Algorithmique de la Planification

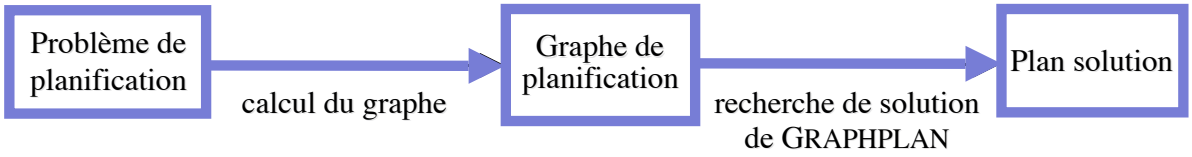
Cours 4 : GRAPHPLAN

Pierre REGNIER
IRIT - Université Paul Sabatier
http://www.irit.fr/~Pierre.Regnier

0. Plan de l'exposé

1.1. Principes du planificateur GRAPHPLAN

GRAPHPLAN [Blum, Furst, 1995]
IPP [Koehler et col., 1997]
STAN [Long, Fox, 1999]
LCGP [Cayrol, Régnier, Vidal, 2000]



Compilation de plans : GRAPHPLAN

1.1. Principes du planificateur GRAPHPLAN

- GRAPHPLAN **sépare la planification en deux procédures :**
  - la construction (complexité polynomiale en temps et en espace par rapport à la taille des données du problème) du graphe de planification. Des optimisations efficaces existent, le temps de construction varie beaucoup suivant les domaines (fonction du nombre d'exclusions mutuelles entre actions et fluents) ;
  - la recherche d'une solution potentielle dans le sous-arbre extrait de ce graphe (NP), qui peut être réalisée par différentes méthodes (recherche arrière, SAT, Davis-Putnam, CSP, recherche locale…) ;
- **Le graphe fournit de nombreuses informations** susceptibles de fournir une **heuristique** pour les méthodes classiques (recherche dans les espaces d'états, de plans…), il peut être adapté pour la prise en compte de ressources et du temps.

© Régnier, IRIT 2006

1.1. Principes du planificateur GRAPHPLAN

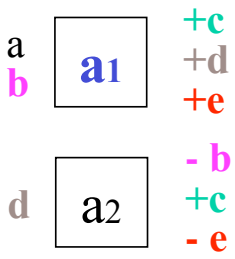
- Difficulté de la planification :
  - Déterminer les actions applicables et pertinentes pour atteindre le but (existentialité) :
    - **Application d'une action :**
      - une action *A* est applicable sur un état E ssi Prec(A) ⊆ E,
      - l'état résultant est l'ensemble de fluents :

$$E \uparrow A = (E - Del(A)) \cup Add(A)$$

    - **Pertinence des actions (guidée par les buts) :**
      - celles qui supportent les préconditions d'actions supportant … les préconditions d'actions qui supportent … le but.
      - autres critères (niveau, nombre préconditions, temps, ressources…) ?
  - Résoudre leurs interactions (ordonnancement)

1.1. Principes du planificateur GRAPHPLAN

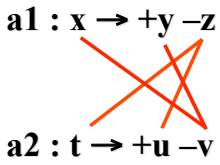
- **Typologie des interactions**



  - **Interactions positives :**
    - **Effets multiples :** action qui produit plusieurs fluents : **action a1**
    - **Add/Add :** ∃ f, f ∈ Add(a1) ∩ Add(a2) : **fluent c**
    - **Add/Prec** : ∃ f, f ∈ Add(a1) ∩ Prec(a2) : **fluent d**
  - **Interactions négatives :**
    - **Effets antagonistes :** ∃ f, f ∈ Add(a1) ∩ Del(a2) : **fluent e**
    - **Interactions croisées :** ∃ f, f ∈ Del(a2) ∩ Prec(a1) : **fluent b**

1.2. Définitions

- Deux **actions** a1, a2 sont **indépendantes** (noté a1 # a2) ssi elles n'ont pas d'interactions négatives càd :
  - Del(a1) ∩ (Prec(a2) ∪ Add(a2)) = ∅ et
  - Del(a2) ∩ (Prec(a1) ∪ Add(a1)) = ∅

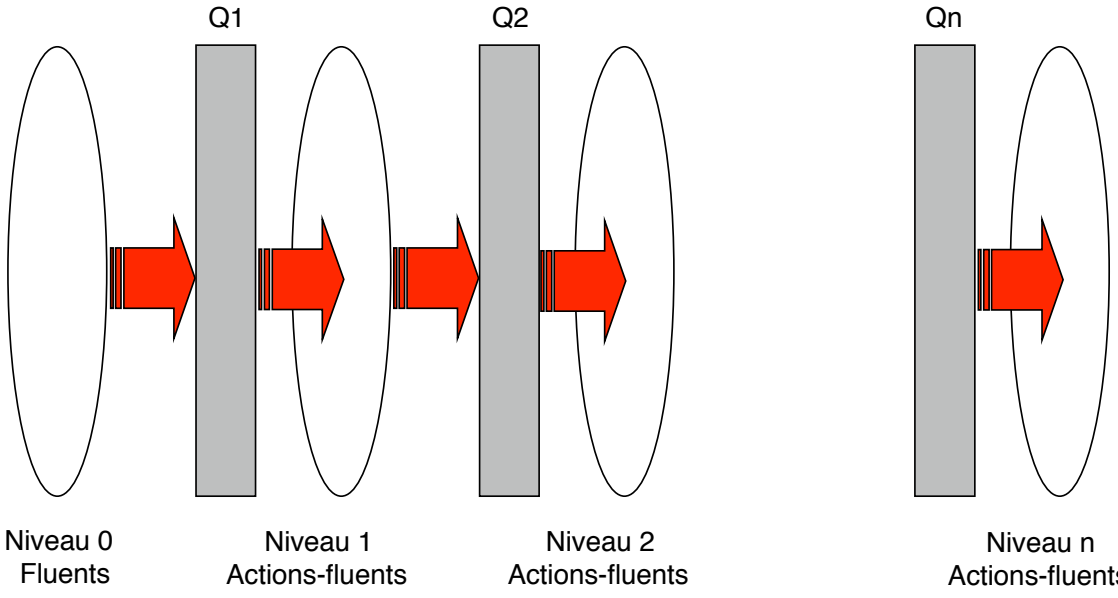


- **Ensemble** d'actions **indépendantes :**
  - Q est un ensemble d'actions indépendantes ou **ensemble indépendant** ssi toutes les actions Ai qui le composent sont indépendantes 2 à 2 ;
  - **Application d'un ensemble indépendant (chaînage avant) :**
    - un ensemble indépendant Q est applicable sur un état E ssi : ∪Prec(Ai) ⊆ E
    - l'état résultant est l'ensemble de fluents :

$$E \uparrow Q = (E - \cup Del(Ai)) \cup (\cup Add(Ai))$$

1.2. Définitions

- Un **plan d'ensembles d'actions** est une séquence finie P = <Q1, Q2,…, Qn> d'ensembles finis d'actions Qi. Un tel plan est noté <Qi>n, n ∈ N.



1.2. Définitions

- L'application $\mathcal{A}$ d'un plan d'ensembles d'actions P à un état E est définie par :

  E $\mathcal{A}$ P = Si P = <> ou E = $\perp$ alors E
  Sinon Si Prec(tête(P)) $\subseteq$ E
  Alors (E$\uparrow$tête(P)) $\mathcal{A}$ reste(P)
  Sinon $\perp$.

- Soit Φ une contrainte sur les ensembles d'actions : dans GRAPHPLAN les **ensembles doivent être indépendants**. Un plan d'ensembles d'actions P est un plan-solution au problème de planification <O, I, B>, relativement à Φ si et seulement si P satisfait les propriétés suivantes :
  - $\forall i \in [1, n]$, $Qi = \{a_1, a_2, \ldots, a_k\}$, $k \in N$ et $\forall m \in [1, k]$, a_m est obtenu par l'instanciation d'un opérateur de O.
  - $\forall i \in [1, n]$, Φ (Qi).
  - B $\subseteq$ I $\mathcal{A}$ P.

1.2. Définitions

- Dans GRAPHPLAN, deux actions d'un même niveau dans le graphe sont **mutuellement exclusives** (mutex) ssi :
  - elles ne sont pas indépendantes ou,
  - elles ont des préconditions mutex au niveau précédent (elles ne peuvent donc pas être déclenchées en même temps) :

    $\exists$ (p,q) $\in$ Prec(a1) x Prec(a2), telles que p et q sont mutex.

    - Deux **fluents** p et q sont **mutex** au niveau i ssi tous les couples d'actions qui les produisent à ce même niveau sont mutex (il n'existe pas de couple d'actions non mutex qui les produise à ce niveau) :
      $\forall$ a1, a2 / p $\in$ Add(a1), q $\in$ Add(a2), a1 et a2 mutex.

© Régnier, IRIT 2006

1.3. Déroulement de l'algorithme

$A : a \rightarrow +b$

$B : a \rightarrow +c\ -a$

$C : b\ c \rightarrow +d$

$NoOps\ \{a,b,c,d\}$

a

Niveau 0

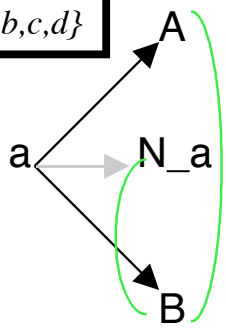
1.3. Déroulement de l'algorithme

$A : a \rightarrow +b$

$B : a \rightarrow +c\ -a$

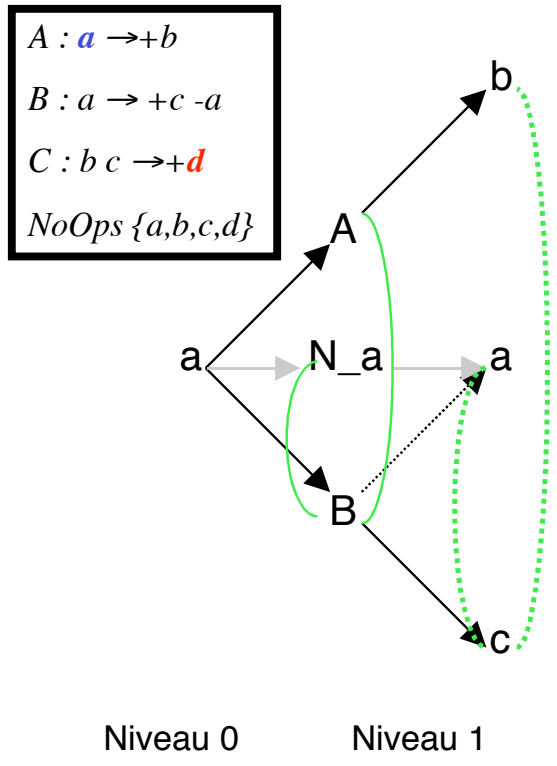
$C : b\ c \rightarrow +d$

$NoOps\ \{a,b,c,d\}$

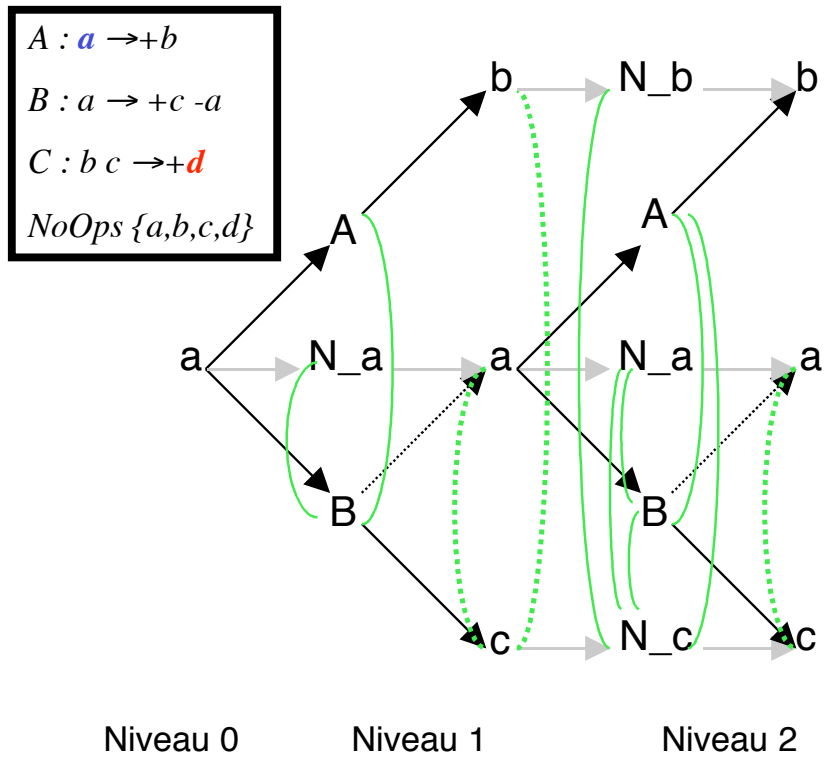


Niveau 0 Niveau 1

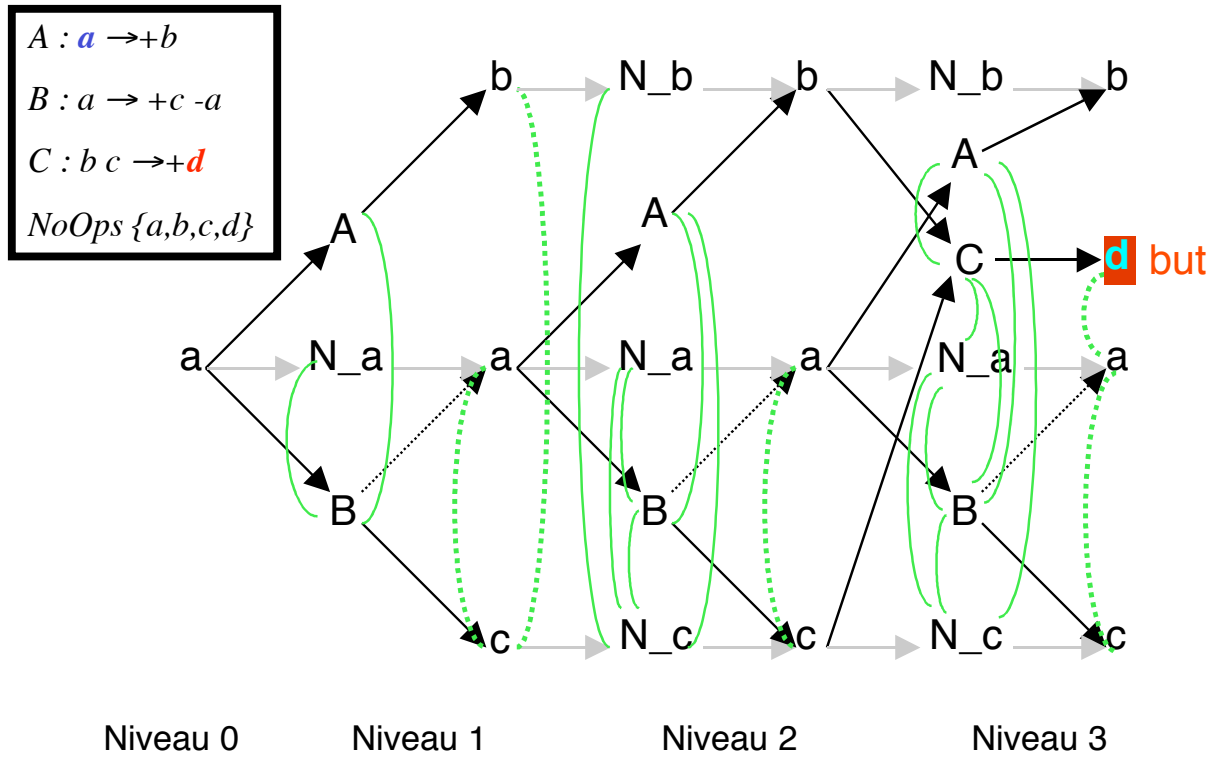
1.3. Déroulement de l'algorithme



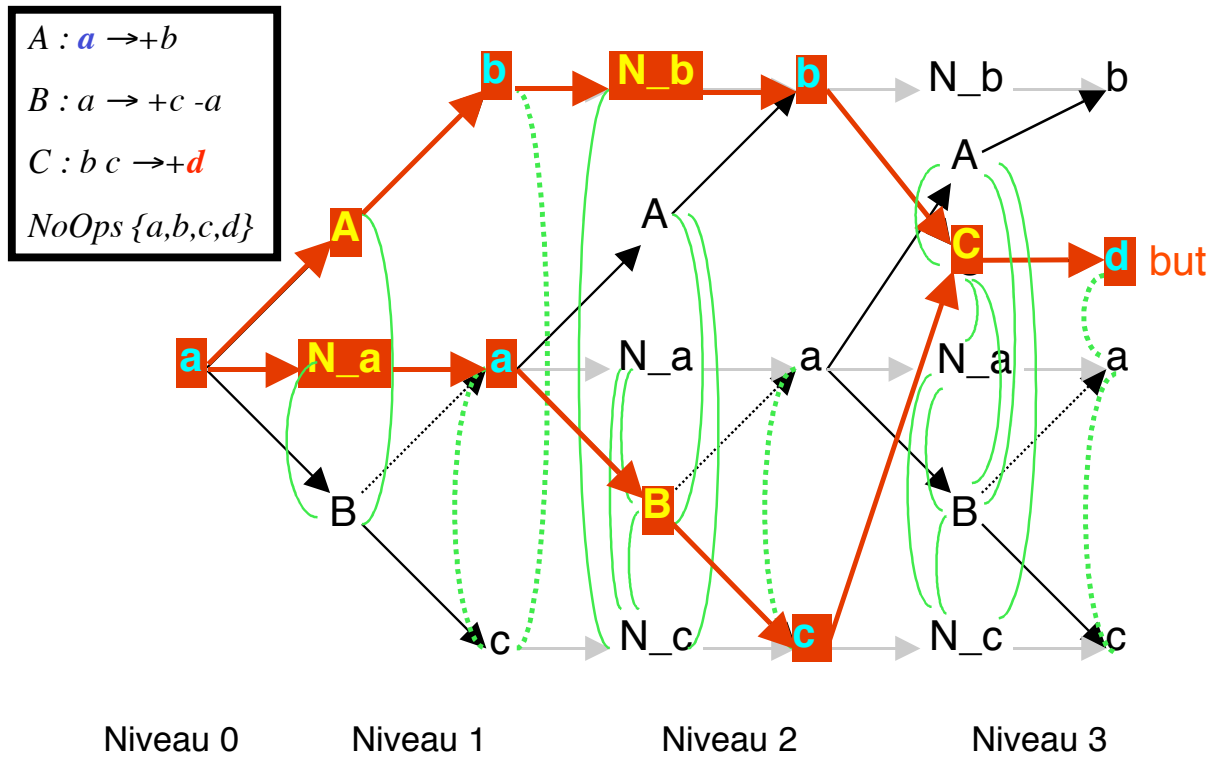
1.3. Déroulement de l'algorithme



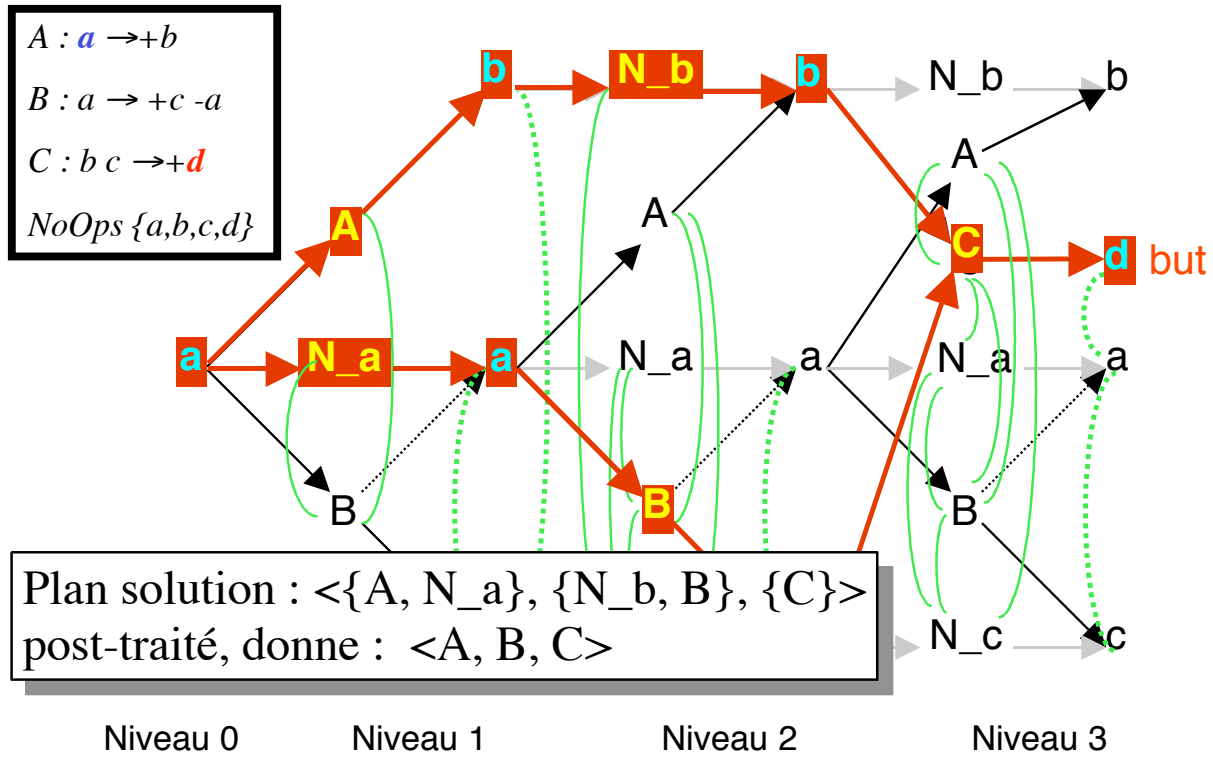
1.3. Déroulement de l'algorithme



1.3. Déroulement de l'algorithme

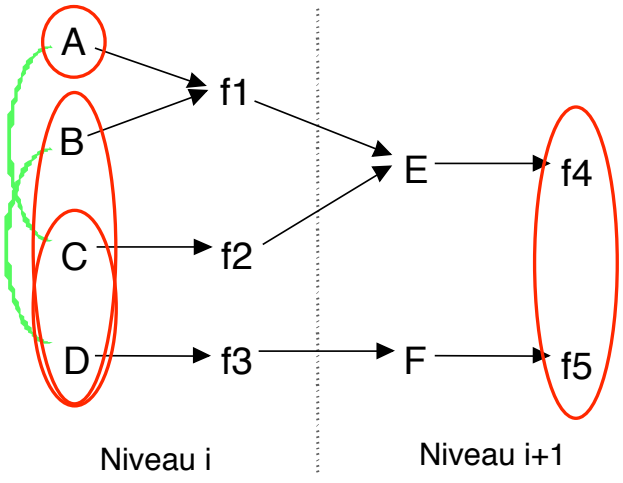


1.3. Déroulement de l'algorithme



1.3. Déroulement de l'algorithme

- **Exemple d'insuffisance des mutex binaires :**



1.4. Conclusion

- **Importance de GRAPHPLAN :**
  - Le plus efficace des algorithmes 1995-1999 ;
  - Optimalité des plans-solutions en nombre de niveaux ;
  - Planificateurs basés sur le graphe de planification :
    - GRAPHPLAN, IPP, STAN, LCGP...
    - BLACKBOX (SAT),
    - DPP (Davis et Putnam),
    - GP-CSP (CSP),
    - LGP (recherche locale) ;
  - Utilisation du graphe de planification pour la recherche heuristique :
    - Dans les espaces d'états (HSP, FF, ALTALT, YAHP…) ;
    - Dans les espaces de plans partiels (REPOP, VHPOP…) ;
  - Flexibilité du graphe de planification :
    - Relaxation, développement → stabilisation,
    - Graphe numérique, temporel…

2.1. Améliorations essentielles

- **Améliorations :**
  - Optimisation de la construction du graphe de planification :
    - **Implémentation, sélection, typage ;**
    - **Relation d'autorisation (LCGP) ;**
  - Optimisation de l'extraction d'une solution :
    - **Relation d'autorisation (LCGP) ;**
    - **Fonctions heuristiques indépendantes du domaine (LCGP, GPCSP) ;**
    - **Procédure de Davis et Putnam sur le graphe (DPPLAN) ;**
    - **Codages SAT du graphe (BLACKBOX, SATPLAN'04) ;**
    - **Codages CSP du graphe (GPCSP) ;**
    - **Techniques de recherche locale (LGP).**

2.1. Améliorations essentielles

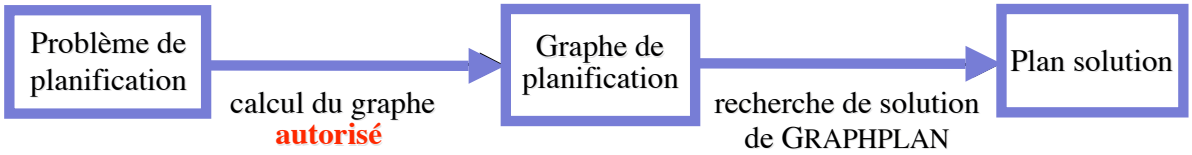
- **Implémentation optimisée :**
  - Structure circulaire à deux niveaux et calcul des mutex ;
  - Tableaux de bits, stockage des mutex TT ;
- **Sélection d'opérateurs et de fluents pertinents :**
  - Typage et instanciation des opérateurs :
    - Suppression des prédicats et fluents inertes (présents à tous les niveaux du graphe) qui ne sont donc pas pertinents ;
    - Module d'Inférence des Types (TIM) ;
  - Construction du graphe guidée par les buts ;
  - Choix heuristique des fluents de l'état initial guidée par le but ;
- **Construction compacte du graphe :** LCGP (Least Committed GRAPHPLAN)
- **Heuristiques pour l'extraction de solution :** LCGP, GPCSP
- **Extraction de solution avec Davis et Putnam, SAT :** DPPLAN, BLACKBOX, SATPLAN'04
- **Extraction de solution avec techniques CSP :** GPCSP

2.2. Relation d'autorisation

- **Première amélioration :**
  - Optimisation de la construction du graphe de planification et de l'extraction d'une solution :
    - **Relation d'autorisation**

2.2. Relation d'autorisation

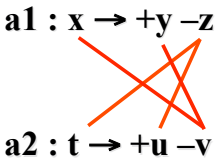
LCGP [Cayrol, Régnier, Vidal, 2000]



Compilation de plans : LCGP

2.2. Relation d'autorisation

- Deux **actions** a1, a2 sont **indépendantes** (noté a1 # a2) ssi elles n'ont pas d'interactions négatives càd :
  - Del(a1) ∩ (Prec(a2) ∪ Add(a2)) = ∅ et
  - Del(a2) ∩ (Prec(a1) ∪ Add(a1)) = ∅



- **Ensemble** d'actions **indépendantes :**
  - Q est un **ensemble indépendant** ssi toutes les actions Ai qui le composent sont indépendantes 2 à 2 ;
  - **Application d'un ensemble indépendant (chaînage avant) :**
    - un ensemble indépendant Q est applicable sur un état E ssi : ∪Prec(Ai) ⊆ E
    - l'état résultant est l'ensemble de fluents :

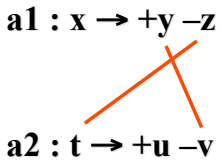
$$E \uparrow Q = (E - \cup Del(Ai)) \cup (\cup Add(Ai))$$

2.2. Relation d'autorisation

- Deux **actions** a1, a2 sont **autorisées** ssi a1 **autorise** a2 (noté a1 ∠ a2) ou a2 **autorise** a1 (noté a2 ∠ a1) ssi :
  - Del(a1) ∩ (Prec(a2) ∪ Add(a2)) = ∅ et
  - Del(a2) ∩ (Prec(a1) ∪ Add(a1)) = ∅

  Del(a1) ∩ Prec(a2) = ∅ et
  Del(a2) ∩ Add(a1) = ∅

  L'exécution des deux actions reste possible dans un ordre donné et produit l'ensemble de leurs effets ;



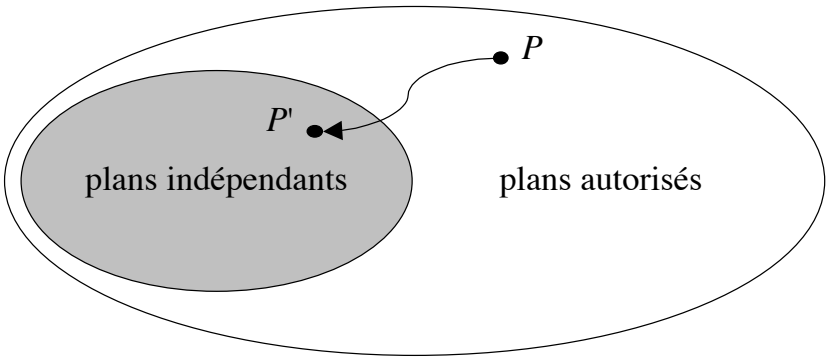
- **Ensemble** d'actions **autorisées :**
  - Q est un **ensemble autorisé** ssi il existe une séquence autorisée de toutes les actions Ai qui le composent ;
  - **Application d'un ensemble autorisé (chaînage avant) :**
    - un ensemble autorisé Q est applicable sur un état E ssi : ∪Prec(Ai) ⊆ E
    - l'état résultant est l'ensemble de fluents :

      E ↑ Q = (E - ∪Del(Ai)) ∪ (∪Add(Ai))

2.2. Relation d'autorisation

- Toutes les linéarisations autorisées d'un plan autorisé mènent au même état résultant, qui correspond à l'application simultanée des actions des ensembles autorisés qui le composent.

- Un plan autorisé peut être réordonnancé en un plan indépendant optimal en nombre de niveaux, en effectuant un tri topologique modifié sur un graphe d'ordre partiel (algorithme PRF polynomial).



© Régnier, IRIT 2006

2.2. Relation d'autorisation

- Dans GRAPHPLAN, deux actions d'un même niveau dans le graphe sont **mutuellement exclusives** (mutex) ssi :
  - elles ne sont pas indépendantes ou,
  - elles ont des préconditions mutex au niveau précédent (elles ne peuvent donc pas être déclenchées en même temps) :

    $\exists$ (p,q) $\in$ Prec(a1) x Prec(a2), telles que p et q sont mutex.

- Dans LCGP, deux actions d'un même niveau dans le graphe sont **mutuellement exclusives** (mutex) ssi :
  - elles ne sont pas autorisées (aucune n'autorise l'autre) ou,
  - elles ont des préconditions mutex au niveau précédent (elles ne peuvent donc pas être déclenchées en même temps).

2.2. Relation d'autorisation

- Les **mutex** du graphe :
  - GRAPHPLAN : a1 et a2 sont mutex au niveau i ssi a1 et a2 ne sont pas indépendantes :

    $\neg(a1 \angle a2)$ **ou** $\neg(a2 \angle a1)$

  - LCGP : a1 et a2 sont mutex au niveau i ssi aucune n'autorise l'autre :

    $\neg(a1 \angle a2)$ **et** $\neg(a2 \angle a1)$

*A : **a** →+b*

B : a → +c -a

*C : b c →+**d***

NoOps {a,b,c,d}

a

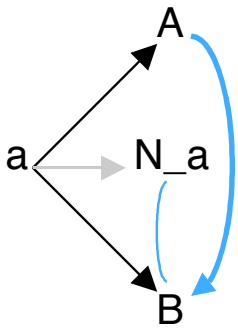
Niveau 0

*A : **a** →+b*

B : a → +c -a

*C : b c →+**d***

NoOps {a,b,c,d}



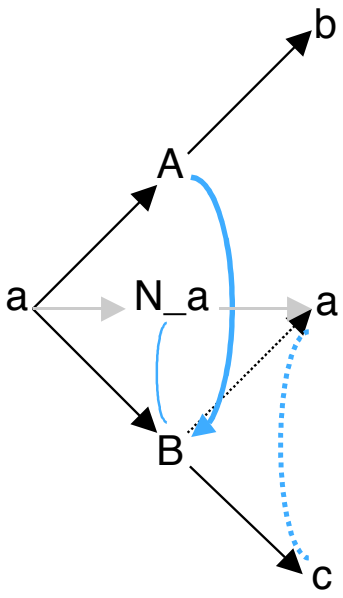
Niveau 0 Niveau 1

$A : a \rightarrow +b$

$B : a \rightarrow +c\ -a$

$C : b\ c \rightarrow +d$

$NoOps\ \{a,b,c,d\}$



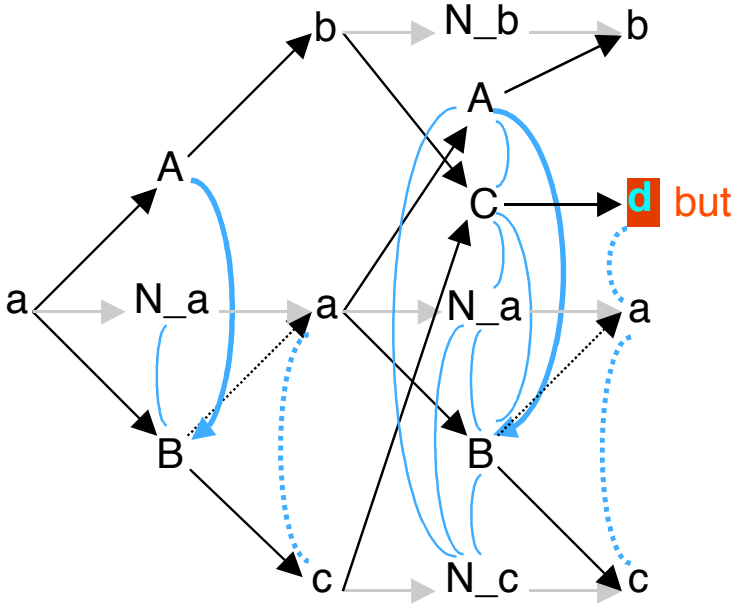
Niveau 0 Niveau 1

$A : a \rightarrow +b$

$B : a \rightarrow +c\ -a$

$C : b\ c \rightarrow +d$

$NoOps\ \{a,b,c,d\}$



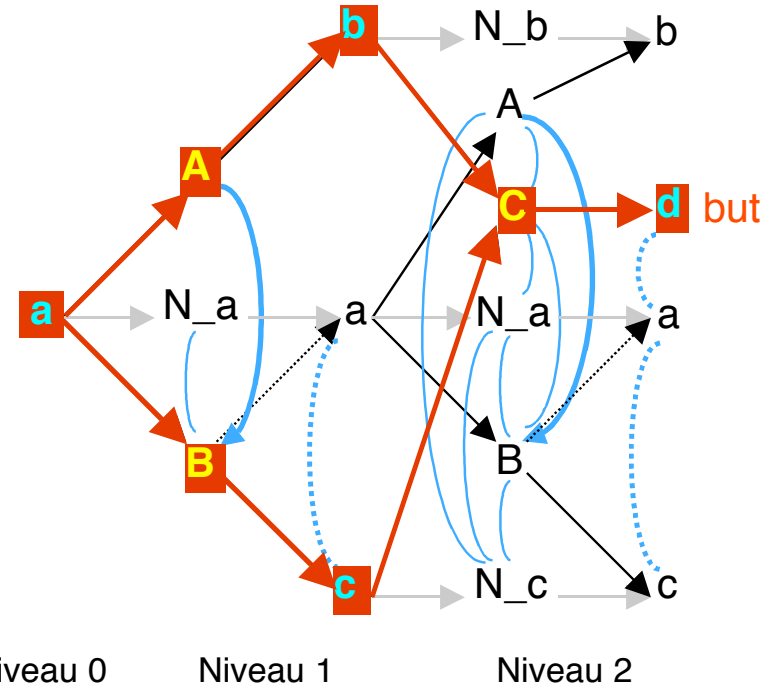
Niveau 0 Niveau 1 Niveau 2

$A : a \rightarrow +b$

$B : a \rightarrow +c\ -a$

$C : b\ c \rightarrow +d$

$NoOps\ \{a,b,c,d\}$



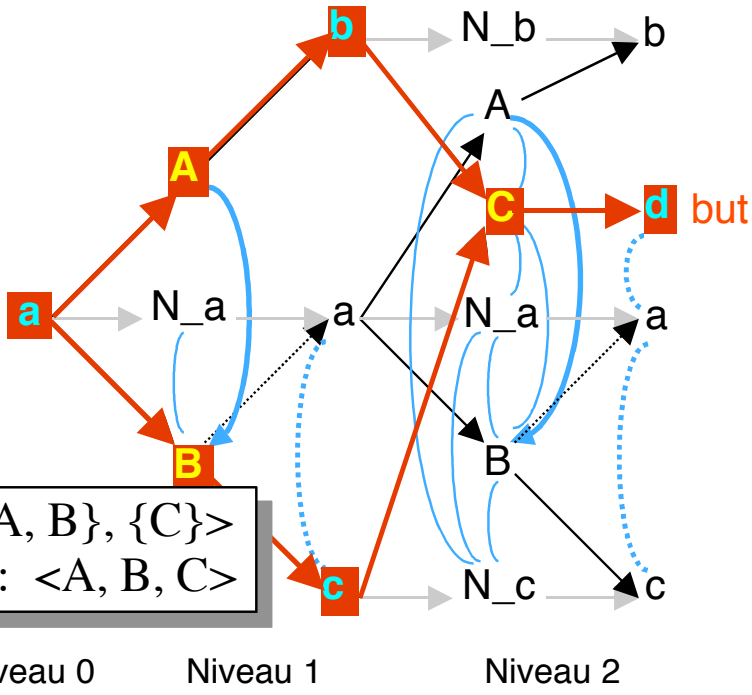
2.2. Relation d'autorisation

$A : a \rightarrow +b$

$B : a \rightarrow +c\ -a$

$C : b\ c \rightarrow +d$

$NoOps\ \{a,b,c,d\}$



Plan solution : <{A, B}, {C}>
post-traité, donne : <A, B, C>

© Régnier, IRIT 2006

2.2. Relation d'autorisation

- **Extraction de la solution :**
  - GRAPHPLAN : un ensemble d'actions est indépendant ssi : (1) il ne contient aucune paire d'actions mutex *(non indépendantes)*.
  - LCGP : un ensemble d'actions est autorisé ssi : (1) il ne contient pas de paire d'actions mutex *(dont aucune n'autorise l'autre)* et (2) s'il en existe une linéarisation autorisée (polynomial).

- **Retour de la solution :**
  - GRAPHPLAN : séquence des ensembles d'actions indépendants déterminés à chaque niveau.
  - LCGP : la séquence des ensembles d'actions autorisés déterminés à chaque niveau, puis son réordonnancement optimal calculé par l'algorithme PRF (polynomial) qui donne un plan indépendant.

2.2. Relation d'autorisation

- **Performances :**
  - LCGP est en moyenne 1800 fois plus rapide que GRAPHPLAN sur les benchmarks de domaines peu contraints (logistics) ;
  - LCGP est plus efficace que GRAPHPLAN sur des domaines Ferry, Gripper, (1 à 10 fois plus rapide quand la difficulté augmente) et BlockWord-no-arm à 3 opérateurs (1 à 3000 fois plus efficace suivant les problèmes) ;
  - Dans les domaines les moins favorables (BlockWord-arm, Mystery, Mystery-prime), LCGP est équivalent à GRAPHPLAN (ratio 0,95) ;
  - Plans de 100 actions, 50 niveaux au maximum en 15 mn environ.

- **Atouts / problèmes :**
  - Qualité des plans-solutions : optimaux mais seulement en nombre de niveaux autorisés ;
  - Perte de l'optimalité en nombre de niveaux indépendants mais non corrélée au nombre d'actions du plan.

© Régnier, IRIT 2006

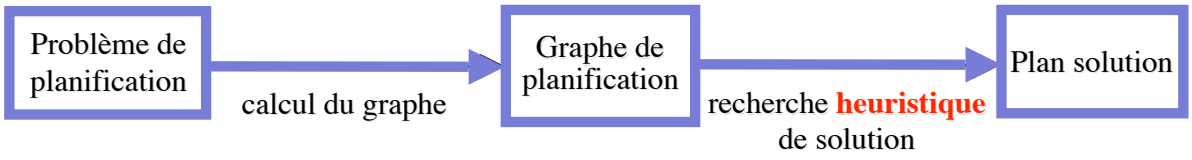
2.3. Fonctions heuristiques indépendantes du domaine

- **Deuxième amélioration :**
  - Optimisation de l'extraction d'une solution :
    - **Fonctions heuristiques indépendantes du domaine**

UNIVERSITE PAUL SABATIER TOULOUSE III

2.3. Fonctions heuristiques indépendantes du domaine

GRAPHPLAN [Blum, Furst, 1995]
LCGP [Cayrol, Régnier, Vidal, 2000]
GPCSP [Kambhampati, Nigenda, 2000]

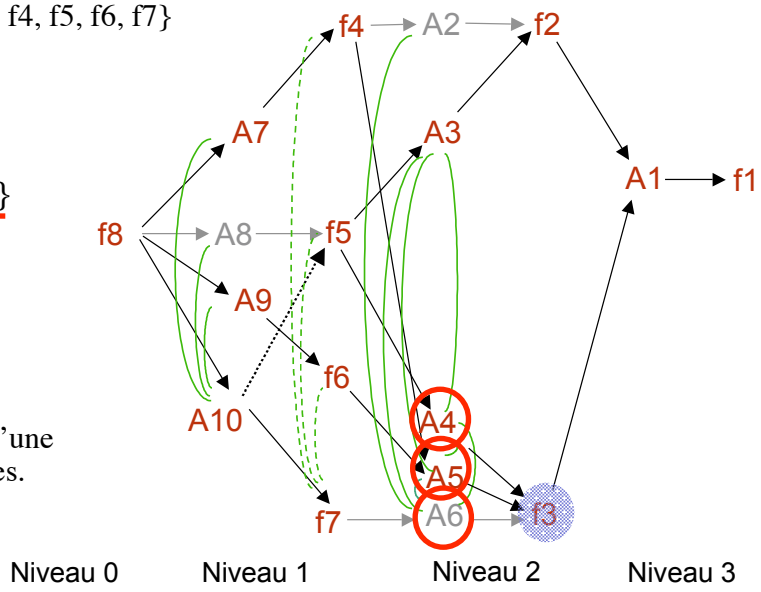


Compilation de plans : GRAPHPLAN, LCGP, GPCSP

© Régnier, IRIT 2006

2.3. Fonctions heuristiques indépendantes du domaine

- Encodage du graphe de planification en CSP
  - Variables = {f1, f2, f3, f4, f5, f6, f7}
  - Domaines :
    D(f1)={A1}
    D(f2)={A2,A3}
    D(f3)={A4, A5, A6}
    D(f4)={A7}
    D(f5)={A8}
    D(f6)={A9}
    D(f7)={A10}
- Heuristiques pour le choix d'une valeur à affecter aux variables.



2.3. Fonctions heuristiques indépendantes du domaine

- **Noops d'abord « noops-first » :** heuristique originale destinée à minimiser le nombre d'actions du plan-solution mais peu efficace ;
- **Taille du domaine des variables :** ×4, peu efficace ;
- **Heuristique basée sur les niveaux « level-based » :** choisit prioritairement les fluents de plus haut niveau d'apparition et, pour les établir, les actions de plus faible niveau d'apparition, ×100, baisse un peu la qualité des plans-solutions ;
- **Heuristique « noops-first, level-based, number mutex » :** choisit prioritairement les fluents de plus haut niveau d'apparition (si égalité, ceux qui ont le plus de mutex) et, pour les établir, utilise les noops d'abord, puis les actions de plus faible niveau d'apparition (si égalité, celles qui ont le moins de mutex) ×100, maintient la qualité des plans-solutions.

2.3. Fonctions heuristiques indépendantes du domaine

- **Résultats des tests :**

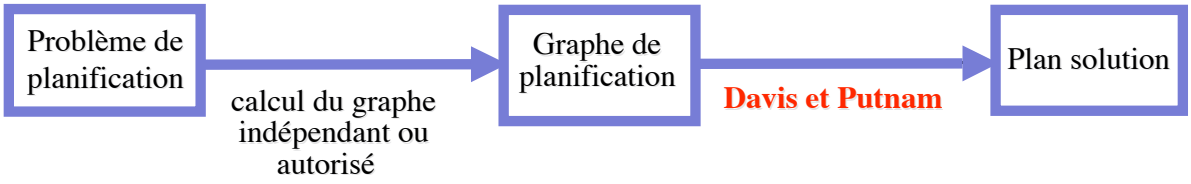
Problems	CPU time (sec.)			Ratio No–opsF/		Actions			Levels	
	No–opsF	Level	LevelN	Level	LevelN	No–opsF	Level	LevelN	(+)	(++)
ferry6	3.05	0.30	0.36	10.17	8.47	23	23	23	23	12
ferry8	387.51	2.51	3.06	154.39	126.64	31	31	31	31	16
gripper6	1.45	0.39	0.49	3.74	2.96	17	17	17	11	6
gripper8	165.81	8.02	7.61	20.68	21.79	23	23	23	15	8
bw-large-a	3.42	2.49	2.57	1.37	1.33	12	12	12	12	12
bw-large-b	257.65	19.13	36.05	13.47	7.15	18	18	18	18	18
log020	578.01	8.38	9.20	68.97	62.83	87	93	87	15	9
log023	90.50	3.77	4.30	24.01	21.05	61	65	61	13	8
hanoi5	8.41	10.48	27.30	0.80	0.31	32	32	32	32	21

2.4. Procédure de Davis et Putnam sur le graphe

- **Troisième amélioration :**
  - Optimisation de l'extraction d'une solution :
    - **Procédure de Davis et Putnam sur le graphe**

2.4. Procédure de Davis et Putnam sur le graphe

DPPLAN [Baioletti, Marcugini, Milani, 2000]
LCDPP [Vidal, 2001]



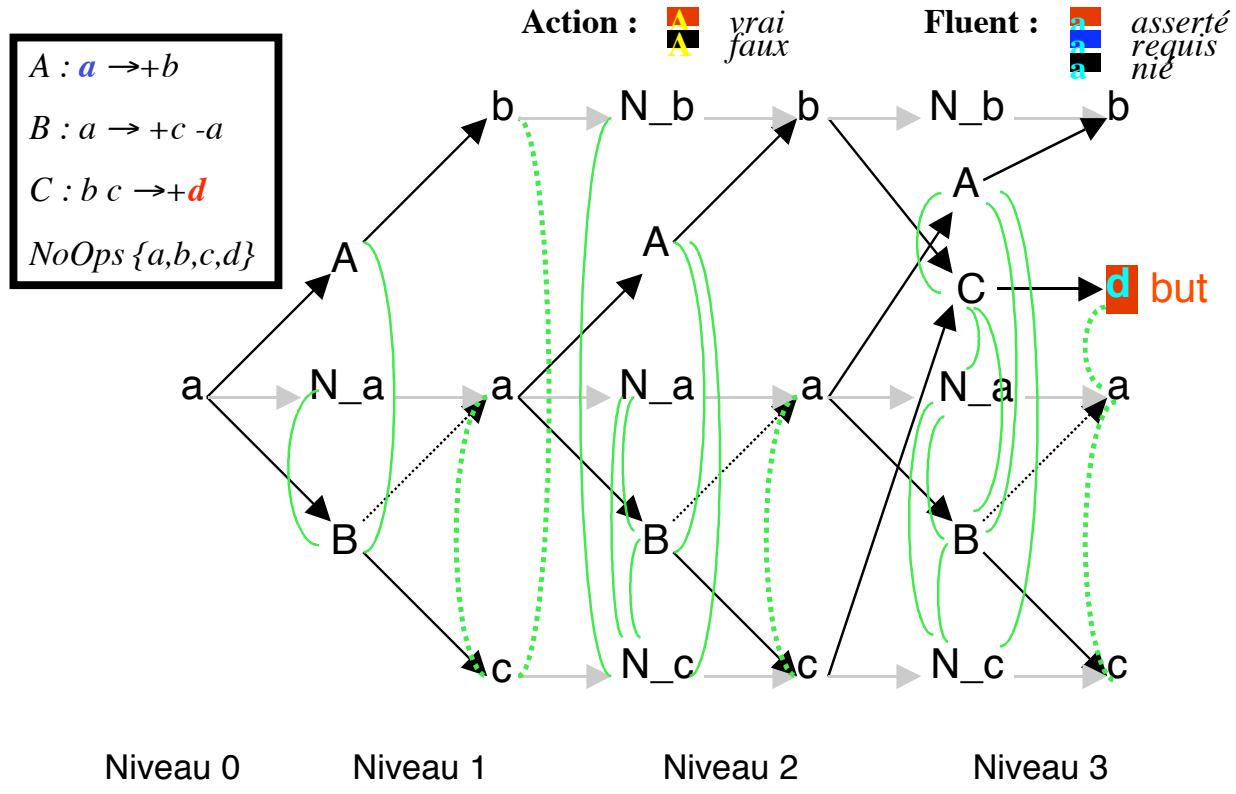
Compilation de plans : DPPLAN

2.4. Procédure de Davis et Putnam sur le graphe

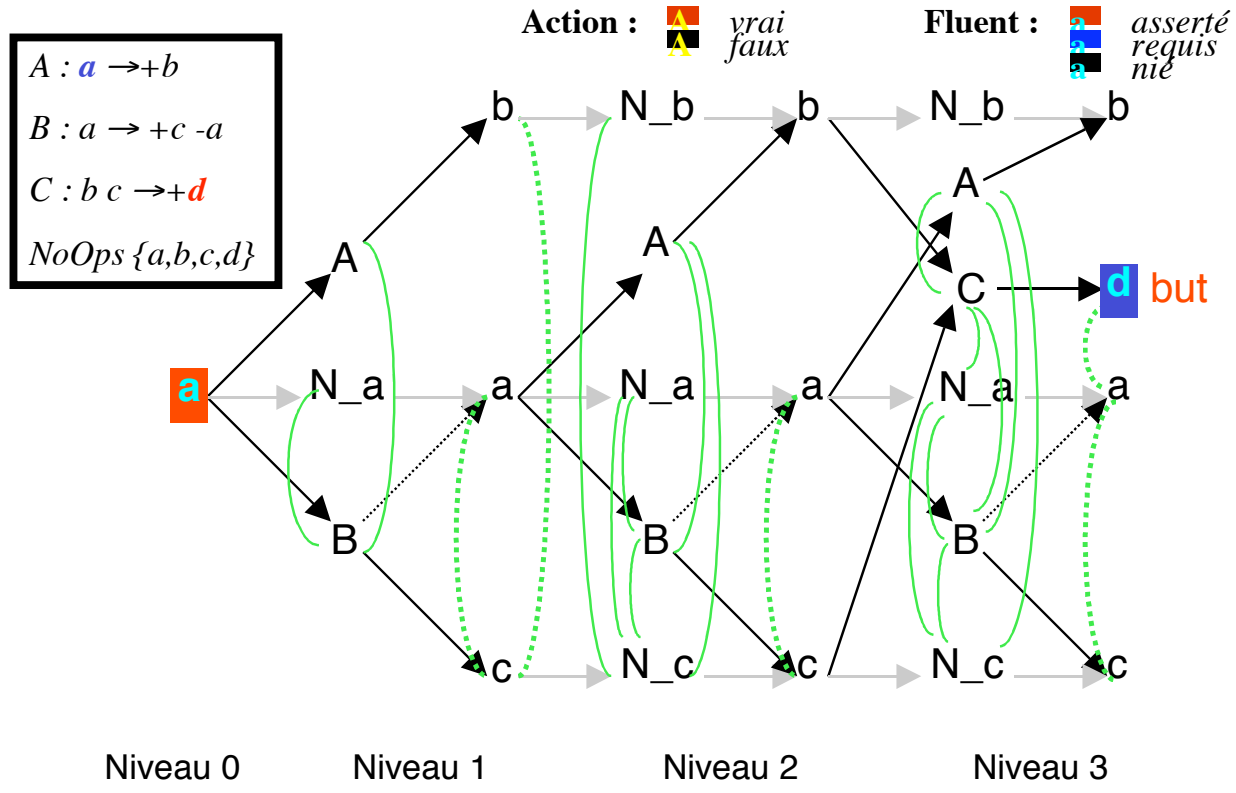
- DPPLAN [Baioletti, Marcugini, Milani, 2000] :
  - Association d'une valeur à chaque noeud du graphe :
    - Fluent :
      - indéfini ;
      - asserté ;
      - requis (valeur positive) ;
      - nié (valeur négative) ;
    - Action :
      - indéfini ;
      - vrai (valeur positive) ;
      - faux (valeur négative) ;
  - Propagation de valeurs sur les nœuds du graphe ;
  - Maintenance d'une liste de buts (fluents ayant la valeur requis), un plan-solution est trouvé lorsque la liste est vide ;
  - Recherche pouvant être guidée par les heuristiques « noops-first, level-based, max mutex » grâce à l'ordonnancement de la liste des buts.

© Régnier, IRIT 2006

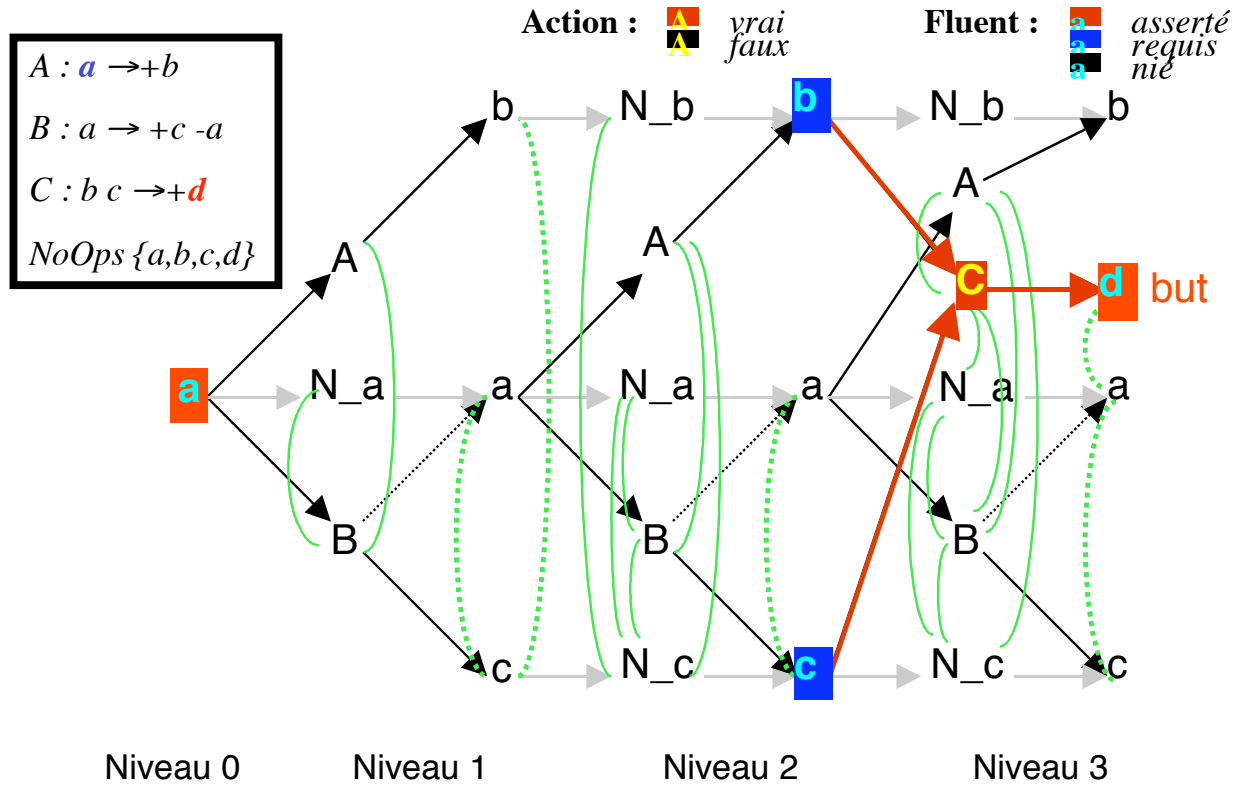
2.4. Procédure de Davis et Putnam sur le graphe



2.4. Procédure de Davis et Putnam sur le graphe

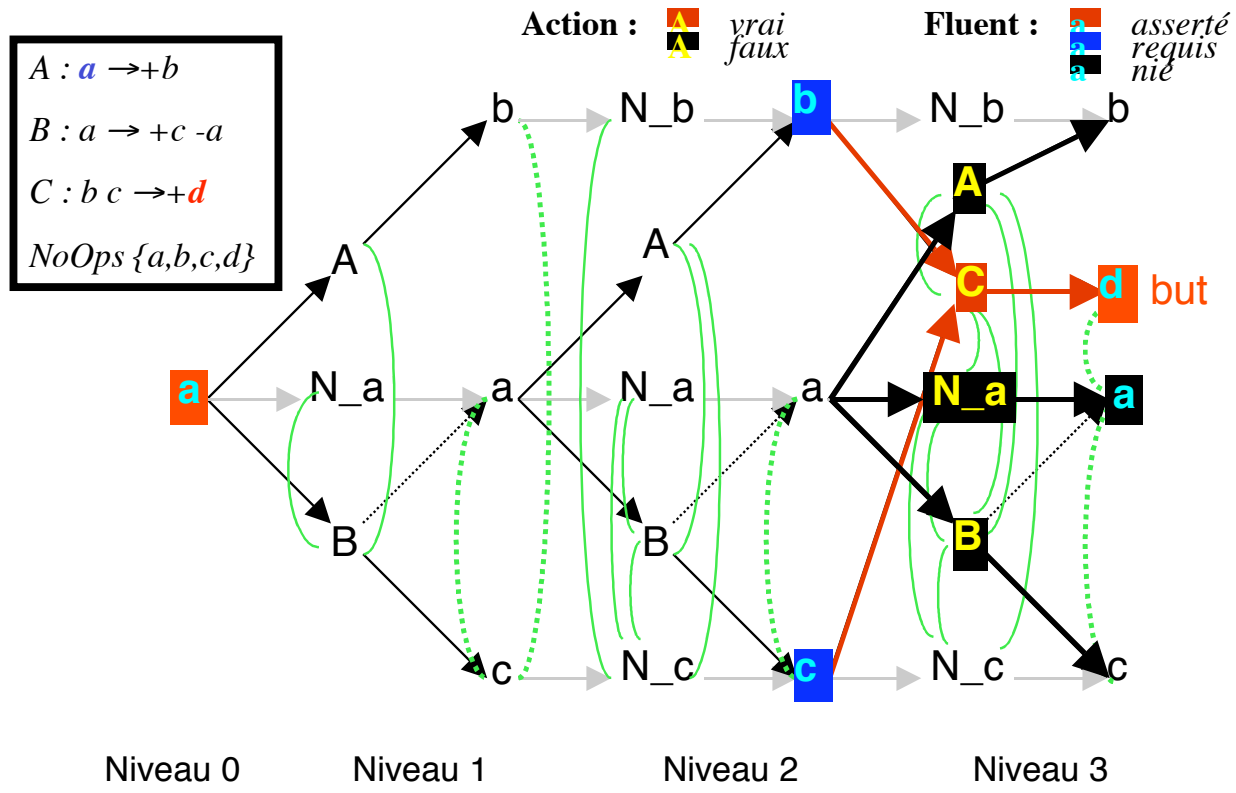


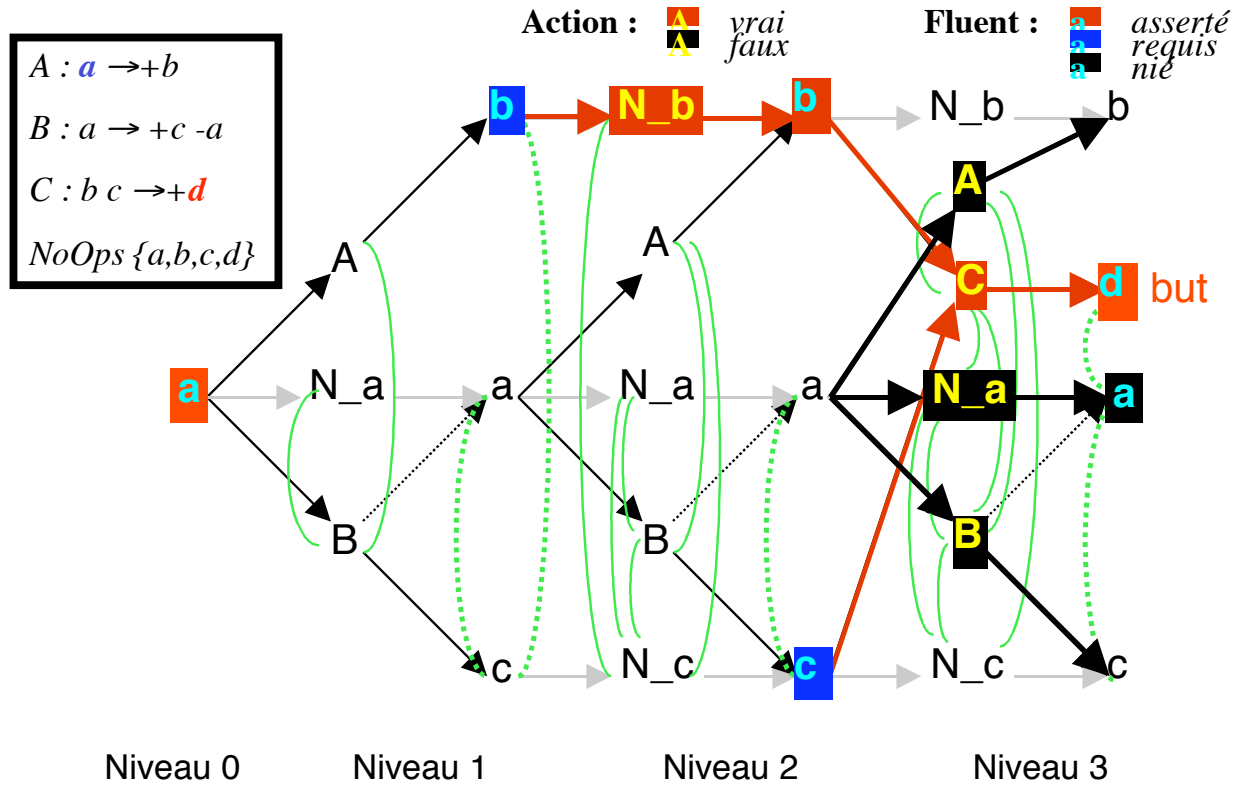
2.4. Procédure de Davis et Putnam sur le graphe



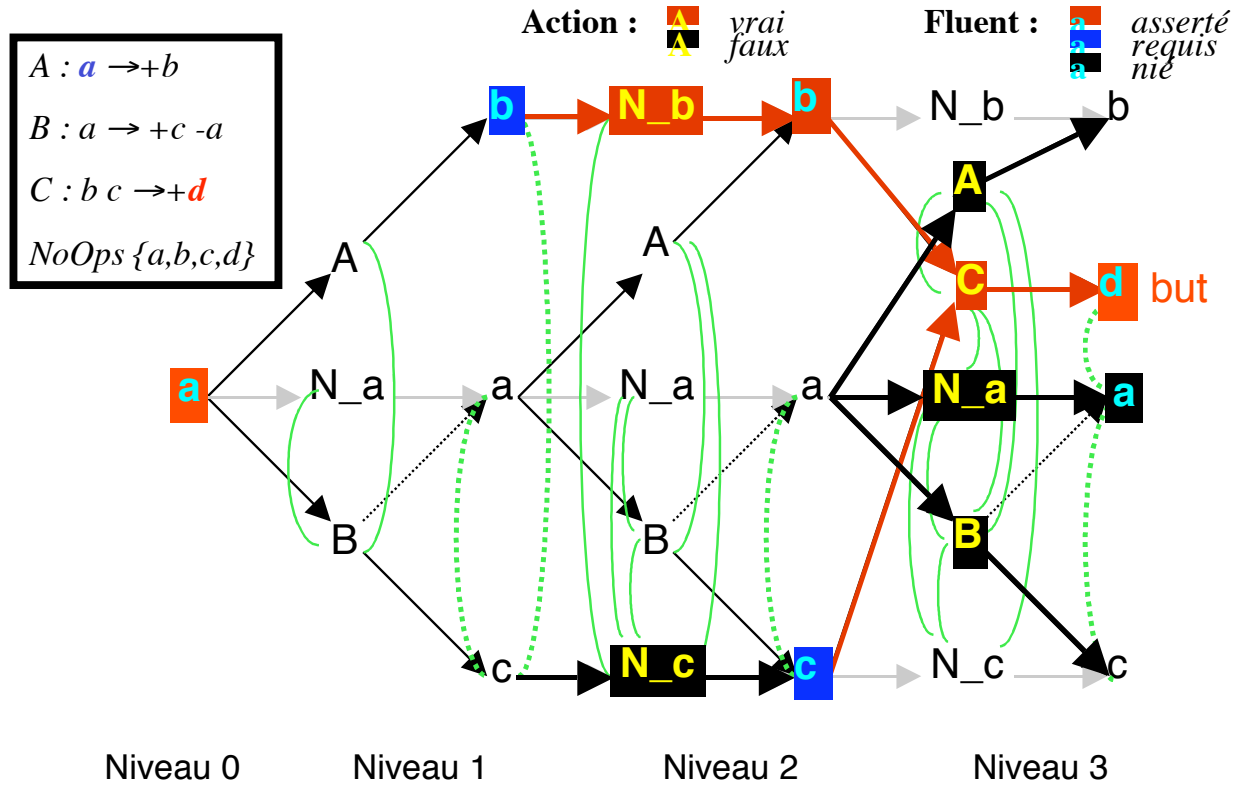
© Régnier, IRIT 2006

2.4. Procédure de Davis et Putnam sur le graphe



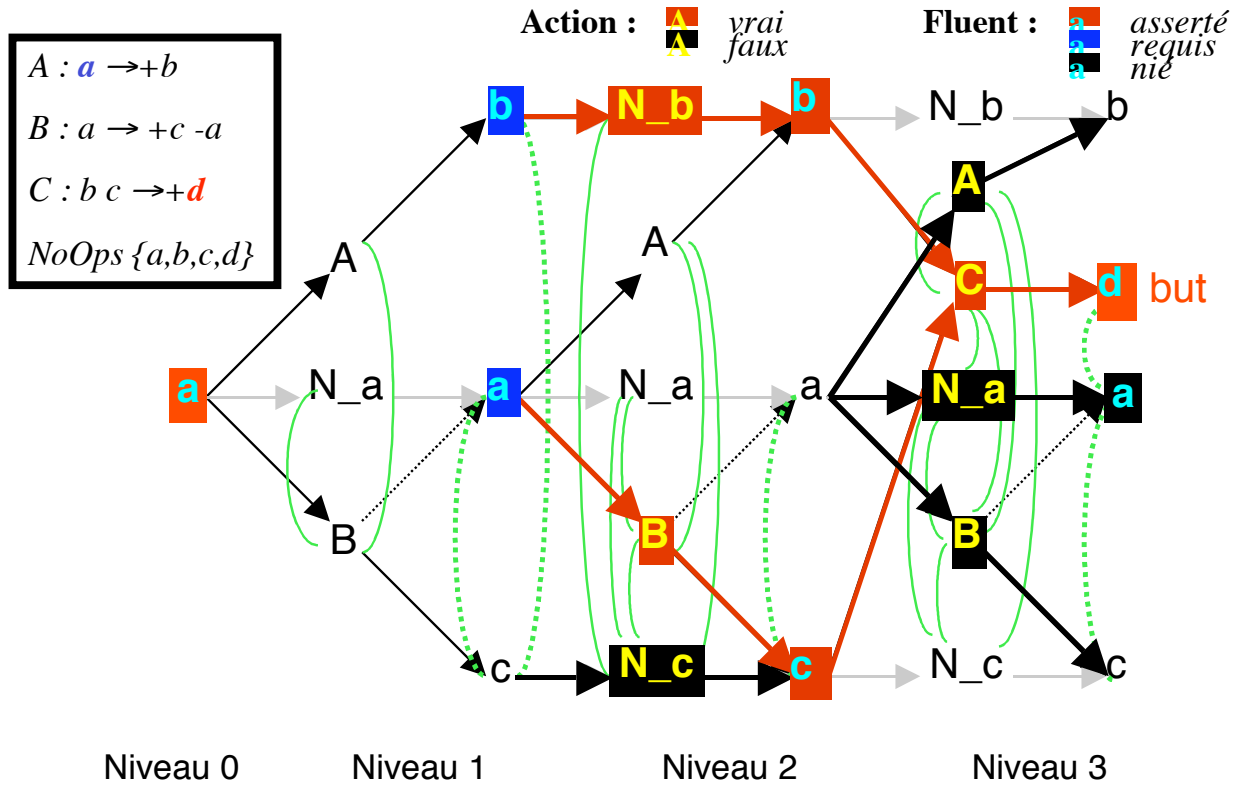


2.4. Procédure de Davis et Putnam sur le graphe

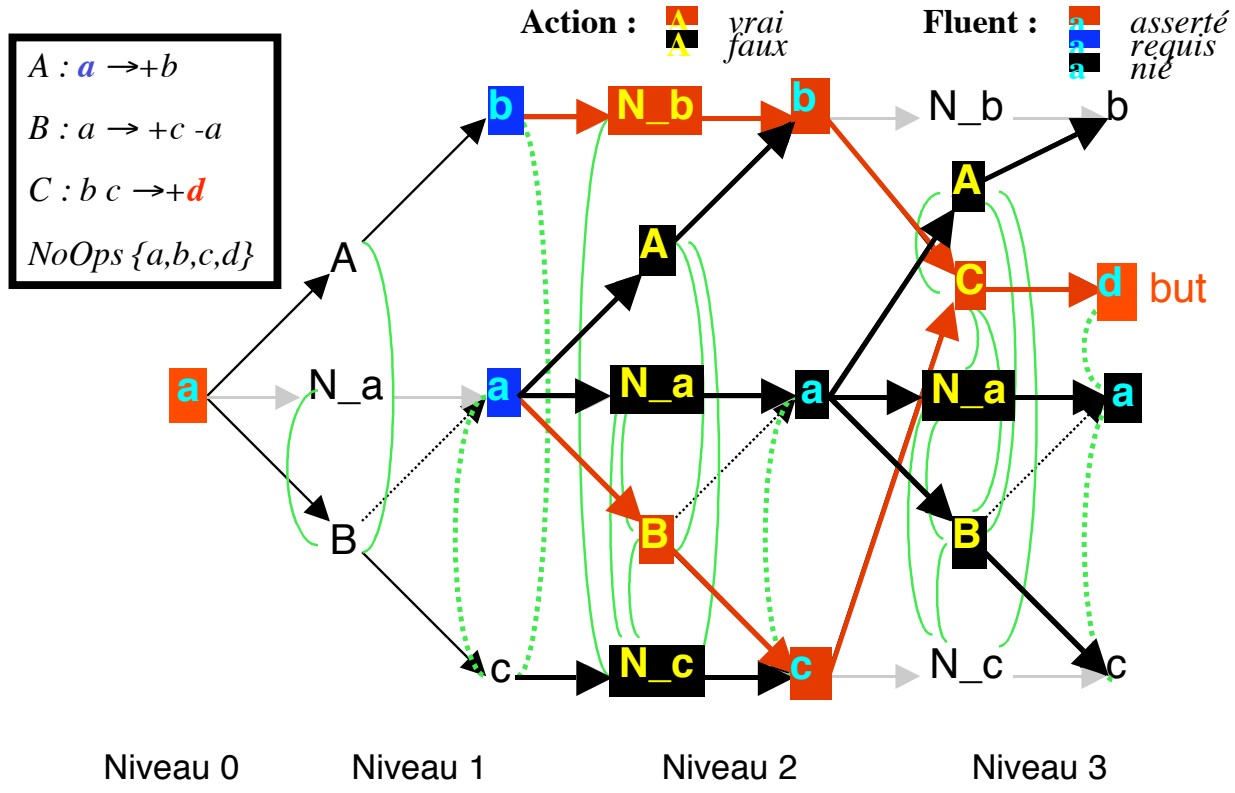


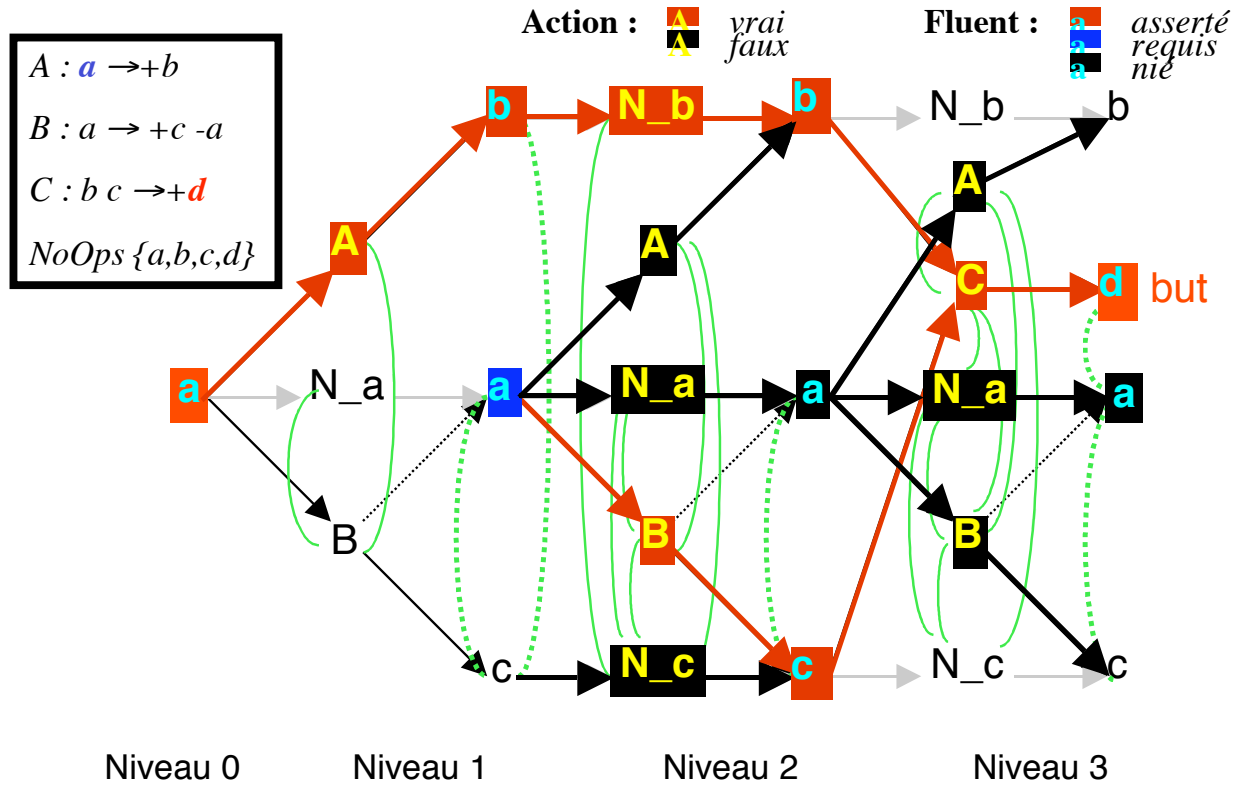
© Régnier, IRIT 2006

2.4. Procédure de Davis et Putnam sur le graphe

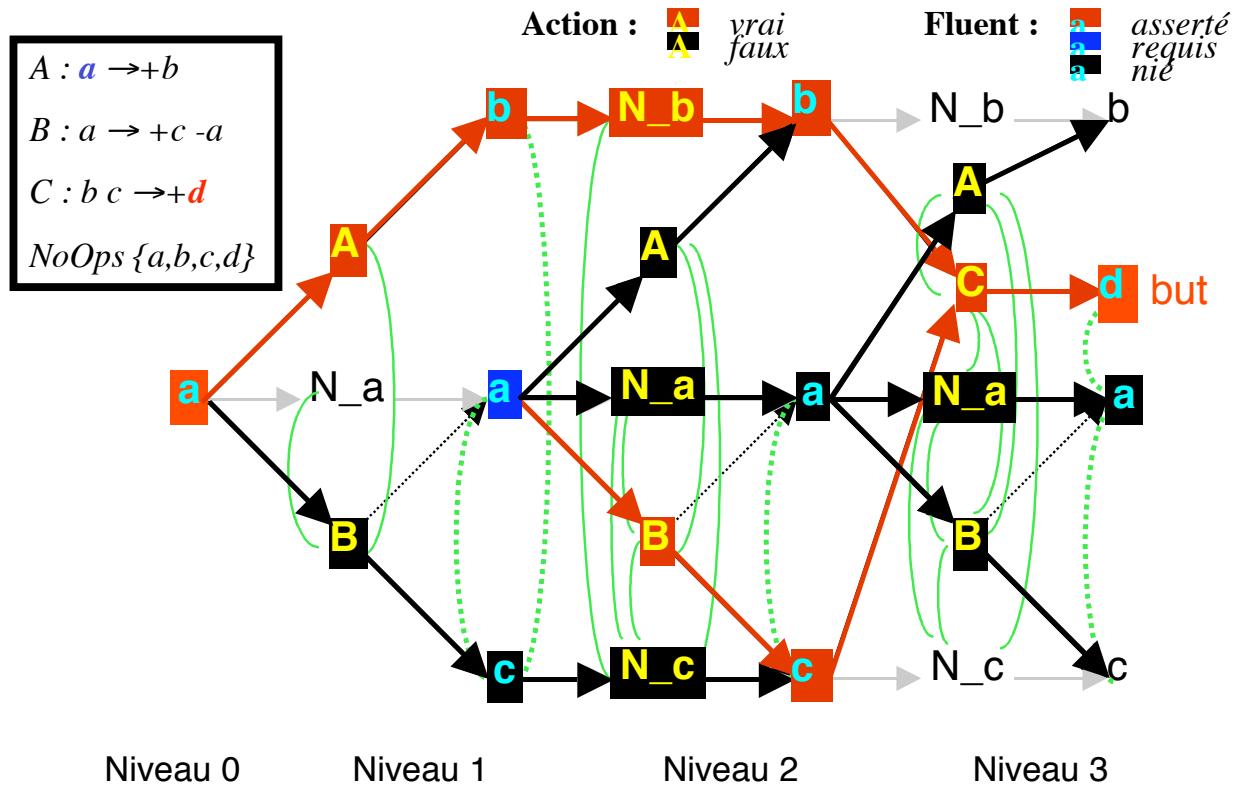


2.4. Procédure de Davis et Putnam sur le graphe

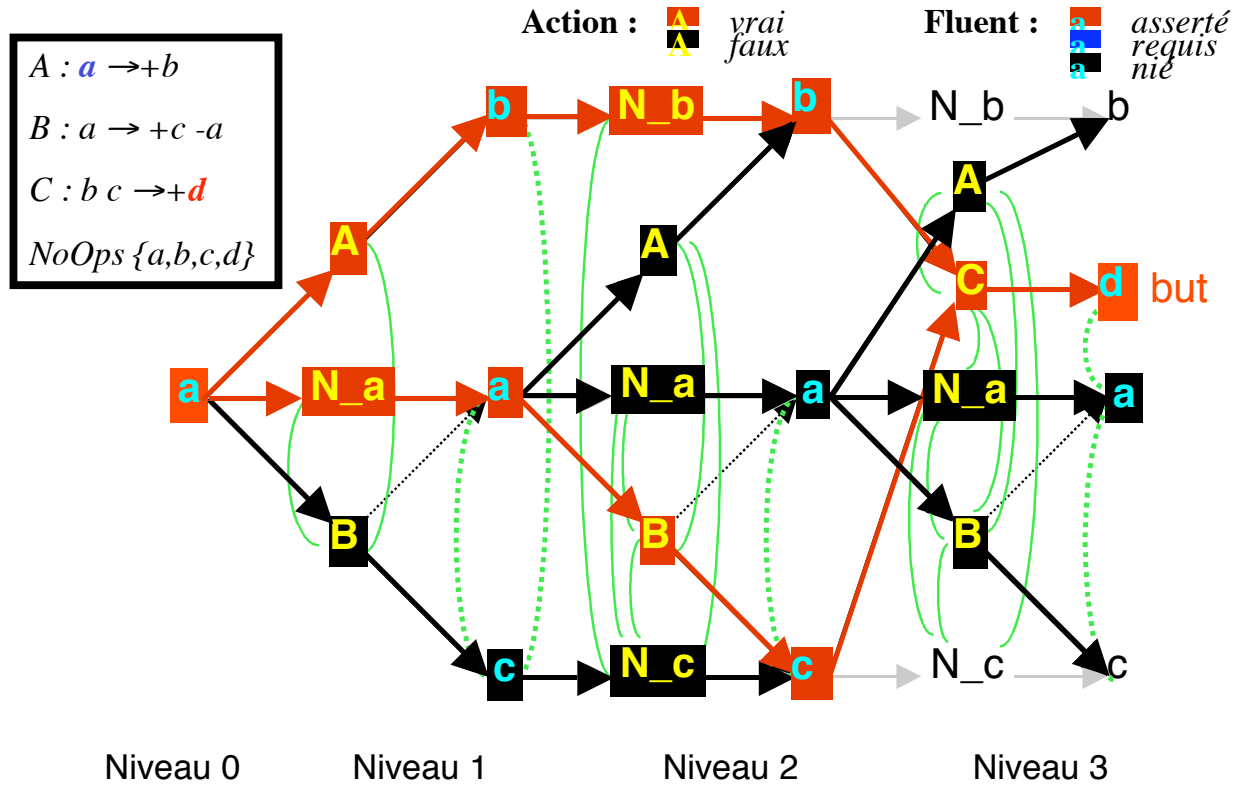


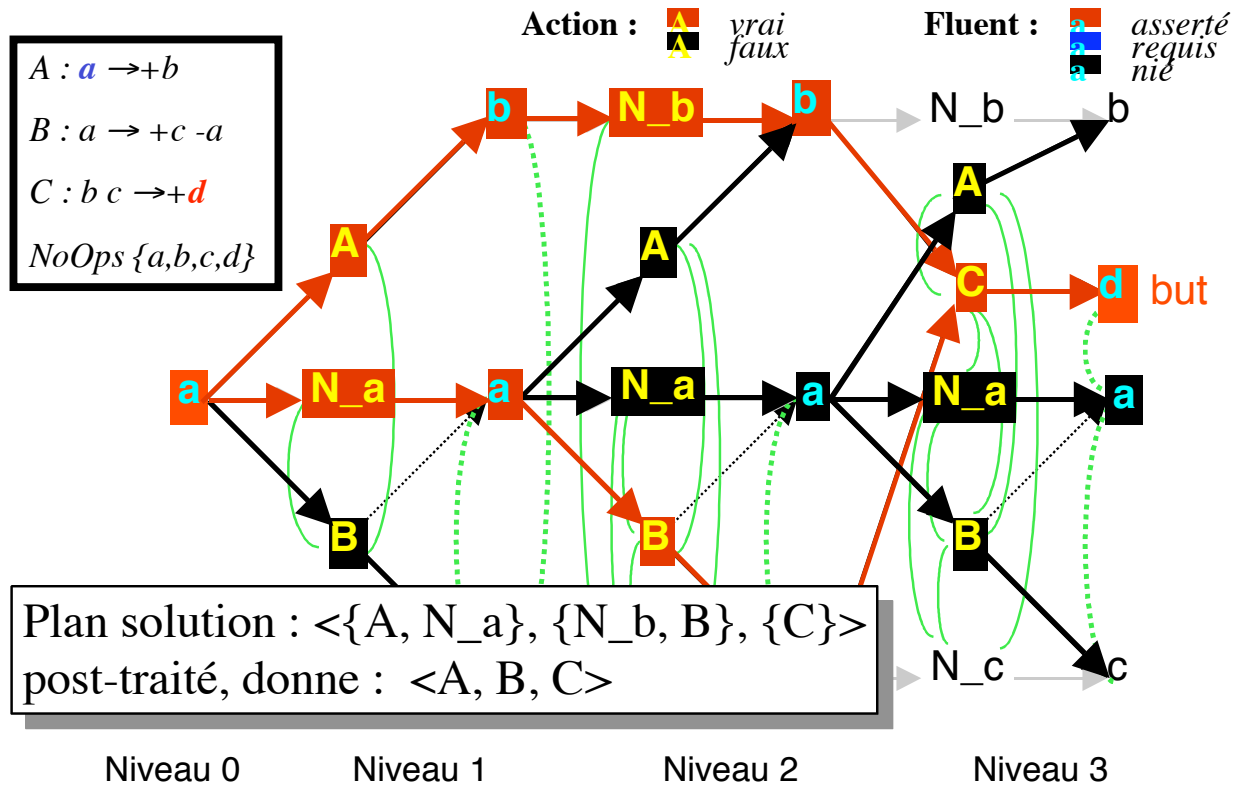


2.4. Procédure de Davis et Putnam sur le graphe



© Régnier, IRIT 2006





2.4. Procédure de Davis et Putnam sur le graphe

- LCDPP :
  - Correction et formalisation de DPPLAN sur la base du codage SAT du graphe de planification ;
  - Introduction de la relation d'autorisation : la contrainte d'autorisation doit être vérifiée à chaque utilisation d'une action ;
  - Modification de certaines règles de propagation lorsqu'un fluent est requis ;
  - Sans mémoization ni backtrack intelligent LCDPP est compétitif avec LCGP-BJ ;
  - Recherche pouvant être guidée par les heuristiques « noops-first, level-based, max mutex » grâce à l'ordonnancement de la liste des buts.

2.4. Procédure de Davis et Putnam sur le graphe

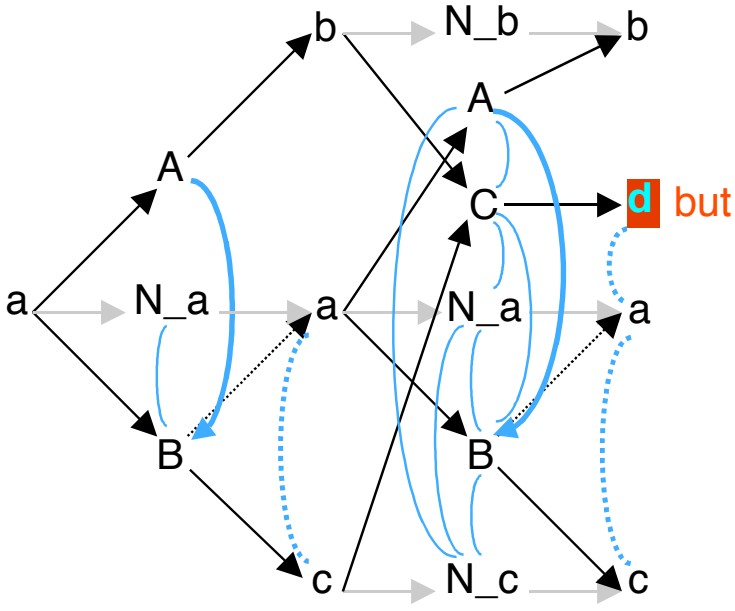
A : a →+b

B : a → +c -a

C : b c →+d

NoOps {a,b,c,d}

Action : *vrai* *faux* **Fluent :** *asserté* *requis* *nié*



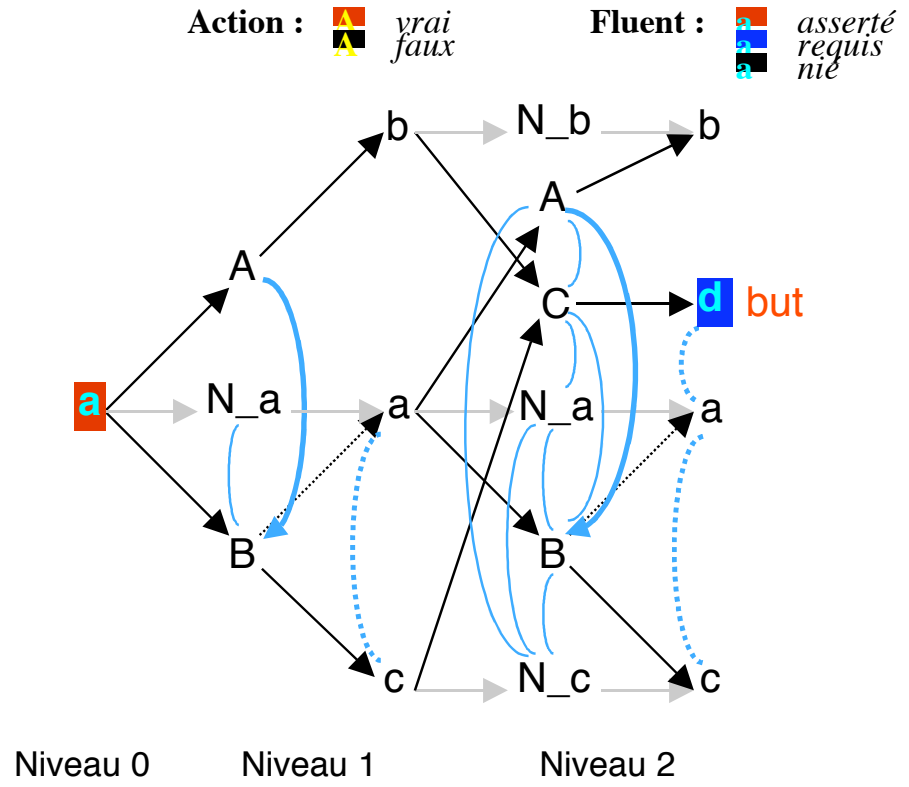
2.4. Procédure de Davis et Putnam sur le graphe

$A : a \rightarrow +b$

$B : a \rightarrow +c\ -a$

$C : b\ c \rightarrow +d$

$NoOps\ \{a,b,c,d\}$



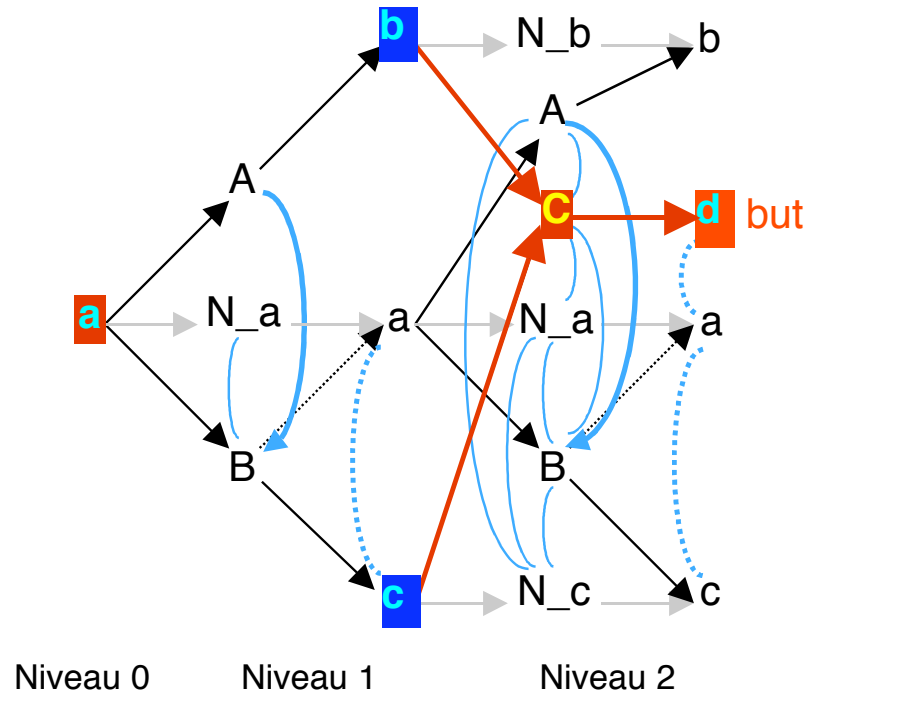
2.4. Procédure de Davis et Putnam sur le graphe

$A : a \rightarrow +b$

$B : a \rightarrow +c\ -a$

$C : b\ c \rightarrow +d$

$NoOps\ \{a,b,c,d\}$



© Régnier, IRIT 2006

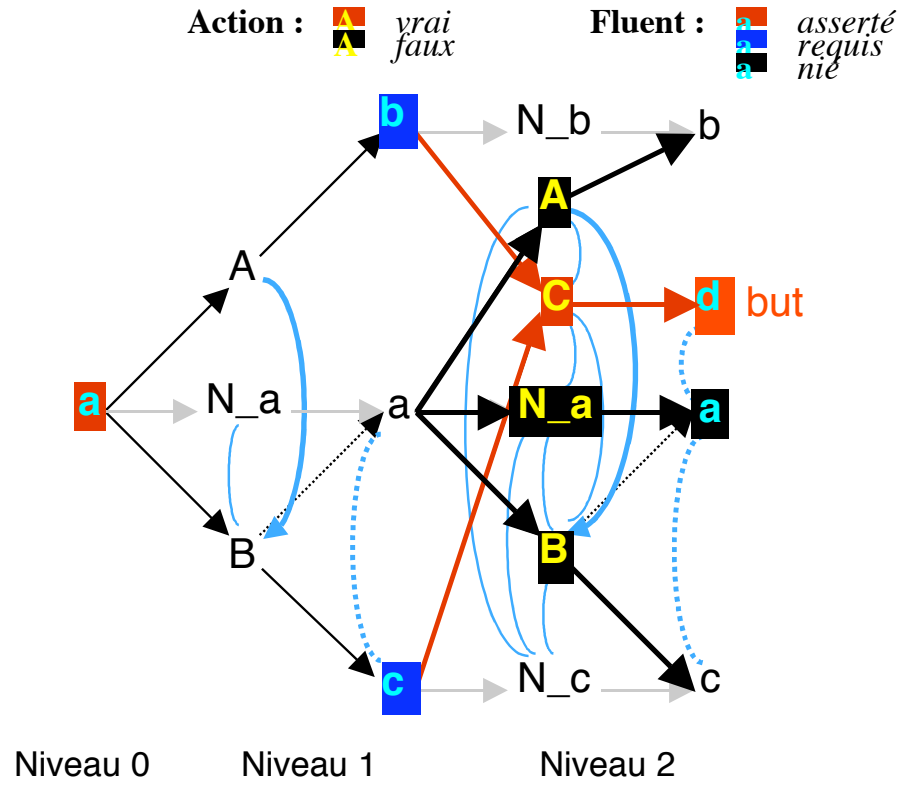
2.4. Procédure de Davis et Putnam sur le graphe

$A : a \rightarrow +b$

$B : a \rightarrow +c\ -a$

$C : b\ c \rightarrow +d$

$NoOps\ \{a,b,c,d\}$



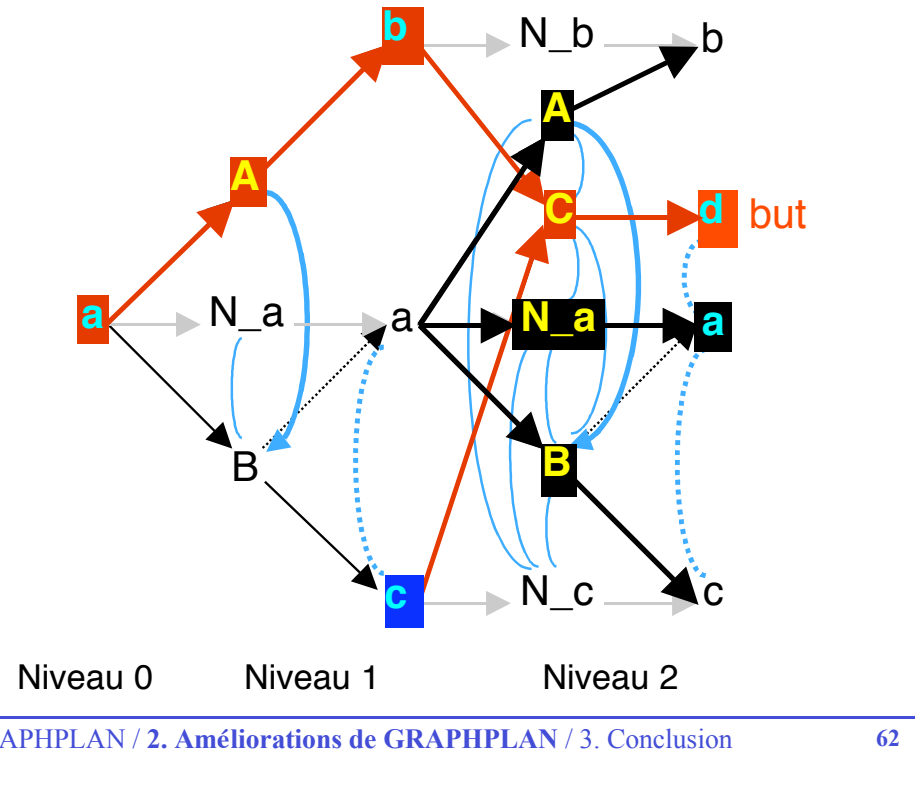
2.4. Procédure de Davis et Putnam sur le graphe

$A : a \rightarrow +b$

$B : a \rightarrow +c\ -a$

$C : b\ c \rightarrow +d$

$NoOps\ \{a,b,c,d\}$



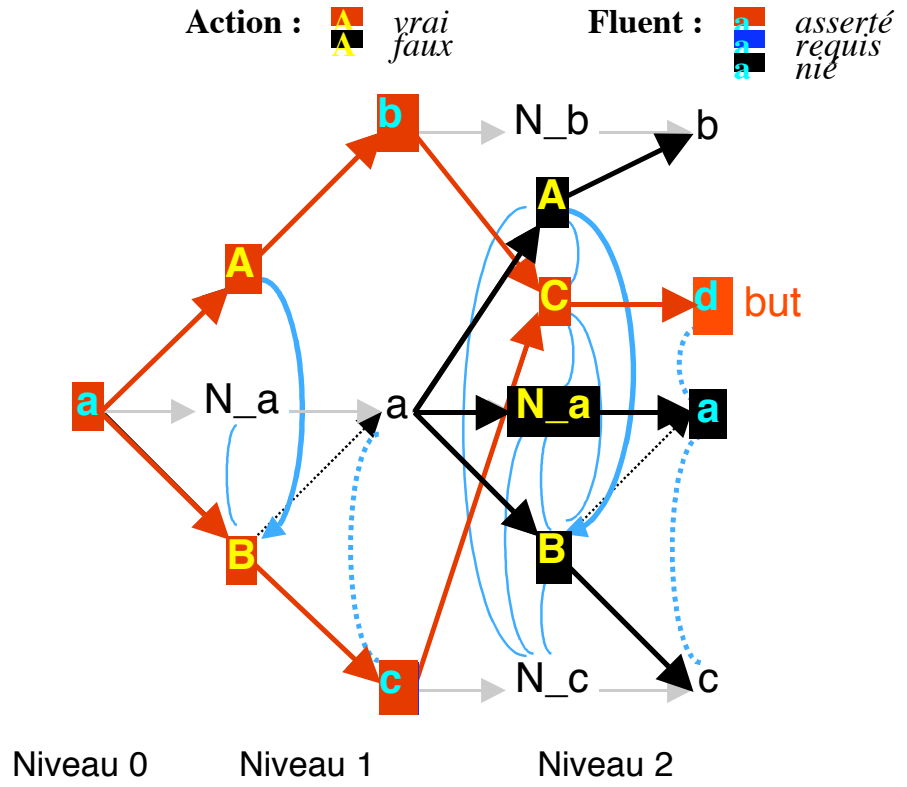
© Régnier, IRIT 2006

$A : a \rightarrow +b$

$B : a \rightarrow +c\ -a$

$C : b\ c \rightarrow +d$

$NoOps\ \{a,b,c,d\}$



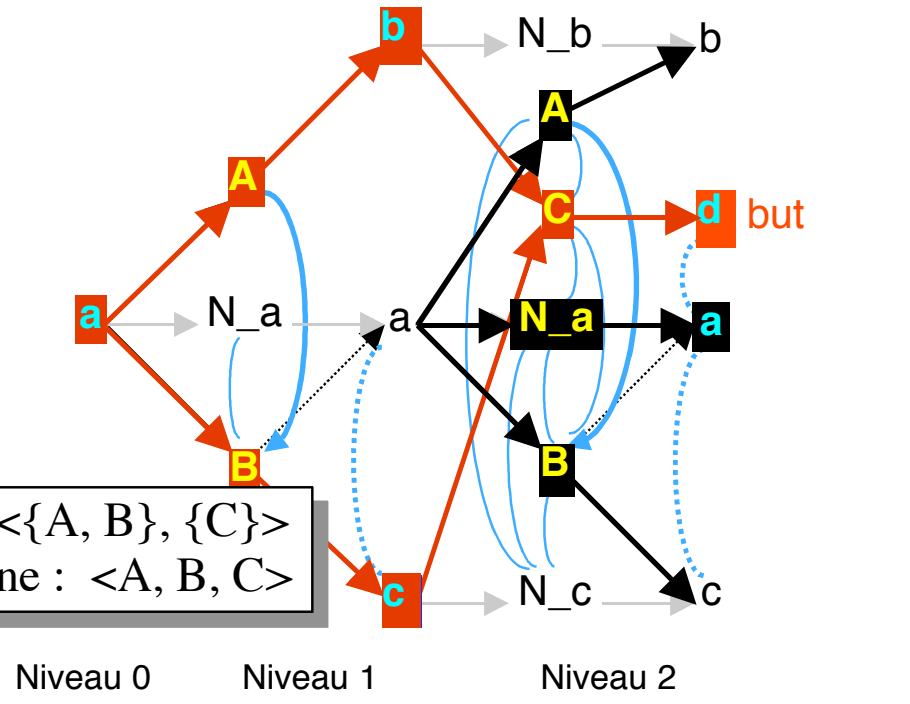
2.4. Procédure de Davis et Putnam sur le graphe

$A : a \rightarrow +b$

$B : a \rightarrow +c\ -a$

$C : b\ c \rightarrow +d$

$NoOps\ \{a,b,c,d\}$



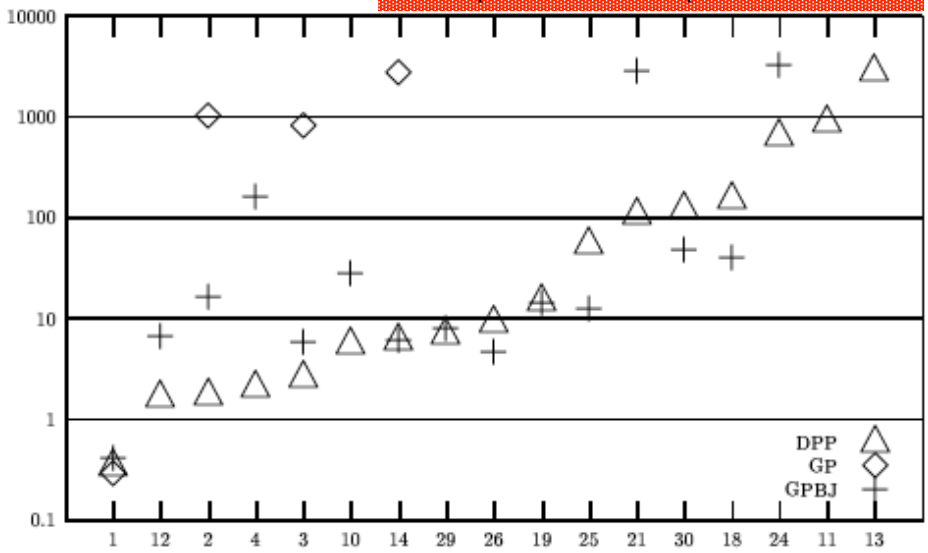
Plan solution : <{A, B}, {C}>
post-traité, donne : <A, B, C>

© Régnier, IRIT 2006

2.4. Procédure de Davis et Putnam sur le graphe

- **Résultats domaine Logistics :**

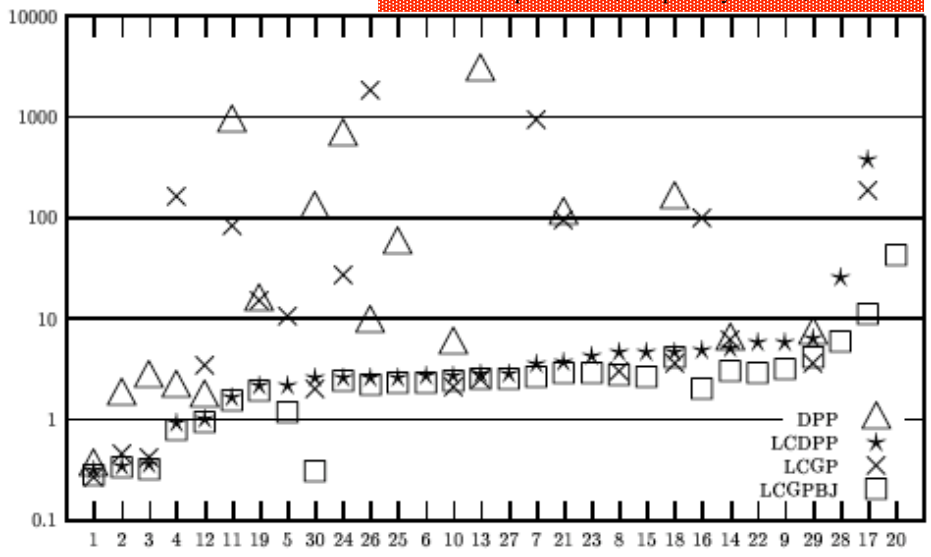
GP : 4 problèmes résolus
GPBJ : 15 problèmes résolus
DPP : le plus efficace avec 17 problèmes résolus



2.4. Procédure de Davis et Putnam sur le graphe

- **Résultats domaine Logistics :**

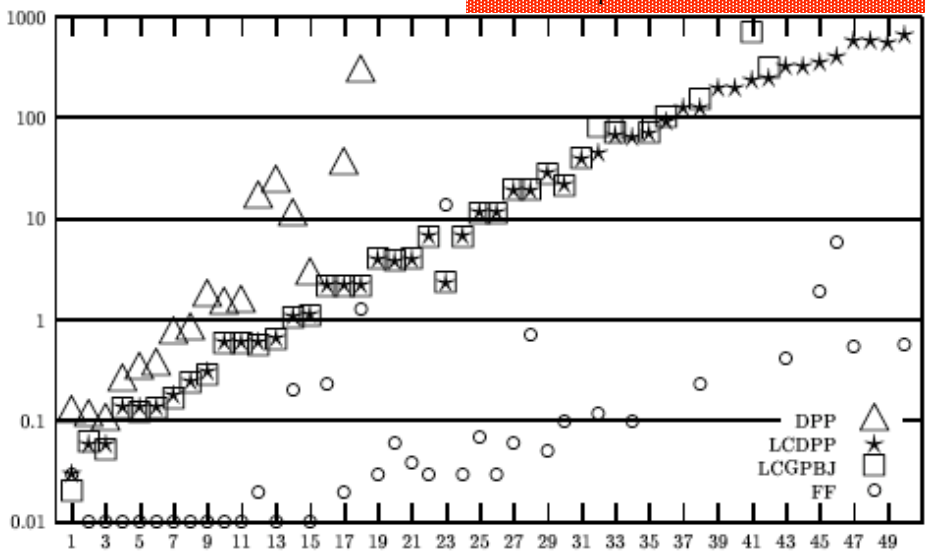
DPP : 17 problèmes, temps moyen de 298 sec.
LCDPP : 29 problèmes, temps moyen de 2,51 sec.



2.4. Procédure de Davis et Putnam sur le graphe

- **Résultats domaine BlocksW-no-arm :**

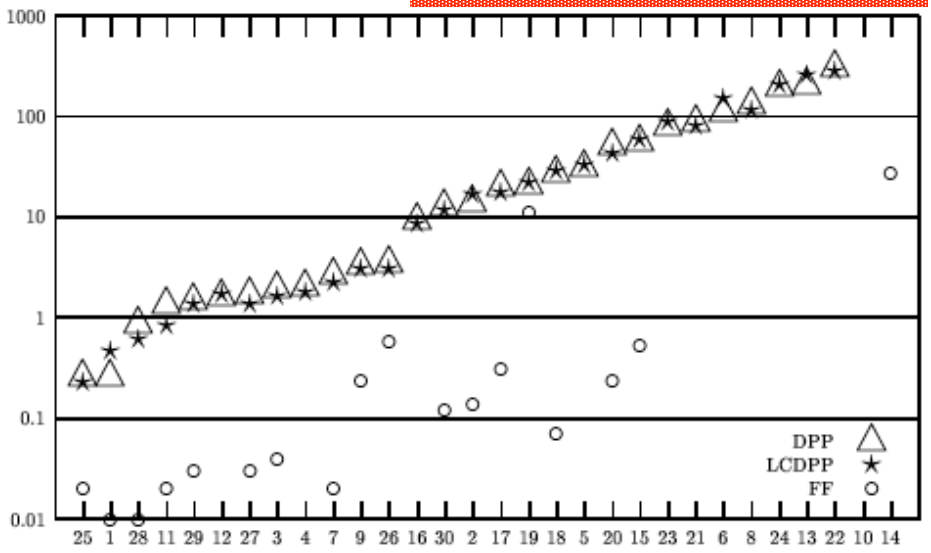
LCDPP : 50 / 50 problèmes résolus
LCGPBJ : 39 / 50 problèmes résolus
FF : 38 / 50 problèmes résolus



2.4. Procédure de Davis et Putnam sur le graphe

- **Résultats domaines Mystery :**

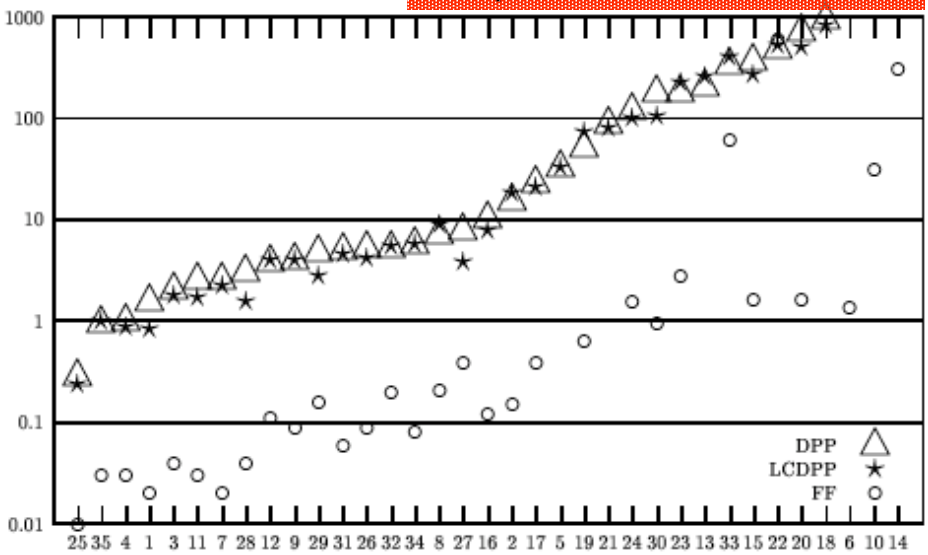
DPP, LCDPP : 29 / 30 problèmes résolus
FF : 18 problèmes résolus



2.4. Procédure de Davis et Putnam sur le graphe

- **Résultats domaines Mystery / MysteryPrime :**

DPP, LCDPP : 32 / 35 problèmes résolus
FF : 31 problèmes résolus

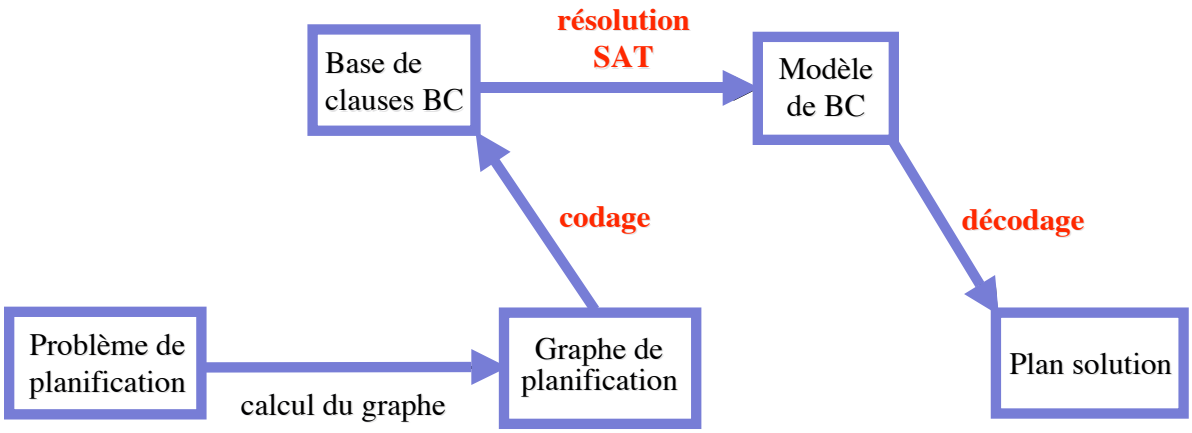


2.5. Codages SAT du graphe

- **Quatrième amélioration :**
  - Optimisation de l'extraction d'une solution :
    - **Codages SAT du graphe**

2.5. Codages SAT du graphe

BLACKBOX [Kautz, Selman, 1999]
CSATPLAN [Baioletti, Marcugini, Milani, 1998]
TSP [Maris, Régnier, Vidal, 2002]
SATPLAN'04 [Kautz, Selman, 2004]



Compilation de plans : BLACKBOX, SATPLAN'04

2.5. Codages SAT du graphe

- **Principe :**
  - **Codage :** transformation du problème de planification en une base de clauses pour une longueur maximale de plan donnée k ;
  - **Résolution :** recherche d'un modèle de la base par un solveur SAT :
    - modèle : décodage qui donne un plan-solution ;
    - pas de modèle :
      - solveur complet : pas de plan de longueur ≤ k ; relance du processus pour k+1 ;
      - solveur incomplet : rien ;
- **Objectif :** bénéficier automatiquement des progrès des techniques SAT ;
- **Différentes approches :**
  - BLACKBOX, SATPLAN'04 : codage du graphe de planification ;
  - SATPLAN : codage direct du problème ;

2.5. Codages SAT du graphe

- 1992 : Kautz, Selman (formulation espaces d'états et Noops)
- 1996 : Kautz, McAllester, Selman (espaces d'états et frame axiomes, graphe de planification, espaces de plans par liens causaux) ;
- 1999 : Kautz, Selman (graphe de planification) ;
- 1999 : Mali, Kambhampati (espaces de plans partiels et white knight, supériorité espaces d'états / espaces de plans) ;
- 2001 : Cayrol, Régnier, Vidal
  - simplifications (clauses redondantes) ;
  - parallélisme pour les codages d'espaces de plans ;
  - graphe de planification codé par les actions ;
- 2002, 2004 : Maris, Régnier, Vidal
  - corrections ;
  - introduction de la relation d'autorisation ;
  - implémentation du système TSP (Tunable SatPlan) et tests comparatifs.

2.5. Codages SAT du graphe

- Codages du graphe de planification :
  - KS99 (indépendant) : fluents, actions, mutex d'actions, relation d'indépendance ;
  - KS99 (autorisé) : fluents, actions, mutex d'actions, relation d'autorisation ;
  - V01 (indépendant) : actions, mutex d'actions, relation d'indépendance ;
  - V01 (autorisé) : actions, mutex d'actions, relation d'autorisation.

© Régnier, IRIT 2006

- **Codage [Kautz, Selman, 1999]**

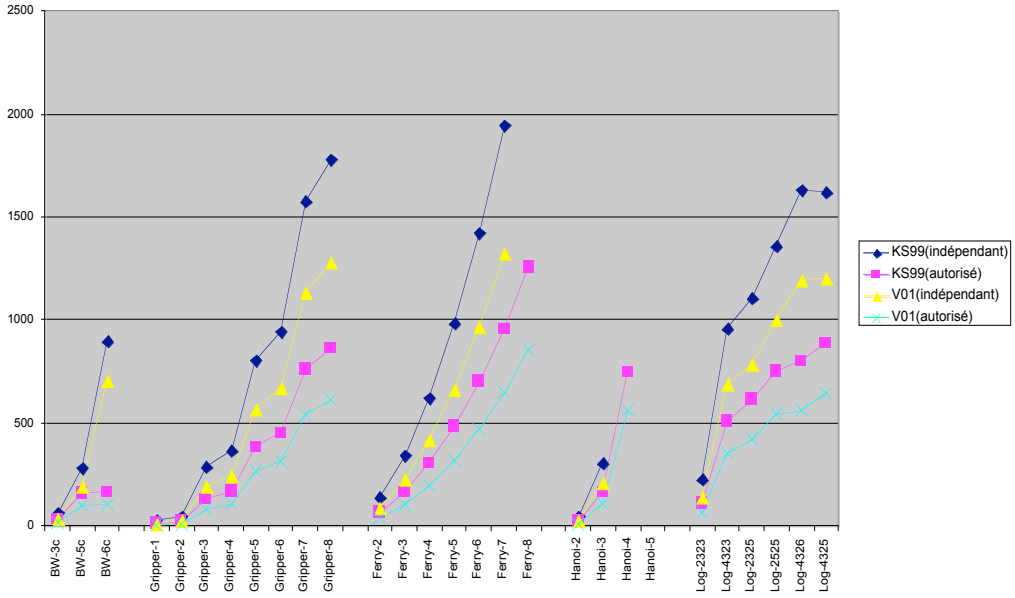
1. $\left(\bigwedge_{n_f \in NoeudsInit(GP)} n_f \right) \wedge \left(\bigwedge_{n_f \in NoeudsBut(GP)} n_f \right)$
2. $\bigwedge_{(n_f, n_a) \in Arcs\,\mathrm{Prec}(GP)} (n_a \Rightarrow n_f)$
3. $\bigwedge_{n_f \in (NoeudsF(GP) - NoeudsInit(GP))} \left(n_f \Rightarrow \bigvee_{(n_a, n_f) \in ArcsAdd(GP)} n_a \right)$
4. $\bigwedge_{\{n_a, n_b\} \in MutexA(GP)} (\neg n_a \vee \neg n_b)$

- **Codage [Vidal, 2001]**

1. $\bigwedge_{n_f \in NoeudsBut(GP)} \left(\bigvee_{(n_a, n_f) \in ArcsAdd(GP)} n_a \right)$
2. $\bigwedge_{(n_a, n_f) \in Arcs\,\mathrm{Prec}(GP) / n_f \notin NoeudsInit(GP)} \left(n_a \Rightarrow \bigvee_{(n_b, n_f) \in ArcsAdd(GP)} n_b \right)$
3. $\bigwedge_{\{n_a, n_b\} \in MutexA(GP)} (\neg n_a \vee \neg n_b)$

2.5. Codages SAT du graphe

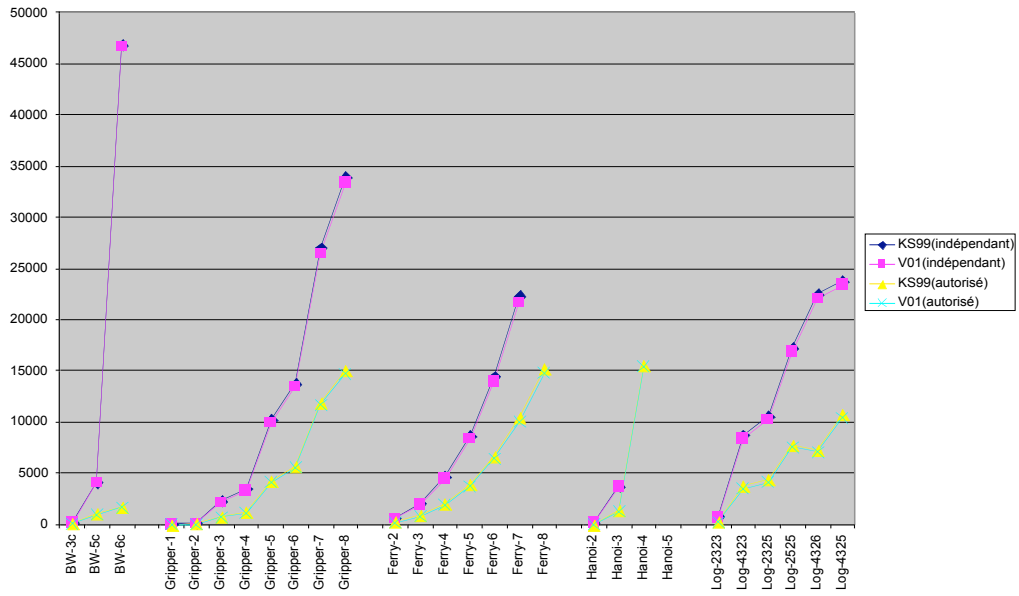
- **Nombre de variables**



UNIVERSITE PAUL SABATIER TOULOUSE III

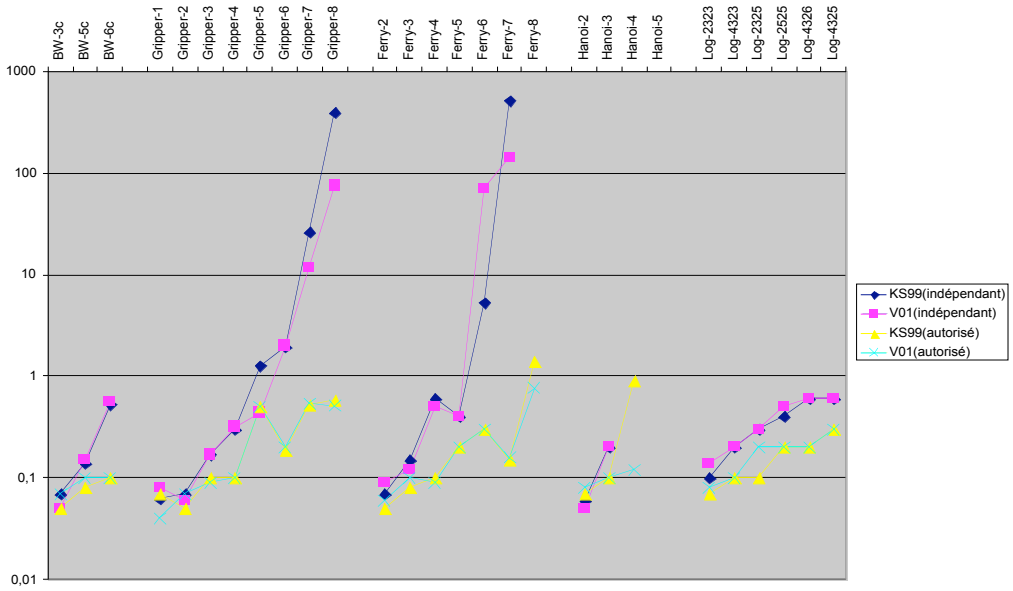
2.5. Codages SAT du graphe

- **Nombre de clauses**

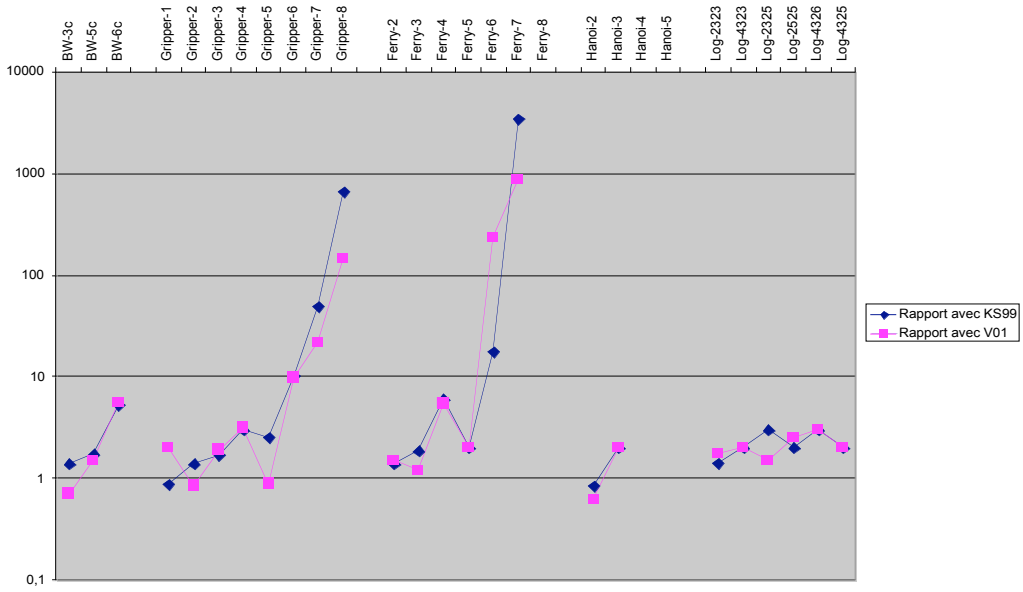


© Régnier, IRIT 2006

- **Temps de résolution**



- **Gains apportés par l'autorisation**



2.5. Codages SAT du graphe

- **Enseignements :**
  - **Codage V01 :** diminution significative du nombre de variables ;
  - **Utilisation de la relation d'autorisation** : diminution importante du nombre de variables, de clauses, et des temps de résolution ;
  - Le **codage V01** autorisé est le **plus compact et performant**, légèrement devant le codage KS99 autorisé.
  - Le système SATPLAN'04, successeur de BLACKBOX, est le plus performant dans le catégorie planificateurs optimaux à l'IPC'04.

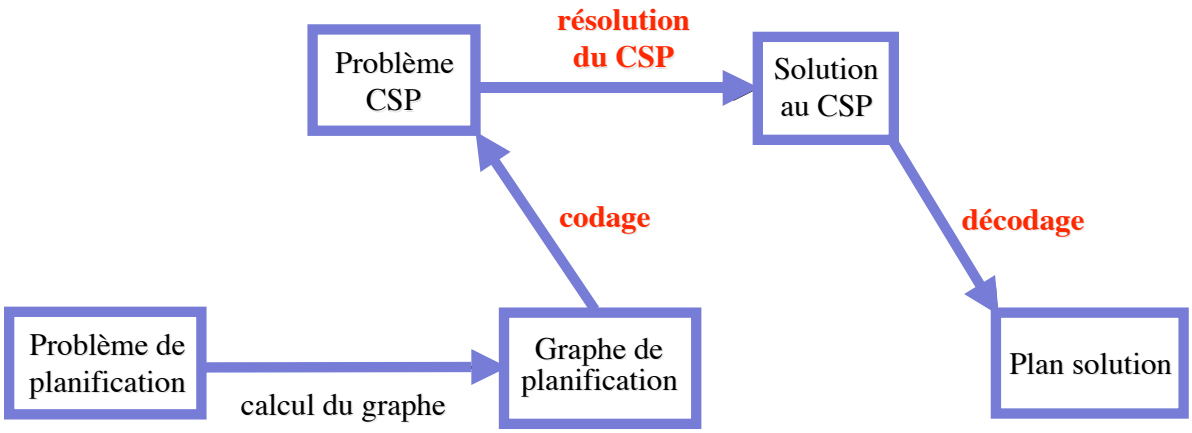
2.5. Codages SAT du graphe

- **Cinquième amélioration :**
  - Optimisation de l'extraction d'une solution :
    - **Codages CSP du graphe**

© Régnier, IRIT 2006

2.6. Codages CSP du graphe

GPCSP [Kambhampati, Nigenda, 2000]



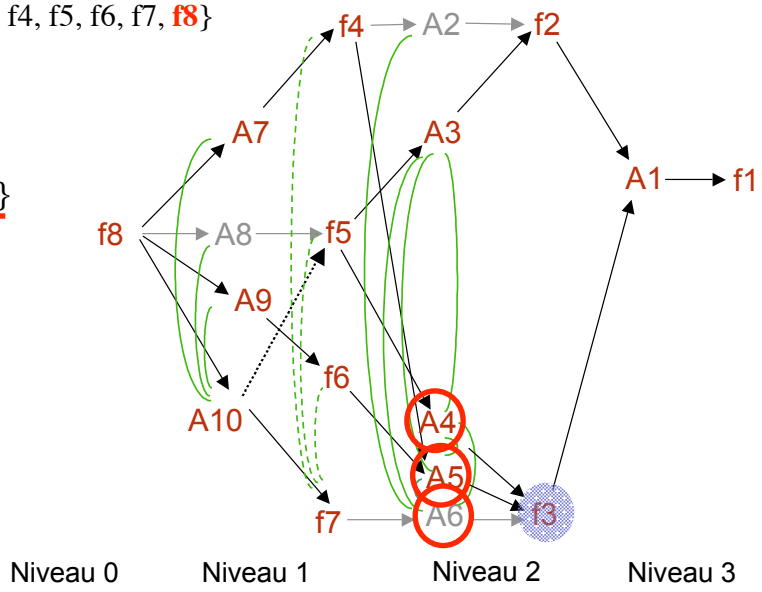
Compilation de plans : GPCSP

2.6. Codages CSP du graphe

- **Principe :**
  - **Codage :** transformation du problème de planification en un problème de satisfaction de contraintes (CSP) pour une longueur maximale de plan donnée k ;
  - **Résolution :** recherche d'une solution par un solveur CSP :
    - solution au CSP : décodage qui donne un plan-solution ;
    - pas de solution au CSP : pas de plan de longueur ≤ k ; relance du processus pour k+1 ;
- **Objectif :** bénéficier automatiquement des progrès des techniques CSP ;
- **Systèmes de référence :**
  - GPCSP.

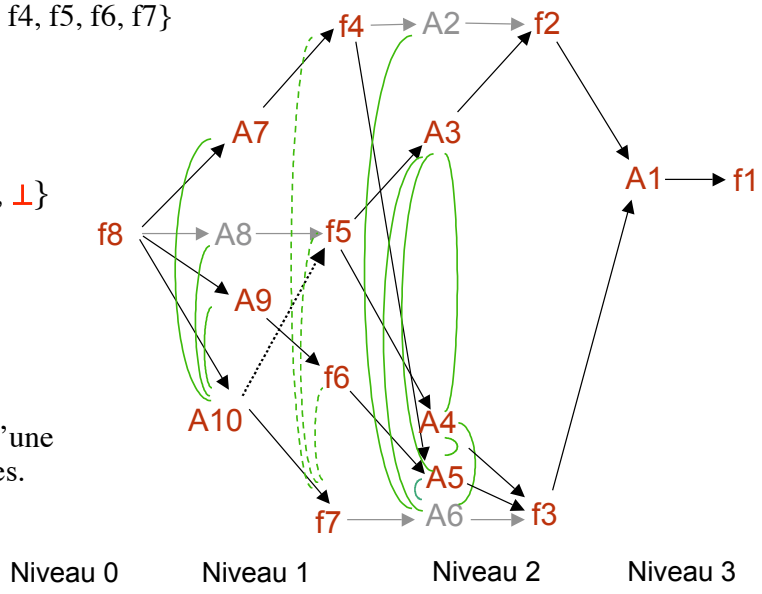
2.6. Codages CSP du graphe

- Encodage du graphe de planification en CSP
  - Variables = {f1, f2, f3, f4, f5, f6, f7, **f8**}
  - Domaines :
    D(f1)={A1}
    D(f2)={A2,A3}
    D(f3)={A4, A5, A6}
    D(f4)={A7}
    D(f5)={A8}
    D(f6)={A9}
    D(f7)={A10}



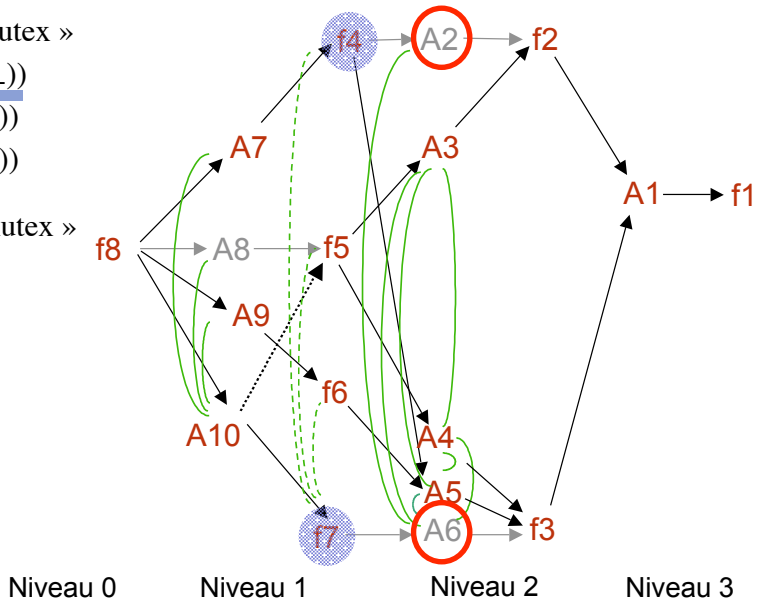
2.6. Codages CSP du graphe

- Encodage du graphe de planification en CSP
  - Variables = {f1, f2, f3, f4, f5, f6, f7}
  - Domaines :
    D(f1)={A1}
    D(f2)={A2,A3, ⊥}
    D(f3)={A4, A5, A6, ⊥}
    D(f4)={A7, ⊥}
    D(f5)={A8, ⊥}
    D(f6)={A9, ⊥}
    D(f7)={A10, ⊥}
- Heuristiques pour le choix d'une valeur à affecter aux variables.



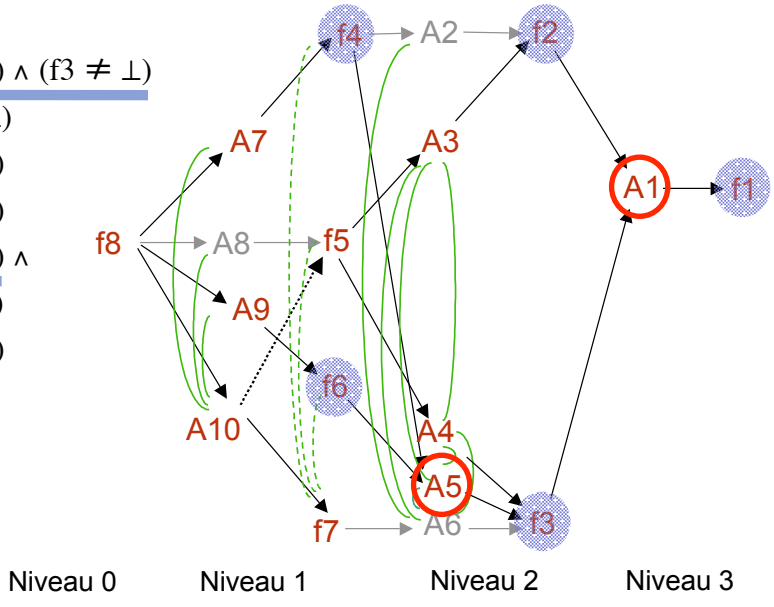
2.6. Codages CSP du graphe

- Encodage du graphe de planification en CSP
  - Contraintes « fluents mutex »
    - ¬ ((f4 ≠ ⊥) ∧ (f7 ≠ ⊥))
    - ¬ ((f5 ≠ ⊥) ∧ (f7 ≠ ⊥))
    - ¬ ((f6 ≠ ⊥) ∧ (f7 ≠ ⊥))
  - Contraintes « actions mutex »
    - (f2=A2) → (f3≠A6)
    - (f2=A3) → (f3≠A4)
    - (f2=A3) → (f3≠A5)
    - (f2=A3) → (f3≠A6)
    - (f4=A7) → (f7≠A10)
    - (f5=A8) → (f7≠A10)
    - (f6=A9) → (f7≠A10)



2.6. Codages CSP du graphe

- Encodage du graphe de planification en CSP
  - Contraintes d'activités
    - (f1=A1) → (f2 ≠ ⊥) ∧ (f3 ≠ ⊥)
    - (f2=A2) → (f4 ≠ ⊥)
    - (f2=A3) → (f5 ≠ ⊥)
    - (f3=A4) → (f5 ≠ ⊥)
    - (f3=A5) → (f4 ≠ ⊥) ∧ (f6 ≠ ⊥)
    - (f3=A6) → (f7 ≠ ⊥)



2.5. Codages SAT du graphe

- **Enseignements :**
  - GPCSP est compétitif avec BLACKBOX'00 ;
  - GPCSP consomme moins de place mémoire que BLACKBOX'00 ;
  - Les heuristiques classiques des CSP sont utilisables et apportent un surcroît d'efficacité (cf. chapitre 2.3) .