Journée stagiaires MIAT
Toulbar2 : Hybrid Best First Search parallelization (HBFS)

Author : A. Beldjilali

Supervisors : Dr. Simon De Givry, Dr. David Allouche

Friday 21th June 2019

ENAC

Plan

Service d'accueil

- Département : Mathématiques et Informatique Appliquées (MIA)
- Unité : Mathématiques et Informatique Appliquées de Toulouse (MIAT)
- Equipe : Statistics and Algorithms for Biology (SaAB)

Service d'accueil

Hybrid Best First Search parallelization (HBFS).

Plan

Structure de valuation

Une structure de valuation V est un Quintuplet tel que :

1. Ensemble de coûts possibles : $E = \{0, ..., k\}$
2. Opérateur d'agrégation $+_k$ tel que $\forall a, b \in E, a +_k b = min(a + b, k)$
3. relation d'ordre : $<$
4. Coût min : 0 $\rightarrow$ sert à imposer une affectation donnée aux variables impliquées dans la contrainte.
5. Coût max : k $\rightarrow$ idem sauf qu'il sert à interdire un tuple de valeurs.

Fonctions de coût

Soit X un ensemble de variables et $S \subseteq X$

- S : support ou Scope de la contrainte valuée. $n = card(S) = |S|$: arité de la contrainte souple
- terminologie : Fonction de coût = contrainte valuée = contrainte souple
- Une Fonction de coûts $w_S(t)$ associe un coût ou un poids (weight) à toute combinaison (tuple) de valeurs affecté aux variables impliquées dans la contrainte.
- Cas binaire : $S = \{x_i, x_j\}, \quad w_S : D_i \times D_j \mapsto E = \{0, ..., k\}$
- Nombre de tuples = card du produit cartésien des domaines : $\prod_{i=1}^{n} |D_i|$
- $w_\varnothing$: contrainte souple d'arité nulle = coût constant quelles que soient les affectations des valeurs aux variables. Si coûts non négatifs, $w_\varnothing$ est une borne inférieure.

Fonction de coût unaire

Ensemble des coûts possibles $E = \{0, .., 5\}$ variable x de domaine {a,b,c}

Modélisation d'une forte préférence pour la valeur a

$f_{\{x\}}(a) = 0$

Modélisation d'une préférence modérée pour la valeur b

$f_{\{x\}}(b) = 3$

Contrainte dure : interdiction de c

$f_{\{x\}}(c) = 5 \iff x \neq c$

Modélisation d'un problème simple

Problème à modéliser : ensemble de variables $X = \{x_1, x_2, x_3\}$ de domaines identiques $D_1 = D_2 = D_3 = \{a, b\}$, avec $a, b \in \mathbb{N}$. La valeur a est préférée à la valeur b et nous avons la contrainte "dures" : $x_1 \neq x_2$

1. a préférée à b $\rightarrow$ E={0,...,k} suffit
2. 3 contraintes souples unaires $f_i(a) = 0$ $f_i(b) = 1$
3. $x_1 \neq x_2$: contrainte dure binaire $f_{\{x_1,x_2\}}(t1, t2) = f_{12}(t1, t2)$
4. $\prod_{i=1}^{n} |D_i| = 2 \times 2 = 4$ coûts possibles pour chaque tuple.
5. f_{12} représentable par une table 2x2

f_{12}	a	b
a	k	0
b	0	k

Réseaux de fonctions de coût

CFN : Cost Network Functions : c'est un quadruplet

1. X : ensemble de variables
2. D : ensemble des domaines des variables de X
3. W : ensemble de fonctions de coût locales
4. V : structure de valuation

Fonction de coût globale ou Valuation du WCSP

Cas arité maximale = 2

$$Val(t) = Val(t_1, ..., t_n) = w_\varnothing + \sum_{i=1}^{n} w_i(t_i) + \sum_{w_{ij} \in C}^{n} w_{ij}(t_i, t_j)$$

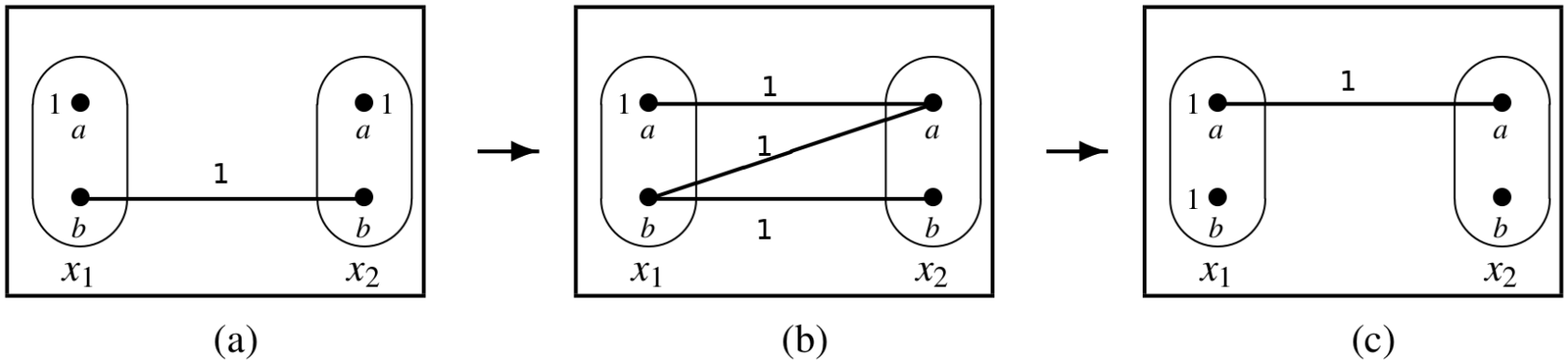
Résoudre un WCSP c'est trouver une affectation de toutes les variables qui minimise sa valuation.

Propagation des contraintes valuées

- Par projections, extensions et projections unaires, un minorant $w_\varnothing$ de la valuation est calculé.
- Heuristique pour trouver le plus grand minorant en un temps raisonnable.
- Minorant qui sera utilisé pour élaguer l'arbre de recherche lors du Branch and Bound.
- HBFS initialisé avec $w_\varnothing$
- En chaque nœud de l'arbre de recherche, une borne inférieure lb est calculée par propagation
- Si pour un nœud donné $lb \geq UB$ où UB est la meilleure solution trouvée à ce stade de la recherche, il est inutile de développer l'arbre de rechercher correspondant car il ne pourra améliorer la solution courante UB.

Propagation des contraintes valuées

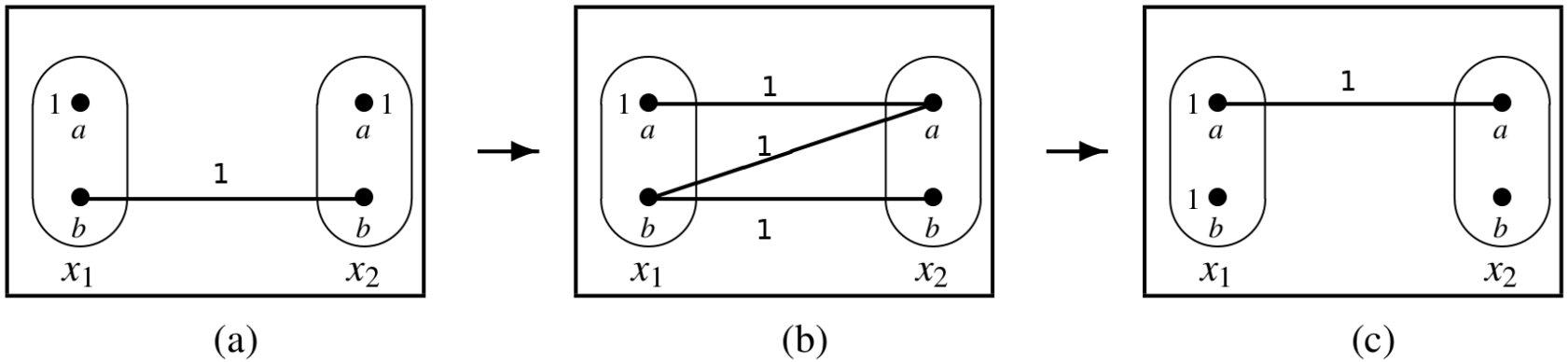
Soient deux variables x_1, x_2 avec les mêmes domaines
$D_1 = D_2 = \{a, b\}$ et $f_1(a) = f_2(a) = 1$. $f_{12}(b, b) = 1$. $E = \{0, 1\}$



$Val(t_1, t_2) = f_\varnothing + \sum_{i=1}^{n} f_i(t_i) + \sum_{f_{ij} \in C}^{n} f_{ij}(t_i, t_j)$
(a) : $Val(a, a) = f_\varnothing + f_1(a) + f_2(a) + f_{12}(a, a) = 0 + 1 + 1 + 0 = 2$

Extension

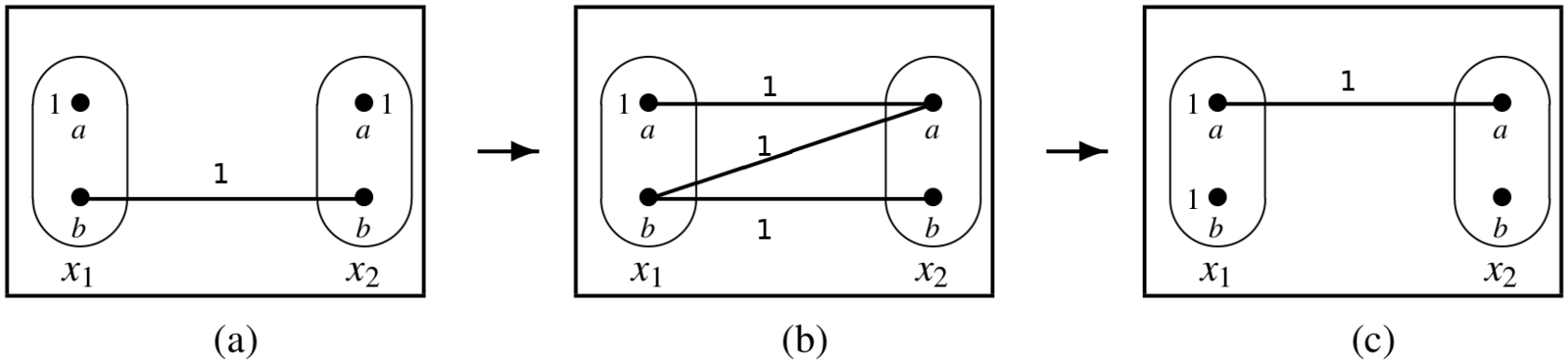
De (a) à (b), extension de $f_2(a)$ en les contraintes $f_{12}(a, a) = 1$ et $f_{12}(b, a) = 1$



(b) : $Val(a, a) = f_\varnothing + f_1(a) + f_2(a) + f_{12}(a, a) = 0 + 1 + 0 + 1 = 2$

Projection

De (b) à (c), projection sur $f_1(b)$ des contraintes $f_{12}(b, a) = 1$ et $f_{12}(b, b) = 1$

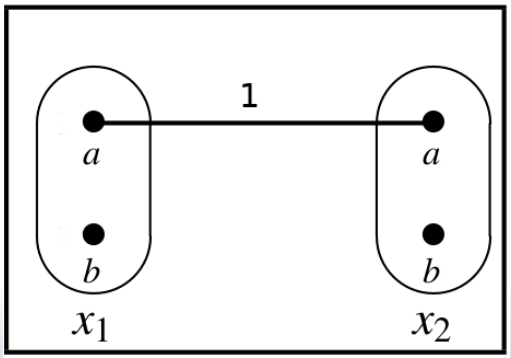


(c) : $Val(a, a) = f_\varnothing + f_1(a) + f_2(a) + f_{12}(a, a) = 0 + 1 + 0 + 1 = 2$

Projection unaire

A partir de (c), projection unaire de f_1 sur la contrainte nulle $w_\varnothing = f_\varnothing = 1$. $Val(t_1, t_2) = f_\varnothing + \sum_{i=1}^{n} f_i(t_i) + \sum_{f_{ij} \in C}^{n} f_{ij}(t_i, t_j)$

$Val(a, a) = f_\varnothing + f_1(a) + f_2(a) + f_{12}(a, a) = 1 + 0 + 0 + 1 = 2$



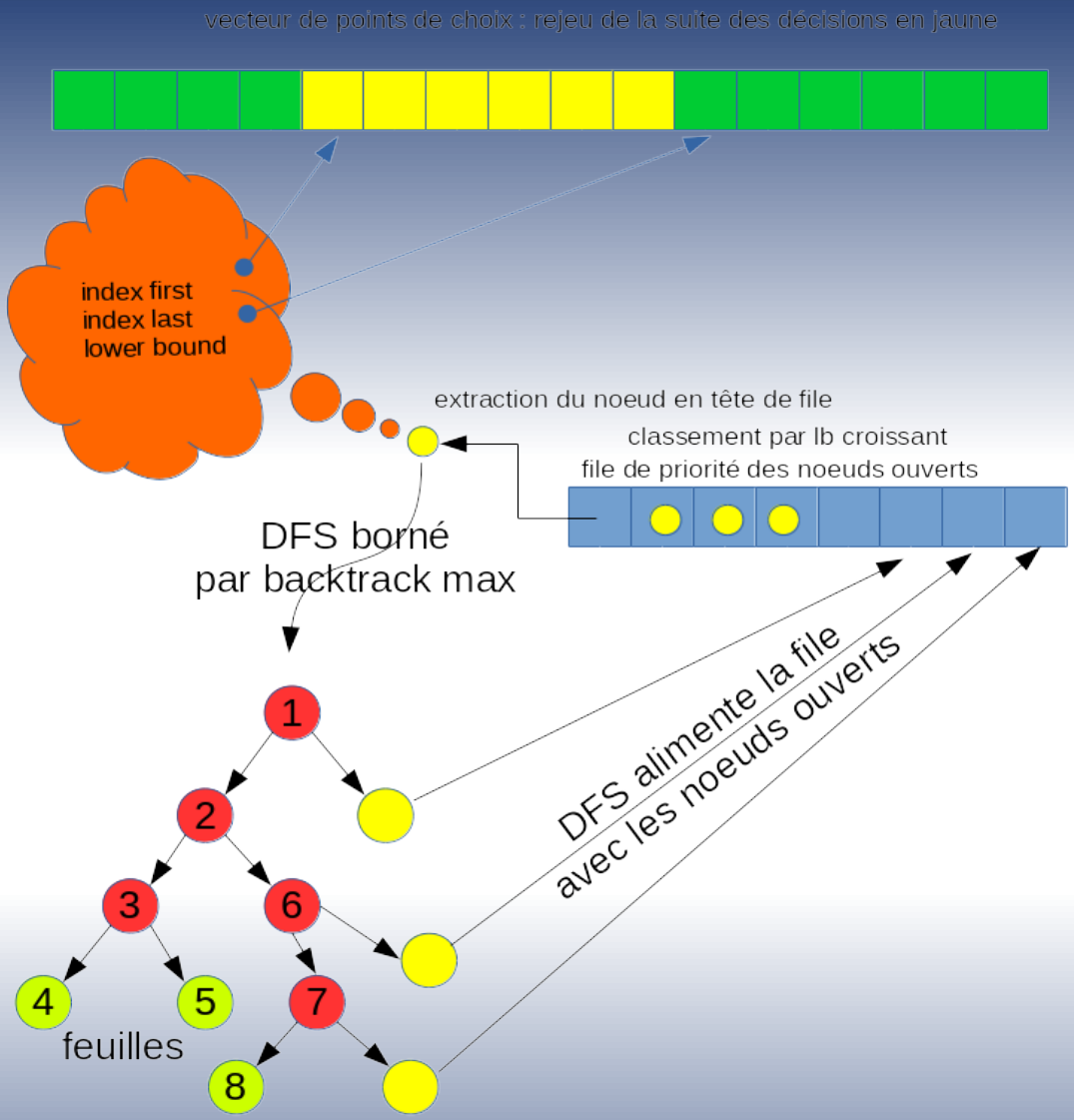
Référence principale

Martin C. Cooper, Simon de Givry, and Thomas Schiex, Valued Constraint Satisfaction Problems,

Plan

Hybrid Best First Search

- Combine Best first search et depth first search
- Initialisation : le nœud racine est placé dans la file de priorité "open" des nœuds ouverts qui représentent des sous problèmes à traiter
- le nœud est retiré de la file
- le "path" du nœud est calculé (restauration)
- Il est donné au DFS
- le DFS effectue une recherche bornée par le nombre de backtracks autorisés Z et renvoie les noeuds ouverts dans la file qui les classe par borne inférieure croissante.
- si un affectation complète est trouvée i.e une feuille de l'arbre de recherche est atteinte, la borne supérieure UB est mise à jour si elle est meilleure que la valeur courante.


vecteur de points de choix : rejeu de la suite des décisions en jaune
index first
index last
lower bound
extraction du noeud en tête de file
classement par lb croissant
file de priorité des noeuds ouverts
DFS borné
par backtrack max
1
2
3
6
4
5
7
8
feuilles
DFS alimente la file
avec les noeuds ouverts

Référence principal

Allouche, David and de Givry, Simon and Katsirelos, George and Schiex, Thomas and Zytnicki, Matthias", title="Anytime Hybrid Best-First Search with Tree Decomposition for Weighted CSP", booktitle="Principles and Practice of Constraint Programming",

Plan

Embarrassingly Parallel Search

- parallélisation naturelle
- division en sous problèmes indépendants
- env. 30 sous problèmes (SP) par cœur sur architecture 4/8
- load balancing assuré par sous problèmes suffisamment "petits"
- Utilisation de la file de priorité des noeuds ouverts de HBFS pour produire ces SP.

Benchmarks

wcsp file	Time (s) of subproblem (sp) generation	Serial Time (s)	Parallel time (s)	speed up	% CPU	obs
1PGB.11p.19aa.usingE	2,5	4,3	6,2	0,69	1422	75 sp
graph11	1,1	550,1	trop long	<1		464 sp*
capmp1	68,3	266,0	96,6	2,76	2226	172 sp – speed up total : 1,61
scen06	0,6	945,0	437,0	2,16		175 sp – speed up total : 1,82
capmo1	4,6	14,2	3,2	4,46	276	85 sp
pedigree18	0,3	340,1	trop long	<1		126 sp
pedigree7	0,9	2,4	4,5	0,54	393	82 sp
nug12	1,0	207,3	49,5	4,19	385	
nug12	0,9	207,0	35,9	5,77	510	153 sp
404.wcsp	0,4	36,7	20,9	1,8	203	731 sp -j60
404,wcsp	0,3	37,1	4,8	7,73	541	141 sp -j+0

Référence principal

Régin, Jean-Charles - Rezgui, Mohamed - Malapert, Arnaud, Chapter : Embarrassingly Parallel Search, book : Principles and Practice of Constraint Programming

Paradigme Master/workers

- un processus maître distribue les sous problèmes aux "workers" i.e. aux cœurs.
- pas d'échanges entre eux suite à l'envoi des SP
- si un worker trouve une solution, il la retourne au master

Supervisors-workers

- Le superviseur envoie le problème root au worker 1
- Le superviseur reçoit les solutions trouvées par les workers et les sous problèmes produits par ces derniers et son rangés dans une file.
- Les workers communiquent périodiquement avec le superviseur.
- le superviseur controle la taille de la file via un message indiquant aux workers qu'il est ou pas en mode collecte de sous problèmes.
- algorithme avec davantage de communications entre worker et superviseur
- Problème de goulot d'étranglement au niveau superviseur

Référence principal

- **Régin, Jean-Charles - Rezgui, Mohamed - Malapert, Arnaud, Chapter : Embarrassingly Parallel Search, book : Principles and Practice of Constraint Programming**
- **Ralphs, Ted and Shinano, Yuji and Berthold, Timo and Koch, Thorsten, Handbook of Parallel Constraint Reasoning, Parallel Solvers for Mixed Integer Linear Optimization, 2018**