Journées d'Intelligence Artificielle Fondamentale

Un algorithme pour la résolution optimale de problèmes de planification avec actions valués

Martin Cooper
Marie de Roquemaurel
Pierre Régnier

IRIT CNRS INPT UPS UT1
Institut de Recherche en Informatique de Toulouse

Université Paul Sabatier
TOULOUSE III

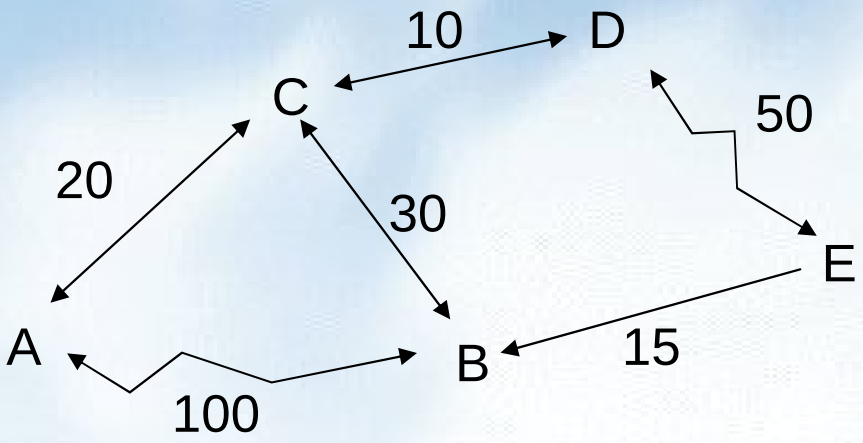
Plan

1. Introduction
2. Construction d’un graphe de planification
3. Codage d’un graphe en WCSP
4. Résolution optimale d’un WCSP
5. Résolution optimale d’un problème valué
6. Conclusions et perspectives

Planification avec actions valués

- Etat : ensemble de propositions (fluent).
- Action a : <prec(a),effet(a),coût(a)>
  - prec(a) : préconditions de a,
  - effet(a) : effets (ajouts, retraits) de a,
  - coût(a) : coût d'application de a.
- Métrique d'un plan : somme des coûts des actions du plan.
- Problème de planification avec actions valués : <A,I,B>
  - A : ensemble d'actions,
  - I : état initial du problème,
  - B : but du problème.
- Obtenir un plan-solution P* de coût optimal C*.
- Cadre de la planification classique (actions instantanées, statiques et déterministes, observabilité totale et agents omniscients) augmenté de la prise en compte des coûts des actions.

Exemple



Actions : $\forall i, j \in \{A,B,C,D,E\}$

trajet_ij = <{v_i}, {v_j, ¬ v_i}, cout_ij>

Prendre_i = <{v_i,c_i}, {c_v,¬v_i}, 5>

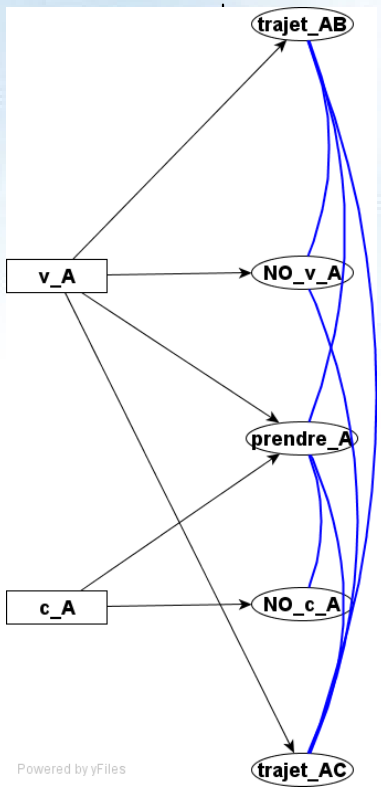
Poser_i = <{v_i,c_v}, {c_i, ¬c_v}, 3>

Etat Initial : I = {v_a, c_a}

But : B = {c_b}

2. Construction d’un graphe de planification

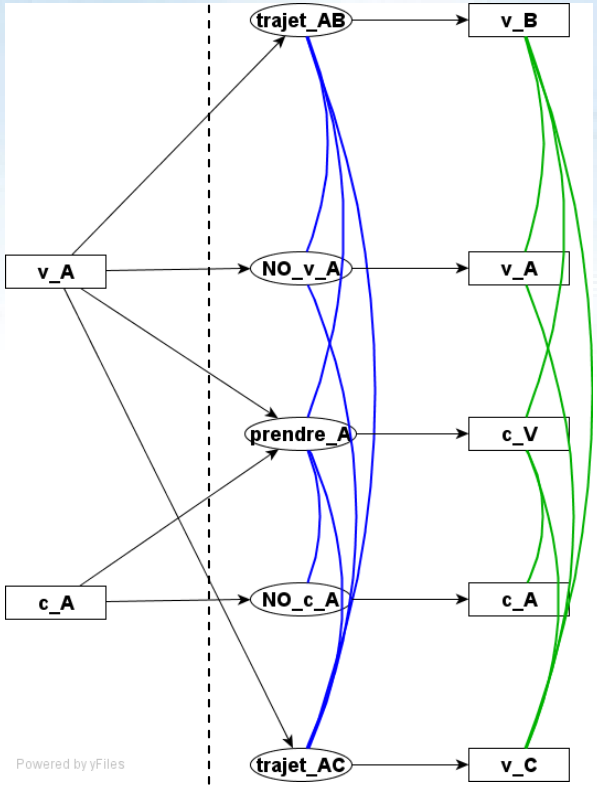
Niveau 0 | Niveau 1 | Niveau 2 | Niveau 3



1. Introduction
2. Construction d’un graphe
3. Codage d’un graphe en WCSP
4. Résolution optimale d’un WCSP
5. Résolution optimale d’un problème valué
6. Conclusions et perspectives

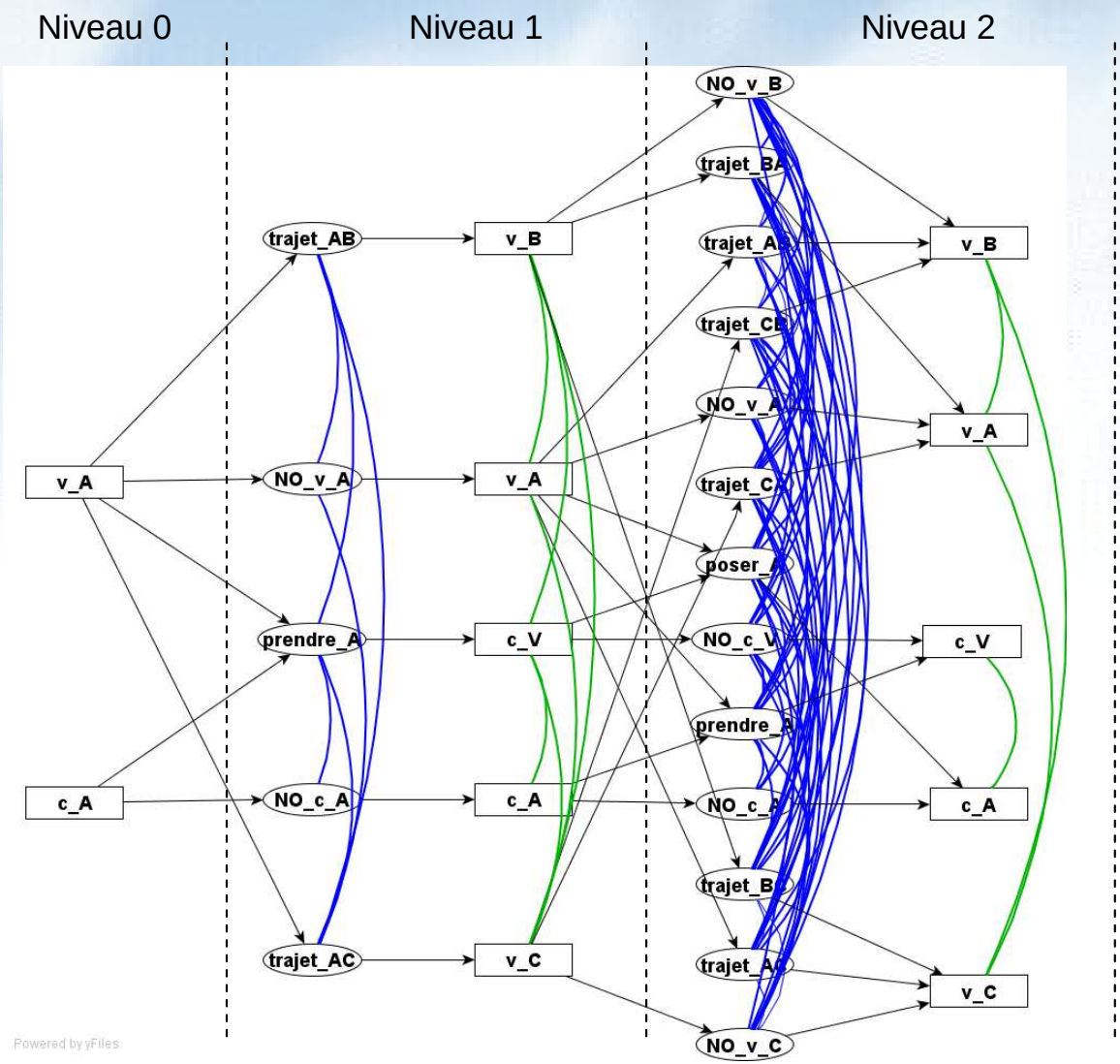
Construction d'un graphe de planification

Niveau 0 | Niveau 1 | Niveau 2 | Niveau 3



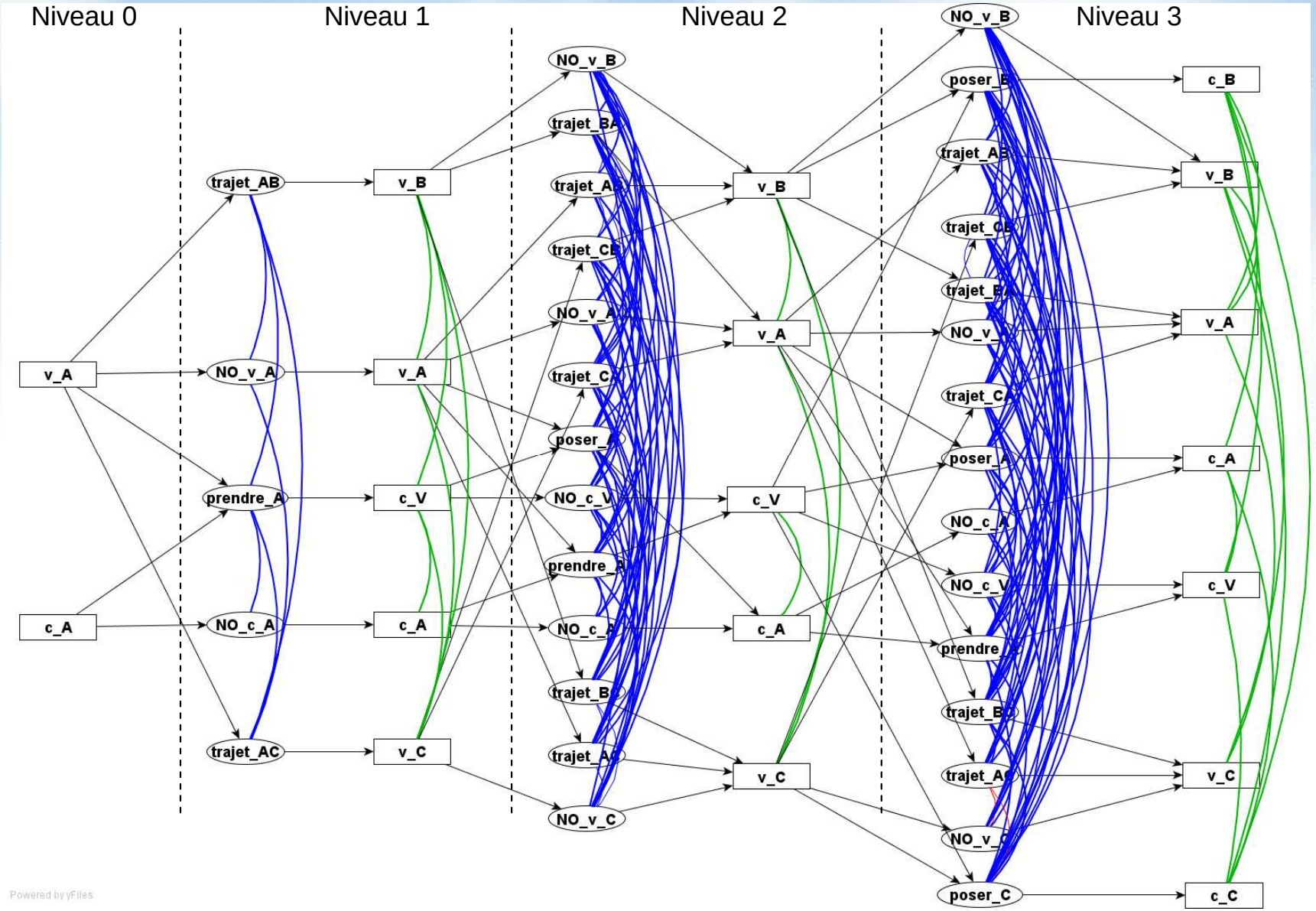
1. Introduction
2. Construction d'un graphe
3. Codage d'un graphe en WCSP
4. Résolution optimale d'un WCSP
5. Résolution optimale d'un problème valué
6. Conclusions et perspectives

Construction d'un graphe de planification



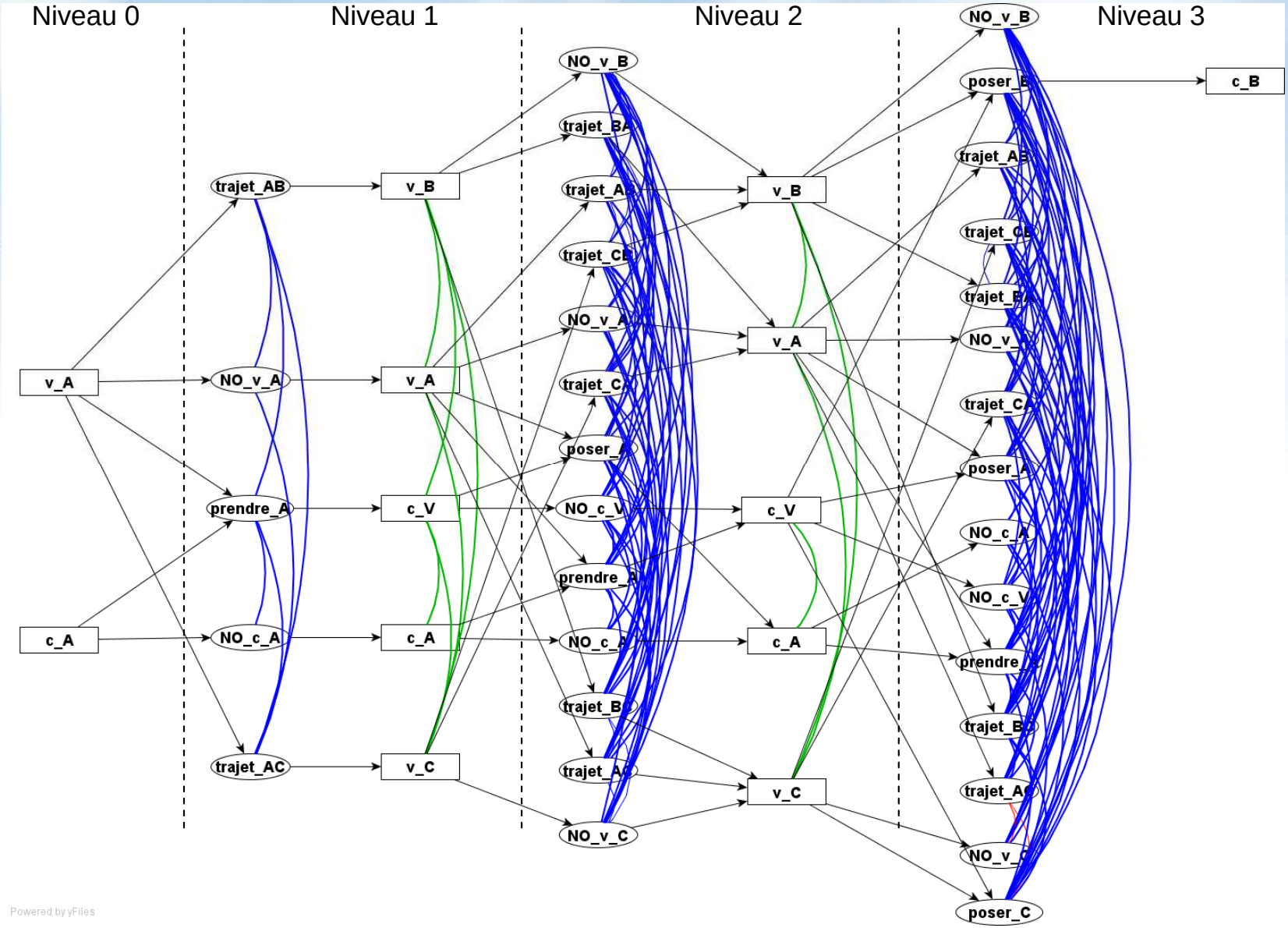
1. Introduction
2. Construction d'un graphe
3. Codage d'un graphe en WCSP
4. Résolution optimale d'un WCSP
5. Résolution optimale d'un problème valué
6. Conclusions et perspectives

Construction d'un graphe de planification



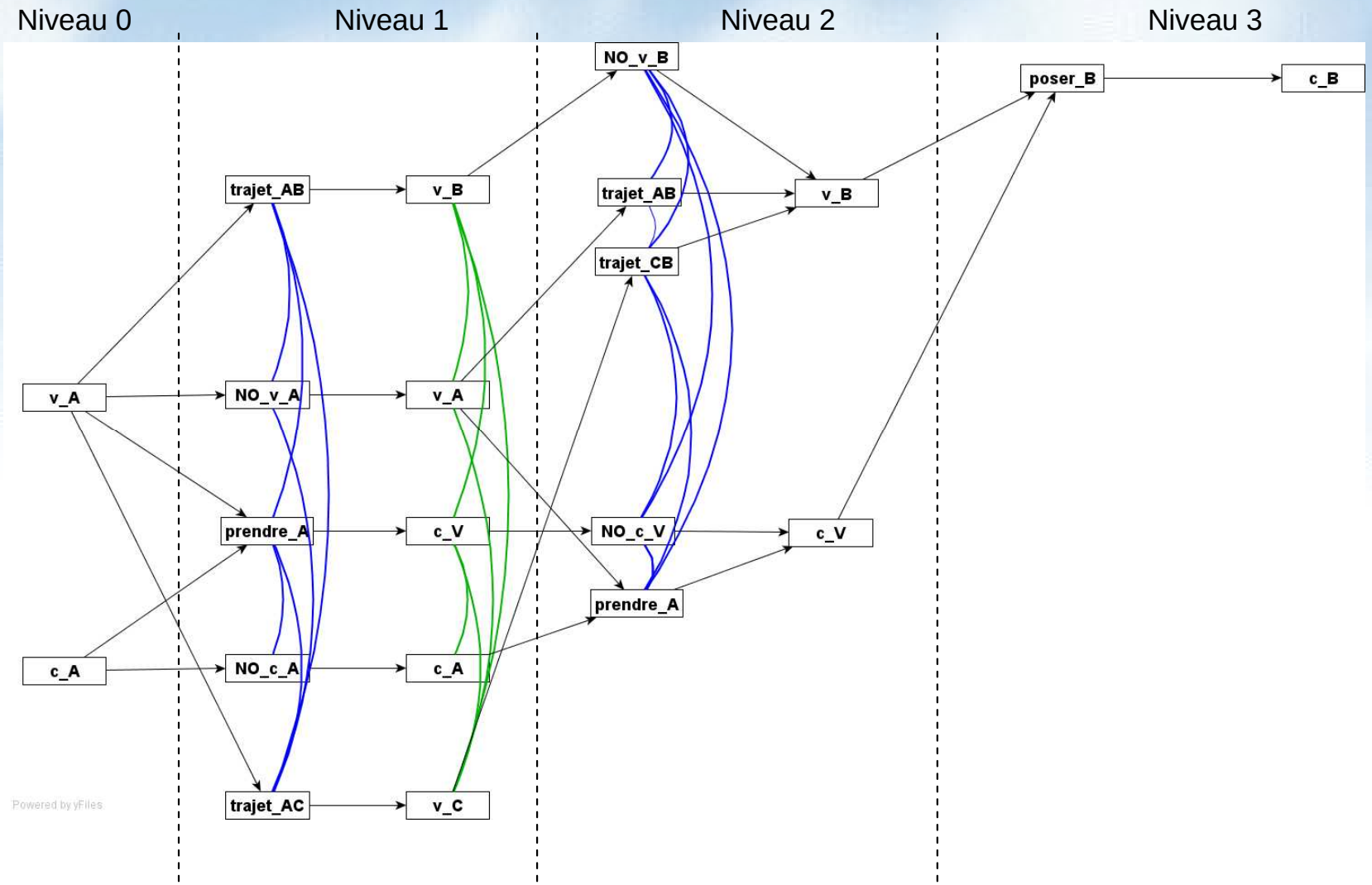
1. Introduction
2. Construction d'un graphe
3. Codage d'un graphe en WCSP
4. Résolution optimale d'un WCSP
5. Résolution optimale d'un problème valué
6. Conclusions et perspectives

Réduction d'un graphe



1. Introduction
2. Construction d'un graphe
3. Codage d'un graphe en WCSP
4. Résolution optimale d'un WCSP
5. Résolution optimale d'un problème valué
6. Conclusions et perspectives

Réduction d'un graphe



1. Introduction
2. Construction d'un graphe
3. Codage d'un graphe en WCSP
4. Résolution optimale d'un WCSP
5. Résolution optimale d'un problème valué
6. Conclusions et perspectives

Codage du graphe en WCSP

- Introduction : Les CSP valués
  - CSP (Constraint Satisfaction Problem) : <X,D,C>
    - $X=\{x_1,\ldots,x_n\}$ ensemble fini de variables
    - $D=\{d_1,\ldots,d_n\}$ ensemble des domaines d_i associés aux variables x_i. Les d_i sont eux-mêmes des ensembles d'entiers.
    - C ensemble de contraintes. Une contrainte est un ensemble de tuples portant sur un sous ensemble de X contenant les affectations non autorisées.
  - Affectation : à chaque variable x_i, on associe une et une seule valeur du domaine d_i.
  - Une solution est une affectation complète et cohérente (toutes les variables de X sont affectées et aucune contrainte n'est violée)
  - Valuation : les contraintes ont un coût. On cherche à minimiser la somme de ces coûts.

Codage du graphe en WCSP

a) Réécriture du graphe : en partant du dernier niveau, on renomme chaque fluent en f_i et on numérote chaque action en j

b) Création des variables et des domaines :

variables = fluents (sauf état initial)

domaines = actions produisant le fluent U {-1} sauf pour les buts

c) Traduction des mutex entre fluents :

Mutex(f_i,f_j) est traduit par ((f_i=-1) ou (f_j=-1))

d) Traduction des mutex entre actions :

Mutex(a,b) est traduit par ((f_i=a) => (f_j≠b)) avec i≠j, f_i Є effet(a) et f_j Є effet(b)

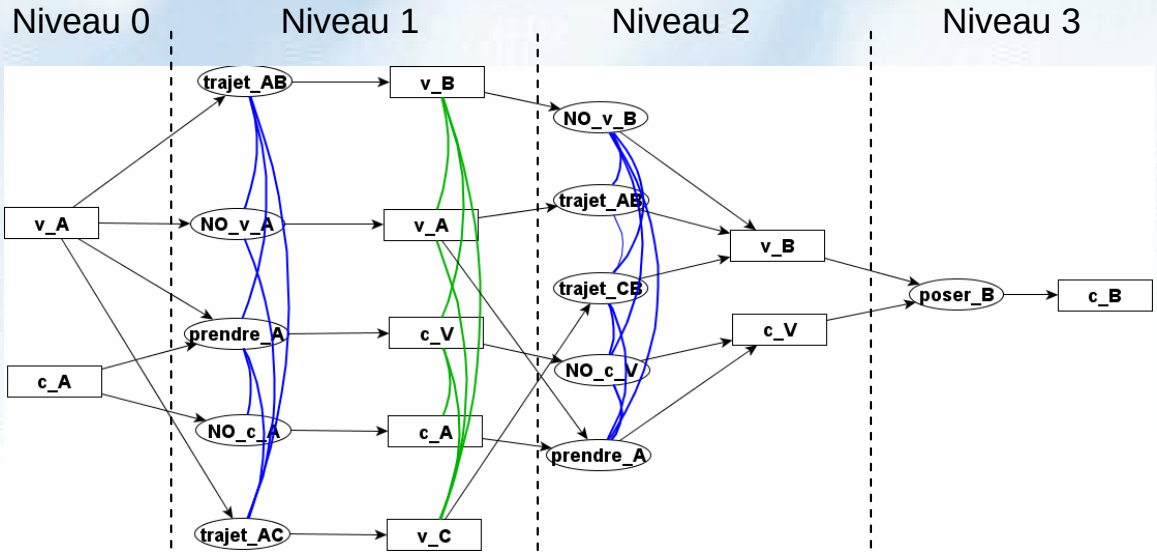
e) Traduction des arcs d'activité : l'activation d'un fluent f_i produit par une action a entraine l'activation des préconditions de a : ((f_i = a) => (f_j ≠ -1)), f_j Є prec(a)

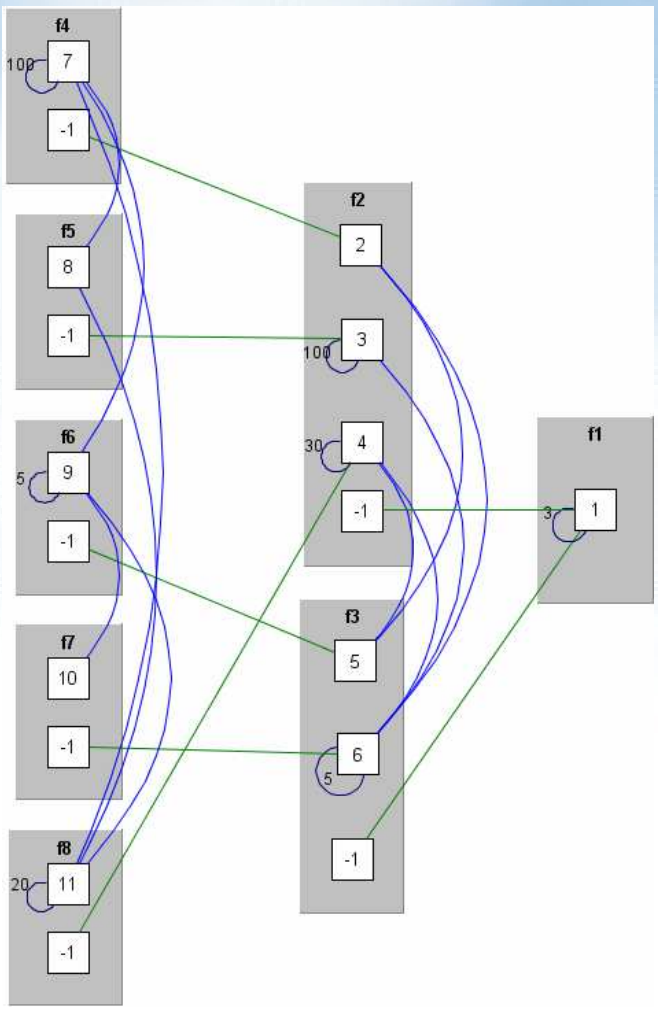
f) Traduction du coût des actions : pour chaque action a de coût non nul, on ajoute la contrainte unaire valuée : f_i≠a (cout(a))

h) Prise en compte des ajouts multiples : pour chaque action a qui produit plusieurs fluents f_i, on crée un fluent fi_int de domaine {a,-1}. Les f_i sont reliés à une nouvelle action a_int de coût nul. On ajoute la contrainte entre fi_int et fi :

(fi_int=-1) => (f_i≠a_int), f_i Є add(a)

Codage du graphe en WCSP

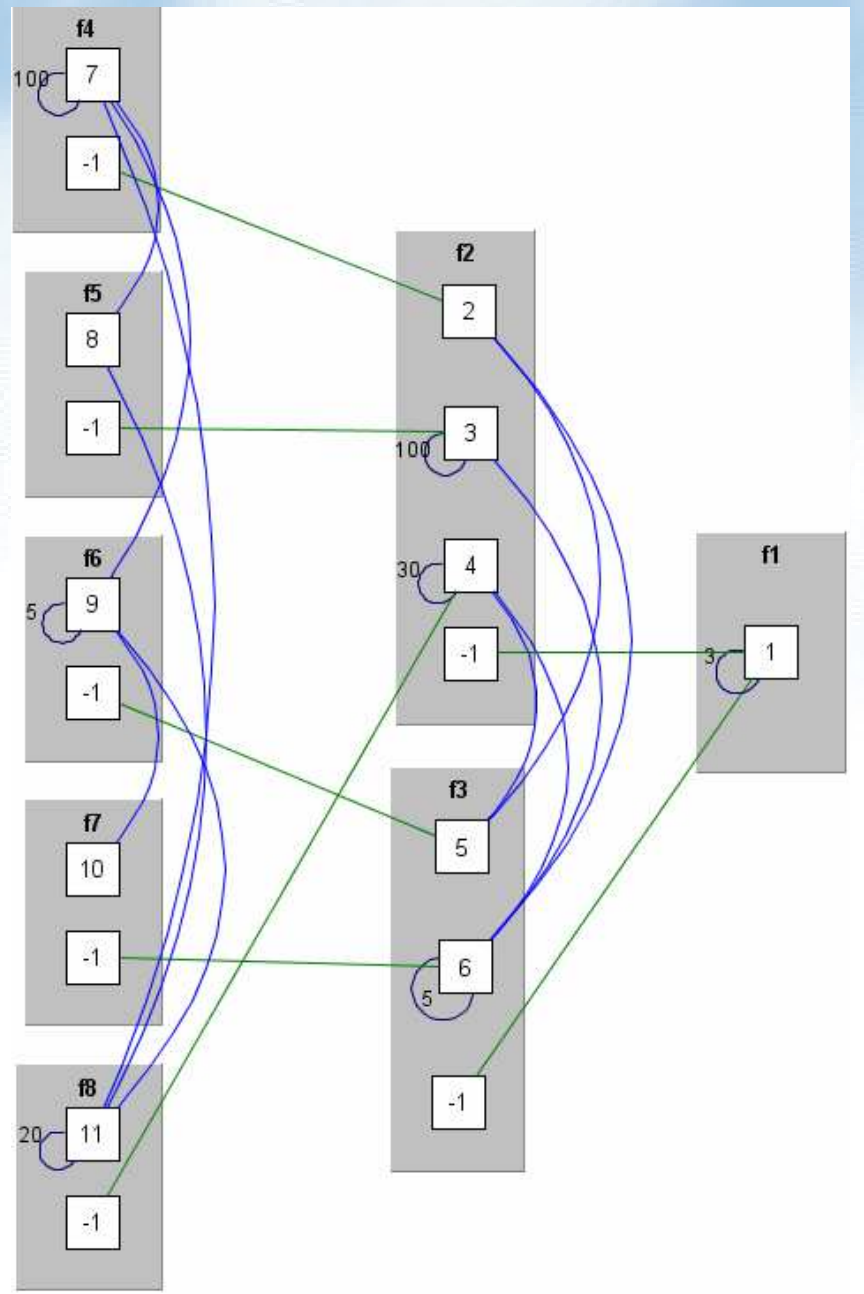




1. Introduction
2. Construction d'un graphe
3. Codage d'un graphe en WCSP
4. Résolution optimale d'un WCSP
5. Résolution optimale d'un problème valué
6. Conclusions et perspectives

Résolution optimale d'un WCSP

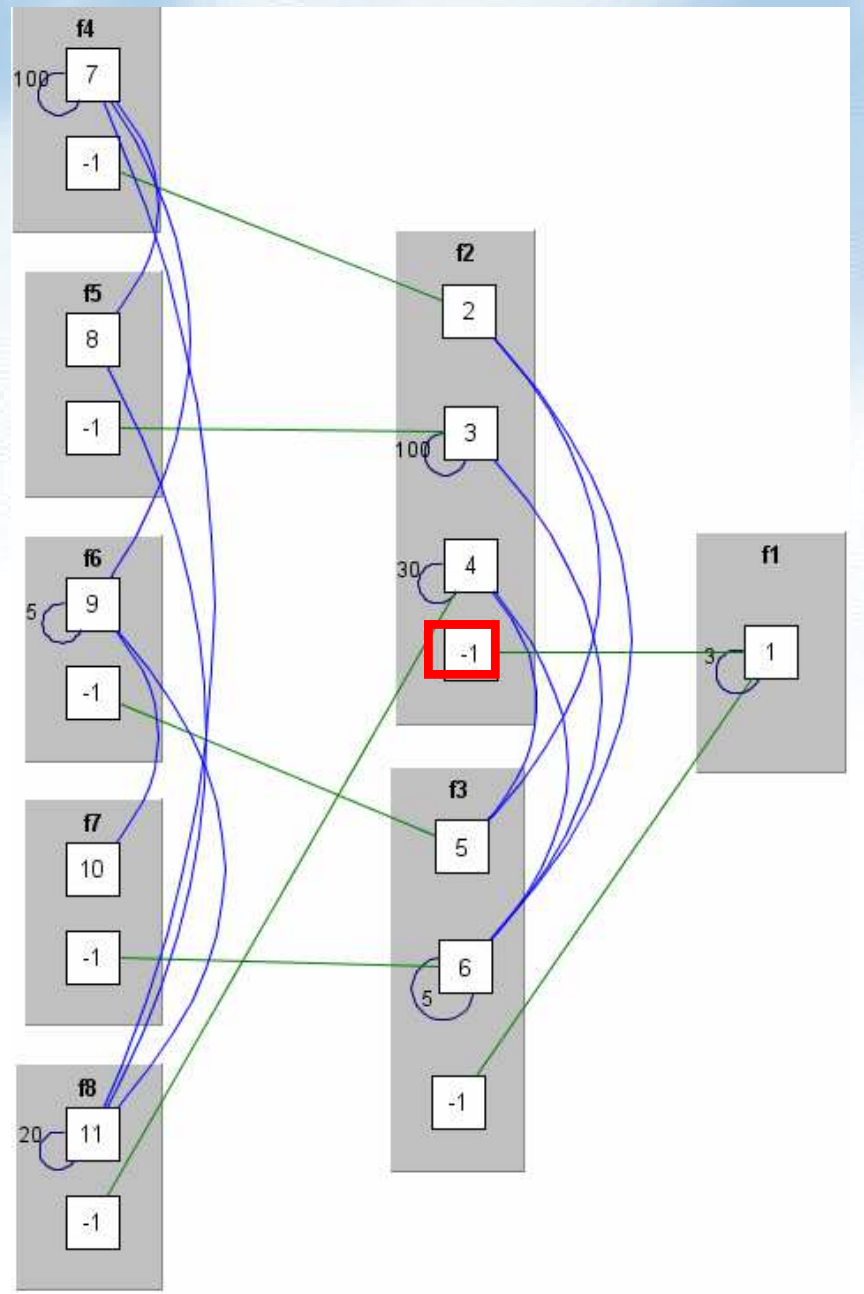
- Algorithme AC (Arc Consistency)
  - Permet simplifier le CSP
  - Supprime les valeurs des domaines n'ayant pas de support avec une autre variable (pas d'affectation possible pour la valeur à supprimer et une autre variable du CSP)



1. Introduction
2. Construction d'un graphe
3. Codage d'un graphe en WCSP
4. Résolution optimale d'un WCSP
5. Résolution optimale d'un problème valué
6. Conclusions et perspectives

Résolution optimale d'un WCSP

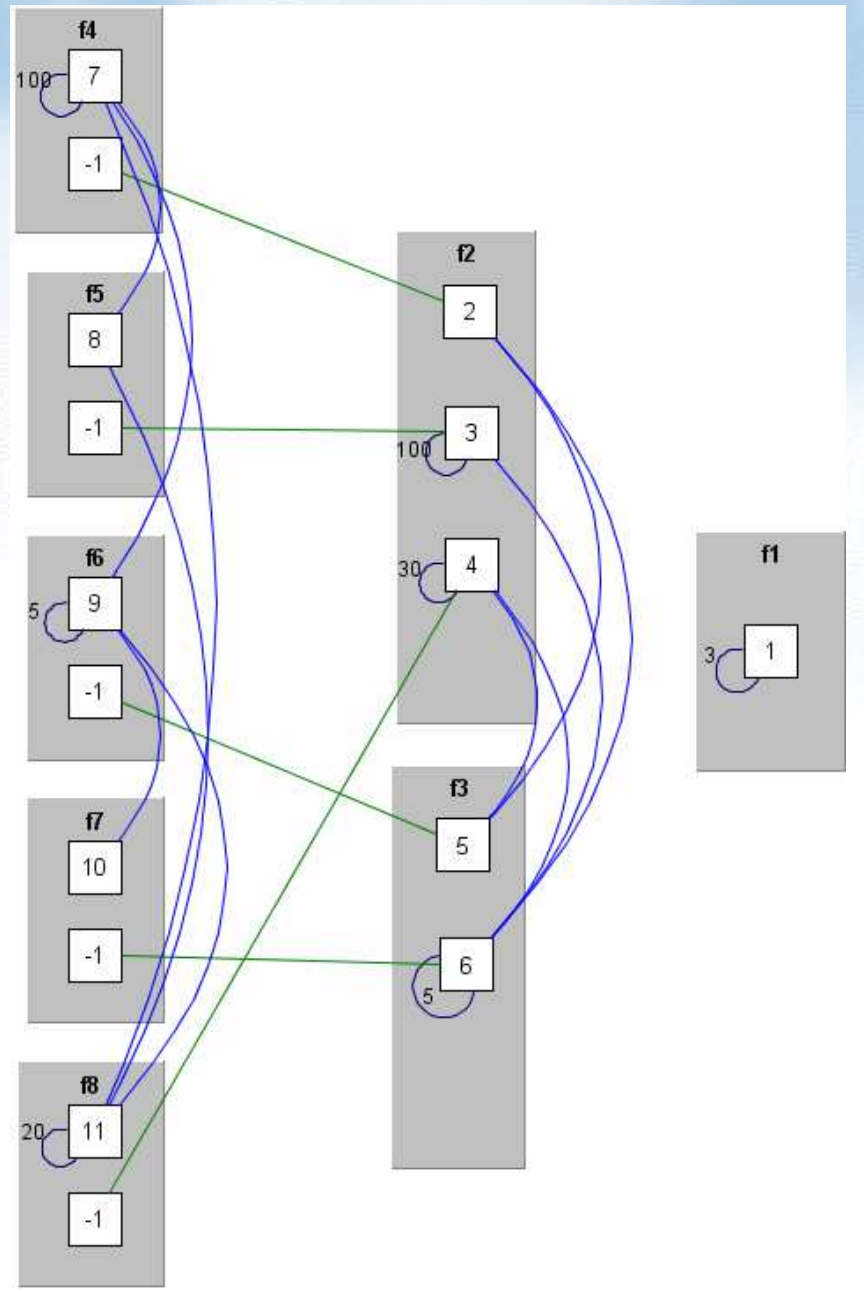
- Algorithme AC (Arc Consistency)
  - Permet simplifier le CSP
  - Supprime les valeurs des domaines n'ayant pas de support avec une autre variable (pas d'affectation possible pour la valeur à supprimer et une autre variable du CSP)



1. Introduction
2. Construction d'un graphe
3. Codage d'un graphe en WCSP
4. Résolution optimale d'un WCSP
5. Résolution optimale d'un problème valué
6. Conclusions et perspectives

Résolution optimale d'un WCSP

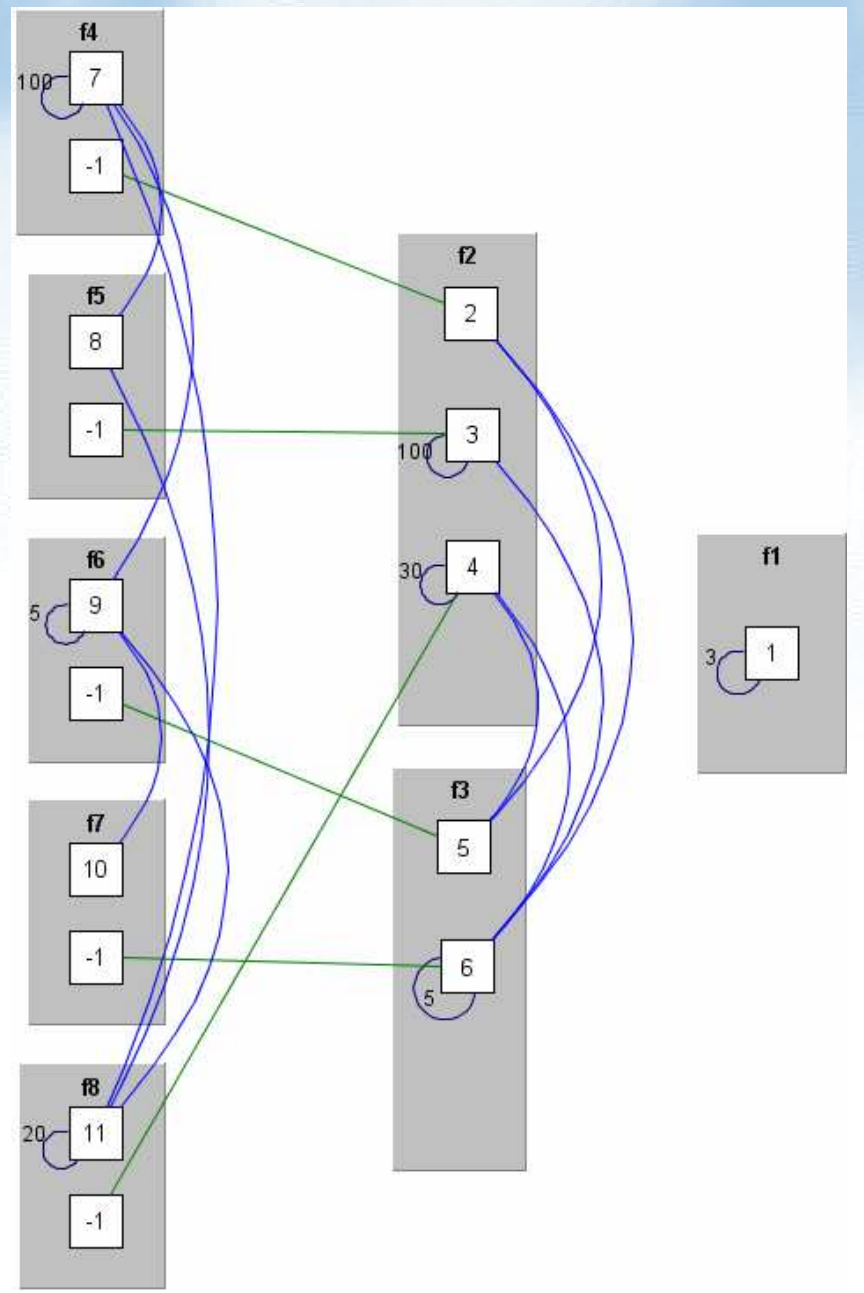
- Algorithme AC (Arc Consistency)
  - Permet simplifier le CSP
  - Supprime les valeurs des domaines n'ayant pas de support avec une autre variable (pas d'affectation possible pour la valeur à supprimer et une autre variable du CSP)



1. Introduction
2. Construction d'un graphe
3. Codage d'un graphe en WCSP
4. Résolution optimale d'un WCSP
5. Résolution optimale d'un problème valué
6. Conclusions et perspectives

Résolution optimale d'un WCSP

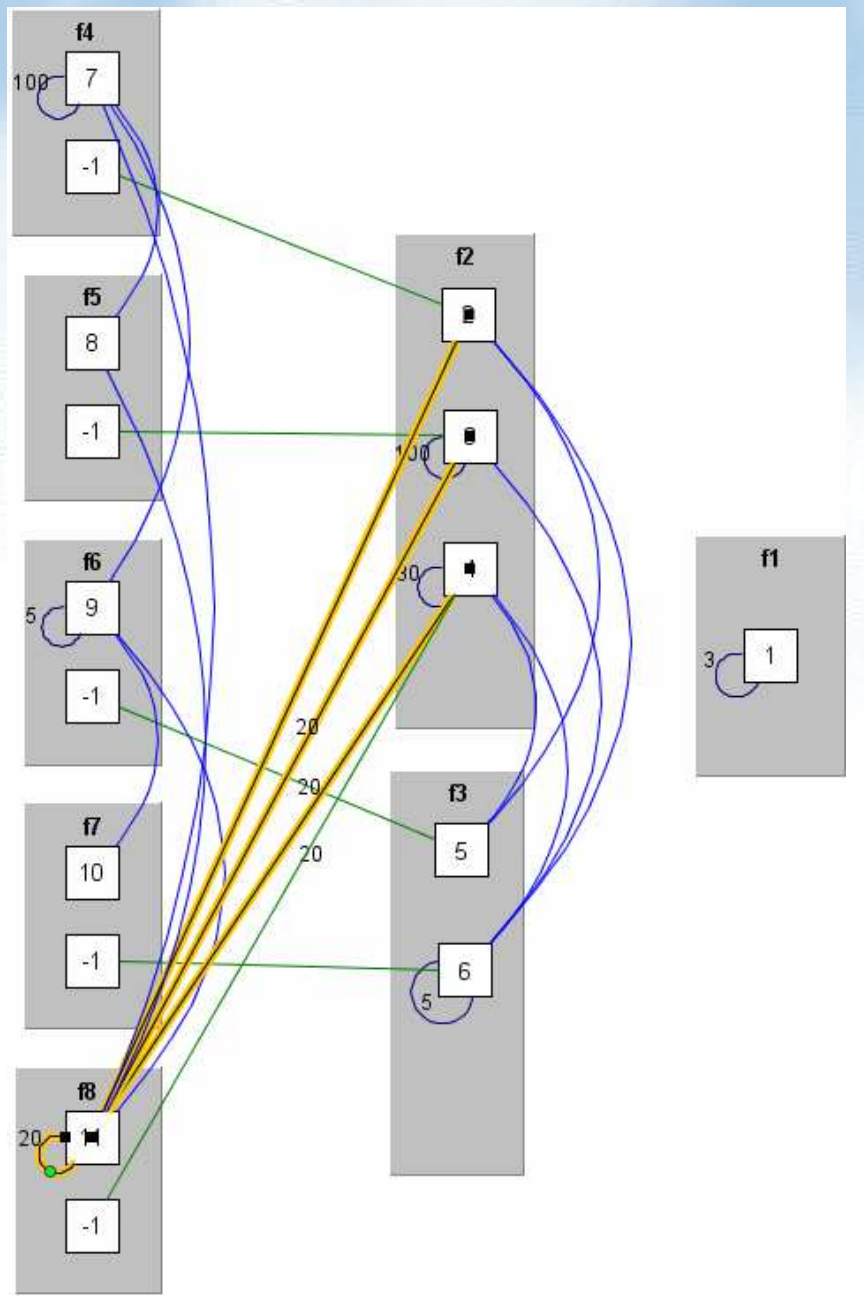
- Algorithme FDAC (Full Directionnal Arc Consistency)
  - Permet de trouver une borne inférieure du coût de la solution par application de AC puis par projection et extension des contraintes
  - Réduit le champ d'exploration des algorithmes de recherche du type branch-and-bound
  - Complexité polynomiale



1. Introduction
2. Construction d'un graphe
3. Codage d'un graphe en WCSP
4. Résolution optimale d'un WCSP
5. Résolution optimale d'un problème valué
6. Conclusions et perspectives

Résolution optimale d'un WCSP

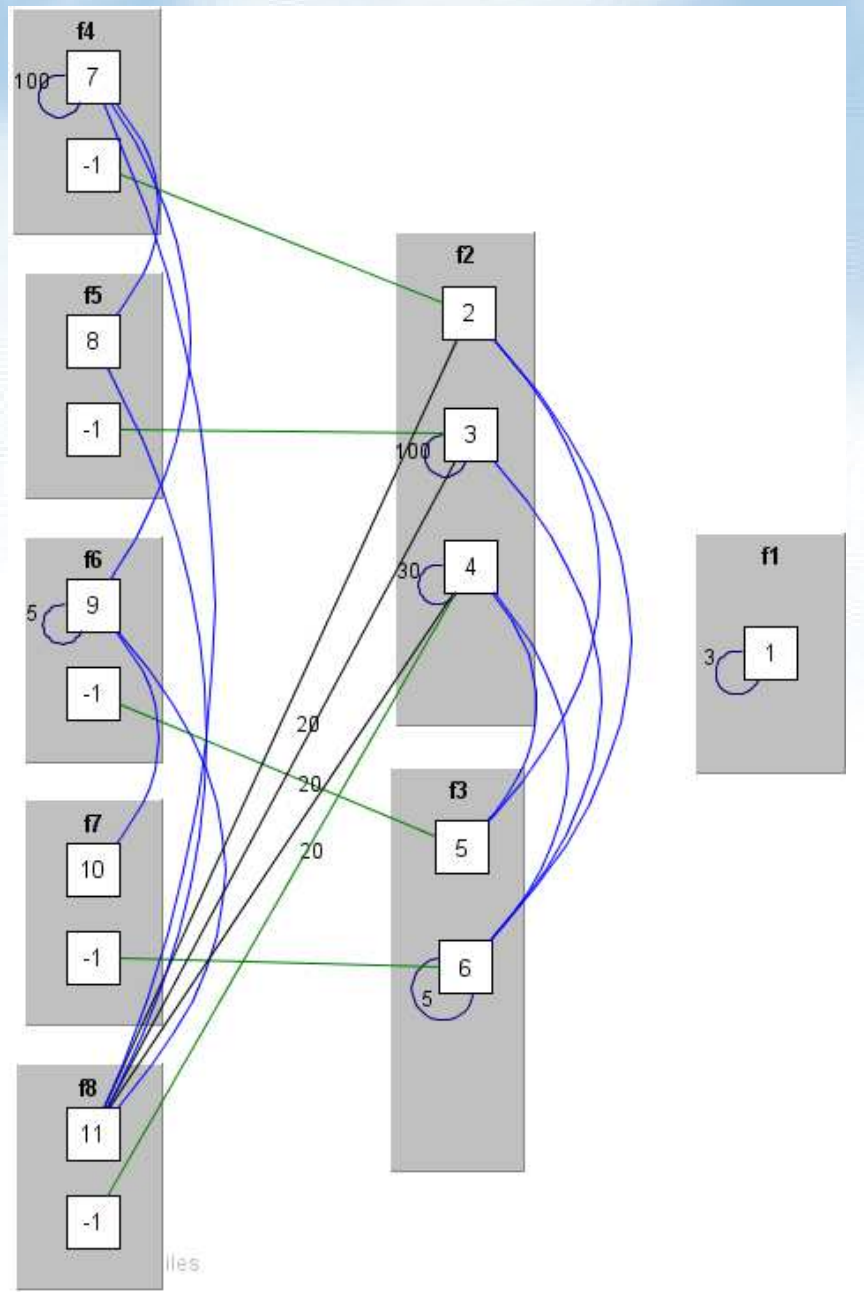
- Algorithme FDAC (Full Directionnal Arc Consistency)
  - Permet de trouver une borne inférieure du coût de la solution par application de AC puis par projection et extension des contraintes
  - Réduit le champ d'exploration des algorithmes de recherche du type branch-and-bound
  - Complexité polynomiale



1. Introduction
2. Construction d'un graphe
3. Codage d'un graphe en WCSP
4. Résolution optimale d'un WCSP
5. Résolution optimale d'un problème valué
6. Conclusions et perspectives

Résolution optimale d'un WCSP

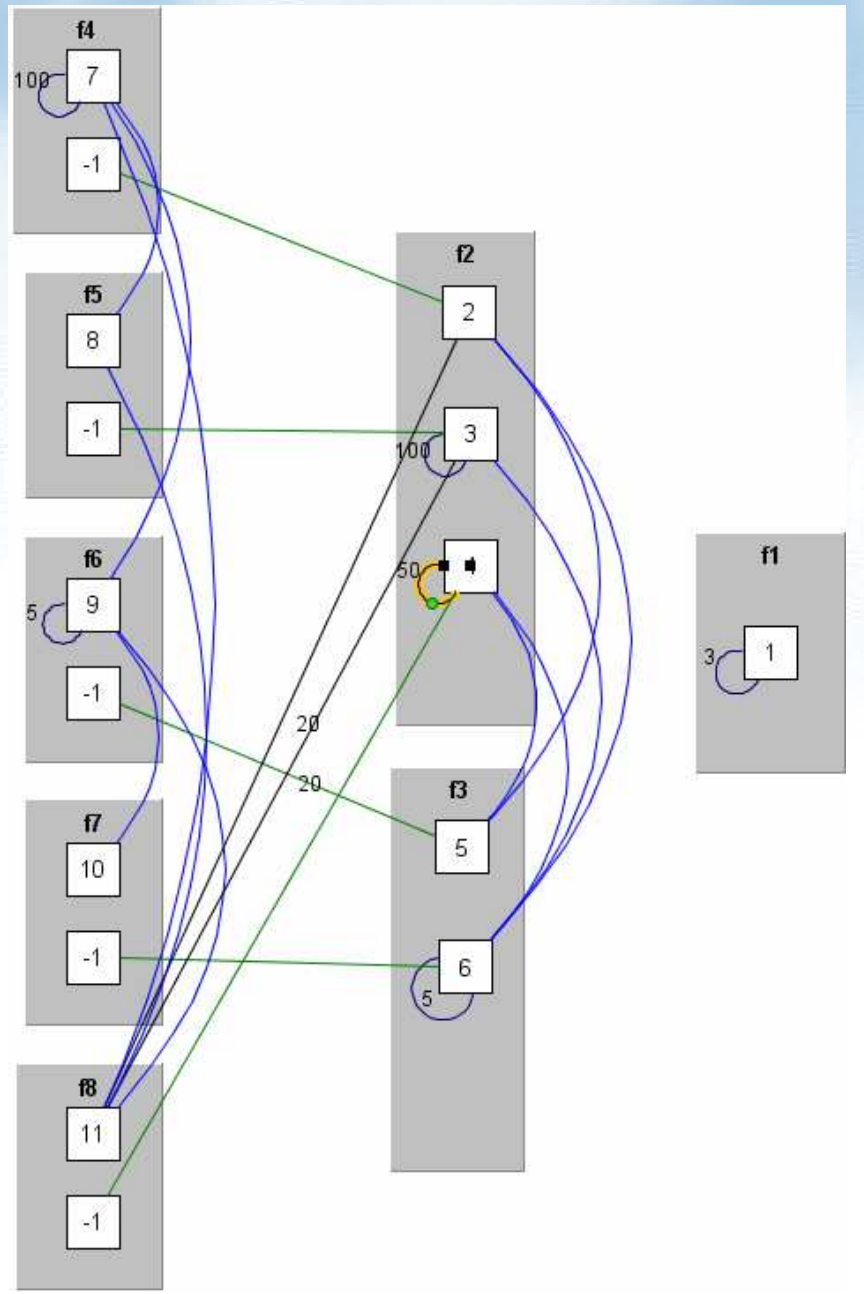
- Algorithme FDAC (Full Directionnal Arc Consistency)
  - Permet de trouver une borne inférieure du coût de la solution par application de AC puis par projection et extension des contraintes
  - Réduit le champ d'exploration des algorithmes de recherche du type branch-and-bound
  - Complexité polynomiale



1. Introduction
2. Construction d'un graphe
3. Codage d'un graphe en WCSP
4. Résolution optimale d'un WCSP
5. Résolution optimale d'un problème valué
6. Conclusions et perspectives

Résolution optimale d'un WCSP

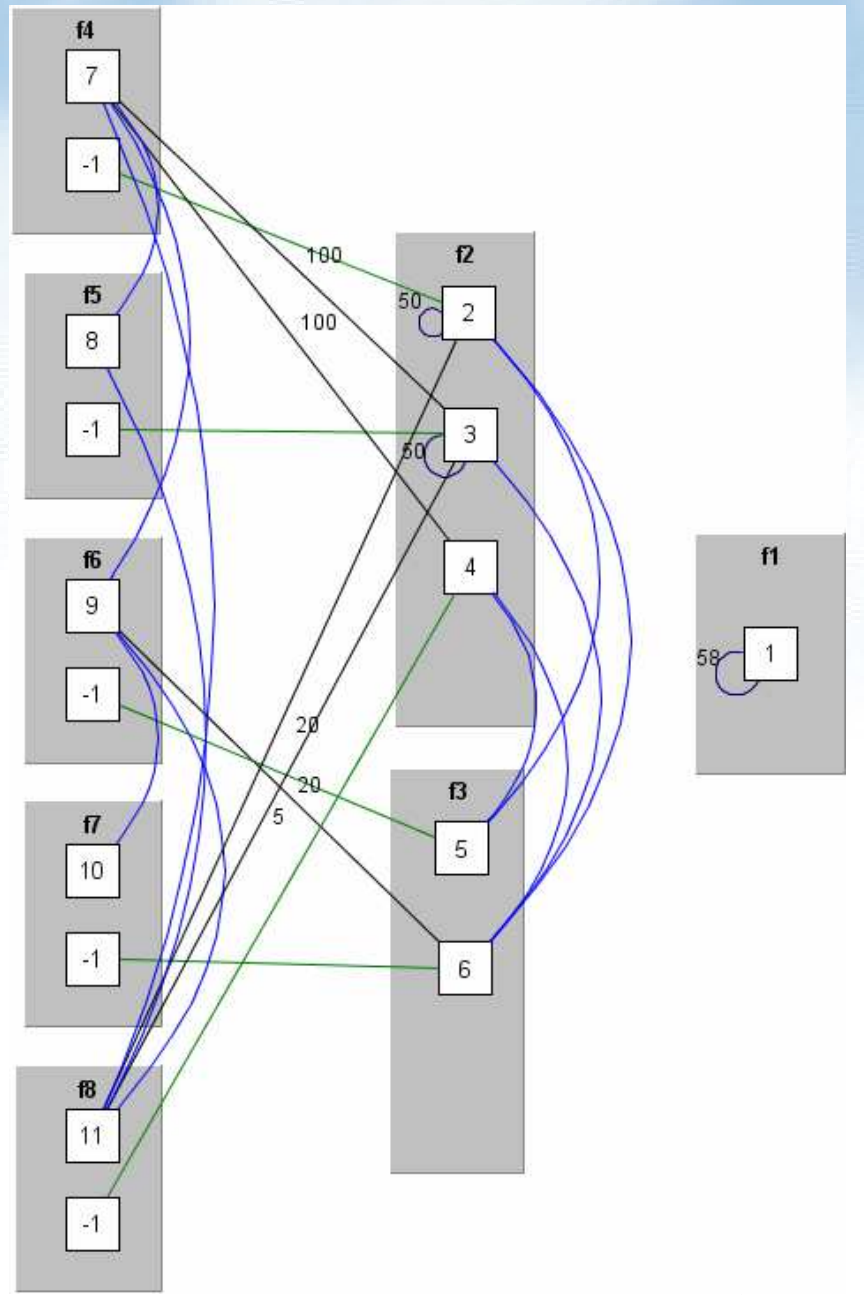
- Algorithme FDAC (Full Directionnal Arc Consistency)
  - Permet de trouver une borne inférieure du coût de la solution par application de AC puis par projection et extension des contraintes
  - Réduit le champ d'exploration des algorithmes de recherche du type branch-and-bound
  - Complexité polynomiale



1. Introduction
2. Construction d'un graphe
3. Codage d'un graphe en WCSP
4. Résolution optimale d'un WCSP
5. Résolution optimale d'un problème valué
6. Conclusions et perspectives

Résolution optimale d'un WCSP

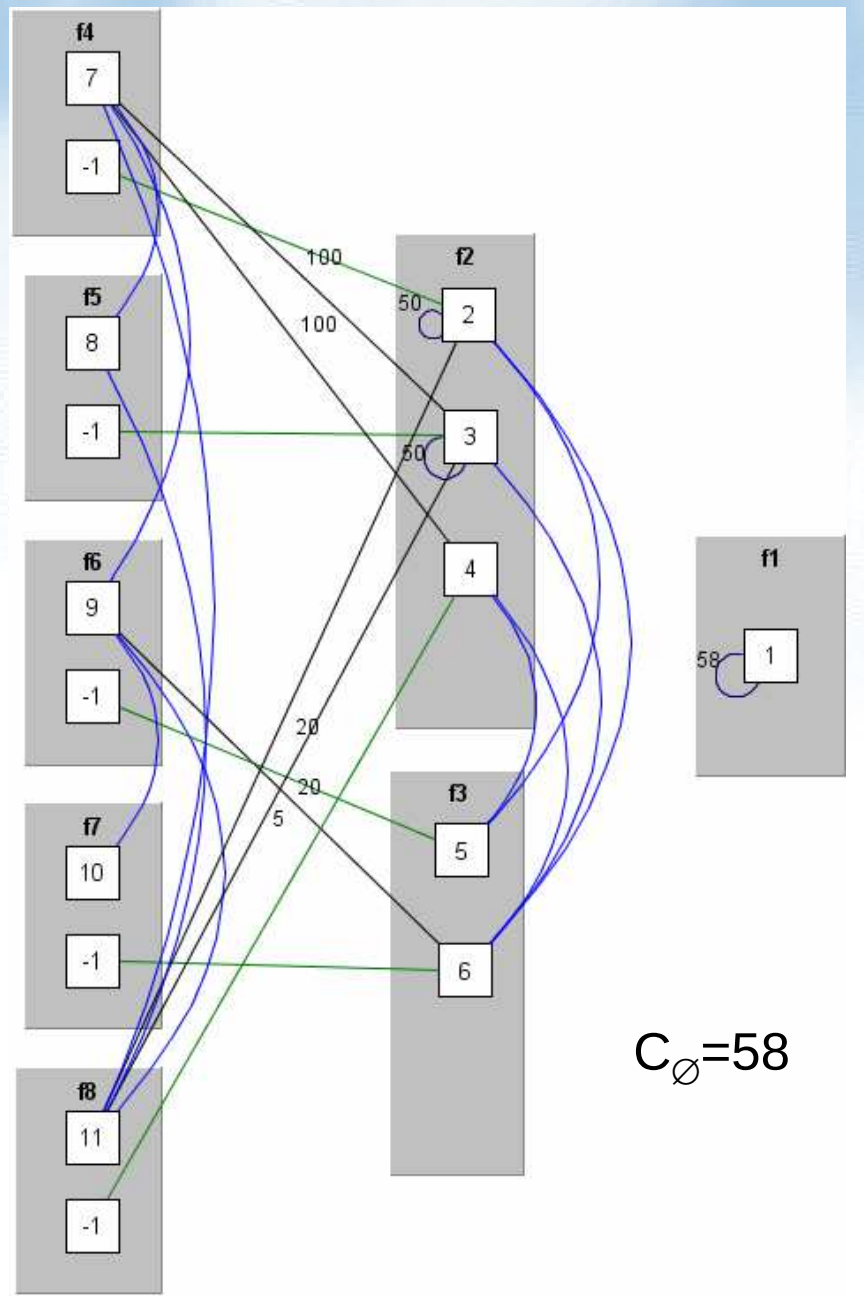
- Algorithme FDAC (Full Directionnal Arc Consistency)
  - Permet de trouver une borne inférieure du coût de la solution par application de AC puis par projection et extension des contraintes
  - Réduit le champ d'exploration des algorithmes de recherche du type branch-and-bound
  - Complexité polynomiale



1. Introduction
2. Construction d'un graphe
3. Codage d'un graphe en WCSP
4. Résolution optimale d'un WCSP
5. Résolution optimale d'un problème valué
6. Conclusions et perspectives

Résolution optimale d'un WCSP

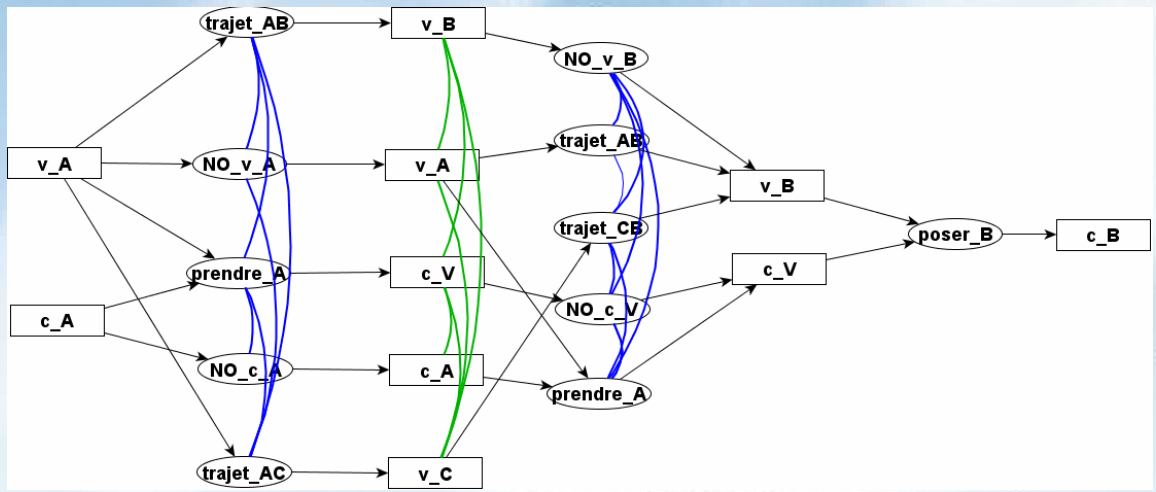
- Algorithme FDAC (Full Directionnal Arc Consistency)
  - Permet de trouver une borne inférieure du coût de la solution par application de AC puis par projection et extension des contraintes
  - Réduit le champ d'exploration des algorithmes de recherche du type branch-and-bound
  - Complexité polynomiale

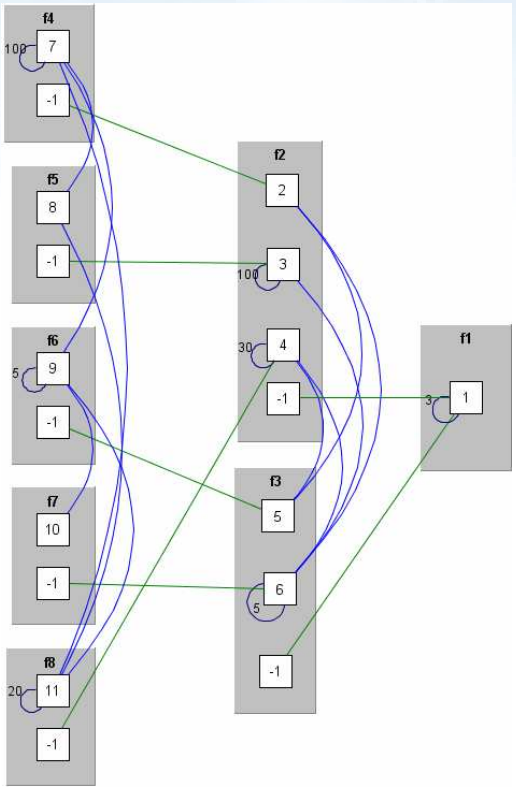


$C_{\varnothing}=58$

1. Introduction
2. Construction d'un graphe
3. Codage d'un graphe en WCSP
4. Résolution optimale d'un WCSP
5. Résolution optimale d'un problème valué
6. Conclusions et perspectives

Résolution optimale d'un WCSP





Solution :

{f1←1, f2←3, f3←5, f4←-1,
f5←8, f6←9, f7←-1,f8←-1}

Plan de coût optimal de niveau 3 :

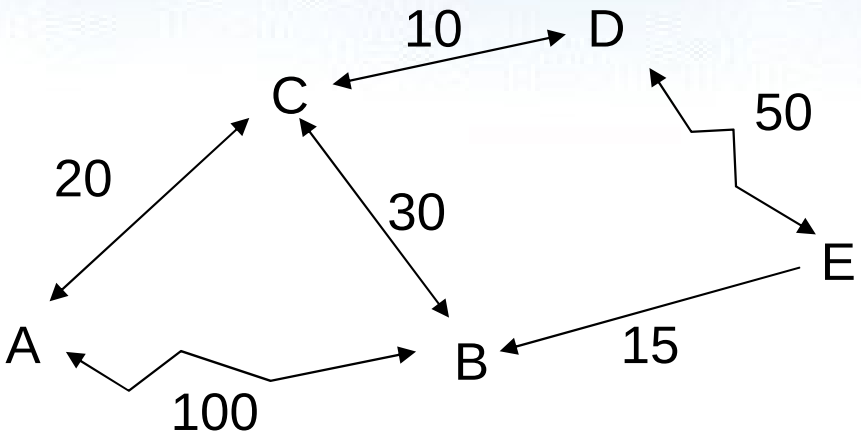
<{prendre_A}, {trajet_AB}, {poser_B}>
Coût = 108

1. Introduction
2. Construction d'un graphe
3. Codage d'un graphe en WCSP
4. Résolution optimale d'un WCSP
5. Résolution optimale d'un problème valué
6. Conclusions et perspectives

Résolution optimale d'un problème valué

- Plan-solution optimal à un niveau donné.

- Recherche d'un plan-solution optimale dans les niveaux suivants du graphe.



Plan de coût optimal de niveau 4 :
<{prendre_A}, {trajet_AC}, {trajet_CB}, {poser_B}>
Coût = 58

Résolution optimale d'un problème valué

Nombre de niveau à construire pour garantir l'obtention d'un plan-solution de coût optimal (NivMax) :

- Cout minimum des actions : $C_{min} = \min_{a \in GrapheReduit} cout(a)$
- Cout du meilleur plan obtenu au niveau k : C^*_k
- Dans le pire des cas :
  C* est un plan séquentiel composé de $(C^*_k - \epsilon)/C_{min}$ actions

$$NivMax = \lfloor (C^*_k - \epsilon)/C_{min} \rfloor = \lceil C^*_k / C_{min} \rceil - 1$$

Dans l'exemple, NivMax = 35

Expérimentations

- Très gros WCSP généré et résolu dans un temps raisonnable :

  WCSP issu d'un graphe de 20 niveaux (blocks05) :

  – Taille de l'espace de recherche : $5*10^{556}$ (10^{36} pour le plus gros WCSP aléatoire résolus en 2005).
  – moyenne géométrique de taille de domaine : 2,79 ;
  – variables : 1273 ;
  – contraintes : 19 005 ;

- Utiliser le coût du plan-solution précédemment obtenu diminue fortement ce temps de calcul :
  – 0,58 secondes avec coût du plan déjà obtenu,
  – 12 885 secondes sans utiliser ce dernier.

6. Conclusions et perspectives

- Travaux effectués :
  - Codage d'un graphe de planification en WCSP
  - Plan-solution optimal à un niveau donné
  - Plan-solution optimal grâce à un algorithme complet
- Perspectives :
  - Diminution de NivMax
  - Comparaison de la qualité des plans avec ceux produits par d'autres approches