# Recherche adaptative de structures de régulation génétique

Mohamed Elati[*,**], Céline Rouveirol[*]

[*]LIPN — CNRS UMR 7030, Université Paris 13
99, av. J-B Clément, F-93430 Villetaneuse
prenom.nom@lipn.univ-paris13.fr
[**]Institut Curie, CNRS UMR 144
26 rue d'Ulm F-75248 Paris

**Résumé.** Nous avons proposé un algorithme original de Fouille de Données, LICORN, afin d'inférer des relations de régulation coopérative à partir de données d'expression. LICORN donne de bons résultats s'il est appliqué à des données de levure, mais le passage à l'échelle sur des données plus complexes (e.g., humaines) est difficile. Dans cet article, nous proposons une extension de LICORN afin qu'il puisse gérer une contrainte de co-régulation *adaptative*. Une évaluation préliminaire sur des données de transcriptome de tumeurs de vessie montre que les réseaux significatifs sont obtenus à l'aide d'une contrainte de corégulation adaptative de manière beaucoup plus efficace, et qu'ils ont des performances de prédiction équivalentes voire meilleures que celles obtenues par LICORN.

## 1 Introduction

Un des principaux objectifs de la biologie moléculaire consiste à comprendre la régulation des gènes d'un organisme vivant dans des contextes biologiques spécifiques. Les facteurs de transcription sont les régulateurs de la transcription qui vont réagir avec les promoteurs de la transcription des gènes cibles. Les techniques récentes d'analyse du transcriptome, telles que les puces à ADN permettent de mesurer simultanément les niveaux d'expression de plusieurs milliers de gènes. Nous avons déjà décrit le système LICORN (Elati et al., 2007a) qui se fonde sur un modèle de régulation locale coopérative : chaque gène peut être régulé par un ensemble des coactivateurs et/ou un ensemble de coinhibiteurs, ces corégulateurs agissent collectivement pour influencer leur(s) gène(s) cible(s). LICORN met en œuvre une approche originale de Fouille de Données afin d'inférer des relations de régulation coopérative à partir de données d'expression. Cet algorithme a été évalué avec succès sur des données publiques de transcriptome de levure. L'application de LICORN sur des données de transcriptome humaines est plus complexe, car le nombre de régulateurs connus est plus important, et nécessite un temps de calcul considérable. En effet, les gènes de faible support vont avoir un nombre très élevé de régulateurs candidats. Nous proposons dans ce travail d'étendre LICORN pour qu'il puisse traiter une contrainte de sélection de corégulateurs candidats adaptative pour chaque gène, prenant en compte le support du gène cible et bornant le nombre de corégulateurs candidats possibles. La suite de cet article est organisée comme suit. Dans la section 2, nous introduisons brièvement le principe de LICORN. Dans la section 3, nous détaillons l'extension de LICORN à la

recherche adaptative. Nous exposons des résultats préliminaires concernant l'application de cette approche à des données de cancer de vessie (section 4) et enfin, nous concluons sur les perspectives ouvertes par ce travail.

## 2 LICORN : Algorithme d'apprentissage

Étant donnés :
- un ensemble de régulateurs $\mathcal{R}$ et un autre de gènes cibles $\mathcal{G}$
- deux matrices d'expression discrétisées $(\text{MG}, \text{MR})$ à valeurs dans $\varepsilon = \{-1, 0, 1\}$, codant les états de $\mathcal{R}$ et $\mathcal{G}$ pour un échantillon $S$ de $n$ conditions : $S = s_1, \ldots, s_n$
- un programme de régulation RP : $\varepsilon \times \varepsilon \rightarrow \varepsilon$, qui associe à un état d'un ensemble d'activateurs $\subseteq \mathcal{R}$ et à un état d'un ensemble d'inhibiteurs $\subseteq \mathcal{R}$ un état du gène cible[1]
- une fonction de score local $h : \varepsilon^n \times \varepsilon^n \rightarrow \mathbb{R}$, permettant d'évaluer un réseau local de régulation,

notre but est de trouver, pour chaque gène cible, l'ensemble de régulateurs qui explique au mieux son niveau d'expression, i.e. le graphe qui minimise la différence entre la valeur d'expression prédite et la valeur d'expression observée pour un gène cible. Pour le résoudre, nous avons formalisé le problème d'inférence de réseaux de régulation comme un problème de recherche dans un espace d'états muni d'une relation de spécialisation, des motifs satisfaisant un ensemble de contraintes anti-monotones (Agrawal et al., 1993). Ceci nous a permis d'optimiser le parcours de l'espace de recherche et de proposer une méthode originale LICORN, capable d'apprendre des réseaux de régulation coopérative. Cet algorithme décompose la tâche d'apprentissage de structures en trois étapes indépendantes. Nous présentons par la suite ces trois étapes, déjà décrites dans Elati et al. (2007a). Dans la suite, chaque gène ou régulateur $g$ est représenté par deux ensembles supports $\subseteq S$, son 1-support $\mathcal{S}_1(g)$ et son $-1$-support $\mathcal{S}_{-1}(g)$. LICORN construit d'abord, en utilisant une extension de l'algorithme *Apriori* (Agrawal et al., 1993) qui manipule en parallèle les deux supports (1 et $-1$-supports), le treillis CL des ensembles de corégulateurs fréquents. Un corégulateur est fréquent s'il est fréquent pour au moins une des valeurs d'intérêt, ici 1 ou -1.

Puis, LICORN calcule pour chaque gène cible $g$ un ensemble de co-régulateurs candidats, de taille réduite par rapport à celle de tous les sous-ensembles possibles de régulateurs. Le critère pour qu'un ensemble de corégulateurs puisse participer au programme de régulation d'un gène cible est d'observer dans les données d'expression une covariation significative de leurs niveaux d'expression. La contrainte de corégulation teste la taille de l'intersection entre les supports du gène et ceux du corégulateur.

**Définition 1 (Contrainte de corégulation)** *Soit une combinaison de co-régulateurs $C$, un gène cible $g$, et leurs supports respectifs $\mathcal{S}_x(C)$ et $\mathcal{S}_y(g)$ pour les états $x, y \in \{-1, 1\}$. $C$ co-régule le gène $g$, noté $C_{coreg}(\mathcal{S}_x(C), \mathcal{S}_y(g))$ si et seulement si $\frac{|\mathcal{S}_y(g) \cap \mathcal{S}_x(C)|}{|\mathcal{S}_y(g)|} \geq S_{coreg}$, où $S_{coreg}$ est un seuil fixé par l'utilisateur.*

Nous distinguons les complexes de régulateurs satisfaisant $C_{coreg}$ pour chaque gène cible selon leur mode de régulation : les complexes d'activateurs candidats $\mathcal{A}(g)$ co-varient positivement

[1] Ce programme de régulation permet de calculer l'expression prédite pour un gène dans un échantillon, étant donnés les états de ses activateurs et de ses inhibiteurs.

avec le gène $g$, en revanche, les complexes d'inhibiteurs candidats $\mathcal{I}(g)$ co-varient négativement avec le gène cible :

$$\mathcal{A}(g) = \{A \in \text{CL} \mid C_{coreg}(\mathcal{S}_x(A), \mathcal{S}_x(g)); x \in \{-1, 1\}\}$$
$$\mathcal{I}(g) = \{I \in \text{CL} \mid C_{coreg}(\mathcal{S}_x(I), \mathcal{S}_{-x}(g)); x \in \{-1, 1\}\}$$

Nous avons décrit dans (Elati et al., 2007a) comment nous exploitons l'anti-monotonie de $C_{coreg}$ pour effectuer une recherche efficace dans le treillis des régulateurs candidats. Ainsi, nous pouvons calculer l'ensemble de tous les graphes de régulation candidats pour chaque gène cible $g$, à partir des complexes d'activateurs et d'inhibiteurs extraits comme suit :

$$\mathcal{C}(g) = \{(A, I) | A \in \mathcal{A}(g), I \in \mathcal{I}(g) \text{ and } A \cap I = \emptyset\}$$

Un GRN candidat pour un gène cible $g$, ou tout simplement un GRN, est un élement de $\mathcal{C}(g)$.

Enfin, à partir de chaque $\mathcal{C}(g)$, LICORN extrait le réseau de régulation (GRN) pour chaque gène cible qui explique au mieux les données d'expression, à l'aide d'une fonction de score (l'erreur absolue moyenne) qui mesure la capacité des régulateurs à prédire le niveau d'expression du gène cible $g$. Ensuite, LICORN sélectionne les GRNs statistiquement significatifs en se fondant sur une méthode (non décrite ici) à base de permutations.

# 3 Génération adaptative de corégulateurs candidats

L'algorithme précédent de génération de corégulateurs candidats ne fournit pas de mécanisme pour limiter le nombre de corégulateurs candidats pour un gène cible. Ceci mène parfois à obtenir trop ou trop peu de corégulateurs candidats, et ainsi à un temps excessif de calcul ou à une faible performance. La plupart des algorithmes classiques d'extraction de motifs fréquents ou contraints ne permettent pas d'encadrer *a priori* le nombre souhaité de résultats. L'extraction des k-meilleurs motifs fréquents a été introduite dans (Fu et al., 2000). Notre objectif ici, est d'extraire pour chaque gène cible, s'il existe, un ensemble de taille modeste de meilleurs corégulateurs au sens de la contrainte de corégulation. Nous proposons, étant donné un intervalle sur le nombre de corégulateurs candidats souhaités, de concevoir un algorithme itératif qui effectue une recherche adaptative des meilleurs corégulateurs candidats.

Cet algorithme, noté AdapSearch (algorithme 1), ajuste le seuil de la contrainte de corégulation pendant la phase d'extraction des corégulateurs candidats, afin d'obtenir un nombre approprié de corégulateurs candidats pour chaque gène cible. Etant donné le treillis de corégulateurs fréquents (voir section 2), un gène cible $g$ et un intervalle $[minReg, maxReg]$ encadrant le nombre de corégulateurs candidats pour $g$, AdapSearch initialise le seuil de corégulation $S_{coreg}$ à $50\%$ et appelle GenCoreg (lignes 1 à 3 de l'algorithme 1).

La procédure GenCoreg (non décrite) effectue une recherche par niveau sur ce treillis de corégulateurs, à partir des *corégulateurs* les plus généraux jusqu'à atteindre les *corégulateurs* les plus spécifiques (i.e., maximaux au sens de l'inclusion) qui satisfont la contrainte de corégulation. Notons que GenCoreg termine le processus de génération dès que le nombre de corégulateurs candidats dépasse $maxReg$. Quand le résultat de GenCoreg est retourné, AdapSearch vérifie d'abord si le nombre des corégulateurs candidats dépasse $maxReg$. Si c'est le cas (ligne 6), c'est-à-dire que le seuil de corégulation est trop bas, AdapSearch augmente

**Algorithme 1** AdapSearch : recherche adaptative de corégulateurs candidats.

**ENTRÉES:** $g$, CL, $minReg$, $maxReg$
**SORTIES:** $R(g)$
1: **Début**
2: $S_{coreg}$ := 0.5
3: $R(g)$ := GenCoreg ($g$,CL,$S_{coreg}$)
4: **TantQue** (not $(minReg \leq |R(g)| \leq maxReg)$ && $(0.3 \leqslant \gamma \leqslant 1)$) **Faire**
5: **Si** $(|R(g)| > maxReg)$ **Alors**
6: $S_{coreg}$ := $S_{coreg}$ + $\gamma$
7: R(g) := AdapMax($g$,$R(g)$,CL, $minReg$, $S_{coreg}$)
8: **FinSi**
9: **Si** $(minReg > |R(g)|)$ **Alors**
10: $S_{coreg}$ := $S_{coreg}$ - $\gamma$
11: R(g) := AdapMin($g$,$R(g)$,CL, $maxReg$, $S_{coreg}$)
12: **FinSi**
13: **Fin TantQue**
14: **Retourner** $R(g)$
15: **Fin**

le seuil de corégulation (algorithme 2, ligne 1) : $S_{coreg}$ est incrémenté d'un pas fixe $\gamma$ et la procédure GenCoreg est appellée itérativement jusqu'à ce que le nombre de corégulateurs soit inférieur à $maxReg$ ou que $S_{coreg}$ dépasse la valeur maximum, i.e., 1. Si l'augmentation du seuil engendre un ensemble de corégulateurs candidats dont la taille est inférieure à $minReg$ (ligne 7 de l'algorithme 2), alors qu'il était supérieur à $minReg$ dans l'itération précédente (ligne 5 de l'algorithme 2), alors l'algorithme oscille entre deux valeurs de $S_{coreg}$ et AdapSearch retourne alors les meilleurs $maxReg$ corégulateurs candidats parmi les deux ensembles de régulateurs résultats.

**Algorithme 2** AdapMax : Adaptation du seuil de corégulation, si $|R(g)| > maxReg$

**ENTRÉES:** $g$, $R(g)$, CL, $minReg$, $S_{coreg}$
**SORTIES:** $R(g)$
1: tmp := $R(g)$
2: $R(g)$ := GenCoreg($g$, CL, $S_{coreg}$)
3: **Si** $(minReg > |R(g)|)$ **Alors**
4: $R(g)$ := maxReg meilleurs corégulateurs de ($R(g) \cup tmp$)
5: **FinSi**
6: **Retourner** $R(g)$

Dualement, AdapSearch vérifie si le nombre de corégulateurs est inférieur à $minReg$. Si c'est le cas, on décrémente le seuil de corégulation de $\gamma$ jusqu'à ce que le nombre de corégulateurs candidats soit supérieur ou égal à $minReg$ ou que $S_{coreg}$ atteigne un seuil minimum (par exemple $30\%$). Dans ce dernier cas, on garantit que les corégulateurs ont un minimum de recouvrement avec le gène cible.

# 4 Expérimentations : LICORN versus LICORN adaptatif

Dans cette étude, nous utilisons des données humaines de 85 échantillons. Ces échantillons se divisent en cinq échantillons normaux (sujets dépourvus de cancer), et quatre-vingts échantillons de carcinomes de vessie à différents stades de la maladie. Nous nous intéressons à trois tendances de variation du niveau d'expression : dans un échantillon tumoral donné, le niveau d'expression d'un gène peut *augmenter* par rapport à son niveau *normal* (noté par 1 dans la matrice discrétisée) ; il peut *diminuer* par rapport à son niveau normal (noté par -1) ; ou encore être *stable* (noté par 0). Nous renvoyons à (Elati et al., 2007b) pour une description de la technique de discrétisation utilisée. Nous obtenons une matrice discrétisée de 8000 gènes et 699 facteurs de transcription.

Dans la partie gauche de la figure 1, nous présentons la distribution de nombre de corégulateurs candidats générés par LICORN avec $S_{coreg}$ fixé à $60\%$. La valeur médiane du nombre de corégulateurs candidats est 1024 et la valeur maximale est 35000. Il y aura donc un sous-ensemble des gènes, souvent de faible fréquence, pour lequel LICORN génère plusieurs milliers de corégulateurs candidats. Ces résultats nous montrent que l'utilisation de LICORN adaptatif avec un intervalle de nombre de corégulateurs candidats autour de la médiane va réduire nettement le volume de corégulateurs candidats engendré, ce qui représente un gain considérable en terme de temps de calcul.

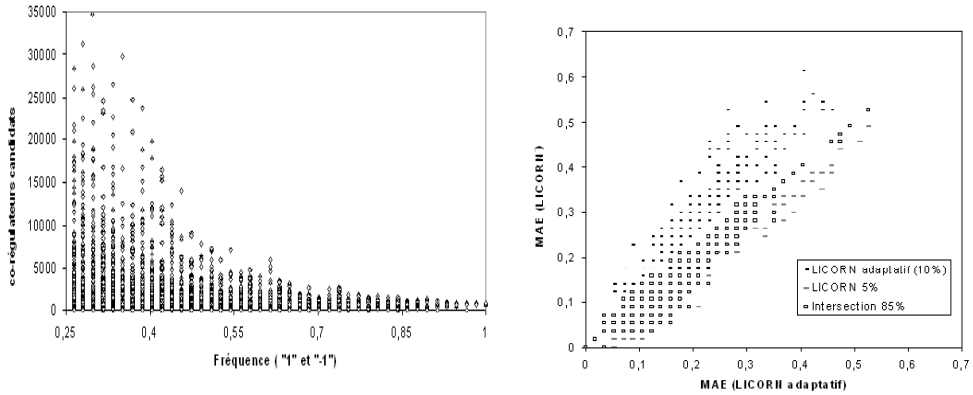


**FIG. 1** – *Partie gauche : chaque point dans le graphique est un gène cible avec sa fréquence d'expression et le nombre de ses corégulateurs candidats générés par* LICORN *(*$S_{coreg} = 60\%$*, voir définition 2). Partie droite : comparaison des GRNs appris par* LICORN *(*$S_{coreg} = 0.6$*) à ceux appris par* LICORN *adaptatif (*$\gamma$ *= 0.1,* $minReg$ *= 200 et* $maxReg$ *= 2000)*

Dans la deuxième phase de validation, nous avons lancé LICORN adaptatif avec les valeurs suivantes de paramètres : $\gamma$ = 0.1, valeur minimum de $S_{coreg}$ = $30\%$ et un intervalle de corégulateurs candidats de $[200, 2000]$. LICORN adaptatif a obtenu des résultats équivalents voire meilleurs que LICORN (voir figure 1, partie droite) : i) $85\%$ de GRNs sont détectés comme significatifs avec un FDR de 5% (voir Elati et al. (2007a)) par les deux méthodes avec $92\%$ d'interactions régulateur/gène en commun. ii) $5\%$ de GRNs détectés par LICORN seulement contre $10\%$ de GRNs détectés par LICORN adaptatif seulement.

# 5 Conclusion

Afin d'améliorer notre algorithme en vue de l'appliquer sur des données humaines de cancer, nous l'avons étendu pour qu'il puisse traiter une contrainte de corégulation adaptative pour chaque gène, prenant en compte le support du gène cible et en bornant le nombre de corégulateurs candidats possibles. Cette recherche adaptative a rendu le calcul plus efficace, tout en engendrant des réseaux dont les performances sont dans une grande majorité des cas équivalentes voire meilleures. Enfin, des structures plus adéquates peuvent être utilisées ou imaginées pour améliorer les performances de la recherche adaptative, par exemple la structure de COFI-tree récemment utilisée par Ngan et al. (2005) pour optimiser l'extraction de k-meilleurs motifs.

**Remerciements**

Nous tenons à remercier tout particulièrement F. Radvanyi, de l'Institut Curie pour son soutien constant au cours de notre collaboration, ainsi que tous les membres de son équipe Oncologie Moléculaire.

# Références

Agrawal, R., T. Imielinski, et A. Swami (1993). Mining association rules between sets of items in large databases. In *Proceedings of the International Conference on Management of Data*, pp. 207–216.

Elati, M., P. Neuvial, M. Bolotin-Fukuhara, E. Barillot, F. Radvanyi, et C. Rouveirol (2007a). LICORN : learning cooperative regulation networks from gene expression data. *Bioinformatics 23*, 2407–2414.

Elati, M., C. Rouveirol, et F. Radvanyi (2007b). Fouille de données pour l'extraction de grands réseaux de régulation génétique. *Technique et Science Informatiques 26*, 173–196.

Fu, A. W.-C., R. Kwong, et J. Tang (2000). Mining n-most interesting itemsets. In *ISMIS*, pp. 59–67.

Ngan, S.-C., T. Lam, R. C.-W. Wong, et A. W.-C. Fu (2005). Mining n-most interesting itemsets without support threshold by the cofi-tree. *Int. J. Business Intelligence and Data Mining 1*, 88–106.

# Summary

We have previously proposed an original Data Mining algorithm LICORN, that infers cooperative regulation relations from expression datasets. LICORN has been successfully applied to public Yeast datasets, but running it on more complex datasets (e.g., human) is problematic. In order to overcome this limitation, we propose here an adaptive coregulation constraint that takes into account the support of the gene to look for a bound number of candidate regulators. Preliminary experiments on human bladder cancer data have shown that the adaptive constraint allows LICORN to learn much more efficiently regulation relations that show a similar (if not better) prediction performance.