MajecSTIC 2009
Avignon, France, du 16 au 18 novembre 2009

Algorithme de décodage généralisé par propagation de croyances

Jean-Christophe Sibel

Laboratoire ETIS UMR 8051-CNRS, ENSEA - Université de Cergy-Pontoise, 6 avenue du Ponceau, 95014 Cergy-Pontoise Cedex - France.

Contact : jean-christophe.sibel@ensea.fr

Résumé
Les communications numériques reposent sur un trio de transmission : émetteur, canal, récepteur. Le canal est l'objet de perturbations physiques qui induisent des erreurs dans le message transmis. Afin de se prémunir contre ces erreurs, on utilise un code correcteur d'erreurs qui consiste au niveau de l'émetteur à ajouter de la redondance au message. Au niveau du récepteur, l'opération de décodage permet alors de récuperer l'information utile. Un des algorithmes de décodage les plus répandus est l'algorithme de propagation de croyances, de type itératif, réputé sous-optimal pour une certaine catégorie de codes, en raison de l'existence de cycles dans le graphe de ces codes. Nous présentons dans cet article une généralisation de la propagation de croyance qui permet de rendre le décodage optimal y compris en présence de cycles, offrant ainsi une meilleure récupération de l'information. Nous décrivons la construction de l'algorithme par analogie avec une technique de physique statistique appliquée à un réseau de spins, puis nous en faisons le lien avec les représentations graphiques des codes LDPC.

Abstract
Digital communications are composed of a transmitter, a channel and a receiver. Errors in the transmitted message come from physical disruptions in the channel. The use of error correcting codes enables to protect information from these errors, it consists in the addition of redundancy in the message before its entrance in the channel. At the receiver, information is got back thanks to a decoding. One of the most widespread decoding algorithms is the Belief Propagation, known for its suboptimal property concerning a particular class of codes whose graphs contain cycles. This article deals with an extension of this algorithm which could make the decoding optimal whatever the cycles would be, so as to get more reliable information. The construction of such an algorithm is based on a technique which comes from statistical mechanics in a spins lattice, that is introduced with an application on LDPC codes.

Mots-clés : décodage, codes LDPC, algorithme itératif, cycles, énergie

Keywords: decoding, LDPC codes, iterative algorithm, cycles, energy.

Introduction

La théorie de Shannon permet aujourd'hui de construire des protections robustes, les codes, contre le bruit du canal dans la transmission de bits d'information. Ces codes sont assez sophistiqués pour exiger un décodage complexe en vue de récupérer l'information. Les algorithmes itératifs modernes de décodage utilisent la représentation des codes en graphe, le plus souvent ceux de Tanner, dont les branches sont le support de transmission de l'information sous forme de messages et dont les noeuds sont les noeuds de variables et noeuds de parité du code. L'idée principale est d'itérer suffisamment de fois la propagation des messages pour faire converger le graphe vers un point fixe qui est l'information utile transmise. Cependant, cette représentation a l'inconvénient majeur de faire apparaître des structures topologiques néfastes réduisant l'efficacité du décodage. L'algorithme de propagation de croyances généralisé vise à absorber ces structures dans le but de récupérer une information plus fidèle. Nous étudions dans un premier

temps l'algorithme fondamental de propagation de croyances ainsi que ses performances après avoir rapidement expliqué ce qu'est un code, les recherches portent ensuite sur les problèmes topologiques, ainsi qu'à la solution proposée.

1. Les codes LDPC

Selon [1], pour protéger l'information dans la transmission, l'idée principale est de plonger l'espace informatif dans un espace de plus grande dimension, dit espace du code. L'information, représentée en bits, devient ainsi redondante telle qu'un mot de taille k devient un mot de taille N où les $N-k$ bits supplémentaires sont les bits de redondance. On représente cette redondance par $N-k$ équations de parité agissant sur les N bits de la transmission, regroupées dans une matice H de vérification de parité de taille $N \times (N-k)$. Un mot de taille N est un mot de code si et seulement s'il appartient au noyau de H.
Un code LDPC a la particularité d'avoir une matrice H de faible densité (LDPC $\equiv$ Low Density Parity Check), autrement dit, le nombre de 1 est très inférieur au nombre de 0, propriété très importante pour le décodage (voir section 3). Voici un exemple simple de code LDPC dit régulier, le nombre d_c de 1 par ligne et le nombre d_v de 1 par colonne étant fixes, noté $(N = 6, d_v = 2, d_c = 3)$:

$$\mathcal{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} \text{ de densité } \rho = 1 - \frac{(N-d_c)d_v}{N(N-k)} = 0.5$$

2. Graphe de Tanner

Les algorithmes de décodage reposent sur une représentation en graphe du code, dit graphe de Tanner, incluant les deux structures essentielles : N bits représentés par N noeuds de variable, M équations de parité représentés par M noeuds de parité. Ces deux types de noeuds sont reliés par des branches selon la règle suivante : une branche joint le noeud de parité E_i et le noeud de variable x_j si et seulement si $H_{ij} = 1$. Le graphe de Tanner du code donné dans la section 1 est présenté figure 1.

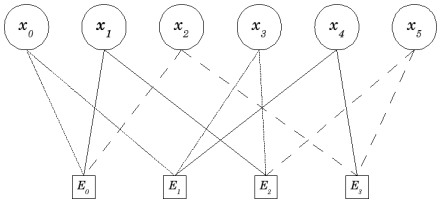
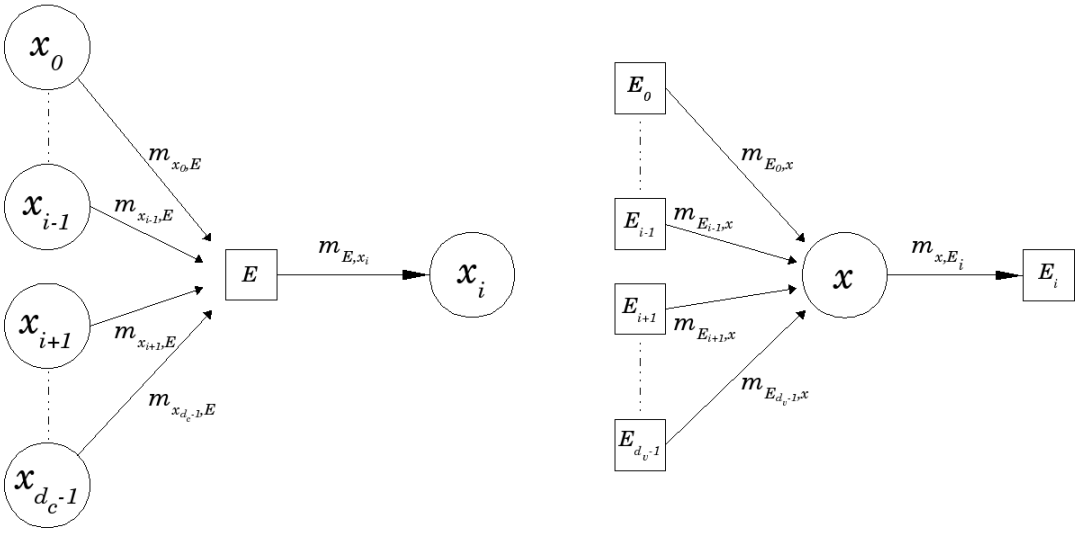


FIG. 1 – Graphe de Tanner du code LDPC $(N = 6, d_v = 2, d_c = 3)$

3. Décodage par propagation de croyances

La branche partant d'un noeud quelconque i et arrivant sur un noeud quelconque j est un message transportant une probabilité sur le noeud j conditionnellement au noeud i et ses voisins. Ces messages sont interprétés comme des propagations d'informations de noeud en noeud, où ces informations, appelées croyances, traduisent la pensée majoritaire sur un noeud selon tous ses voisins, d'après [2] (voir figure 2).
Le décodeur fonctionne de manière itérative, c'est-à-dire que les quantités calculées dans l'algorithme à l'itération (i) dépendent directement de ces mêmes quantités à l'itération précédente $(i-1)$, selon les équations ci-dessous :



(a) Noeud de parité → noeud de variable (b) Noeud de variable → noeud de parité

FIG. 2 – Mise à jour des messages

$$m^{(i)}_{x_k \to E} = f_2(m^{(i-1)}_{E \to x_i})$$
$$m^{(i)}_{E \to x_i} = f_1(m^{(i)}_{x_k \to E})$$

D'après [3], ces équations se définissent comme suit :

$$m_{x \to E_i} \propto \prod_{k \neq i}^{d_v - 1} m_{E_k \to x} \tag{1}$$

$$m_{E \to x_i} \propto \sum_{x_0} \dots \sum_{x_{i-1}} \sum_{x_{i+1}} \dots \sum_{x_{d_c}-1} \left(\prod_{k \neq i}^{d_c - 1} m_{x_k \to E} \right) \left(\sum_{k=0}^{d_c - 1} x_k \right)) \tag{2}$$

Remarque : Les sommes sont faites modulo 2 car on travaille dans le corps de Gallois $GF(2)$.

Le but du décodage est, rappelons-le, de retrouver les valeurs des bits ou noeuds de variable en sortie du canal de transmission. Au fur et à mesure des propagations dans le graphe, les valeurs des noeuds de variable convergent vers un point fixe qui est le mot de code. Dans le cas idéal (canal non perturbatif), le mot en sortie de l'algorithme est exactement un mot de code correspondant au mot en entrée du canal, mais en pratique, le canal peut fortement perturber le signal entrant, et l'algorithme peut ne pas converger vers un mot de code : le taux d'erreurs binaire (rapport du nombre de bits en erreur sur la taille du mot) dépend directement du rapport signal à bruit (voir figure 3).
A première vue, il suffirait alors de travailler à de forts rapports signal à bruit pour obtenir de bons décodages, mais cette solution n'est pas optimale. Premièrement, un fort rapport signal à bruit implique un coût matériel élevé, deuxièmement, il existe d'autres causes de dégradations des performances de décodage bien plus difficiles à contrer (voir section 4).

Remarque : D'un point de vue algorithmique, d'après [1], la propagation de croyances traverse toutes les branches un même nombre de fois, ce qui permet d'affirmer que le nombre d'opérations est linéaire en le nombre de noeuds. Ainsi, plus le code est creux (à faible densité), plus le nombre de branche diminue ce qui rend le décodage plus rapide. Il existe cependant d'autres algorithmes plus rapides mais moins précis,

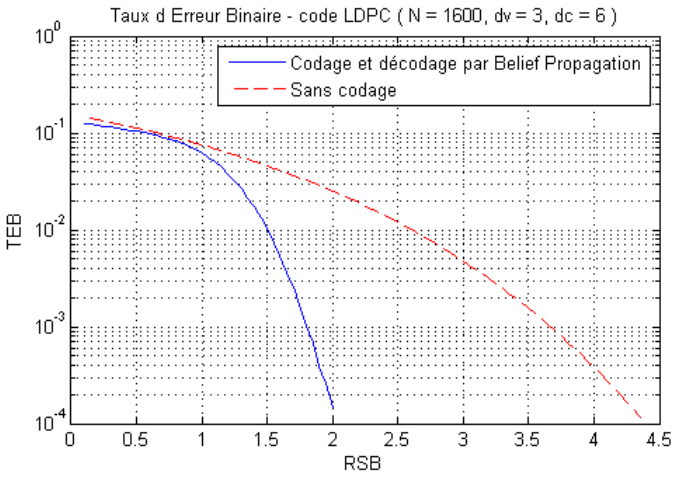


FIG. 3 – Taux d'erreur binaire du code LDPC ($N = 1600, d_v = 3, d_c = 6$) décodé par propagation de croyances

tels que les algorithmes de décodage à états finis (algorithme de Gallager-B). On retrouve ici le compromis complexité/temps de calcul qui est un des enjeux du décodage.

4. Cycles et trapping sets

Dans l'exemple de la section 2, le graphe présente des structures topologiques néfastes appelées cycles.

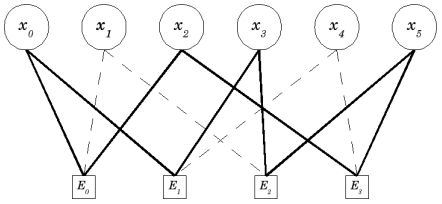


FIG. 4 – Cycles dans le code LDPC ($N = 6, d_v = 2, d_c = 3$)

Ces cycles sont tels que les messages bouclent sur eux-mêmes. Dans un premier cas de figure, ce bouclage peut faire osciller des croyances entre plusieurs valeurs sans jamais les faire converger vers un point fixe. Par conséquent, le mot de sortie a une très forte probabilité de ne pas être un mot de code, quelque soit le nombre d'itérations de l'algorithme. Dans un second cas de figure, ces croyances convergent vers des valeurs fausses donnant lieu à des faux mots de codes, dits pseudo-mots de code, dégradant également les performances du décodage. Le taux d'erreurs binaire devient trop élevé pour des valeurs de rapport signal à bruit encore trop importantes. Pour contrer ces effets, l'idée est de passer par une approche de la physique statistique.

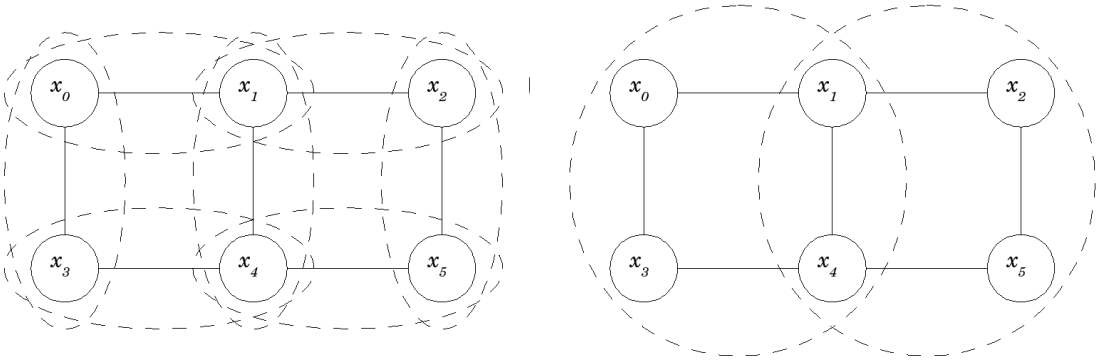
5. Physique statistique et codage

On peut modéliser des mots de code par un réseau de spins plongé dans un champ magnétique uniforme, où chaque spin représente un noeud de variable du graphe, les noeuds de parité eux ne sont pas représentés car physiquement, ce sont des interactions. L'état de chaque spin prend ses

valeurs dans $\{-1,+1\}$, correspondant simplement à une modulation BPSK [1], et les états de tous les spins du réseau déterminent l'énergie globale E du réseau. En effet, selon [4], l'énergie du réseau se décompose linéairement selon chaque spin i avec deux types d'énergies : l'énergie E_i échangée entre le spin i et le champ magnétique, l'énergie E_{ij} échangée entre le spin i et le spin j du réseau.

$$E = \sum_i E_i + \sum_i \sum_j E_{ij} \tag{3}$$

D'après [3], on peut déduire de cette équation les mises à jour des messages notées à la section 3. Cependant, ce modèle, appelé approximation de Bethe, n'est pas rigoureusement exact. En effet, si un nombre $N_S \geq 3$ de spins sont liés par des interactions, ne considérer que les interactions par paires revient à supposer la paire indépendante des $N_S - 2$ autres spins. Autrement dit, ce modèle ne considère pas d'interactions d'ordre supérieur à 2. Par conséquent, lorsqu'on transpose ce modèle dans le graphe d'un code, les structures de types cycles comprenant plus de deux noeuds de variable ne sont pas gérées correctement, il est nécessaire de considérer une approche plus fine. Une approximation, dite de Kikuchi, plus rigoureuse consiste à considérer des interactions d'ordre 4.



(a) Modèle de Bethe (b) Modèle de Kikuchi

FIG. 5 – Approximations de l'énergie d'un réseau de spins

Dans cette approximation, chaque groupe de 4 spins possède une énergie propre, ce qui permet d'assimiler un groupe à une entité élémentaire d'un nouveau réseau (voir figure 5). En poursuivant l'approximation de Kikuchi, on peut finalement considérer tout type de groupe de toute taille et gérer dans le graphe du code les structures cycliques de plus grandes tailles. Cette nouvelle gestion repose sur l'élaboration d'un autre graphe, le graphe des régions (voir section 6).

6. Le graphe des régions

Ce graphe regroupe les noeuds de variable en interaction, c'est-à-dire les noeuds de variables appartenant à une même équation de parité. Chaque équation de parité du code constitue ainsi ce qu'on appelle une région, correspondant dans la version physique statistique à un groupe de spins de taille maximale. Certains noeuds de variables étant présents dans plusieurs équations, ils forment d'autres interactions constituant de nouvelles intersections entre les régions que l'on apelle des groupes. La construction se poursuit donc en cherchant ces interactions entre les régions, ceci jusqu'à tomber sur des groupes de noeuds élémentaires formés donc d'un seul noeud de variable. Une région R_G d'un groupe G dont les noeuds de variables sont l'ensemble $V_G = \{v_0, ..., v_j | j = \mathtt{card}(V)\}$ contient tous les groupes ascendants dont les noeuds de variables

[1] Binary Phase Shift Keying

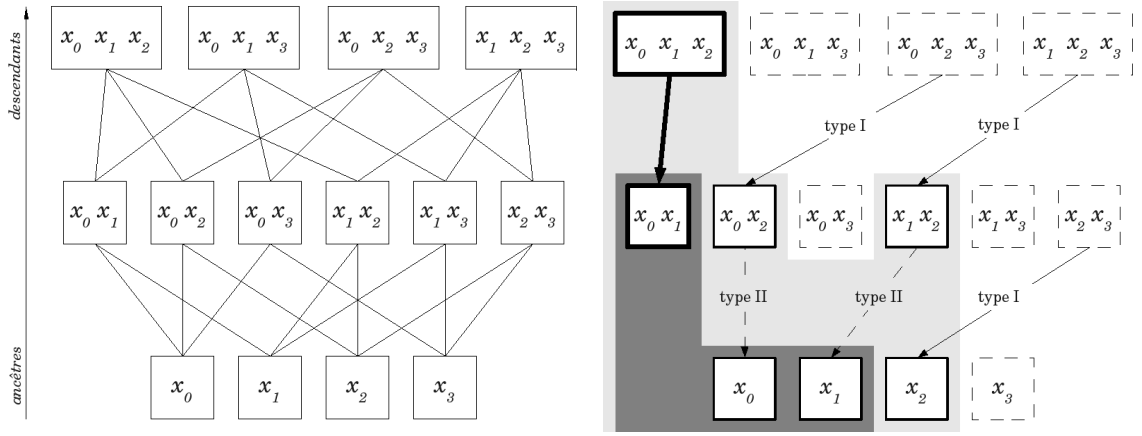
sont des sous-ensembles de V. La figure 6 (a) montre un exemple de graphe des régions pour le code $(N = 4, d_v = 3, d_c = 3)$ dont la matrice est

$$\mathcal{H} = \begin{pmatrix} \boxed{1} & \boxed{1} & 1 & 0 \\ \boxed{1} & \boxed{1} & 0 & 1 \\ 1 & 0 & \boxed{1} & \boxed{1} \\ 0 & 1 & \boxed{1} & \boxed{1} \end{pmatrix}$$

Les éléments de H encadrés représentent deux intersections de taille 2 :
– $\{x_0, x_1\}$ pour les lignes 0 et 1 ;
– $\{x_2, x_3\}$ pour les lignes 2 et 3.

Il existe au total, pour ce code, 6 intersections de taille 2 dont celles citées ci-dessus et les suivantes :
– $\{x_0, x_2\}$ pour les lignes 0 et 2 ;
– $\{x_0, x_3\}$ pour les lignes 1 et 2 ;
– $\{x_1, x_2\}$ pour les lignes 0 et 3 ;
– $\{x_1, x_3\}$ pour les lignes 1 et 3.

Remarque : Ce code n'est pas exploitable en pratique en raison de sa très forte densité, mais le graphe des régions associé est complet, ce qui permet une bonne compréhension.



(a) Graphe des régions

(b) Mise à jour du message $m_{\{x_0,x_1,x_2\}\to\{x_0,x_1\}}$ dans le graphe des régions

FIG. 6 – Graphe des régions et mise à jour d'un message

Algorithme - Message
La mise à jour des messages est calculée d'après [3], que l'on peut interpréter comme un algorithme de parcours du graphe des régions. Le calcul est tel qu'on ne considère plus de messages bidirectionnels comme dans la propagation de croyances simple (voir section[4]) mais des messages unidirectionnels se propageant à l'intérieur des régions dans le sens ascendant (des enfants vers les parents). Pour le message allant d'un groupe G_E ($\{x_0, x_1, x_2\}$ sur la figure 6 (b)) vers un groupe G_P ($\{x_0, x_1\}$ sur la figure), le principe de l'algorithme est de considérer tous les messages de type I entrant dans la région R_{G_E} de G_E (en gris clair sur la figure) et les messages de type II entrant dans la région R_{G_P} de G_P (en gris foncé sur la figure). On différencie deux types de messages car l'action sur la mise à jour n'est pas la même (voir Algorithme 1).

Algorithme - Croyance
Le but du décodeur est de trouver les croyances sur les noeuds de variables élémentaires, dont le calcul est expliqué dans [2]. De manière synthétique, il suffit, pour un groupe G quelconque du graphe, de trouver tous les messages entrant dans la région de G (voir Algorithme 2).

Algorithme 1 : Mise à jour du message de l'enfant E vers le parent P

début
  $R_{G_E} \longleftarrow$ Région de l'enfant
  $R_{G_P} \longleftarrow$ Région du parent
  $M = 1 \longleftarrow$ valeur d'initialisation du message
  pour chaque *niveau* n *du graphe* **faire**
    pour chaque *groupe* g *du niveau* n **faire**
      si $g \in R_{G_E} \backslash R_{G_P}$ **alors**
        Message de type I
        pour chaque *enfant* e_g *du groupe* g **faire**
          si $e_g \notin R_{G_E}$ **alors**
            $M = M \times m_{e_g \to g}$
      si $g \in R_{G_P}$ **alors**
        Message de type II
        pour chaque *enfant* e_g *du groupe* g **faire**
          si $e_g \in R_{G_E} \backslash R_{G_P}$ **alors**
            $M = \frac{M}{m_{e_g \to g}}$
fin

Algorithme 2 : Calcul de la croyance sur une région R

début
  $R \longleftarrow$ Région du groupe considéré
  pour chaque *niveau* n *du graphe* **faire**
    pour chaque *groupe* g *du niveau* **faire**
      si $g \notin R$ **alors**
        pour chaque *parent* p_g *de* g **faire**
          si $p_g \in R$ **alors**
            $M = M \times m_{g \to p_g}$
fin

Pour quelques codes, comme le précédent, la mise à jour est simple, mais pour des codes de plus en plus creux (de densité de plus en plus faible), le graphe s'élague et devient irrégulier. En effet, certains groupes peuvent ne pas avoir d'enfants tandis que d'autres du même niveau en ont. C'est le cas pour le code $(N = 10, d_v = 3, d_c = 6)$:

$$\mathcal{H} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

On observe en effet une première intersection de taille 4 entre les équations de parité des lignes 0 et 3 : $\{x_0, x_2, x_4, x_5\}$, et une seconde intersection de taille 4 entre les équations de parité des lignes 1 et 2 : $\{x_1, x_7, x_8, x_9\}$. Ce sont les deux seules intersections de taille maximale, on ne peut pas trouver d'intersection de taille plus grande. L'irrégularité vient du fait que ces deux intersections ne contiennent pas à elles-deux les noeuds de variables tels que $\{x_3, x_6\}$. Ainsi, au niveau inférieur où les intersections sont de taille 3, les groupes contenant les noeuds de variables x_3 et x_6 n'ont pas d'enfants, contrairement à tous les autres groupes ne contenant pas ces noeuds (voir figure 7). En revanche, certaines irrégularités sont impossibles. On ne peut pas trouver de trous dans le graphe, c'est-à-dire une liaison entre deux groupes qui ont un lien de parenté de degré supérieur à 2 (grand-parent et petit-enfant sans parent entre eux). La raison est que s'il existe une intersection $\{a_0, ..., a_{L-1} | L \leq d_c\}$ alors par inclusion, toute intersection $\{a_0, ..., a_{l-1} | l \leq L\}$ existe nécessairement. Les irrégularités sont strictement des espaces vides à partir du niveau de dernière descendance.
Dans la construction du graphe des régions, il est ainsi plus judicieux de commencer par les noeuds de plus faible dimension, et de poursuivre vers les noeuds de plus grande dimension. On pourra ainsi déterminer les performances de l'algorithme en fonction de la hauteur du graphe puisque la hauteur détermine le nombre de cycles absorbés (voir section 5). En terme d'implémentation, il faut donc veiller à considérer le bon début d'une région, en tenant compte des irrégularités. L'algorithme est itératif, tout comme la propagation de croyances simple, mais sa complexité n'est pas calculable aussi simplement puisqu'elle dépend de la hauteur du graphe et de ses irrégularités.

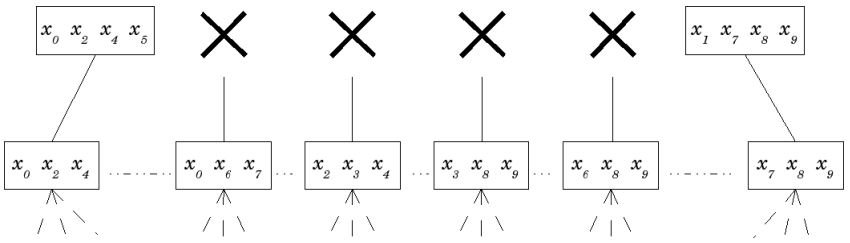


FIG. 7 – Graphe irrégulier

Conclusion

L'étude de la propagation de croyances nécessite encore quelques approfondissement, spécifiquement pour la méoire. En effet, pour éviter les explosions combinatoires dans la mémoire, la structure même du graphe des régions doit être la plus économe possible ; le temps de calculs pour le parcours dans le graphe grandissant exponentiellement en fonction de la hauteur, et du nombre d'enfants pour chaque groupe, le parcours-même doit être le plus simple possible. Ces deux aspects sont complexes, et demandent une recherche approfondie sur les équations-même de l'algorithme de propagation de croyances généralisé pour suplanter la propagation de croyances simple.

Bibliographie

1. Amin Shokrollahi. LDPC Codes : An Introduction. avril 2003.

2. Jonathan S. Yedida, William T. Freeman, and Yair Weiss. Understanding Belief Propagation and its Generalizations. janvier 2002.
3. Jonathan S. Yedida, William T. Freeman, and Yair Weiss. Constructing free energy approximations and Generalized Belief Propagation algorithms. décembre 2004.
4. Hidetoshi Nishimori. *Statistical Physics of Spin Glasses and Information Processing : An Introduction*. Clarendon Press, 2001.

Merci à Sylvain Reynal, maître de conférence à l'Université de Cergy-Pontoise et à l'ENSEA ainsi qu'à ceux qui m'ont aidé dans la rédaction de cet article.
