Efficient Semidefinite Bounds for Multi-Label Discrete Graphical Models

Valentin Durante, Thomas Schiex, George Katsirelos

Modèles Graphiques

Idée

Description d'une fonction de plusieurs variables comme une combinaison de fonctions simples.

De manière formelle

Un modèle graphique $M = \langle V, \Phi \rangle$ definit une fonction jointe:

$$\Phi_M(v) = \bigoplus_{\theta_S \in \Phi} \theta_S(v[S])$$

- V ensemble de n variables discrètes
- Φ un ensemble de fonctions

Étude d'un modèle graphique particulier

Un réseau de fonction de coût (CFN) est un triplet $M = \langle V, C, u \rangle$. Il définit une fonction jointe:

$$C_M(v) = \sum_{c_S \in C} c_S(v[S])$$

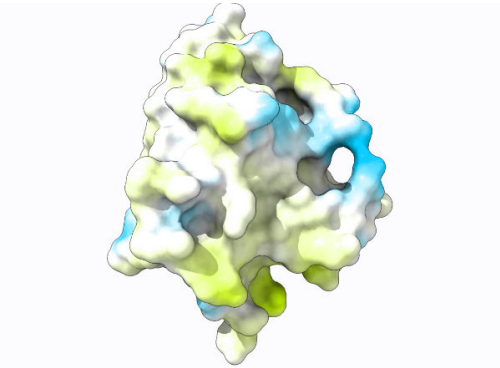
- V un ensemble de n variables discrètes
- C un ensemble de fonctions
- u un majorant de la fonction jointe

Les fonctions de coût $c_S \in C$ sont à valeurs dans $\overline{\mathbb{N}}_u$ ($\{0, +\infty\}$ pour les CSP)

Weighted Constraint Satisfaction Problem (WCSP)

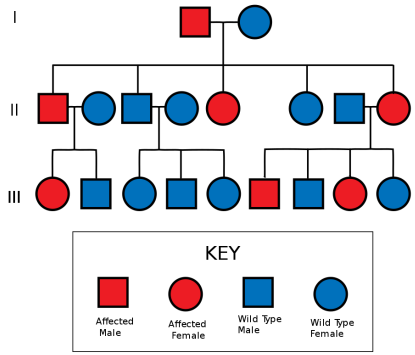
- Trouver le minimum de la fonction jointe.
- Problème NP-Difficile.

Quelles applications?



Design de protéines

Assignation de fréquences



Analyse de pédigrée, . . .

Exemple

X	$f_1(X)$
a	0
b	1

X	Y	$f_2(X,Y)$
a	a	0
b	a	1
a	b	2
b	b	1

Y	$f_3(Y)$
a	1
b	0

Minimiser $f_1(X) + f_2(X,Y) + f_3(Y)$

Travaux antérieurs — VAC, TRWS

"Polytope local" [Sch76; Kos99; Wer07] (sans la dernière ligne)

$$\min \sum_{i,a} c_i(a) \cdot x_{ia} + \sum_{\substack{c_{ij} \in C \\ a \in D_i, b \in D_j}} c_{ij}(a,b) \cdot y_{iajb} \quad \text{tel que}$$

$$\sum_{a \in D_i} x_{ia} = 1 \qquad \forall i \in \{1, \ldots, n\}$$

$$\sum_{b \in D_j} y_{iajb} = x_{ia} \qquad \forall c_{ij} \in C, \forall a \in D_i$$

$$\sum_{a \in D_i} y_{iajb} = x_{jb} \qquad \forall c_{ij} \in C, \forall b \in D_j$$

$$x_{ia} \in \{0,1\} \qquad \forall i \in \{1, \ldots, n\}$$

Le problème WCSP peut se réduire à un programme linéaire en nombres entiers (ILP).

TRWS, VAC (Virtual Arc Consistency)

Trouver une solution approchée du dual du problème ILP relaché.

Notre but

Calculer des bornes fortes pour un algorithme de Branch & Bound.

Utiliser ces bornes dans un outil d'optimisation exact (Toulbar2 [Hur+16]).

Pour cela

Utiliser la puissance de la programmation semi-definie (SDP).

$$\begin{aligned} \min\ & tr(QX) \\ \text{s.c}\ & X \in \mathscr{C} \\ & X \succeq 0 \end{aligned} \tag{1}$$

Les relaxations SDP pour les problèmes discrets sont largement étudiées depuis les travaux précurseurs de Goemans et Williamson [GW95].

Problèmes

Coûts en mémoire et nombre d'opérations élevés.

Utilisation limitée à des problèmes de taille modérée ou possédant des structures particulières.

Formulation SDP du problème WCSP

- 1-hot encoding: $x_i \rightarrow \begin{pmatrix} 0 & \dots & 0 & 1 & 0 & \dots & 0 \end{pmatrix}$.
- programme quadratique 0/1.
- encoder la contrainte de valeur unique grâce à une contrainte linéaire.

$$\begin{aligned} \min_x \ & \sum_{i,j} x_i^T c_{ij} x_j + \sum_i c_i^T x_i \\ \text{s.t } & Ax = b \\ & x \in \{0,1\}^d \end{aligned} \tag{2}$$

2 formulations SDP différentes

Par pénalisation des contraintes [Las16] nous obtenons une relaxation SDP d'un problème MAXCUT particulier.

$$\begin{aligned} \min_{x} \; & x^T C x + x^T g + \rho ||Ax - b||^2 \\ & x \in \{-1, 1\}^d \end{aligned} \tag{3}$$

$$\min_{X} \{\langle Q, X \rangle : X \succeq 0;\; X_{ii} = 1,\; i = 1, \ldots, d+1\} \tag{4}$$

Méthode de rang faible

Transformation de Burer-Montero: $X \rightarrow VV^T$ avec V une matrice de rang $r \ll d+1$.

Pour un rang r assez grand ($r > \sqrt{2 \times m}$) les optimums sont les mêmes.

- (4) Résolu par une méthode de descente en coordonnées [WCK17].

Mixing method

Résolution de SDP avec contraintes sur la diagonale: $X_{ii} = 1 \, \forall i$.

Descente en coordonnées: raisonner sur les colonnes de la matrice V.

Contraintes sur les colonnes de V, $||V_i|| = 1$.

$$\min_{V \in \mathbb{R}^{(d+1) \times r}} \langle Q, VV^T \rangle \text{ s.c } ||V_i|| = 1 \, , \, i = 1, \ldots, d+1 \tag{5}$$

LR-BCD

$$\min_X \langle R, X \rangle \text{ s.c } \begin{cases} \text{diag}(X) = 1_{d+1} \\ \langle F_i, X \rangle = 2 - d_i, i = 1, \ldots, N \\ X \succeq 0 \end{cases}$$

Profite de la structure particulière de la matrice de coût R.

$$R = \begin{pmatrix} 0^{d_1 \times d_1} & & \\ & \ddots & \\ & & 0^{d_n \times d_n} \\ & & & \alpha \end{pmatrix}$$

Transformation de Burer-Montero:

$$\min \sum_{i=s_k}^{s_k+d_k-1} V_i^T g_i \ \text{ s.t.} \begin{cases} V_{d+1}^T \left(\sum_{j=s_k}^{s_k+d_k-1} V_j \right) = 2 - d_k \\ \|V_i\| = 1,\ s_k \leq i < s_{k+1} \end{cases} \tag{6}$$

Descente par blocs: raisonner sur des blocs de colonnes de la matrice V.

Méthode de rounding

1. Construire un vecteur unaire aléatoire V_r.
2. Mettre à 1 la valeur qui correspond à l'indice du vecteur qui maximise $V_r^T V_i$.

Le rounding de la solution SDP permet de produire une solution entière réalisable pour le problème discret.

Une simple méthode de recherche greedy pour améliorer la solution.

Résultats

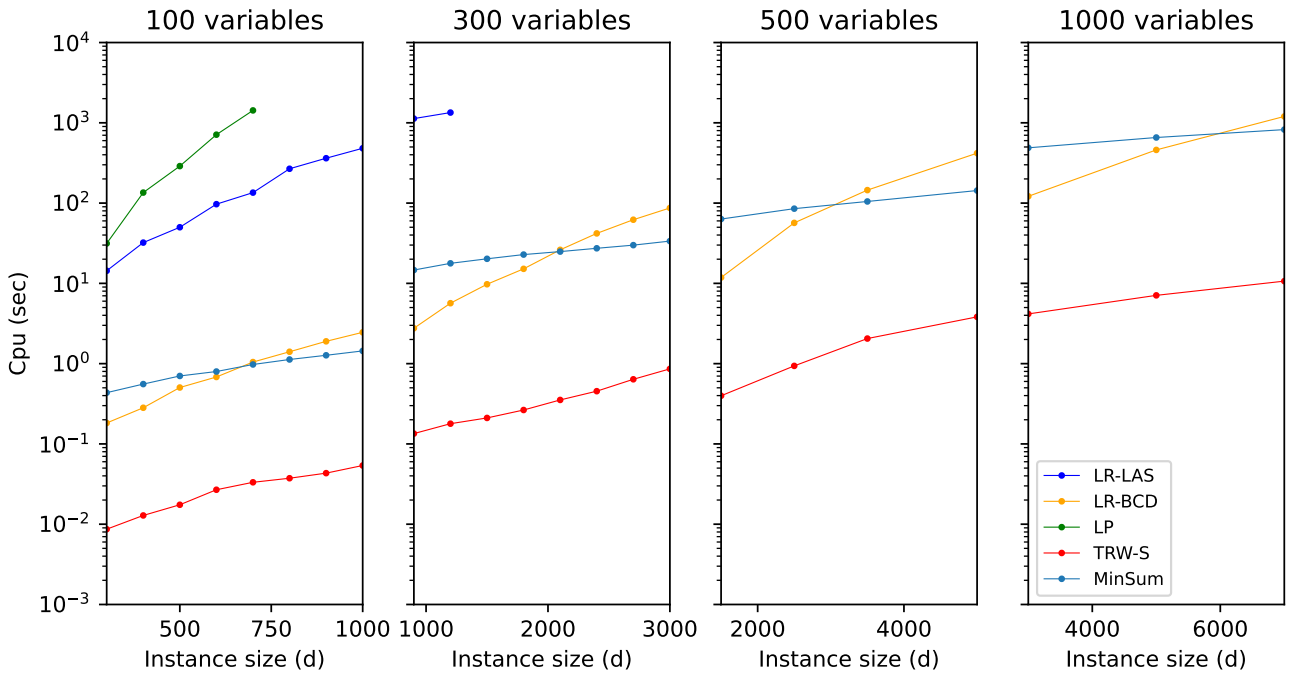


Figure: Temps cpu pour les 4 solvers sur des problèmes de taille croissante (100, 300, 500 et 1000 variables) fonction de la taille des problèmes (nombre d'états × nombre de variables).

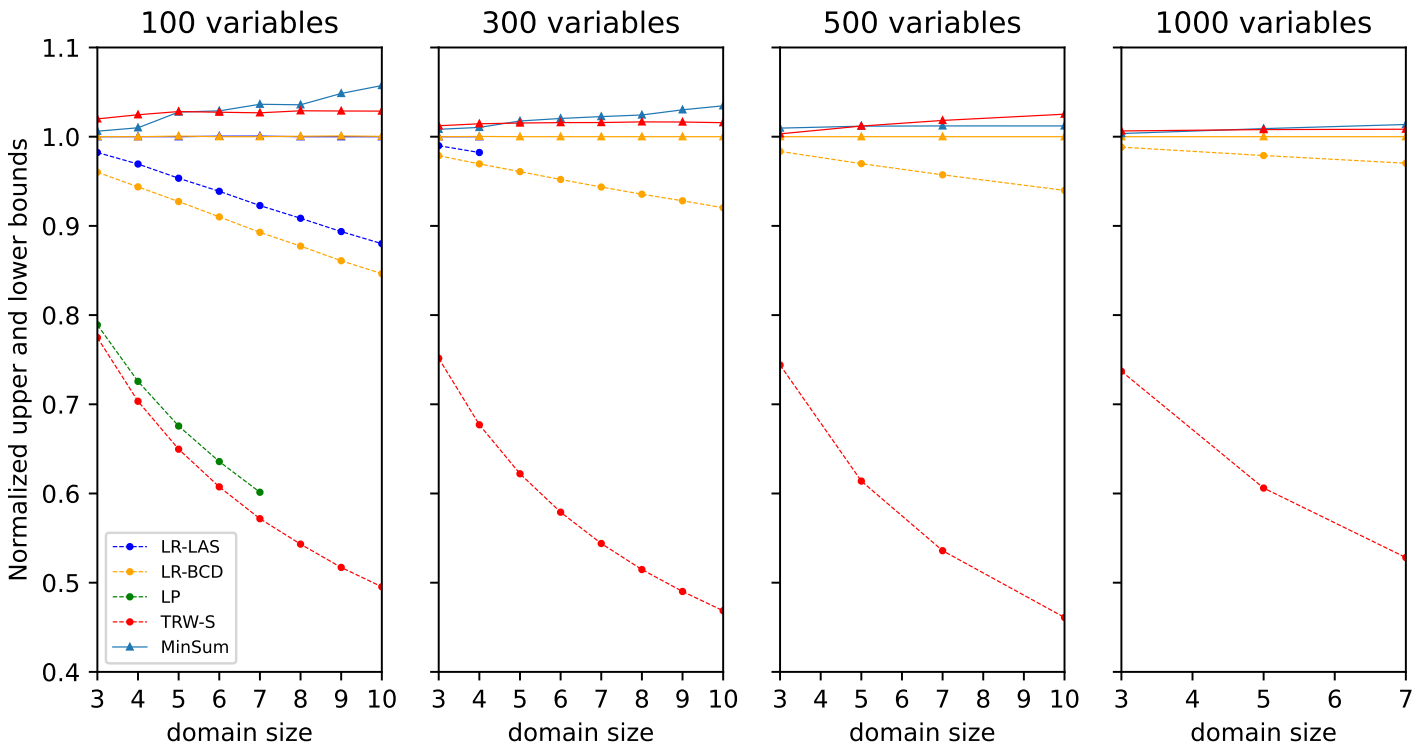


Figure: Majorants et minorants normalisés sur des problèmes de taille croissante pour les 4 solvers (100, 300, 500 et 1000 variables) fonction du nombre d'états (domain size). La valeur du coût de la meilleure solution entière est prise comme référence.

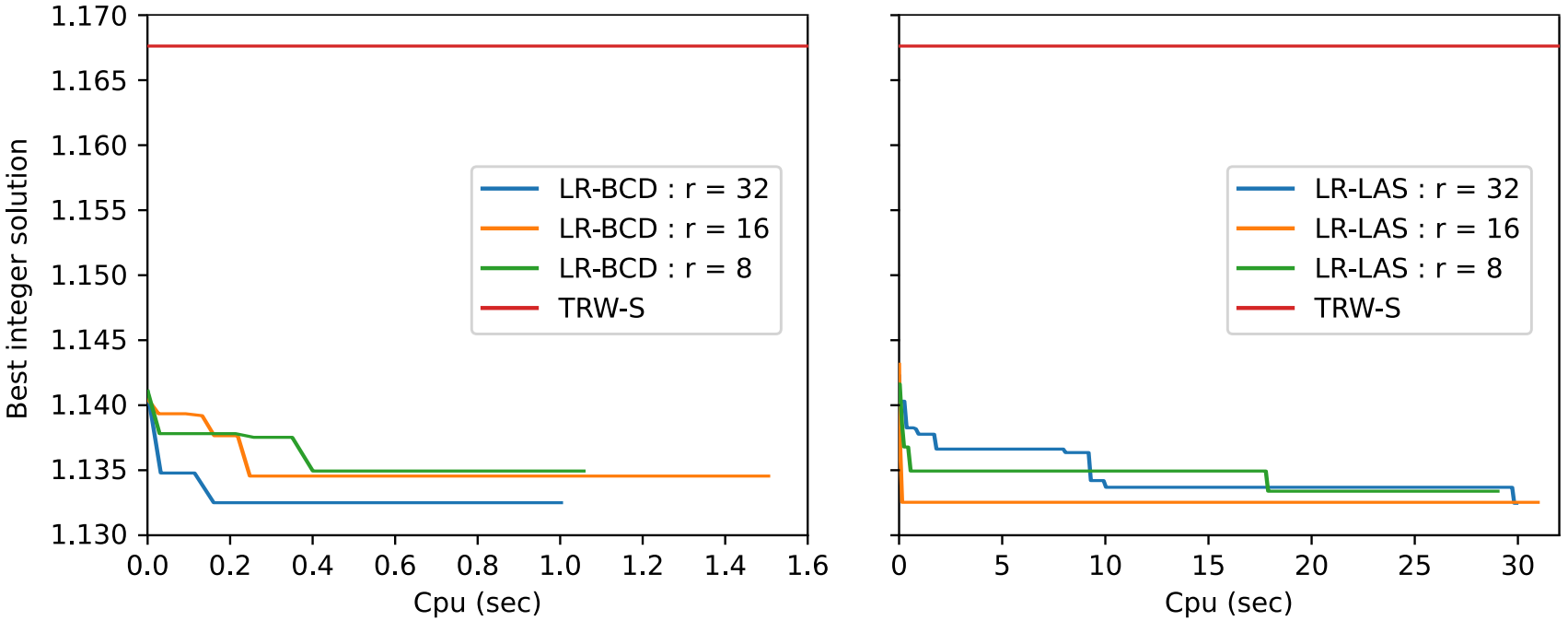


Figure: Comparaison de la meilleur solution entière fonction du temps entre TRW-S et LR-BCD (gauche) et TRW-S et LR-LAS (droite) sur une instance de 100 variables avec 5 états.

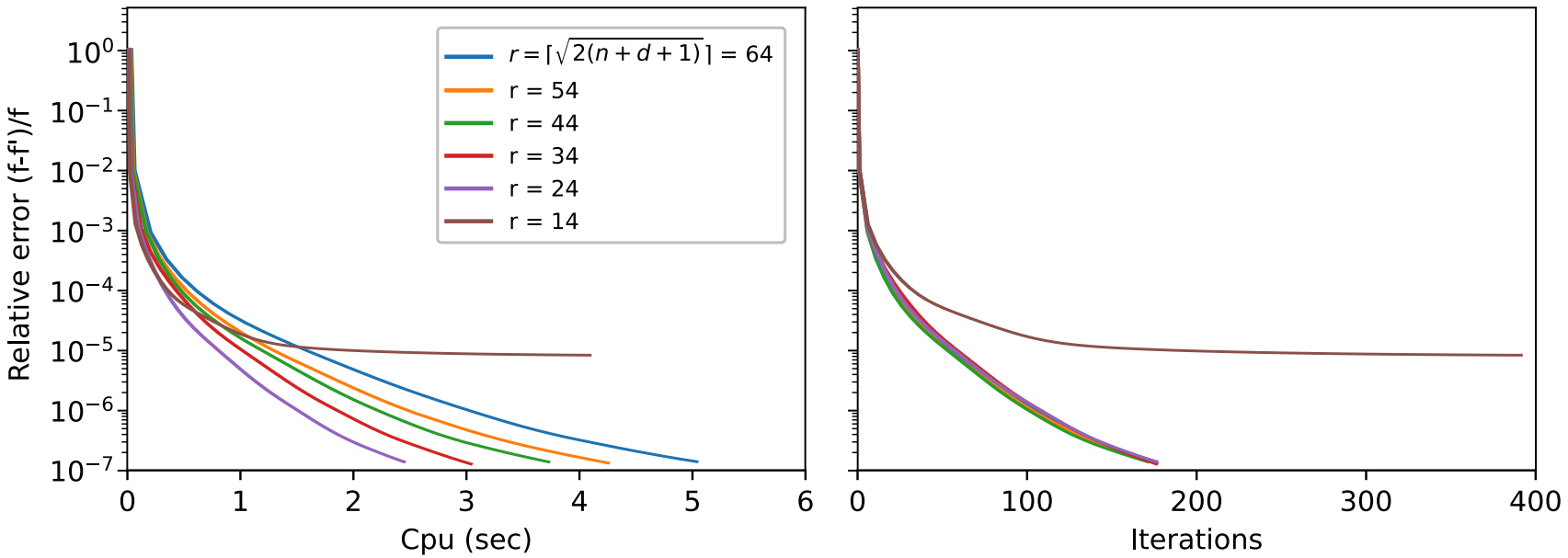


Figure: Erreur relative avec une solution de bonne qualité vs. temps cpu (gauche) et nombre d'iterations (mise à jour complète de la matrice, droite) en utilisant des rangs différents pour la relaxation. Le test a été fait sur une instance de 500 variables avec 3 états ($d = 1500$) et est représentatif du comportement sur tout le benchmark.

Merci pour votre attention!