s'explique	$poubelles^0 \wedge \neg poubelles^1 \rightarrow Chariot^0 \vee Sortir^0$
Faire au moins une action à chaque étape	$Cuisiner^0 \vee Envelopper^0 \vee Sortir^0 \vee Chariot^0$
Empêcher les exclusions mutuelles implicites	$\neg Cuisiner^0 \vee \neg Sortir^0$

D'AUTRES IDÉES

Des exclusions mutuelles



QUELQUES AUTRES IDÉES

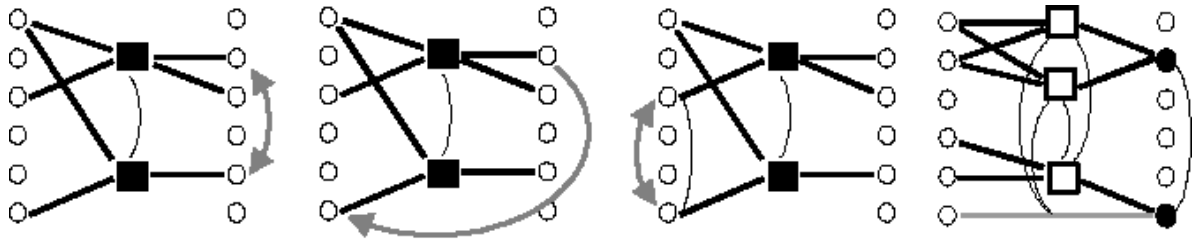
- Découper les actions avec beaucoup d'arguments
  embarquer(*voilier*, 8*h*15, *Runion*, *capitaine*)
- Encoder le temps et/ou diverses mesures diverses en *log*

1992 la recherche locale donnent de bons résultats
2001 bénéficie de zchaff, berkmin et siege
2004 premier prix de la compétition (domaines prop.)
2006 idem, ex aequo avec un autre SAT-planificateur

A ouvert la porte aux encodages SAT de problèmes *industriels*.

D'AUTRES IDÉES

Des exclusions mutuelles



QUELQUES AUTRES IDÉES

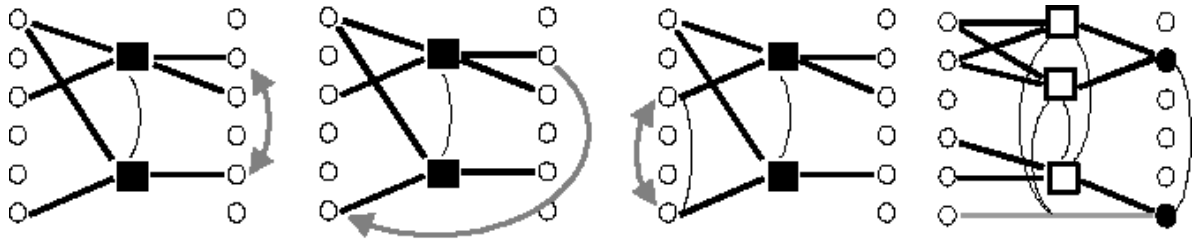
- Découper les actions avec beaucoup d'arguments
  embarquer(*voilier*, 8*h*15, *Runion*, *capitaine*)
- Encoder le temps et/ou diverses mesures diverses en *log*

1992 la recherche locale donnent de bons résultats
2001 bénéficie de zchaff, berkmin et siege
2004 premier prix de la compétition (domaines prop.)
2006 idem, ex aequo avec un autre SAT-planificateur

A ouvert la porte aux encodages SAT de problèmes *industriels*.

D'AUTRES IDÉES

Des exclusions mutuelles



QUELQUES AUTRES IDÉES

- Découper les actions avec beaucoup d'arguments
  embarquer(*voilier*, 8*h*15, *Runion*, *capitaine*)
- Encoder le temps et/ou diverses mesures diverses en *log*

1992 la recherche locale donnent de bons résultats
2001 bénéficie de zchaff, berkmin et siege
2004 premier prix de la compétition (domaines prop.)
2006 idem, ex aequo avec un autre SAT-planificateur

A ouvert la porte aux encodages SAT de problèmes *industriels*.

Principes du Model Checking

Étant donné un automate décrivant les états possibles d'un système

Vérifier qu'un état n'est jamais atteint

C'est généralement la recherche de bug. Un état spécial « erreur » est utilisé dans la modélisation.

Vérifier que le système ne bloque jamais

Tout état doit être accessible depuis tout état, à n'importe quel moment du futur (impossible de construire une boucle infinie).

A des liens étroits avec les logiques temporelles.
Avant SAT, les BDD étaient utilisés pour résoudre ces problèmes.

Principes du (Bounded) Model Checking

Comme pour la planification, on se borne à un k fixé, qu'on incrémente à loisir.

- L'automate est représenté par la fonction caractéristique T de ses transitions entre états.
  Exemple (2-bit additionneur) : $(a' \leftrightarrow \neg a) \wedge (b' \leftrightarrow a \oplus b)$
- La propriété à vérifier est une formule logique sur l'état du système
  Exemple : $a \wedge b$ (l'état (11) peut-il être atteint ?)
- L'état initial sera une affectation le codant au temps 0

DÉROULEMENT DE BOUCLES

Vérifions si l'état (11) est atteignable en deux itérations dans notre 2-bit additionneur, depuis (00)

$I(s_0) = \neg a_0 \wedge \neg b_0$

$T(s_0, s_1) = (a_1 \leftrightarrow \neg a_0) \wedge (b_1 \leftrightarrow a_0 \oplus b_0)$

$T(s_1, s_2) = (a_2 \leftrightarrow \neg a_1) \wedge (b_2 \leftrightarrow a_1 \oplus b_1)$

$p(s_2) = a_2 \wedge b_2$

$p(s_0) = a_0 \wedge b_0$

$p(s_2) = a_1 \wedge b_1$

AU FINAL

$(\neg a_0 \wedge \neg b_0) \wedge ((a_1 \leftrightarrow \neg a_0) \wedge (b_1 \leftrightarrow a_0 \oplus b_0)) \wedge ((a_2 \leftrightarrow \neg a_1) \wedge (b_2 \leftrightarrow a_1 \oplus b_1)) \wedge (a_0 \wedge b_0) \wedge (a_1 \wedge b_1) \wedge (a_2 \wedge b_2)$ est-elle satisfiable ?

DÉROULEMENT DE BOUCLES

Vérifions si l'état (11) est atteignable en deux itérations dans notre 2-bit additionneur, depuis (00)

$I(s_0) = \neg a_0 \wedge \neg b_0$

$T(s_0, s_1) = (a_1 \leftrightarrow \neg a_0) \wedge (b_1 \leftrightarrow a_0 \oplus b_0)$

$T(s_1, s_2) = (a_2 \leftrightarrow \neg a_1) \wedge (b_2 \leftrightarrow a_1 \oplus b_1)$

$p(s_2) = a_2 \wedge b_2$

$p(s_0) = a_0 \wedge b_0$

$p(s_2) = a_1 \wedge b_1$

AU FINAL

$(\neg a_0 \wedge \neg b_0) \wedge ((a_1 \leftrightarrow \neg a_0) \wedge (b_1 \leftrightarrow a_0 \oplus b_0)) \wedge ((a_2 \leftrightarrow \neg a_1) \wedge (b_2 \leftrightarrow a_1 \oplus b_1)) \wedge (a_0 \wedge b_0) \wedge (a_1 \wedge b_1) \wedge (a_2 \wedge b_2)$ est-elle satisfiable ?

Déroulement de boucles

Vérifions si l'état (11) est atteignable en deux itérations dans notre 2-bit additionneur, depuis (00)

$I(s_0) = \neg a_0 \wedge \neg b_0$
$T(s_0, s_1) = (a_1 \leftrightarrow \neg a_0) \wedge (b_1 \leftrightarrow a_0 \oplus b_0)$
$T(s_1, s_2) = (a_2 \leftrightarrow \neg a_1) \wedge (b_2 \leftrightarrow a_1 \oplus b_1)$
$p(s_2) = a_2 \wedge b_2$
$p(s_0) = a_0 \wedge b_0$
$p(s_2) = a_1 \wedge b_1$

Au final

$(\neg a_0 \wedge \neg b_0) \wedge ((a_1 \leftrightarrow \neg a_0) \wedge (b_1 \leftrightarrow a_0 \oplus b_0)) \wedge ((a_2 \leftrightarrow \neg a_1) \wedge (b_2 \leftrightarrow a_1 \oplus b_1)) \wedge (a_0 \wedge b_0) \wedge (a_1 \wedge b_1) \wedge (a_2 \wedge b_2)$ est-elle satisfiable ?

Exemples de taille

http://www.miroslav-velev.com/sat_benchmarks.html

D'AUTRES PROBLÈMES, D'AUTRES ENCODAGES

- Sudoku (pas encore un problème industriel)
- Recherche d'haplotype dans une population
- (Bounded) Model Checking
- Logiques modales
- Calcul pour les ontologies
- Organisation de tournois de golf
- ...

POURQUOI VOUS DEVEZ AUSSI VOUS Y METTRE

- Soit ça marche, et votre problème est résolu
- Soit ça ne marche pas et votre problème devient un benchmark de valeur

Si en plus des industriels sont intéressés...

D'AUTRES PROBLÈMES, D'AUTRES ENCODAGES

- Sudoku (pas encore un problème industriel)
- Recherche d'haplotype dans une population
- (Bounded) Model Checking
- Logiques modales
- Calcul pour les ontologies
- Organisation de tournois de golf
- ...

POURQUOI VOUS DEVEZ AUSSI VOUS Y METTRE

- Soit ça marche, et votre problème est résolu
- Soit ça ne marche pas et votre problème devient un benchmark de valeur

Si en plus des industriels sont intéressés...

D'AUTRES PROBLÈMES, D'AUTRES ENCODAGES

- Sudoku (pas encore un problème industriel)
- Recherche d'haplotype dans une population
- (Bounded) Model Checking
- Logiques modales
- Calcul pour les ontologies
- Organisation de tournois de golf
- ...

POURQUOI VOUS DEVEZ AUSSI VOUS Y METTRE

- Soit ça marche, et votre problème est résolu
- Soit ça ne marche pas et votre problème devient un benchmark de valeur

Si en plus des industriels sont intéressés...

D'AUTRES PROBLÈMES, D'AUTRES ENCODAGES

- Sudoku (pas encore un problème industriel)
- Recherche d'haplotype dans une population
- (Bounded) Model Checking
- Logiques modales
- Calcul pour les ontologies
- Organisation de tournois de golf
- ...

POURQUOI VOUS DEVEZ AUSSI VOUS Y METTRE

- Soit ça marche, et votre problème est résolu
- Soit ça ne marche pas et votre problème devient un benchmark de valeur

Si en plus des industriels sont intéressés...

Plan de l'exposé

Principale amélioration des performances

[Marques-Silva and Sakallah, 1996] a introduit dans le formalisme SAT l'idée d'apprentissage au niveau des conflits, lors de la recherche arborescente.

Cette idée, déjà bien connue en CSP, a révolutionné les démonstrateurs SAT de manière profonde.

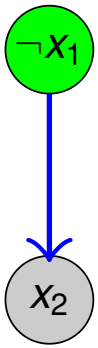
Toute l'architecture des solveurs est tournée vers l'apprentissage

- Les heuristiques de choix
- Les retours arrières / la complétude
- Le redémarrage rapide

Retour arrière

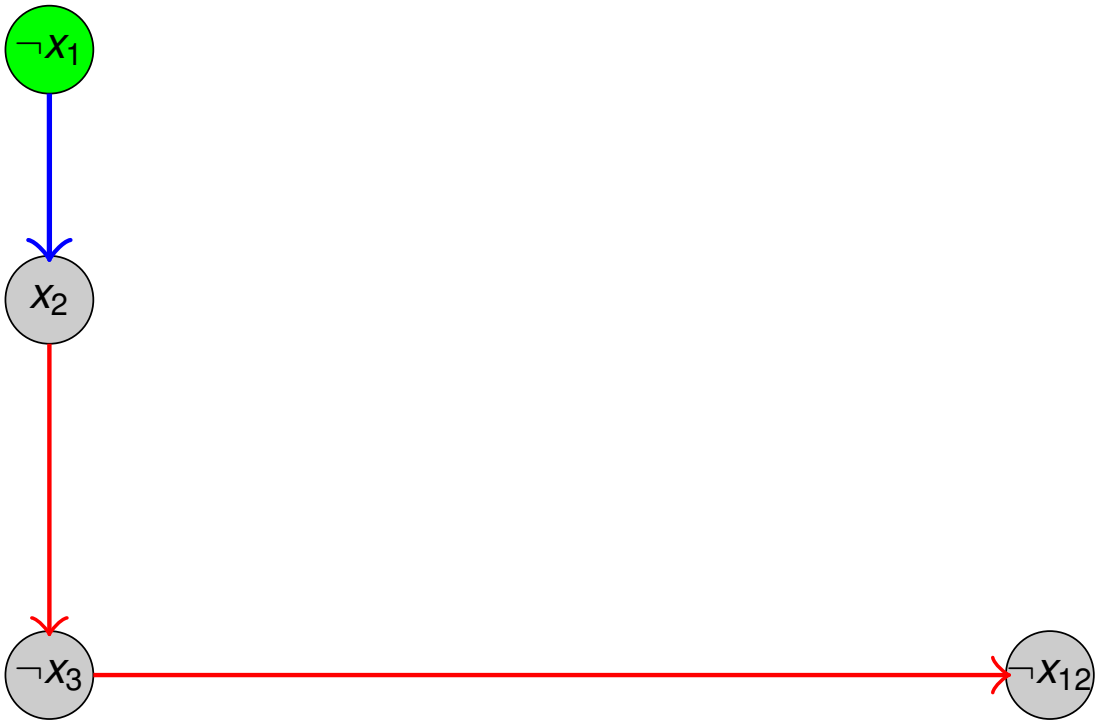
C_1: $x_1 \vee x_2$
C_2: $\neg x_2 \vee \neg x_3$
C_3: $x_3 \vee \neg x_{12}$
C_4: $\neg x_2 \vee \neg x_4 \vee x_5$
C_5: $x_3 \vee x_5 \vee x_6 \vee x_7$
C_6: $\neg x_7 \vee x_8 \vee \neg x_9$
C_7: $x_6 \vee \neg x_7 \vee x_9 \vee x_{10}$
C_8: $\neg x_7 \vee \neg x_{10} \vee x_8 \vee x_{11}$
C_9: $x_8 \vee \neg x_7 \vee x_{11}$
C_{10}: $\neg x_{11} \vee \neg x_{13}$
C_{11}: $\neg x_{11} \vee x_{14}$
C_{12}: $x_{12} \vee x_{15}$
C_{13}: $x_{13} \vee \neg x_{14} \vee \neg x_{15}$

Retour arrière



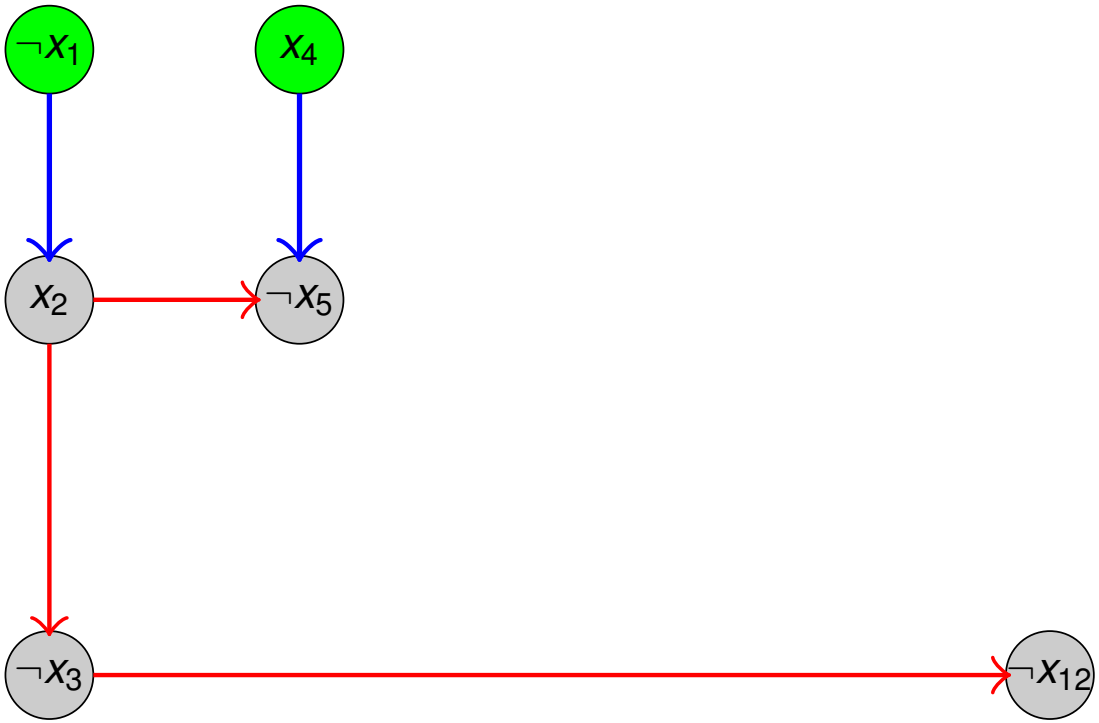
C_1: $x_1 \vee x_2$
C_2: $\neg x_2 \vee \neg x_3$
C_3: $x_3 \vee \neg x_{12}$
C_4: $\neg x_2 \vee \neg x_4 \vee x_5$
C_5: $x_3 \vee x_5 \vee x_6 \vee x_7$
C_6: $\neg x_7 \vee x_8 \vee \neg x_9$
C_7: $x_6 \vee \neg x_7 \vee x_9 \vee x_{10}$
C_8: $\neg x_7 \vee \neg x_{10} \vee x_8 \vee x_{11}$
C_9: $x_8 \vee \neg x_7 \vee x_{11}$
C_{10}: $\neg x_{11} \vee \neg x_{13}$
C_{11}: $\neg x_{11} \vee x_{14}$
C_{12}: $x_{12} \vee x_{15}$
C_{13}: $x_{13} \vee \neg x_{14} \vee \neg x_{15}$

Retour arrière



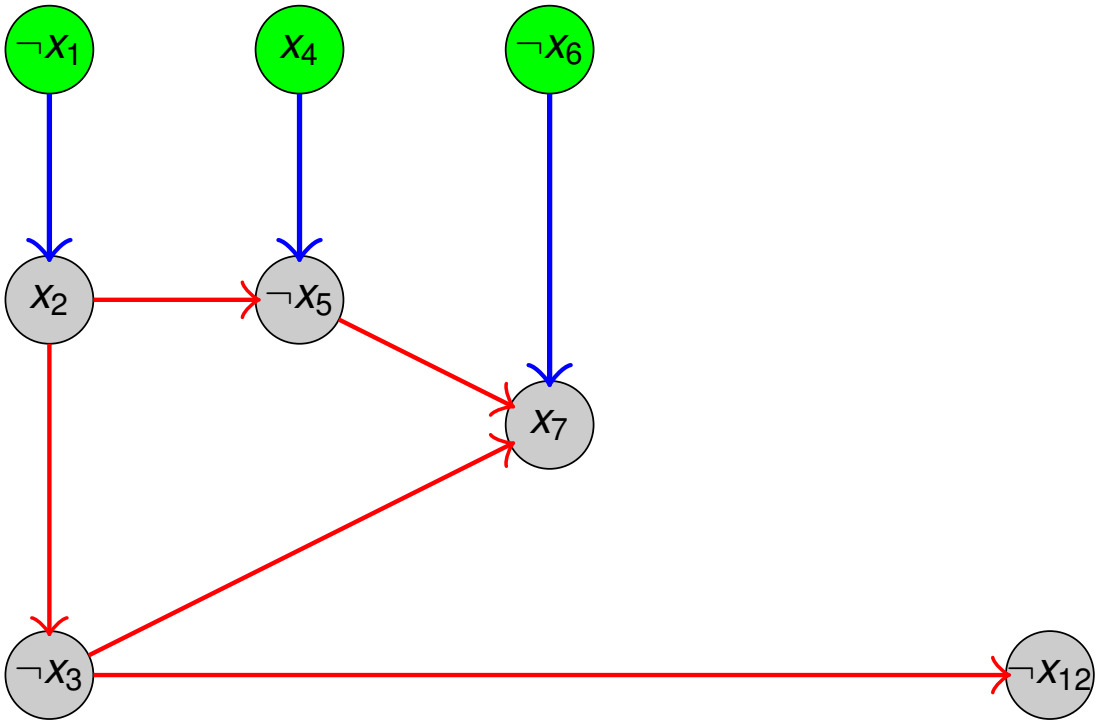
C_1: $x_1 \vee x_2$
C_2: $\neg x_2 \vee \neg x_3$
C_3: $x_3 \vee \neg x_{12}$
C_4: $\neg x_2 \vee \neg x_4 \vee x_5$
C_5: $x_3 \vee x_5 \vee x_6 \vee x_7$
C_6: $\neg x_7 \vee x_8 \vee \neg x_9$
C_7: $x_6 \vee \neg x_7 \vee x_9 \vee x_{10}$
C_8: $\neg x_7 \vee \neg x_{10} \vee x_8 \vee x_{11}$
C_9: $x_8 \vee \neg x_7 \vee x_{11}$
C_{10}: $\neg x_{11} \vee \neg x_{13}$
C_{11}: $\neg x_{11} \vee x_{14}$
C_{12}: $x_{12} \vee x_{15}$
C_{13}: $x_{13} \vee \neg x_{14} \vee \neg x_{15}$

Retour arrière



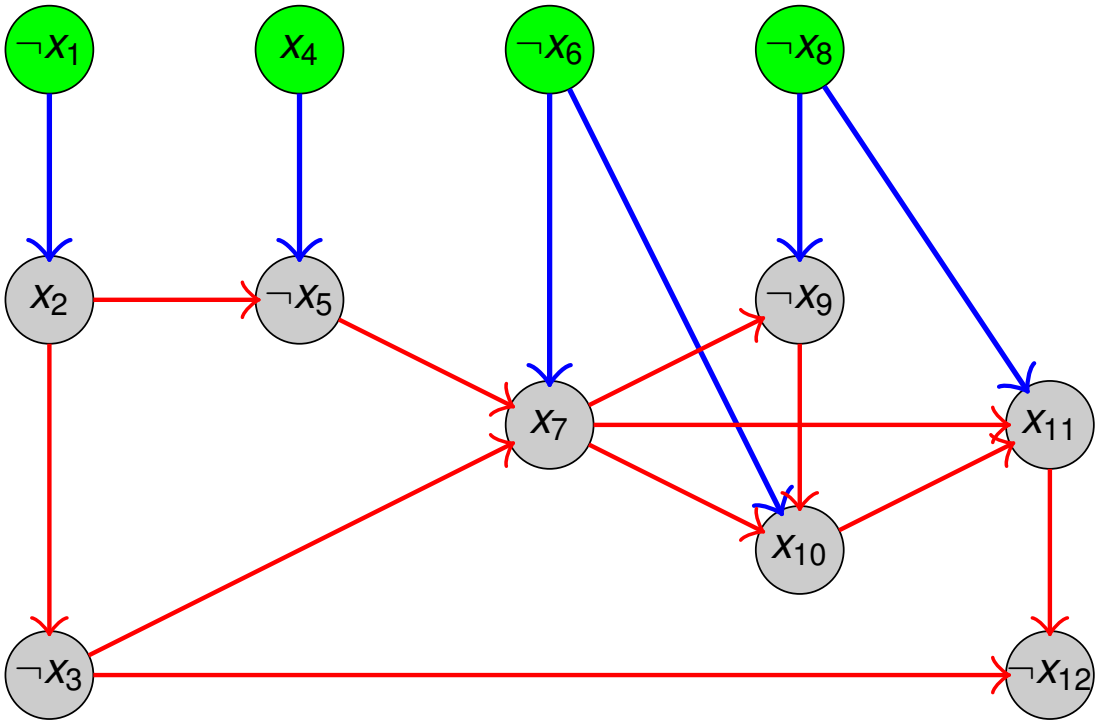
C_1: $x_1 \vee x_2$
C_2: $\neg x_2 \vee \neg x_3$
C_3: $x_3 \vee \neg x_{12}$
C_4: $\neg x_2 \vee \neg x_4 \vee x_5$
C_5: $x_3 \vee x_5 \vee x_6 \vee x_7$
C_6: $\neg x_7 \vee x_8 \vee \neg x_9$
C_7: $x_6 \vee \neg x_7 \vee x_9 \vee x_{10}$
C_8: $\neg x_7 \vee \neg x_{10} \vee x_8 \vee x_{11}$
C_9: $x_8 \vee \neg x_7 \vee x_{11}$
C_{10}: $\neg x_{11} \vee \neg x_{13}$
C_{11}: $\neg x_{11} \vee x_{14}$
C_{12}: $x_{12} \vee x_{15}$
C_{13}: $x_{13} \vee \neg x_{14} \vee \neg x_{15}$

Retour arrière



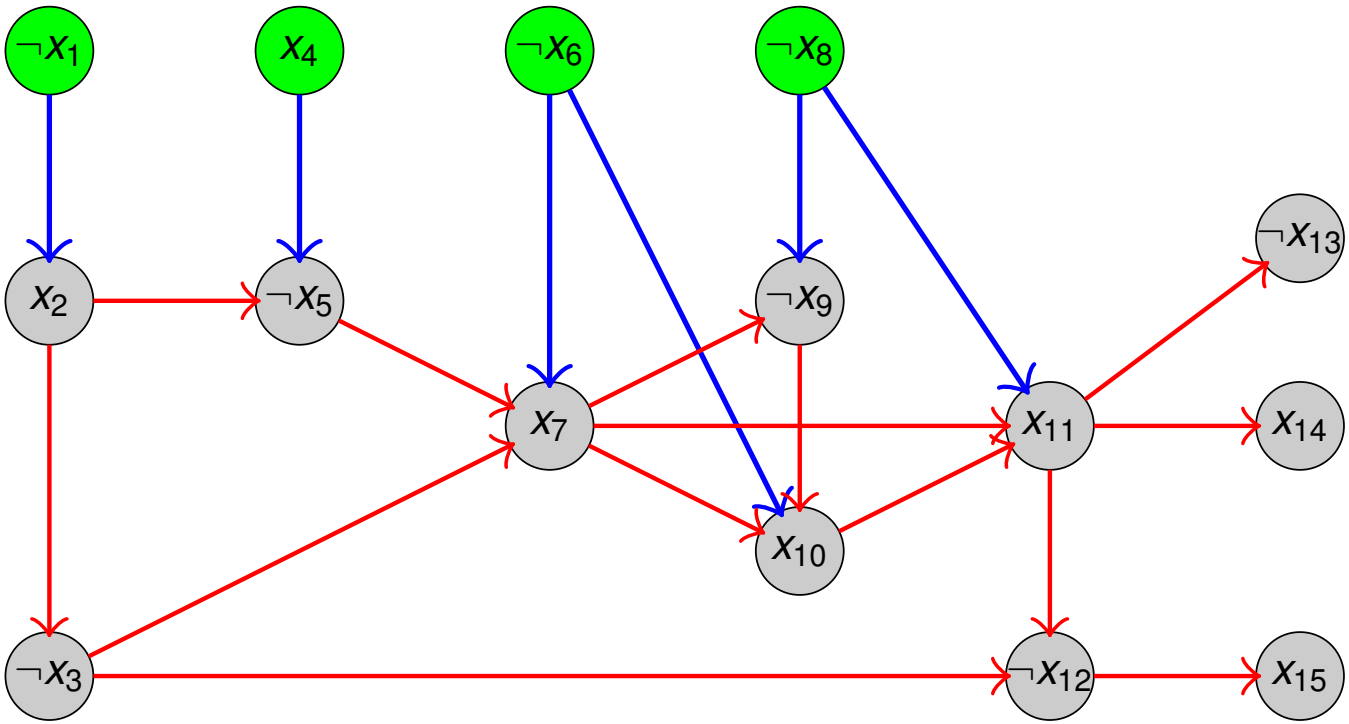
C_1: $x_1 \vee x_2$
C_2: $\neg x_2 \vee \neg x_3$
C_3: $x_3 \vee \neg x_{12}$
C_4: $\neg x_2 \vee \neg x_4 \vee x_5$
C_5: $x_3 \vee x_5 \vee x_6 \vee x_7$
C_6: $\neg x_7 \vee x_8 \vee \neg x_9$
C_7: $x_6 \vee \neg x_7 \vee x_9 \vee x_{10}$
C_8: $\neg x_7 \vee \neg x_{10} \vee x_8 \vee x_{11}$
C_9: $x_8 \vee \neg x_7 \vee x_{11}$
C_{10}: $\neg x_{11} \vee \neg x_{13}$
C_{11}: $\neg x_{11} \vee x_{14}$
C_{12}: $x_{12} \vee x_{15}$
C_{13}: $x_{13} \vee \neg x_{14} \vee \neg x_{15}$

Retour arrière



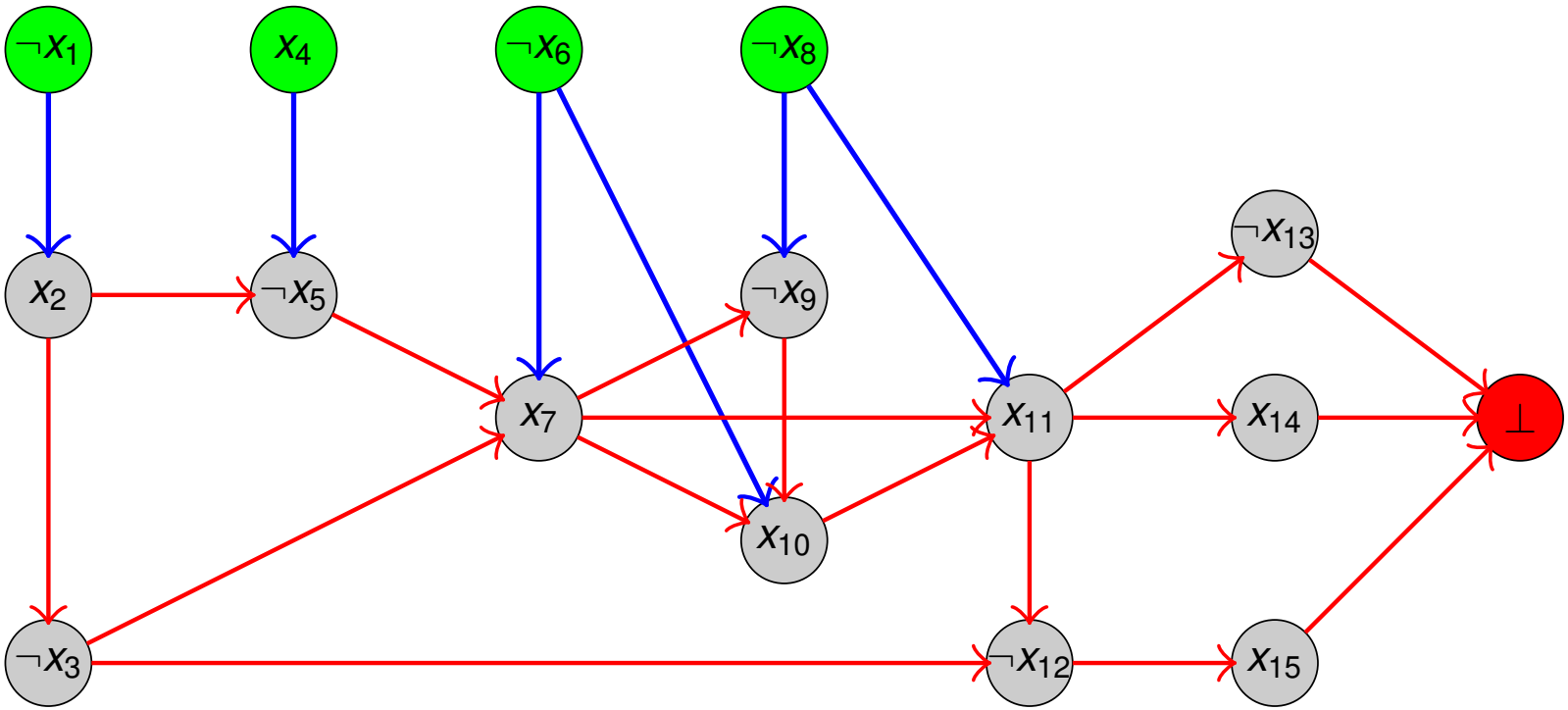
C_1: $x_1 \vee x_2$
C_2: $\neg x_2 \vee \neg x_3$
C_3: $x_3 \vee \neg x_{12}$
C_4: $\neg x_2 \vee \neg x_4 \vee x_5$
C_5: $x_3 \vee x_5 \vee x_6 \vee x_7$
C_6: $\neg x_7 \vee x_8 \vee \neg x_9$
C_7: $x_6 \vee \neg x_7 \vee x_9 \vee x_{10}$
C_8: $\neg x_7 \vee \neg x_{10} \vee x_8 \vee x_{11}$
C_9: $x_8 \vee \neg x_7 \vee x_{11}$
C_{10}: $\neg x_{11} \vee \neg x_{13}$
C_{11}: $\neg x_{11} \vee x_{14}$
C_{12}: $x_{12} \vee x_{15}$
C_{13}: $x_{13} \vee \neg x_{14} \vee \neg x_{15}$

Retour arrière



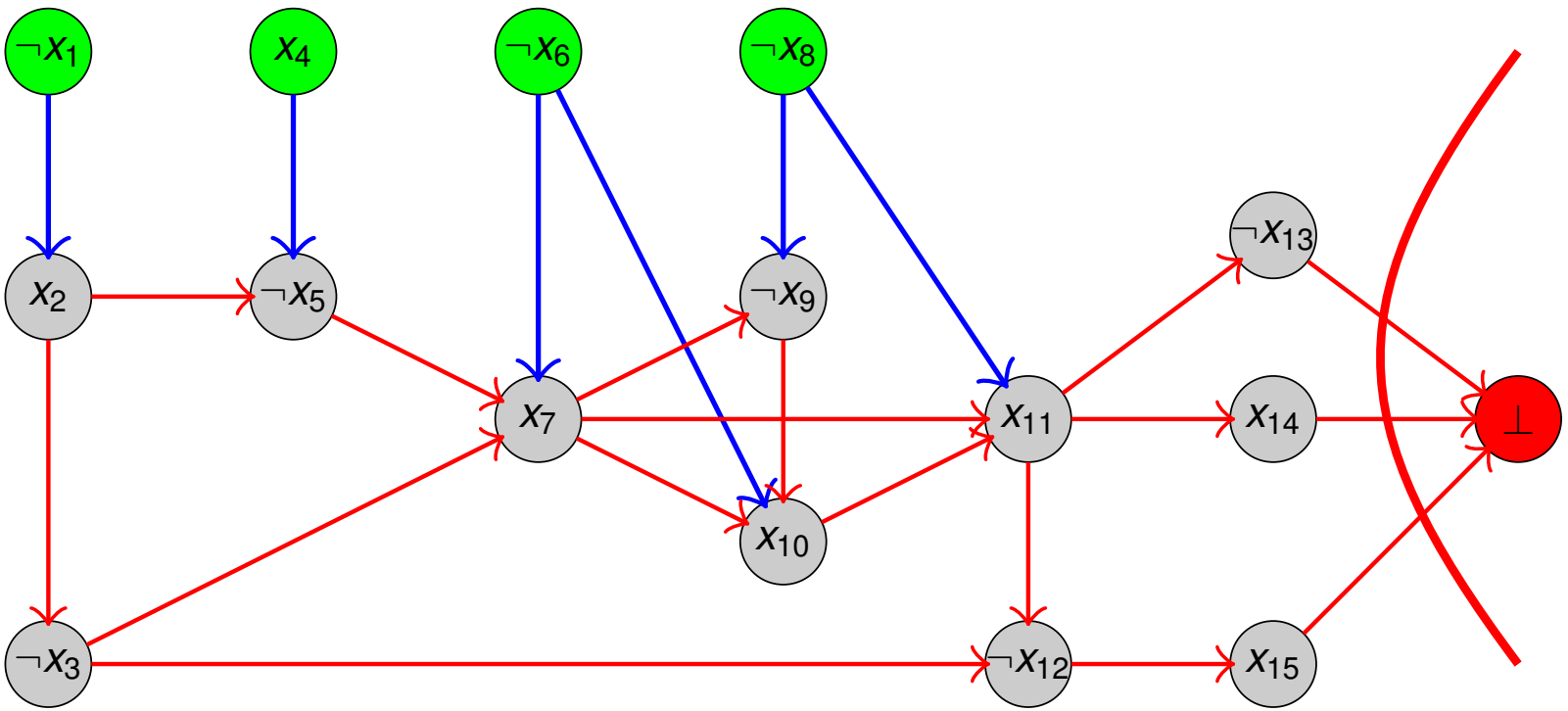
C_1: $x_1 \vee x_2$
C_2: $\neg x_2 \vee \neg x_3$
C_3: $x_3 \vee \neg x_{12}$
C_4: $\neg x_2 \vee \neg x_4 \vee x_5$
C_5: $x_3 \vee x_5 \vee x_6 \vee x_7$
C_6: $\neg x_7 \vee x_8 \vee \neg x_9$
C_7: $x_6 \vee \neg x_7 \vee x_9 \vee x_{10}$
C_8: $\neg x_7 \vee \neg x_{10} \vee x_8 \vee x_{11}$
C_9: $x_8 \vee \neg x_7 \vee x_{11}$
C_{10}: $\neg x_{11} \vee \neg x_{13}$
C_{11}: $\neg x_{11} \vee x_{14}$
C_{12}: $x_{12} \vee x_{15}$
C_{13}: $x_{13} \vee \neg x_{14} \vee \neg x_{15}$

Retour arrière



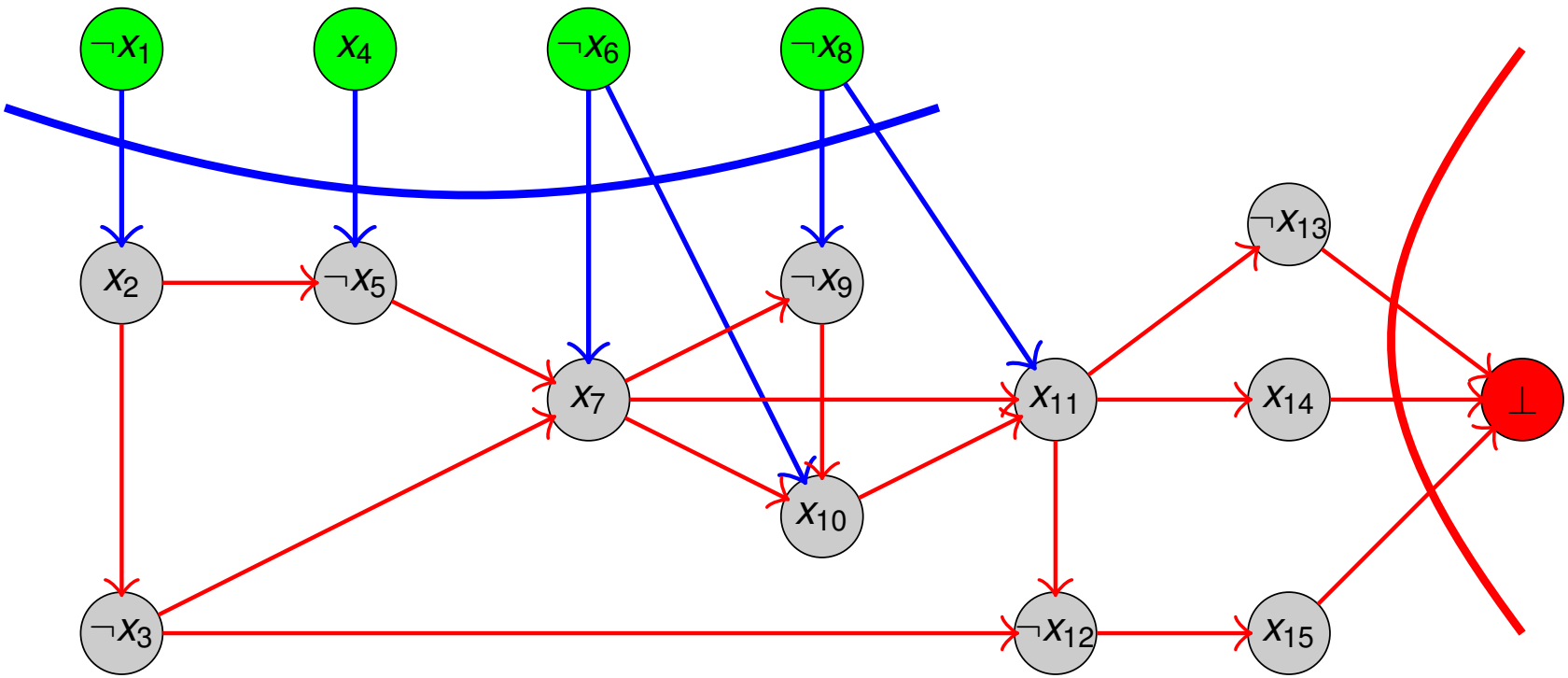
C_1: $x_1 \vee x_2$
C_2: $\neg x_2 \vee \neg x_3$
C_3: $x_3 \vee \neg x_{12}$
C_4: $\neg x_2 \vee \neg x_4 \vee x_5$
C_5: $x_3 \vee x_5 \vee x_6 \vee x_7$
C_6: $\neg x_7 \vee x_8 \vee \neg x_9$
C_7: $x_6 \vee \neg x_7 \vee x_9 \vee x_{10}$
C_8: $\neg x_7 \vee \neg x_{10} \vee x_8 \vee x_{11}$
C_9: $x_8 \vee \neg x_7 \vee x_{11}$
C_{10}: $\neg x_{11} \vee \neg x_{13}$
C_{11}: $\neg x_{11} \vee x_{14}$
C_{12}: $x_{12} \vee x_{15}$
C_{13}: $x_{13} \vee \neg x_{14} \vee \neg x_{15}$

Retour arrière



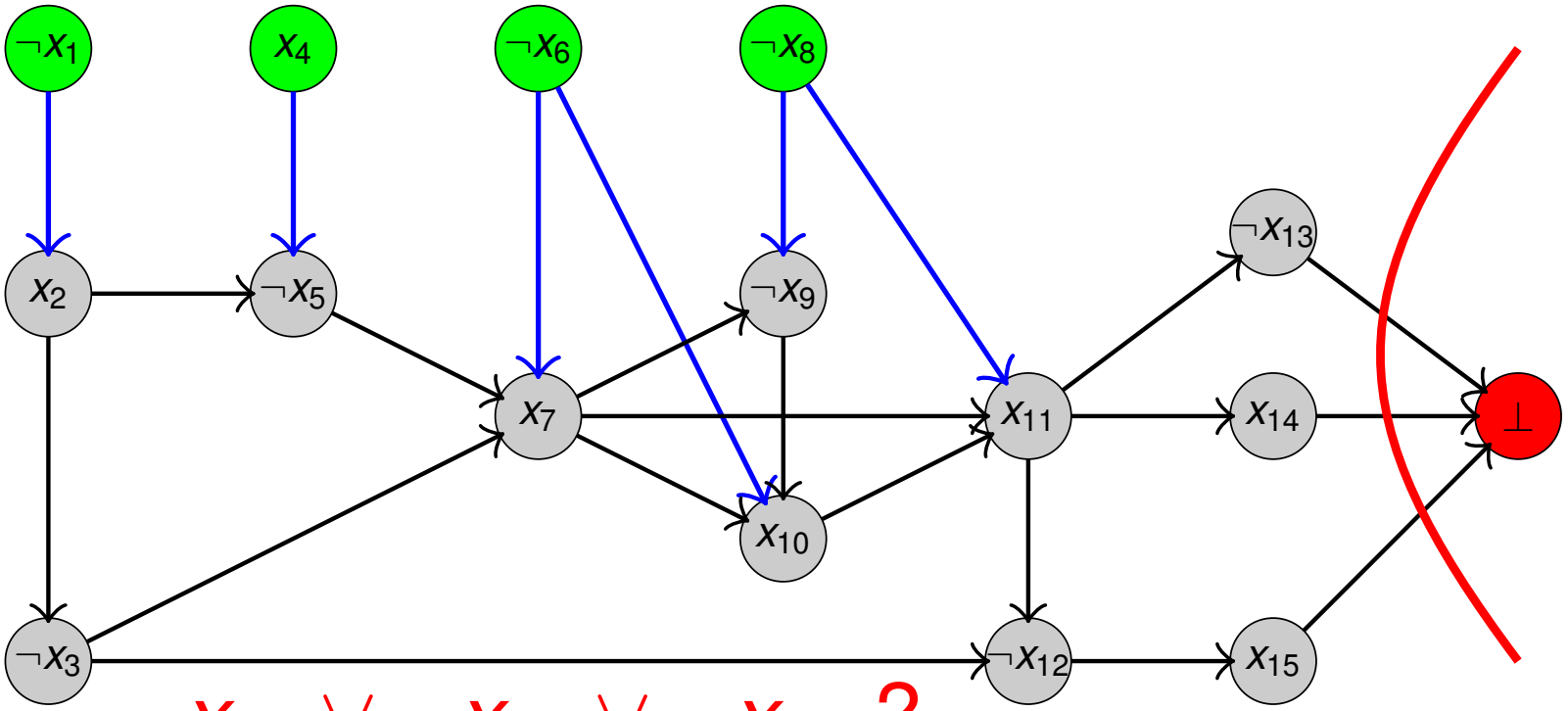
C_1: $x_1 \vee x_2$
C_2: $\neg x_2 \vee \neg x_3$
C_3: $x_3 \vee \neg x_{12}$
C_4: $\neg x_2 \vee \neg x_4 \vee x_5$
C_5: $x_3 \vee x_5 \vee x_6 \vee x_7$
C_6: $\neg x_7 \vee x_8 \vee \neg x_9$
C_7: $x_6 \vee \neg x_7 \vee x_9 \vee x_{10}$
C_8: $\neg x_7 \vee \neg x_{10} \vee x_8 \vee x_{11}$
C_9: $x_8 \vee \neg x_7 \vee x_{11}$
C_{10}: $\neg x_{11} \vee \neg x_{13}$
C_{11}: $\neg x_{11} \vee x_{14}$
C_{12}: $x_{12} \vee x_{15}$
C_{13}: $x_{13} \vee \neg x_{14} \vee \neg x_{15}$

Retour arrière



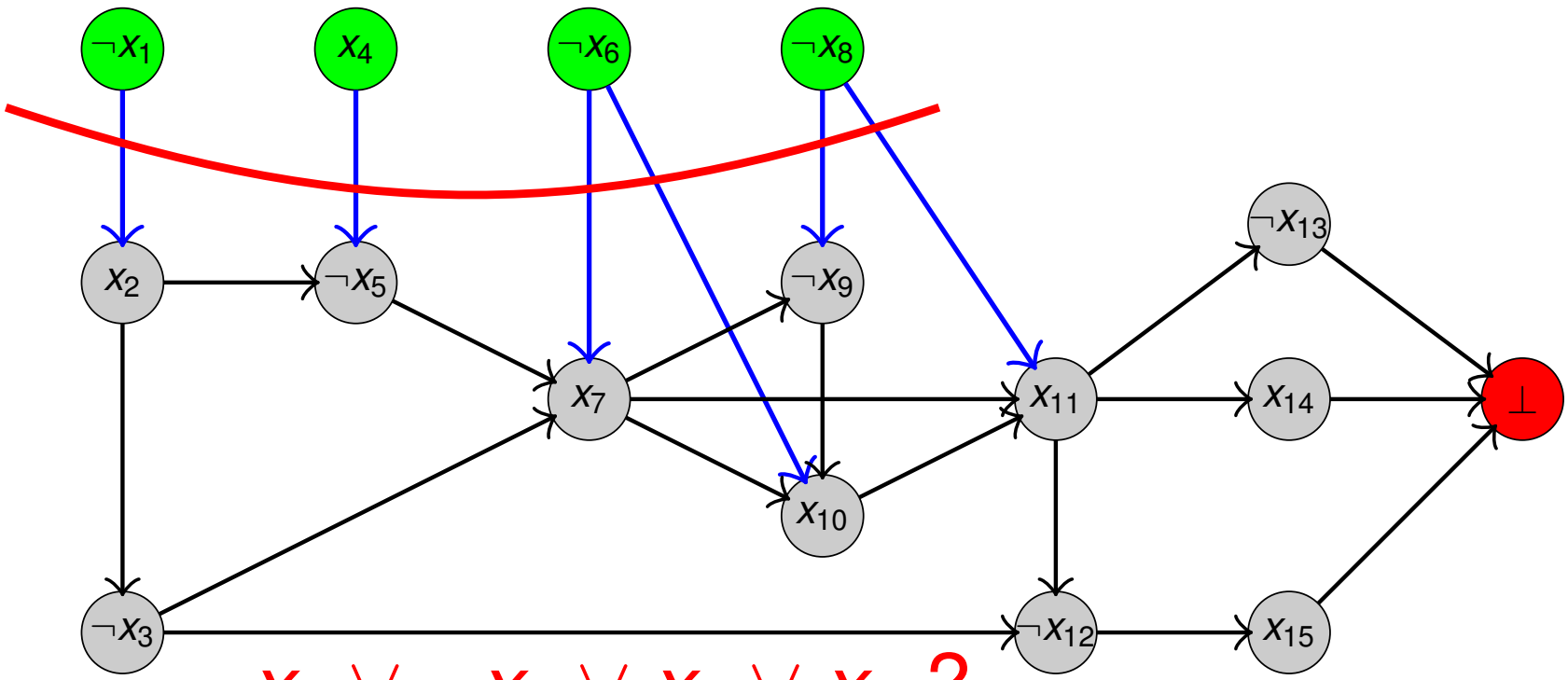
C_1: $x_1 \vee x_2$
C_2: $\neg x_2 \vee \neg x_3$
C_3: $x_3 \vee \neg x_{12}$
C_4: $\neg x_2 \vee \neg x_4 \vee x_5$
C_5: $x_3 \vee x_5 \vee x_6 \vee x_7$
C_6: $\neg x_7 \vee x_8 \vee \neg x_9$
C_7: $x_6 \vee \neg x_7 \vee x_9 \vee x_{10}$
C_8: $\neg x_7 \vee \neg x_{10} \vee x_8 \vee x_{11}$
C_9: $x_8 \vee \neg x_7 \vee x_{11}$
C_{10}: $\neg x_{11} \vee \neg x_{13}$
C_{11}: $\neg x_{11} \vee x_{14}$
C_{12}: $x_{12} \vee x_{15}$
C_{13}: $x_{13} \vee \neg x_{14} \vee \neg x_{15}$

TROUVER DES EXPLICATIONS



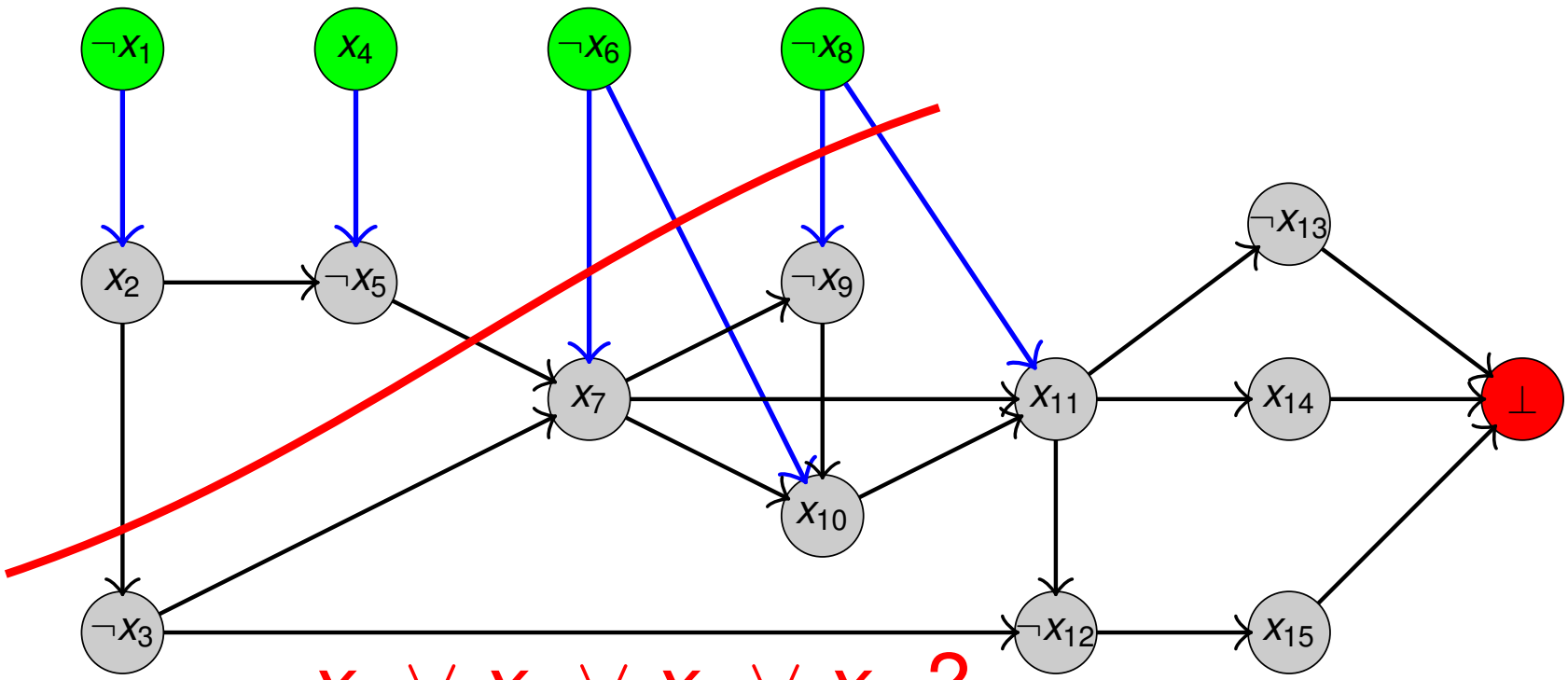
C_1: $x_1 \vee x_2$
C_2: $\neg x_2 \vee \neg x_3$
C_3: $x_3 \vee \neg x_{12}$
C_4: $\neg x_2 \vee \neg x_4 \vee x_5$
C_5: $x_3 \vee x_5 \vee x_6 \vee x_7$
C_6: $\neg x_7 \vee x_8 \vee \neg x_9$
C_7: $x_6 \vee \neg x_7 \vee x_9 \vee x_{10}$
C_8: $\neg x_7 \vee \neg x_{10} \vee x_8 \vee x_{11}$
C_9: $x_8 \vee \neg x_7 \vee x_{11}$
C_{10}: $\neg x_{11} \vee \neg x_{13}$
C_{11}: $\neg x_{11} \vee x_{14}$
C_{12}: $x_{12} \vee x_{15}$
C_{13}: $x_{13} \vee \neg x_{14} \vee \neg x_{15}$

TROUVER DES EXPLICATIONS



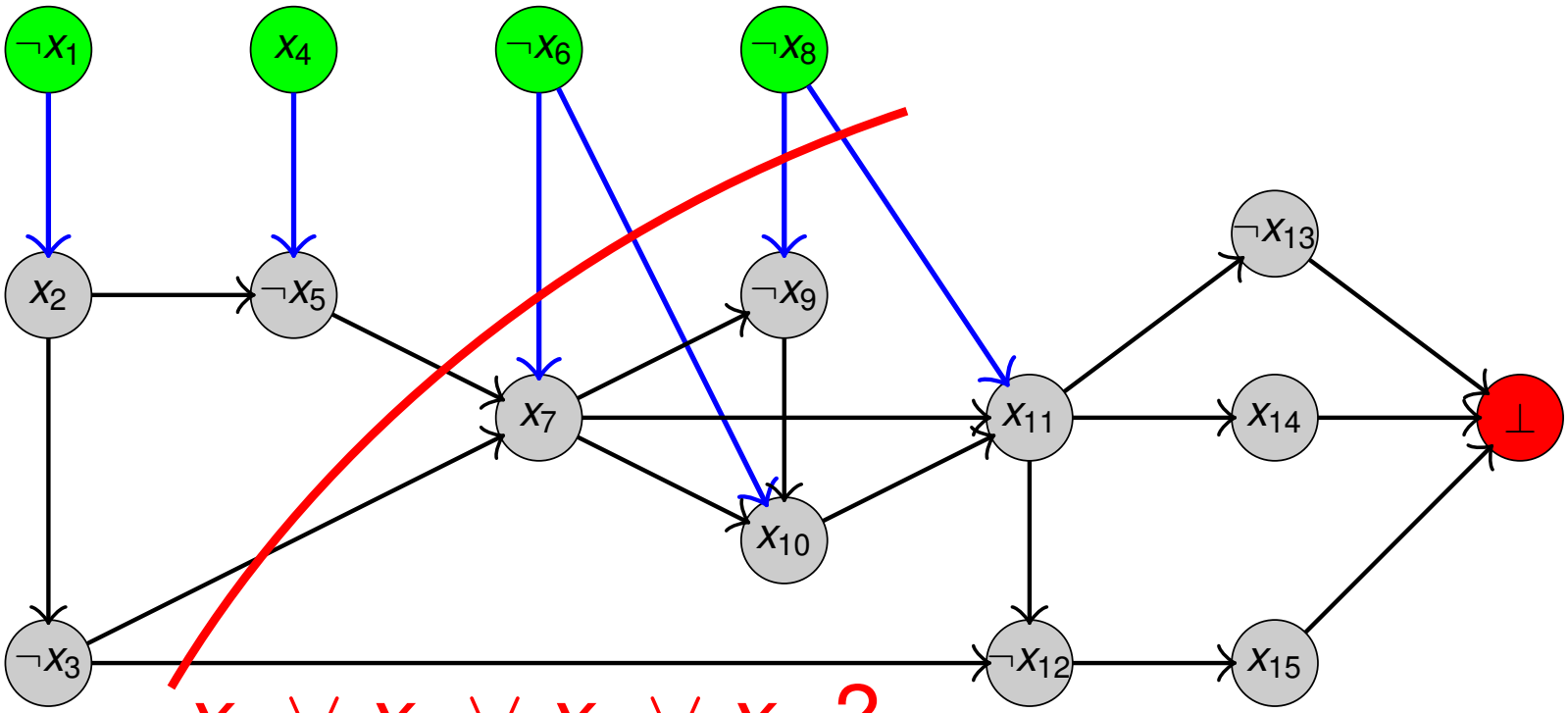
C_1: $x_1 \vee x_2$
C_2: $\neg x_2 \vee \neg x_3$
C_3: $x_3 \vee \neg x_{12}$
C_4: $\neg x_2 \vee \neg x_4 \vee x_5$
C_5: $x_3 \vee x_5 \vee x_6 \vee x_7$
C_6: $\neg x_7 \vee x_8 \vee \neg x_9$
C_7: $x_6 \vee \neg x_7 \vee x_9 \vee x_{10}$
C_8: $\neg x_7 \vee \neg x_{10} \vee x_8 \vee x_{11}$
C_9: $x_8 \vee \neg x_7 \vee x_{11}$
C_{10}: $\neg x_{11} \vee \neg x_{13}$
C_{11}: $\neg x_{11} \vee x_{14}$
C_{12}: $x_{12} \vee x_{15}$
C_{13}: $x_{13} \vee \neg x_{14} \vee \neg x_{15}$

TROUVER DES EXPLICATIONS



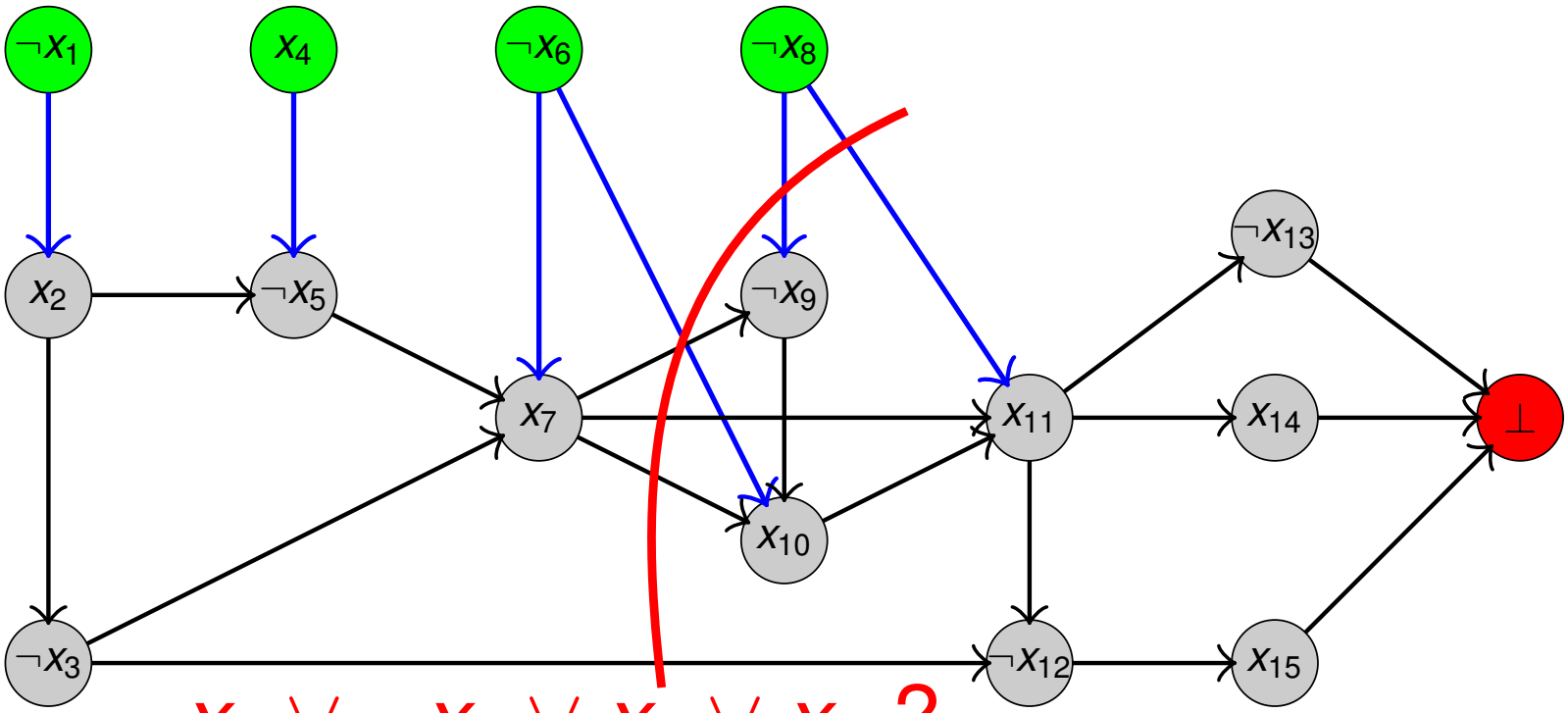
C_1: $x_1 \vee x_2$
C_2: $\neg x_2 \vee \neg x_3$
C_3: $x_3 \vee \neg x_{12}$
C_4: $\neg x_2 \vee \neg x_4 \vee x_5$
C_5: $x_3 \vee x_5 \vee x_6 \vee x_7$
C_6: $\neg x_7 \vee x_8 \vee \neg x_9$
C_7: $x_6 \vee \neg x_7 \vee x_9 \vee x_{10}$
C_8: $\neg x_7 \vee \neg x_{10} \vee x_8 \vee x_{11}$
C_9: $x_8 \vee \neg x_7 \vee x_{11}$
C_{10}: $\neg x_{11} \vee \neg x_{13}$
C_{11}: $\neg x_{11} \vee x_{14}$
C_{12}: $x_{12} \vee x_{15}$
C_{13}: $x_{13} \vee \neg x_{14} \vee \neg x_{15}$

TROUVER DES EXPLICATIONS



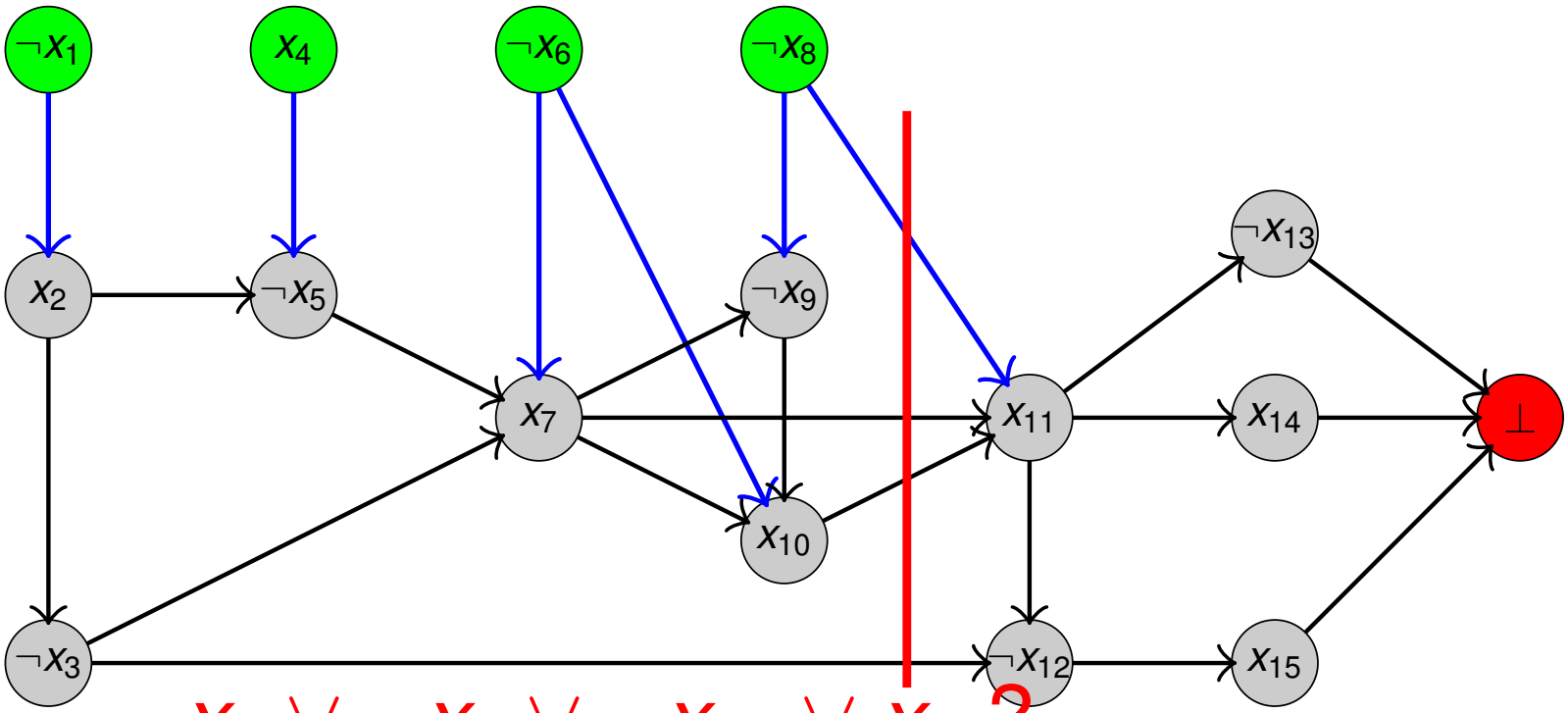
C_1: $x_1 \vee x_2$
C_2: $\neg x_2 \vee \neg x_3$
C_3: $x_3 \vee \neg x_{12}$
C_4: $\neg x_2 \vee \neg x_4 \vee x_5$
C_5: $x_3 \vee x_5 \vee x_6 \vee x_7$
C_6: $\neg x_7 \vee x_8 \vee \neg x_9$
C_7: $x_6 \vee \neg x_7 \vee x_9 \vee x_{10}$
C_8: $\neg x_7 \vee \neg x_{10} \vee x_8 \vee x_{11}$
C_9: $x_8 \vee \neg x_7 \vee x_{11}$
C_{10}: $\neg x_{11} \vee \neg x_{13}$
C_{11}: $\neg x_{11} \vee x_{14}$
C_{12}: $x_{12} \vee x_{15}$
C_{13}: $x_{13} \vee \neg x_{14} \vee \neg x_{15}$

TROUVER DES EXPLICATIONS



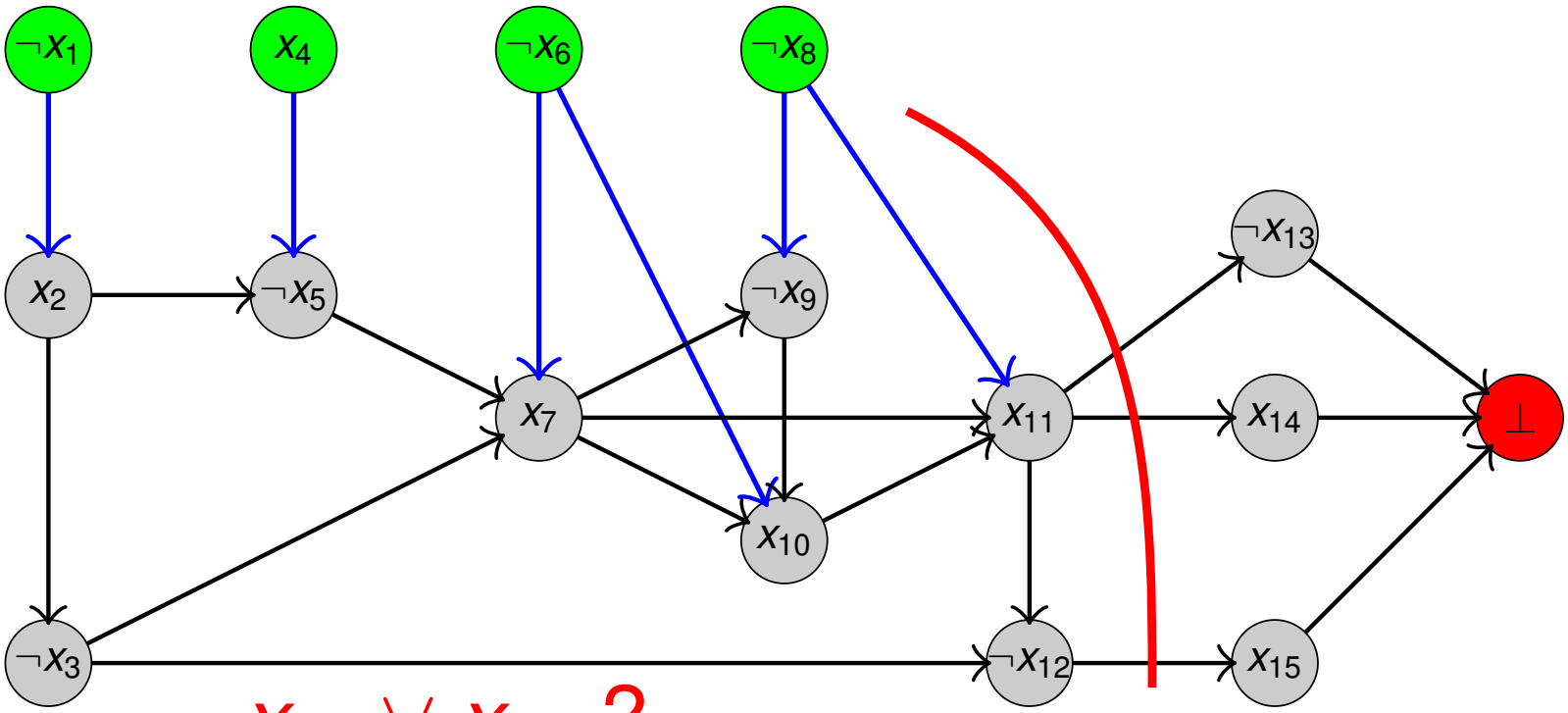
C_1: $x_1 \vee x_2$
C_2: $\neg x_2 \vee \neg x_3$
C_3: $x_3 \vee \neg x_{12}$
C_4: $\neg x_2 \vee \neg x_4 \vee x_5$
C_5: $x_3 \vee x_5 \vee x_6 \vee x_7$
C_6: $\neg x_7 \vee x_8 \vee \neg x_9$
C_7: $x_6 \vee \neg x_7 \vee x_9 \vee x_{10}$
C_8: $\neg x_7 \vee \neg x_{10} \vee x_8 \vee x_{11}$
C_9: $x_8 \vee \neg x_7 \vee x_{11}$
C_{10}: $\neg x_{11} \vee \neg x_{13}$
C_{11}: $\neg x_{11} \vee x_{14}$
C_{12}: $x_{12} \vee x_{15}$
C_{13}: $x_{13} \vee \neg x_{14} \vee \neg x_{15}$

TROUVER DES EXPLICATIONS



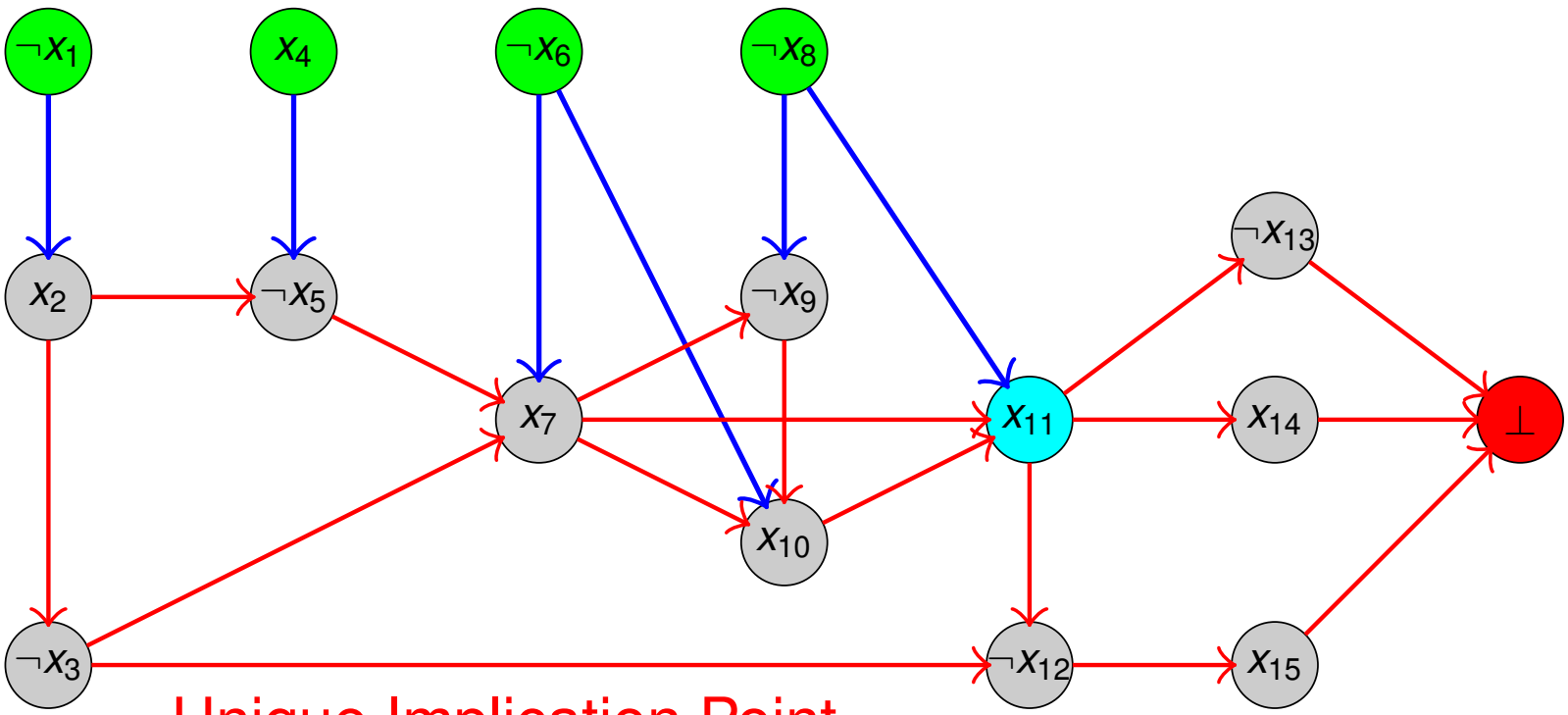
C_1: $x_1 \vee x_2$
C_2: $\neg x_2 \vee \neg x_3$
C_3: $x_3 \vee \neg x_{12}$
C_4: $\neg x_2 \vee \neg x_4 \vee x_5$
C_5: $x_3 \vee x_5 \vee x_6 \vee x_7$
C_6: $\neg x_7 \vee x_8 \vee \neg x_9$
C_7: $x_6 \vee \neg x_7 \vee x_9 \vee x_{10}$
C_8: $\neg x_7 \vee \neg x_{10} \vee x_8 \vee x_{11}$
C_9: $x_8 \vee \neg x_7 \vee x_{11}$
C_{10}: $\neg x_{11} \vee \neg x_{13}$
C_{11}: $\neg x_{11} \vee x_{14}$
C_{12}: $x_{12} \vee x_{15}$
C_{13}: $x_{13} \vee \neg x_{14} \vee \neg x_{15}$

TROUVER DES EXPLICATIONS



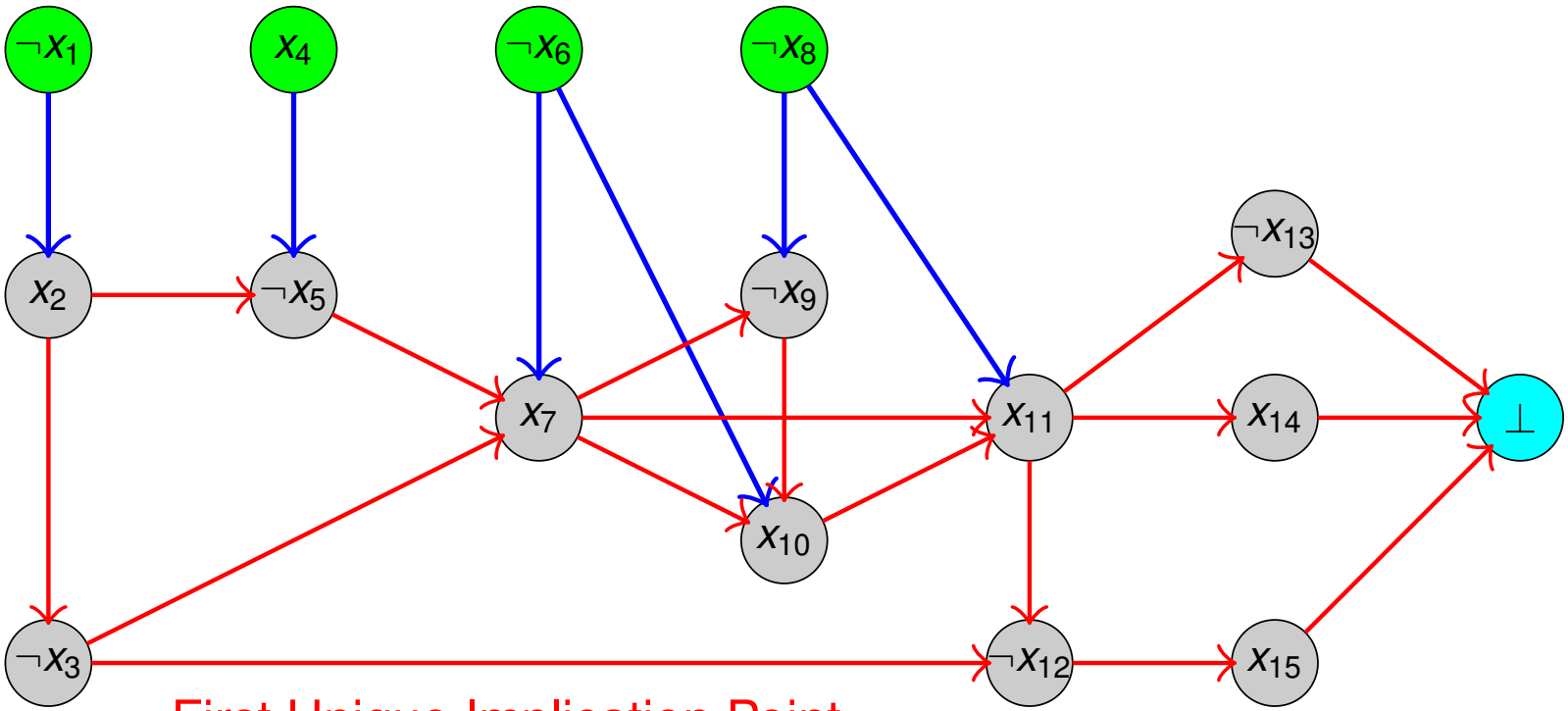
C_1: $x_1 \vee x_2$
C_2: $\neg x_2 \vee \neg x_3$
C_3: $x_3 \vee \neg x_{12}$
C_4: $\neg x_2 \vee \neg x_4 \vee x_5$
C_5: $x_3 \vee x_5 \vee x_6 \vee x_7$
C_6: $\neg x_7 \vee x_8 \vee \neg x_9$
C_7: $x_6 \vee \neg x_7 \vee x_9 \vee x_{10}$
C_8: $\neg x_7 \vee \neg x_{10} \vee x_8 \vee x_{11}$
C_9: $x_8 \vee \neg x_7 \vee x_{11}$
C_{10}: $\neg x_{11} \vee \neg x_{13}$
C_{11}: $\neg x_{11} \vee x_{14}$
C_{12}: $x_{12} \vee x_{15}$
C_{13}: $x_{13} \vee \neg x_{14} \vee \neg x_{15}$

First UIP



C_1: $x_1 \vee x_2$
C_2: $\neg x_2 \vee \neg x_3$
C_3: $x_3 \vee \neg x_{12}$
C_4: $\neg x_2 \vee \neg x_4 \vee x_5$
C_5: $x_3 \vee x_5 \vee x_6 \vee x_7$
C_6: $\neg x_7 \vee x_8 \vee \neg x_9$
C_7: $x_6 \vee \neg x_7 \vee x_9 \vee x_{10}$
C_8: $\neg x_7 \vee \neg x_{10} \vee x_8 \vee x_{11}$
C_9: $x_8 \vee \neg x_7 \vee x_{11}$
C_{10}: $\neg x_{11} \vee \neg x_{13}$
C_{11}: $\neg x_{11} \vee x_{14}$
C_{12}: $x_{12} \vee x_{15}$
C_{13}: $x_{13} \vee \neg x_{14} \vee \neg x_{15}$

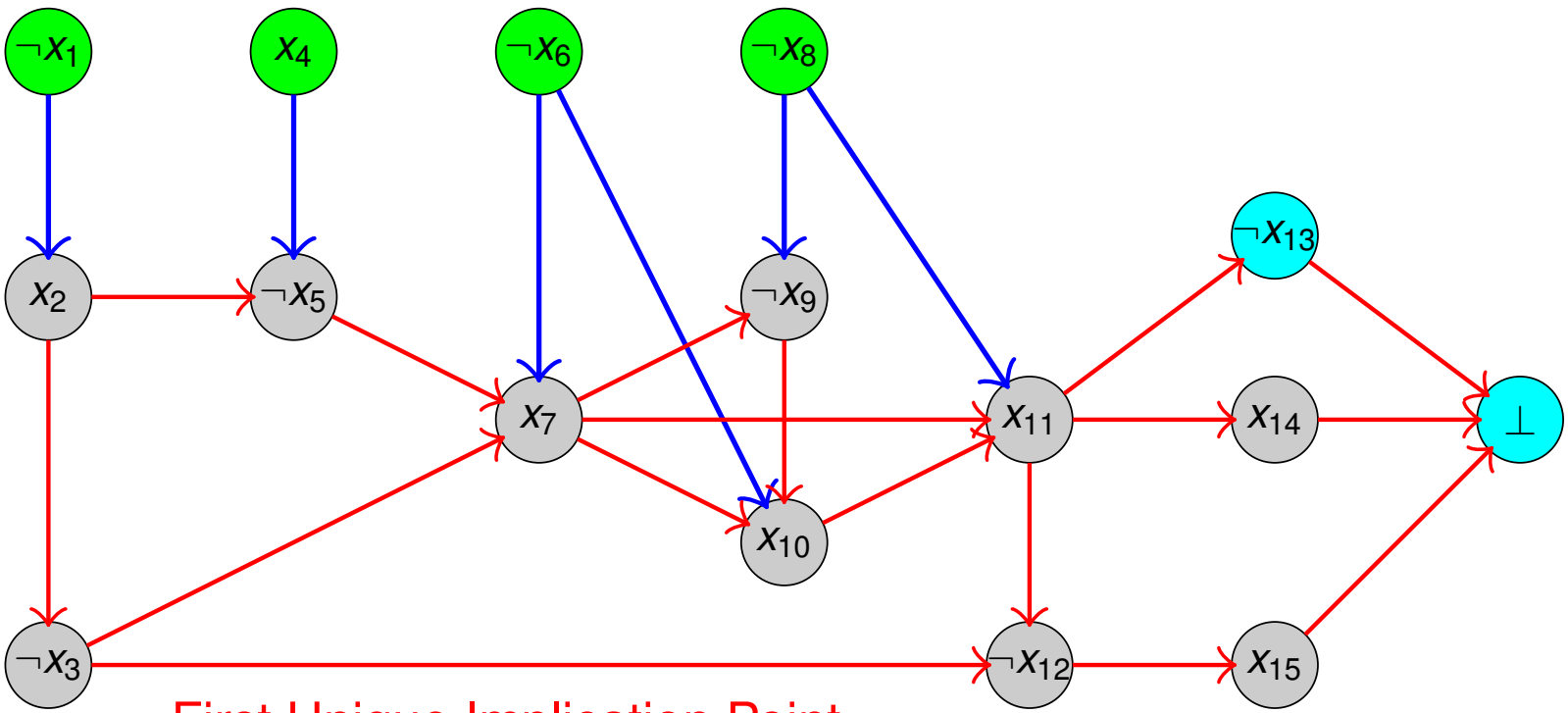
First UIP



First Unique Implication Point

C_1: $x_1 \vee x_2$
C_2: $\neg x_2 \vee \neg x_3$
C_3: $x_3 \vee \neg x_{12}$
C_4: $\neg x_2 \vee \neg x_4 \vee x_5$
C_5: $x_3 \vee x_5 \vee x_6 \vee x_7$
C_6: $\neg x_7 \vee x_8 \vee \neg x_9$
C_7: $x_6 \vee \neg x_7 \vee x_9 \vee x_{10}$
C_8: $\neg x_7 \vee \neg x_{10} \vee x_8 \vee x_{11}$
C_9: $x_8 \vee \neg x_7 \vee x_{11}$
C_{10}: $\neg x_{11} \vee \neg x_{13}$
C_{11}: $\neg x_{11} \vee x_{14}$
C_{12}: $x_{12} \vee x_{15}$
C_{13}: $x_{13} \vee \neg x_{14} \vee \neg x_{15}$

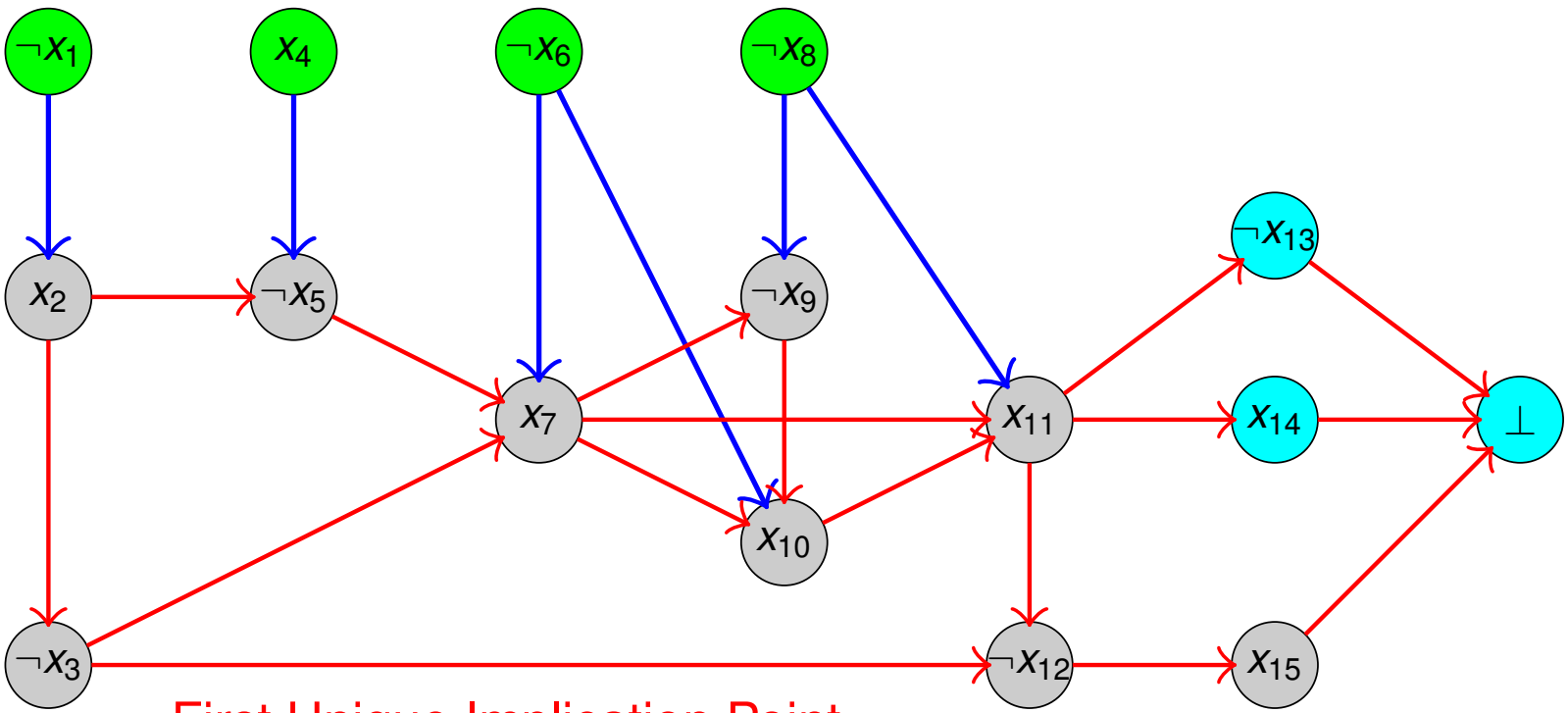
First UIP



First Unique Implication Point

C_1: $x_1 \vee x_2$
C_2: $\neg x_2 \vee \neg x_3$
C_3: $x_3 \vee \neg x_{12}$
C_4: $\neg x_2 \vee \neg x_4 \vee x_5$
C_5: $x_3 \vee x_5 \vee x_6 \vee x_7$
C_6: $\neg x_7 \vee x_8 \vee \neg x_9$
C_7: $x_6 \vee \neg x_7 \vee x_9 \vee x_{10}$
C_8: $\neg x_7 \vee \neg x_{10} \vee x_8 \vee x_{11}$
C_9: $x_8 \vee \neg x_7 \vee x_{11}$
C_{10}: $\neg x_{11} \vee \neg x_{13}$
C_{11}: $\neg x_{11} \vee x_{14}$
C_{12}: $x_{12} \vee x_{15}$
C_{13}: $x_{13} \vee \neg x_{14} \vee \neg x_{15}$

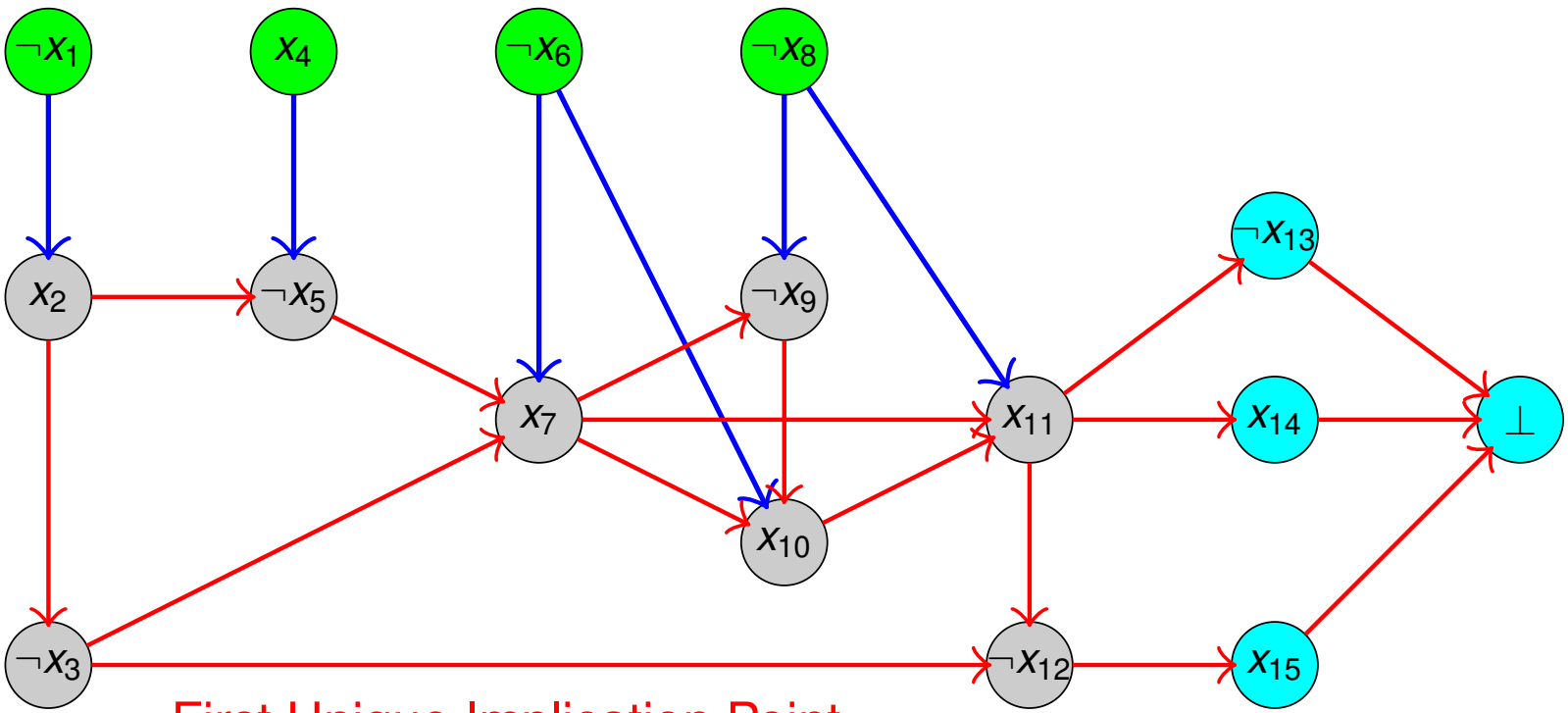
First UIP



First Unique Implication Point

C_1: $x_1 \vee x_2$
C_2: $\neg x_2 \vee \neg x_3$
C_3: $x_3 \vee \neg x_{12}$
C_4: $\neg x_2 \vee \neg x_4 \vee x_5$
C_5: $x_3 \vee x_5 \vee x_6 \vee x_7$
C_6: $\neg x_7 \vee x_8 \vee \neg x_9$
C_7: $x_6 \vee \neg x_7 \vee x_9 \vee x_{10}$
C_8: $\neg x_7 \vee \neg x_{10} \vee x_8 \vee x_{11}$
C_9: $x_8 \vee \neg x_7 \vee x_{11}$
C_{10}: $\neg x_{11} \vee \neg x_{13}$
C_{11}: $\neg x_{11} \vee x_{14}$
C_{12}: $x_{12} \vee x_{15}$
C_{13}: $x_{13} \vee \neg x_{14} \vee \neg x_{15}$

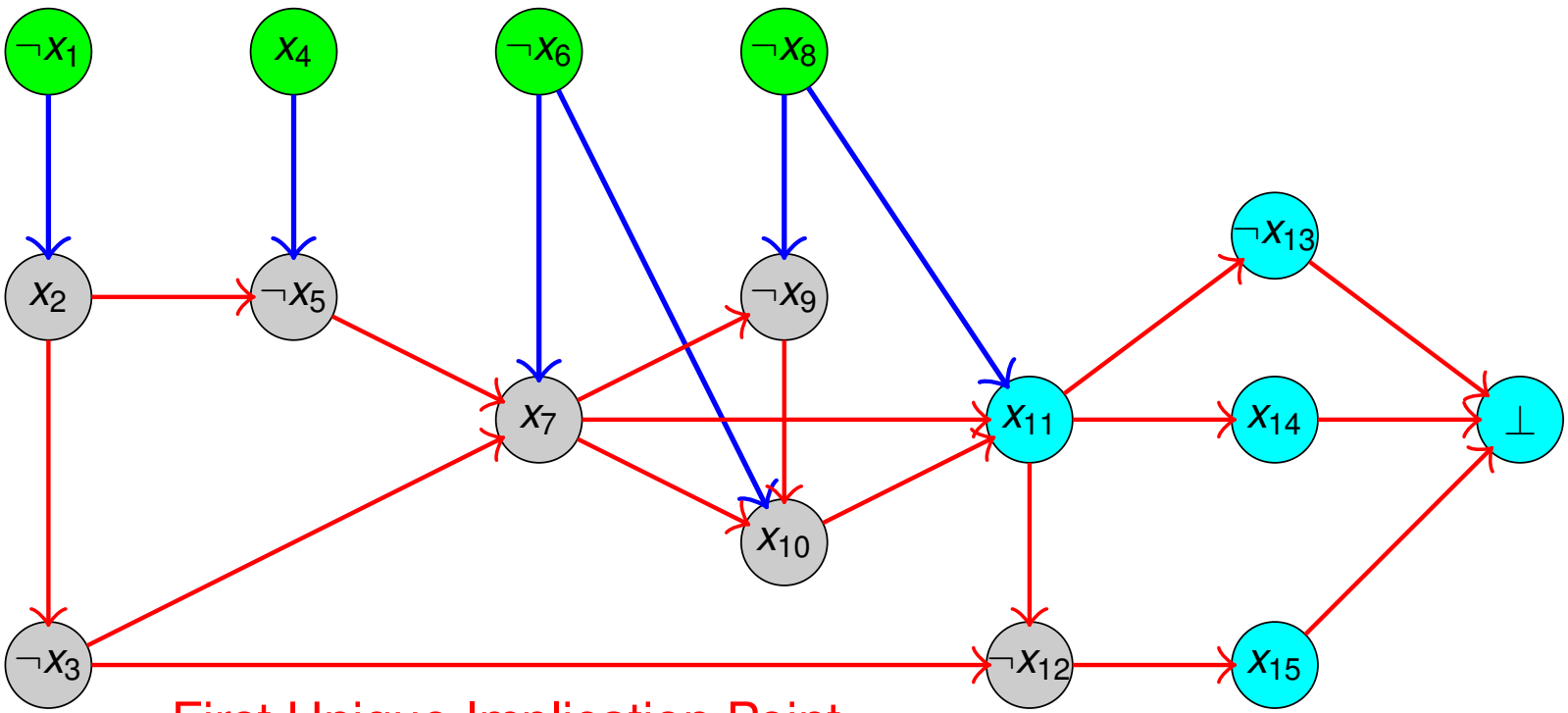
First UIP



First Unique Implication Point

C_1: $x_1 \vee x_2$
C_2: $\neg x_2 \vee \neg x_3$
C_3: $x_3 \vee \neg x_{12}$
C_4: $\neg x_2 \vee \neg x_4 \vee x_5$
C_5: $x_3 \vee x_5 \vee x_6 \vee x_7$
C_6: $\neg x_7 \vee x_8 \vee \neg x_9$
C_7: $x_6 \vee \neg x_7 \vee x_9 \vee x_{10}$
C_8: $\neg x_7 \vee \neg x_{10} \vee x_8 \vee x_{11}$
C_9: $x_8 \vee \neg x_7 \vee x_{11}$
C_{10}: $\neg x_{11} \vee \neg x_{13}$
C_{11}: $\neg x_{11} \vee x_{14}$
C_{12}: $x_{12} \vee x_{15}$
C_{13}: $x_{13} \vee \neg x_{14} \vee \neg x_{15}$

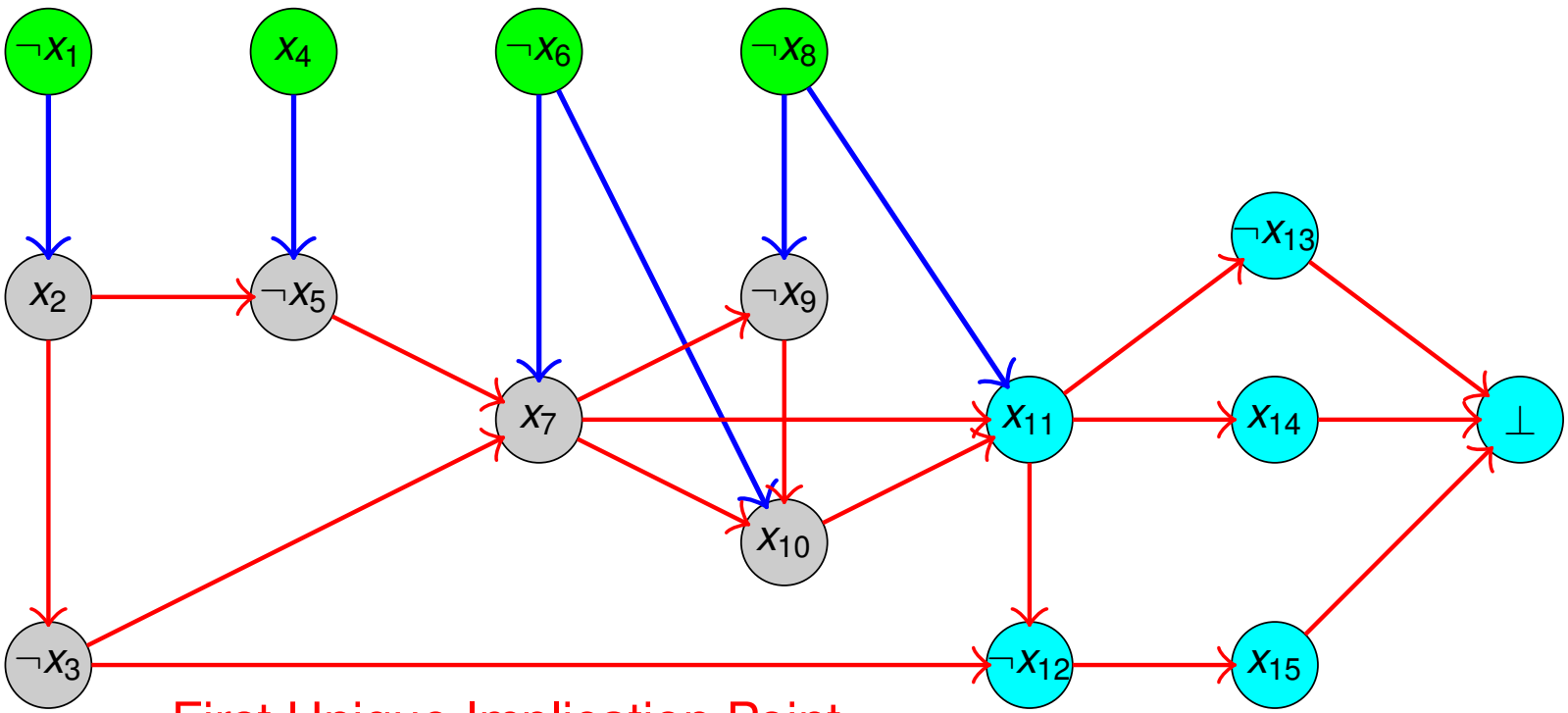
First UIP



First Unique Implication Point

C_1: $x_1 \vee x_2$
C_2: $\neg x_2 \vee \neg x_3$
C_3: $x_3 \vee \neg x_{12}$
C_4: $\neg x_2 \vee \neg x_4 \vee x_5$
C_5: $x_3 \vee x_5 \vee x_6 \vee x_7$
C_6: $\neg x_7 \vee x_8 \vee \neg x_9$
C_7: $x_6 \vee \neg x_7 \vee x_9 \vee x_{10}$
C_8: $\neg x_7 \vee \neg x_{10} \vee x_8 \vee x_{11}$
C_9: $x_8 \vee \neg x_7 \vee x_{11}$
C_{10}: $\neg x_{11} \vee \neg x_{13}$
C_{11}: $\neg x_{11} \vee x_{14}$
C_{12}: $x_{12} \vee x_{15}$
C_{13}: $x_{13} \vee \neg x_{14} \vee \neg x_{15}$

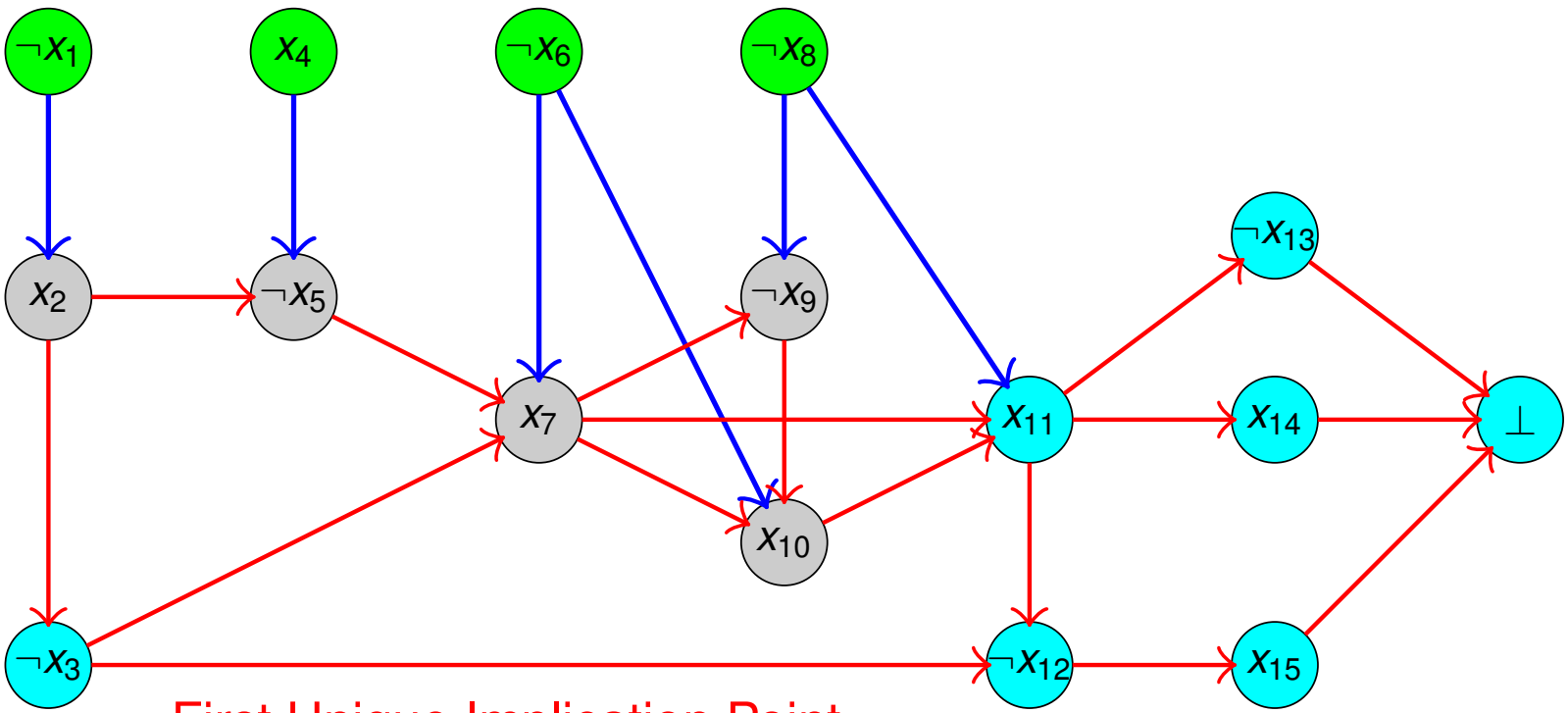
First UIP



First Unique Implication Point

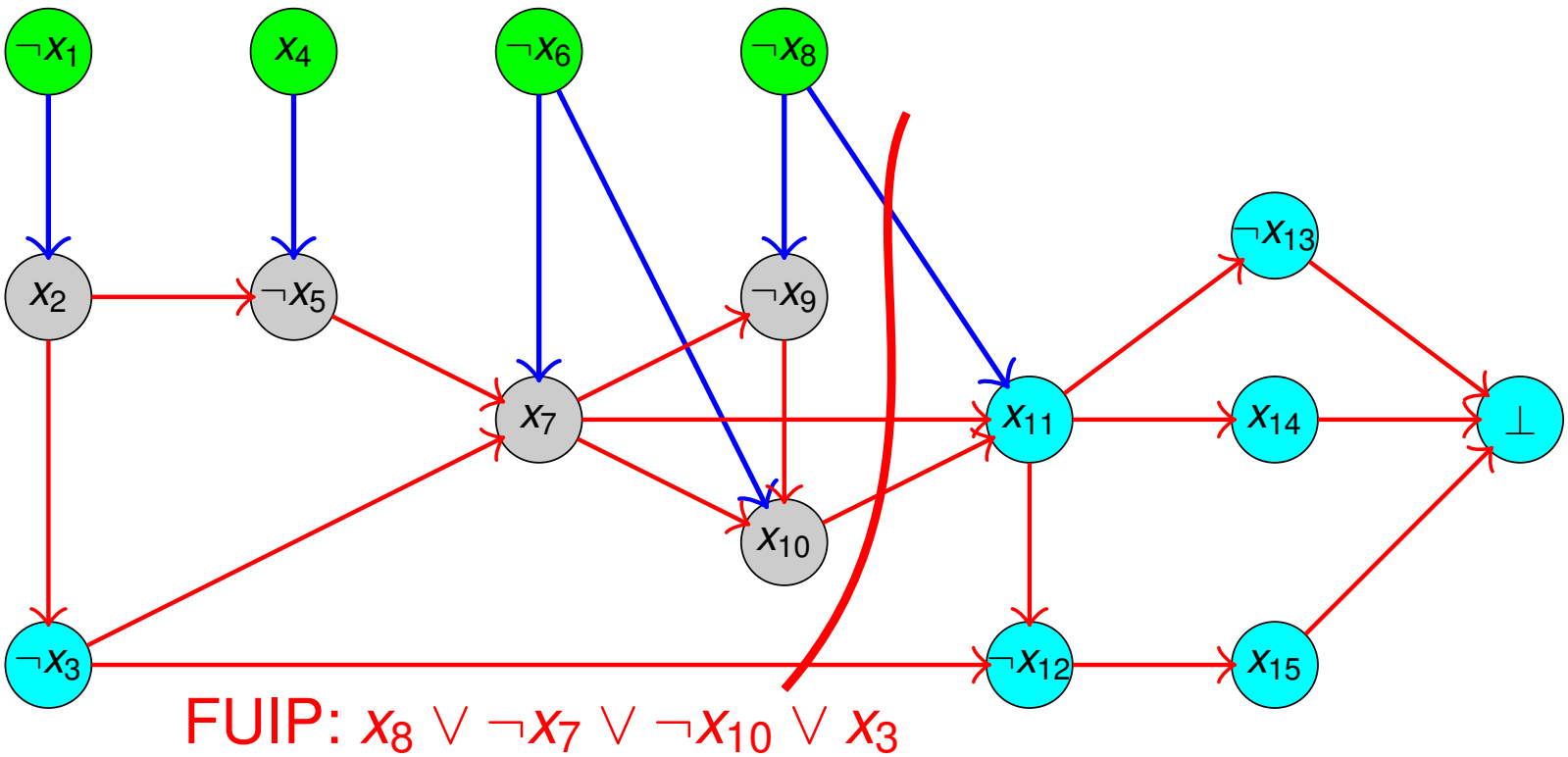
C_1: $x_1 \vee x_2$
C_2: $\neg x_2 \vee \neg x_3$
C_3: $x_3 \vee \neg x_{12}$
C_4: $\neg x_2 \vee \neg x_4 \vee x_5$
C_5: $x_3 \vee x_5 \vee x_6 \vee x_7$
C_6: $\neg x_7 \vee x_8 \vee \neg x_9$
C_7: $x_6 \vee \neg x_7 \vee x_9 \vee x_{10}$
C_8: $\neg x_7 \vee \neg x_{10} \vee x_8 \vee x_{11}$
C_9: $x_8 \vee \neg x_7 \vee x_{11}$
C_{10}: $\neg x_{11} \vee \neg x_{13}$
C_{11}: $\neg x_{11} \vee x_{14}$
C_{12}: $x_{12} \vee x_{15}$
C_{13}: $x_{13} \vee \neg x_{14} \vee \neg x_{15}$

First UIP



C_1: $x_1 \vee x_2$
C_2: $\neg x_2 \vee \neg x_3$
C_3: $x_3 \vee \neg x_{12}$
C_4: $\neg x_2 \vee \neg x_4 \vee x_5$
C_5: $x_3 \vee x_5 \vee x_6 \vee x_7$
C_6: $\neg x_7 \vee x_8 \vee \neg x_9$
C_7: $x_6 \vee \neg x_7 \vee x_9 \vee x_{10}$
C_8: $\neg x_7 \vee \neg x_{10} \vee x_8 \vee x_{11}$
C_9: $x_8 \vee \neg x_7 \vee x_{11}$
C_{10}: $\neg x_{11} \vee \neg x_{13}$
C_{11}: $\neg x_{11} \vee x_{14}$
C_{12}: $x_{12} \vee x_{15}$
C_{13}: $x_{13} \vee \neg x_{14} \vee \neg x_{15}$

First UIP



C_1: $x_1 \vee x_2$
C_2: $\neg x_2 \vee \neg x_3$
C_3: $x_3 \vee \neg x_{12}$
C_4: $\neg x_2 \vee \neg x_4 \vee x_5$
C_5: $x_3 \vee x_5 \vee x_6 \vee x_7$
C_6: $\neg x_7 \vee x_8 \vee \neg x_9$
C_7: $x_6 \vee \neg x_7 \vee x_9 \vee x_{10}$
C_8: $\neg x_7 \vee \neg x_{10} \vee x_8 \vee x_{11}$
C_9: $x_8 \vee \neg x_7 \vee x_{11}$
C_{10}: $\neg x_{11} \vee \neg x_{13}$
C_{11}: $\neg x_{11} \vee x_{14}$
C_{12}: $x_{12} \vee x_{15}$
C_{13}: $x_{13} \vee \neg x_{14} \vee \neg x_{15}$

VOICI (ENFIN) LA VRAIE PROCÉDURE SAT

EN ANGLAIS DANS LE TEXTE

Algorithme 3 : Conflict-Driven Clause Learning (CDCL)

$\mathcal{I} = \emptyset$, $level = 0$;
while *True* **do**
    Execute UP on $(\Sigma, \mathcal{I})$;
    if *a Conflict was reached* **then**
        if $level = 0$ **then** Retourner UNSAT ;
        C = the derived conflict clause;
        l = the sole literal of C set at the conflict level;
        $level = max\{level(x) : x \in C/\{l\}\}$;
        $\mathcal{I} = \mathcal{I}$ less all assignments made at level greater than $level$;
        $(\Sigma, \mathcal{I}) = (\Sigma \cup \{C\}, \mathcal{I}.l)$;
    end
    if $\mathcal{I}$ *is total* **then** Return SAT ;
    Choose a decision literal l occuring in $\Sigma|\mathcal{I}$;
    $\mathcal{I} = \mathcal{I}.l$;
    Increment *level*
end

Oublier un peu

Mais pas trop

Problème des chaff-like

L'apprentissage induit une consommation mémoire trop importante. Souvent, les algorithmes rendent l'âme à cause d'elle.

Point important : Oublier les clauses inintéressantes.

Les clauses trop longues ? Les clauses les moins actives ?

Ne jamais effacer les clauses garantissant la complétude.

Dans Minisat

L'effacement est aggressif. La moitié des clauses apprises sont régulièrement effacées, en prenant d'abord les clauses d'activité moindre.

Oublier un peu

Mais pas trop

Problème des chaff-like

L'apprentissage induit une consommation mémoire trop importante.
Souvent, les algorithmes rendent l'âme à cause d'elle.

Point important : Oublier les clauses inintéressantes.
Les clauses trop longues ? Les clauses les moins actives ?
Ne jamais effacer les clauses garantissant la complétude.

Dans Minisat

L'effacement est aggressif. La moitié des clauses apprises sont régulièrement effacées, en prenant d'abord les clauses d'activité moindre.

Oublier un peu

Mais pas trop

Problème des chaff-like

L'apprentissage induit une consommation mémoire trop importante. Souvent, les algorithmes rendent l'âme à cause d'elle.

Point important : Oublier les clauses inintéressantes.
Les clauses trop longues ? Les clauses les moins actives ?
Ne jamais effacer les clauses garantissant la complétude.

Dans Minisat

L'effacement est aggressif. La moitié des clauses apprises sont régulièrement effacées, en prenant d'abord les clauses d'activité moindre.

Des justifications expérimentales

Avec une goutte de théorie

Indispensable dans les instances structurées

L'apprentissage semble faire ressortir de l'information de qualité. Plus pertinente que si elle était ajoutée à la main ?

En théorie

L'apprentissage, couplé au redémarrage correspond à un système de preuves de la puissance de la résolution générale.

Vers une meilleur compréhension ?

Bons résultats expérimentaux, mais :

- Pourquoi l'apprentissage marche si bien ?
- Si on réinjecte les clauses apprisent en début de calcul, on perd en performance.
- Si on réordonne les clauses, les clauses apprises ne sont pas les mêmes, même si les performances restent identique.

Des justifications expérimentales

Avec une goutte de théorie

Indispensable dans les instances structurées

L'apprentissage semble faire ressortir de l'information de qualité. Plus pertinente que si elle était ajoutée à la main ?

En théorie

L'apprentissage, couplé au redémarrage correspond à un système de preuves de la puissance de la résolution générale.

Vers une meilleur compréhension ?

Bons résultats expérimentaux, mais :

- Pourquoi l'apprentissage marche si bien ?
- Si on réinjecte les clauses apprisent en début de calcul, on perd en performance.
- Si on réordonne les clauses, les clauses apprises ne sont pas les mêmes, même si les performances restent identique.

Des justifications expérimentales

Avec une goutte de théorie

Indispensable dans les instances structurées

L'apprentissage semble faire ressortir de l'information de qualité. Plus pertinente que si elle était ajoutée à la main ?

En théorie

L'apprentissage, couplé au redémarrage correspond à un système de preuves de la puissance de la résolution générale.

Vers une meilleur compréhension ?

Bons résultats expérimentaux, mais :

- Pourquoi l'apprentissage marche si bien ?
- Si on réinjecte les clauses apprisent en début de calcul, on perd en performance.
- Si on réordonne les clauses, les clauses apprises ne sont pas les mêmes, même si les performances restent identique.

UNE PREMIÈRE HEURISTIQUE :BOHM

[BURO AND KLEINE-BÜNING, 1992]

$$H_i(x) = \alpha max(h_i(x), h_i(\neg x)) + \beta min(h_i(x), h_i(\neg x))$$

$h_i(x)$ est le nombre de clauses non résolues contenant x, et de longueur i Il est suggéré de prendre $\alpha = 1$ et $\beta = 2$

IDÉE

On préfère satisfaire en priorité les clauses courtes, ou obtenir encore plus de clauses courtes.

UNE PREMIÈRE HEURISTIQUE :BOHM

[BURO AND KLEINE-BÜNING, 1992]

$$H_i(x) = \alpha max(h_i(x), h_i(\neg x)) + \beta min(h_i(x), h_i(\neg x))$$

$h_i(x)$ est le nombre de clauses non résolues contenant x, et de longueur i Il est suggéré de prendre $\alpha = 1$ et $\beta = 2$

IDÉE

On préfère satisfaire en priorité les clauses courtes, ou obtenir encore plus de clauses courtes.

PRIVILÉGIER LES CLAUSES COURTES : MOM

MAXIMUM OCCURENCES ON CLAUSES OF MINIMUM SIZE [DUBOIS ET AL., 1996]

$$[f^*(x) + f^*(\neg x)] * 2^k + f^*(x) * f^*(\neg x)$$

IDÉE

Les clauses courtes pèsent *exponentiellement* plus lourd, mais la recherche reste équilibrée.

PRIVILÉGIER LES CLAUSES COURTES : MOM

MAXIMUM OCCURENCES ON CLAUSES OF MINIMUM SIZE [DUBOIS ET AL., 1996]

$$[f^*(x) + f^*(\neg x)] * 2^k + f^*(x) * f^*(\neg x)$$

IDÉE

Les clauses courtes pèsent *exponentiellement* plus lourd, mais la recherche reste équilibrée.

JEROSLOW-WANG
[JEROSLOW AND WANG, 1990]

$$J(l) = \Sigma 2^{-|\omega|}$$

IDÉE

Toujours un poids exponentiellement plus important aux clauses courtes.

On peut mesurer le littéral avec le plus grand $J(l)$ ou la variable dont la somme $J(x) + J(\neg x)$ est la plus grande (2-Sided JW).

JEROSLOW-WANG

[JEROSLOW AND WANG, 1990]

$$J(l) = \Sigma 2^{-|\omega|}$$

IDÉE

Toujours un poids exponentiellement plus important aux clauses courtes.

On peut mesurer le littéral avec le plus grand $J(l)$ ou la variable dont la somme $J(x) + J(\neg x)$ est la plus grande (2-Sided JW).

DLIS

Dynamic Largest Individual Sum [Marques, 1999]

Tout simplement :

$$C_p + C_n$$

où C_p est le nombre de clauses non résolues... simplification de BOHM.

Observation

Donne des résultats pas si mauvais que cela... et même parfois bons.

DLIS

Dynamic Largest Individual Sum [Marques, 1999]

Tout simplement :

$$C_p + C_n$$

où C_p est le nombre de clauses non résolues... simplification de BOHM.

Observation

Donne des résultats pas si mauvais que cela... et même parfois bons.

VSIDS

Variable State Independant Decaying Sum

L'entonnoir heuristique

Toutes les heuristiques dépendent de l'affectation courante. Comme les performances des démonstrateurs se sont accrues, les algorithmes passaient tout leur temps à maintenir les valeurs heuristiques.

Idée [Zhang, 01]

- Au départ, $h(l)$ contient son nombre d'occurences dans la formule
- À chaque fois qu'une clause de conflit c est apprise, $h(l)$ est incrémenté d'une constante si c contient l.
- Régulièrement, les compteurs sont divisés par une constante

VSIDS

Variable State Independant Decaying Sum

L'entonnoir heuristique

Toutes les heuristiques dépendent de l'affectation courante. Comme les performances des démonstrateurs se sont accrues, les algorithmes passaient tout leur temps à maintenir les valeurs heuristiques.

Idée [Zhang, 01]

- Au départ, $h(l)$ contient son nombre d'occurences dans la formule
- À chaque fois qu'une clause de conflit c est apprise, $h(l)$ est incrémenté d'une constante si c contient l.
- Régulièrement, les compteurs sont divisés par une constante

Dans Minisat

Problème : On sait sur quelle variable brancher, mais quelle valeur lui donner ? Réponse de Minisat :

```
check(assume(~Lit(next)));  // Arbitrarly default to ne
```

Incroyable : toutes les variables sont assignées d'abord à faux.

DANS RSAT

GAGNANT DE LA COMPÉTITION 2007

UNE IDÉE ULTRA-SIMPLE

Méthode de cache pour VSIDS. On mémorise la dernière branche prise pour cette variable.

Donne de très grosses améliorations sur certains problèmes !

DANS RSAT

GAGNANT DE LA COMPÉTITION 2007

UNE IDÉE ULTRA-SIMPLE

Méthode de cache pour VSIDS. On mémorise la dernière branche prise pour cette variable.

Donne de très grosses améliorations sur certains problèmes !

Les redémarrages rapides

Une technique fondamentale

Principes

De temps en temps, on reprend la recherche à zéro, en gardant tout de même les meilleurs clauses apprises.

Dans Zchaff, à l'origine

Un redémarrage tous les 10000 conflits.

Aujourd'hui : environ tous les **100** conflits.

Les redémarrages rapides

Une technique fondamentale

Principes

De temps en temps, on reprend la recherche à zéro, en gardant tout de même les meilleurs clauses apprises.

Dans Zchaff, à l'origine

Un redémarrage tous les 10000 conflits.

Aujourd'hui : environ tous les **100** conflits.

Les redémarrages rapides

Une technique fondamentale

Principes

De temps en temps, on reprend la recherche à zéro, en gardant tout de même les meilleurs clauses apprises.

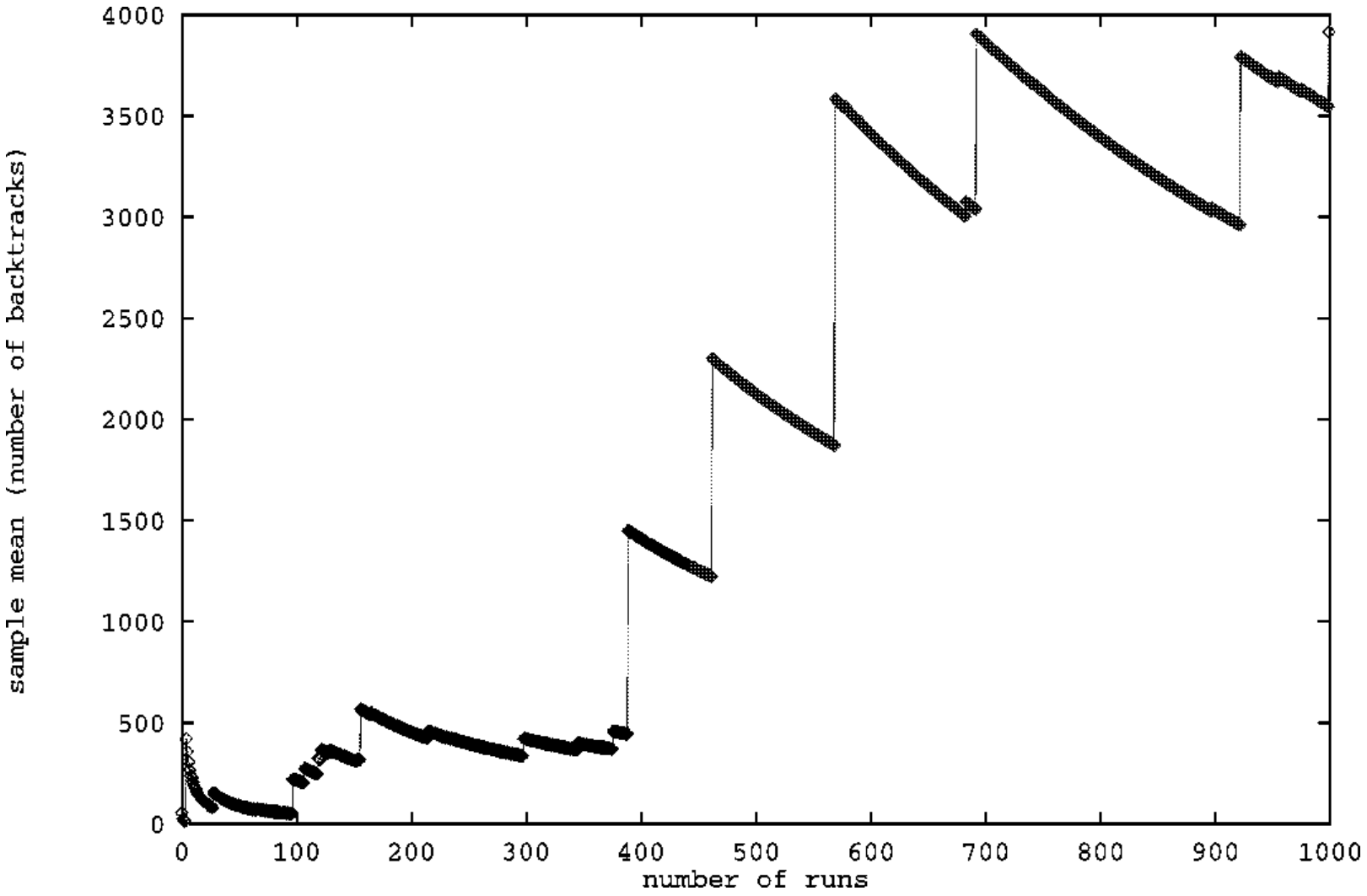
Dans Zchaff, à l'origine

Un redémarrage tous les 10000 conflits.

Aujourd'hui : environ tous les **100** conflits.

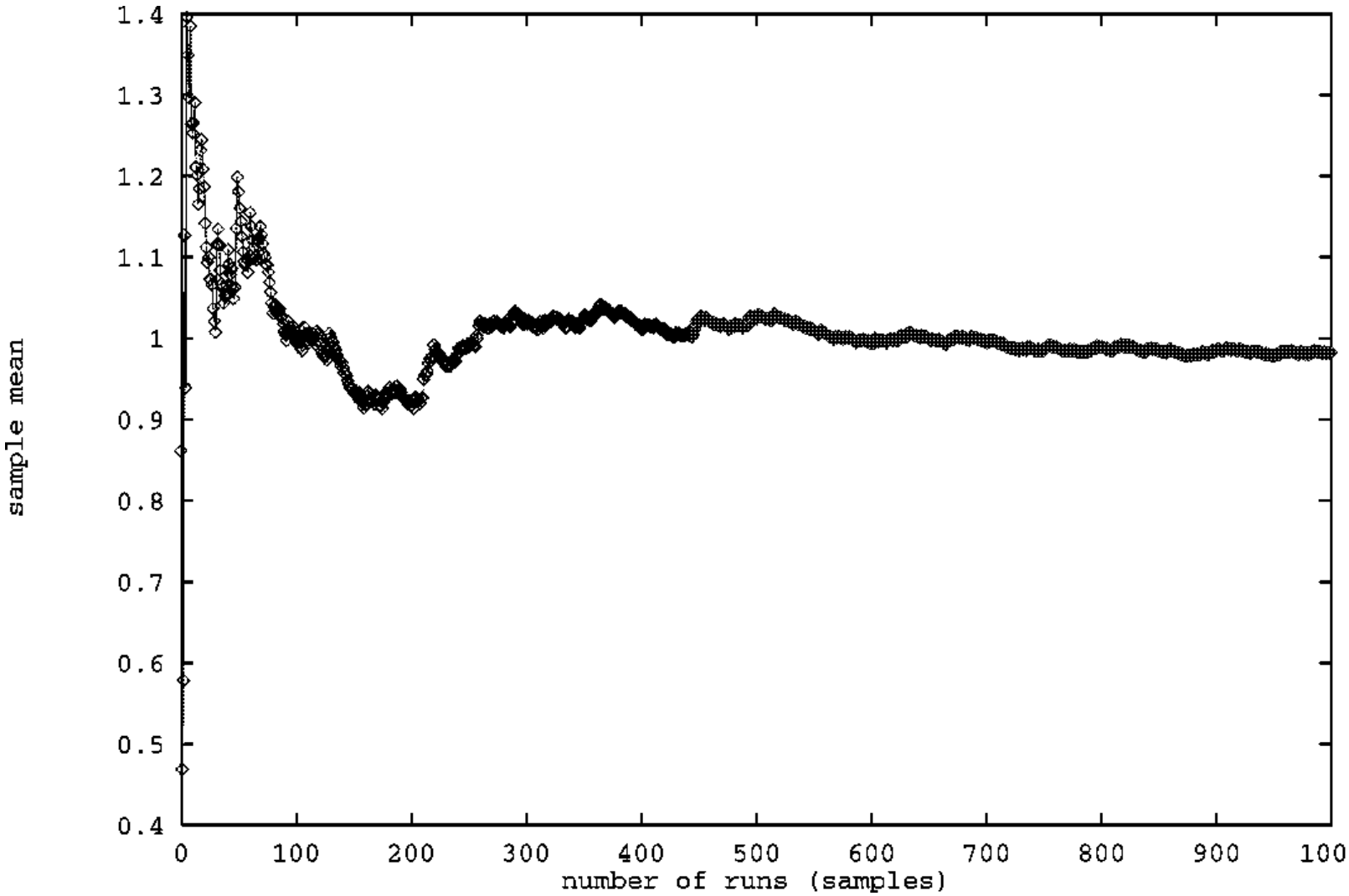
EXPLICATIONS SUCCINCTES

MOYENNE DE TEMPS CPU SUR PLUSIEURS LANCEMENTS

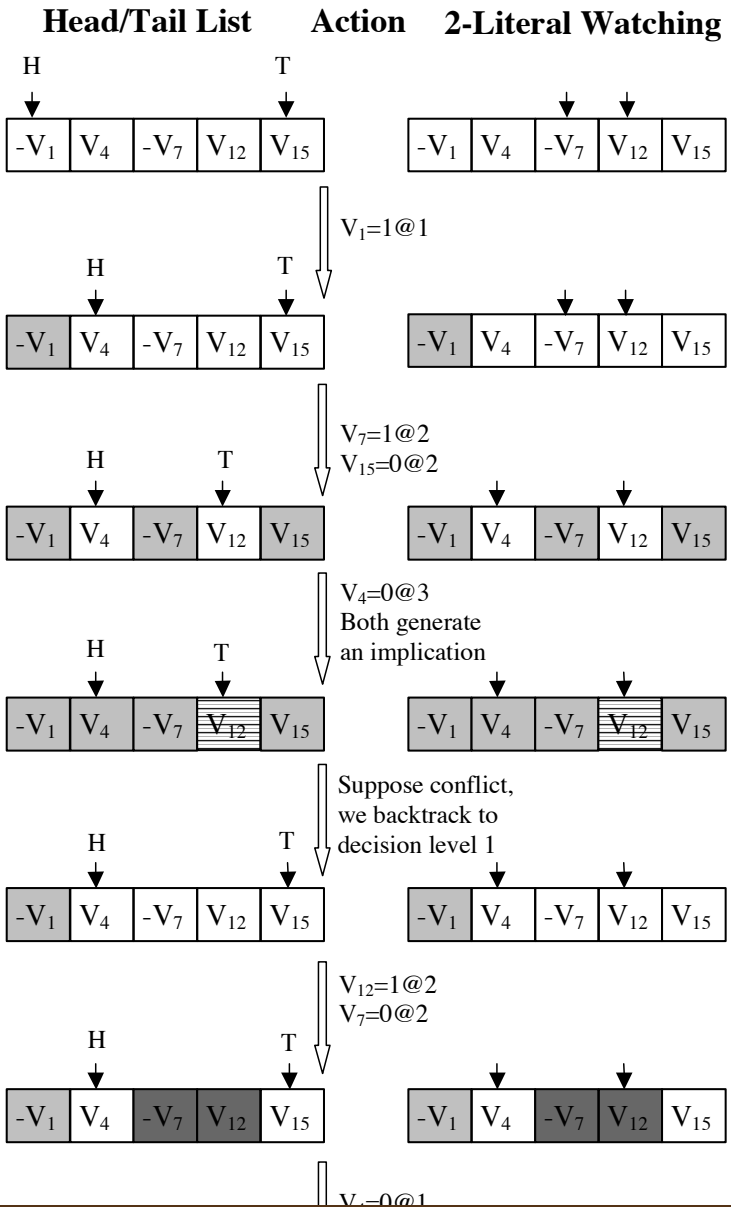


EXPLICATIONS SUCCINCTES

MÉDIANE DE TEMPS CPU SUR PLUSIEURS LANCEMENTS



FAIRE JUSTE ASSEZ, DÉFAIRE LE MOINS POSSIBLE



Implantation proche de la machine

	zChaff	SATO	GRASP
# Decisions	3166	3771	1795
# Instructions	86.6M	630.4M	1415.9M
# L1/L2 accesses	24M / 1.7M	188M / 79M	416M / 153M
% L1/L2 misses	4.8% / 4.6%	36.8% / 9.7%	32.9% / 50.3%
# Seconds	0.22	4.41	11.78

Plan de l'exposé

Small is Beautiful

Là où les ordinateurs sont forts

- Beaucoup d'efforts, mais des progrès réels et impressionnants
- Des pièges partout (réordonnancement, fausses bonnes idées, expérimentations douteuses)
- Vers une vraie expérimentation des solveurs ?
- Nouveaux paradigmes parallèle

Un challenge : appréhender les preuves UNSAT courtes.

Small is Beautiful

Là où les ordinateurs sont forts

- Beaucoup d'efforts, mais des progrès réels et impressionnants
- Des pièges partout (réordonnancement, fausses bonnes idées, expérimentations douteuses)
- Vers une vraie expérimentation des solveurs ?
- Nouveaux paradigmes parallèle

Un challenge : appréhender les preuves UNSAT courtes.

SMALL IS BEAUTIFUL

LÀ OÙ LES ORDINATEURS SONT FORTS

- Beaucoup d'efforts, mais des progrès réels et impressionnants
- Des pièges partout (réordonnancement, fausses bonnes idées, expérimentations douteuses)
- Vers une vraie expérimentation des solveurs ?
- Nouveaux paradigmes parallèle

Un challenge : appréhender les preuves UNSAT courtes.

SMALL IS BEAUTIFUL

LÀ OÙ LES ORDINATEURS SONT FORTS

- Beaucoup d'efforts, mais des progrès réels et impressionnants
- Des pièges partout (réordonnancement, fausses bonnes idées, expérimentations douteuses)
- Vers une vraie expérimentation des solveurs ?
- Nouveaux paradigmes parallèle

Un challenge : appréhender les preuves UNSAT courtes.

SMALL IS BEAUTIFUL

LÀ OÙ LES ORDINATEURS SONT FORTS

- Beaucoup d'efforts, mais des progrès réels et impressionnants
- Des pièges partout (réordonnancement, fausses bonnes idées, expérimentations douteuses)
- Vers une vraie expérimentation des solveurs ?
- Nouveaux paradigmes parallèle

Un challenge : appréhender les preuves UNSAT courtes.

SMALL IS BEAUTIFUL

LÀ OÙ LES ORDINATEURS SONT FORTS

- Beaucoup d'efforts, mais des progrès réels et impressionnants
- Des pièges partout (réordonnancement, fausses bonnes idées, expérimentations douteuses)
- Vers une vraie expérimentation des solveurs ?
- Nouveaux paradigmes parallèle

Un challenge : appréhender les preuves UNSAT courtes.

Bibliographie I

Buro, M. and Kleine-Büning, H. (1992).

Report on a SAT competition.
Technical report, University of Paderborn.

Cook, S. (1971).

The complexity of theorem proving procedures.
In *ACM Symposium of Theory of Computing*, pages 151–158.

Davis, M., Logemann, G., and Loveland, D. (1962).

A machine program for theorem proving.
Communications of the ACM, 5(7):394–397.

Davis, M. and Putnam, H. (1960).

A computing procedure for quantification theory.
Journal of the ACM, 7(3):201–215.

Dubois, O., André, P., Boufkhad, Y., and Carlier, J. (1996).

SAT versus UNSAT.
In *Second DIMACS implementation challenge : cliques, coloring and satisfiability*, volume 26 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 415–436. American Mathematical Society.

Dubois, O. and Dequen, G. (2001).

A backbone-search heuristic for efficient solving of hard 3-sat formulae.
In *17th International Joint Conference on Artificial Intelligence (IJCAI'01)*, pages 248–253, Seattle, Washington, USA.

Garey, M. and Johnson, D. (1979).

Computers and Intractability: A guide to the Theory of NP-Completeness.
W.H. Freeman and Company.

Bibliographie II

Jeroslow, R. J. and Wang, J. (1990).

Solving propositional satisfiability problems.
Annals of Mathematics and Artificial Intelligence, 1:167–188.

Li, C.-M. and Anbulagan (1997).

Heuristics based on unit propagation for satisfiability problems.
In *Proceedings of Internation Joint conference on Artificial Intelligence (IJCAI'97)*, pages 366–371.

Li, C.-M. and Gérard, S. (2000).

On the limit of branching rules for hard random unsatisfiable 3-SAT.
In *Proceedings of the 14th European Conference on Artificial Intelligence (ECAI-2000)*, pages 98–102, Berlin.

Marques (1999).

The impact of Branching Heuristics in Propositional Satisfiability Algorithms.

Marques-Silva, J. P. and Sakallah, K. A. (1996).

GRASP - A New Search Algorithm for Satisfiability.
In *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pages 220–227.

Robinson, J. (1965).

A machine-oriented logic based on the resolution principle.
JACM, 12:23–41.

Selman, B., Kautz, H., and Cohen, B. (1996).

Local search strategies for satisfiability testing.
In *Second DIMACS implementation challenge : cliques, coloring and satisfiability*, volume 26 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 521–532. American Mathematical Society.

BIBLIOGRAPHIE III

Selman, B., Levesque, H., and Mitchell, D. (1992).

A new method for solving hard satisfiability problems.

In *Proceedings of the National Conference on Artificial Intelligence (AAAI'92)*, pages 440–446.