Des millions d'images pour la stéganalyse : inutile !

Jérôme Pasquet, Sandra Bringay, Marc Chaumont

Des millions d'images pour la stéganalyse : inutile !

J. Pasquet[2,3] S. Bringay[2,3,4] M. Chaumont[1,2,3]

[1] UNIVERSITE DE NIMES, F-30021 Nîmes Cedex 1, France
[2] UNIVERSITE MONTPELLIER 2, UMR5506-LIRMM, F-34095 Montpellier Cedex 5, France
[3] CNRS, UMR5506-LIRMM, F-34392 Montpellier Cedex 5, France
[4] AMIS, UNIVERSITE MONTPELLIER 3, Route de Mende 34199 Montpellier Cedex 5, France
{jerome.pasquet, sandra.bringay, marc.chaumont}@lirmm.fr

Résumé

Cet article fait suite aux travaux de Lubenko et Ker [7]. Nous comparons des méthodes de stéganalyse de la littérature permettant de lutter contre le cover-source mismatch avec un nombre réduit d'images d'apprentissage. De plus, nous proposons une méthode réduisant l'impact du cover-source mismatch sur les classifieurs en partitionnant la base de données d'apprentissage en îlots. Des expérimentations sur des jeux de données réelles montrent un gain de performance non négligeable lorsque l'on cherche à traiter le problème sans utiliser des millions d'images.

Mots clefs

Stéganalyse, cover-source mismatch, îlot, clustering, FLD, Ensemble classifieur

1 Introduction

La stéganographie est la science permettant de dissimuler de façon anodine un message dans un document. La stéganalyse est l'art de déceler ce message. Les technologies utilisées pour la stéganographie se sont considérablement améliorées ces dernières années. Le plus souvent, le stéganalyste d'images se place dans un cadre favorable pour décider si une image est *stego*[1] ou *cover*[2]. Pour cela, il pose des hypothèses fortes décrivant le scénario. Le scénario le plus utilisé est celui de *clairvoyance*[8]. Dans celui-ci, le stéganalyste connaît l'algorithme d'insertion utilisé, le type des images (appareil photo, lieux, etc), la quantité de bits à insérer et la dimension des images. Dans le cas d'une analyse automatique de trafic, c'est-à-dire où deux individus échangent librement des documents, ces hypothèses ne sont pas réalistes. En particulier, le stéganalyste ne peut pas avoir une connaissance parfaite du type des images utilisées par le stéganographe. Il y a forcément des différences entre les types d'images que le stéganalyste connaît et celles utilisées pour l'insertion. Ce problème s'appelle le *cover-source mismatch*.

Dans cet article, nous reprenons les conclusions obtenues par [6, 7] : 1) Il est intéressant d'utiliser de grandes bases de données d'apprentissage pour traiter le problème du *cover-source mismatch* ; 2) L'utilisation de classifieurs simples est plus robuste dans le cas du *cover-source mismatch*. Dans [7], afin de vérifier la première hypothèse, les auteurs utilisent une grande base de données d'images (environ 1 million d'images) mais provenant d'un nombre limité d'auteurs (environ 1800 auteurs). Dans notre scénario d'analyse automatique de trafic, nous sélectionnons des images provenant du site web TwitPic[3]. Ces images proviennent d'un grand nombre d'utilisateurs et sont très hétérogènes.

Notre contribution est triple. Nous confirmons les conclusions de [6, 7] précédemment décrites avec une base de données encore plus hétérogène. Nous proposons deux prétraitements : le premier permet d'amplifier la base de données en générant plusieurs images stego à partir d'une image *cover* et donc de travailler avec 'peu' d'images en entrée du processus (les cover). Le deuxième consiste à partitionner la base d'apprentissage en groupes d'images similaires, les ilots, afin d'appliquer la classification par ilot sur des données homogènes. Dans la section 2, nous présentons les enjeux liés au développement de méthodes de stéganalyse. Dans la section 3, nous exposons brièvement les classifieurs utilisés. Dans la section 4, nous introduisons les deux prétraitements utilisés. Enfin, dans la section 5, nous présentons les résultats expérimentaux associés à ces classifieurs dans le cas d'un mismatch total. Finalement, nous concluons et donnons des perspectives

1. Une image *stego* est une image contenant un message.
2. Une image *covers* est une image ne contenant pas de message.
3. http ://twitpic.com/

dans la dernière section.

2 Objectifs détaillés et notations

Dans cet article, nous souhaitons nous rapprocher d'une automatisation de la détection de messages cachés lors d'échanges d'images entre utilisateurs. Dans la réalité, les utilisateurs peuvent insérer un message d'une taille variable, dans une image quelconque et avec un algorithme qui leur est propre. Nous allons toutefois, restreindre ce scénario en imposant la taille du message inséré ainsi que l'algorithme d'insertion. Nous utilisons l'algorithme d'insertion HUGO [9] avec un *payload*[4] de 0.35 bit par pixel. Pour le stéganalyste, la principale difficulté réside dans le fait que les images utilisées lors des échanges sont inconnues à l'avance. Pour résoudre ce problème, deux solutions existent :

- Soit le stéganalyste est capable de faire un algorithme capable de s'adapter après avoir appris sur une base de données différentes de la base de données d'apprentissage utilisée par le stéganographe ;
- Soit l'apprentissage doit être réalisé sur un grand nombre d'images censées couvrir l'immense diversité de celles existantes dans la réalité et réduisant ainsi le problème du *cover-source mismatch*.

La deuxième solution a été utilisée dans [7] avec une très grande base de données d'apprentissage et un ensemble de classifieurs de faibles complexités. Notre proposition se rapproche de la première solution, avec peu d'images d'apprentissage. Dépendre de moins d'images comporte plusieurs avantages : l'espace de stockage nécessaire est moins important ; il est plus facile d'obtenir la base de données ; il est possible d'utiliser des algorithmes plus complexes en coût de calculs.

Afin de comparer les méthodes de la littérature et les nôtres nous avons constitué une base de données d'images. Notre "grande" base de données d'images comporte N images très variées : images synthétiques et photographiques, taux de compression différents, aux contenus sémantiques variés... À partir de cette base de données, nous extrayons des vecteurs caractéristiques de type RichModel [3] . Ces vecteurs, notés $\mathbf{f}$, se divisent alors en deux catégories : les vecteurs pour les images *covers*, notés $\mathbf{f}_x$, et ceux pour les images *stego*, notés $\mathbf{f}_y$.

3 Classifieurs

Dans cette section, nous présentons trois méthodes de la littérature : l'average perceptron, l'ensemble (classifieur et average perceptron) et le discriminant linéaire de Fisher avec une sélection de caractérisitiques.

3.1 Average Perceptron

Le perceptron est une méthode d'apprentissage *online*[5] utisant des données d'apprentissage et des données de tests. Ce classifieur permet sur les grands ensembles de données de créer une frontière linéaire. Lors de la phase d'apprentissage, cet algorithme repose sur un vecteur initial, appelé vecteur poids, noté $\mathbf{w} = \overrightarrow{\mathbf{0}}$, sur un vecteur représentant la somme des poids, noté $\mathbf{w}^{sum}$ et sur un vecteur moyen, équation (1).

$$\mathbf{w}^{avg} = \frac{\mathbf{w}^{sum}}{N_{app}} \tag{1}$$

avec N_{app} le nombre d'apprentissages que le perceptron a réalisé.

Le perceptron prédit le label à partir d'un vecteur caractéristique $\mathbf{f}_i$, auquel on veut associer un label $l_i \in \{cover = 1,\ stego = -1\}$. La prédiction notée z est calculée selon l'équation (2). Le calcul de z se fait par un produit scalaire entre le vecteur caractéristique d'entrée $\mathbf{f}_i$, et le vecteur moyen des poids, c'est-à-dire $\mathbf{w}^{avg}$. En fonction du signe de ce produit scalaire, on détermine de quel côté de l'hyperplan orthogonal à $\mathbf{w}^{avg}$ se trouve $\mathbf{f}_i$ et on en déduit la classe du vecteur (*stego* ou *cover*).

Lors de l'apprentissage, pour chaque vecteur caractéristique $\mathbf{f}_i$, de classe réelle l_i, si la prédiction est correcte, aucune mise à jour n'est effectuée sur le vecteur poids $\mathbf{w}$, sinon le vecteur $\mathbf{w}$ est mis à jour, voir équation (3). Une fois la mise à jour effectuée, on met à jour le vecteur $\mathbf{w}^{sum}$, voir équation (4), puis le processus recommence avec un nouveau vecteur caractéristique en entrée jusqu'à la stabilisation du vecteur de poids. Ce vecteur de poids représente la séparation entre les différentes classes, dans notre cas il s'agit de la frontière entre les images *stego* et *cover*.

$$z = sign(\mathbf{w}^{avg}.\mathbf{f}_i) \tag{2}$$

$$\begin{cases} si\ l_i = z\ \ alors\ aucune\ mise\ à\ jour \\ si\ l_i \neq z\ \ alors\ \mathbf{w} = \mathbf{w} + l_i.\mathbf{f}_i \end{cases} \tag{3}$$

$$\mathbf{w}^{sum} = \mathbf{w}^{sum} + \mathbf{w} \tag{4}$$

3.2 Ensemble classifieur et ensemble average perceptron

Dans [4, 5], Kodovský et al. proposent une nouvelle approche basée sur un ensemble de L classifieurs faibles. Lors d'une prise de décision de l'ensemble, le résultat final est issu du vote des L classifieurs faibles. Chaque classifieur faible effectue un apprentissage uniquement sur un sous-échantillonnage du vecteur caractéristique $\mathbf{f}$, noté

4. *Le payload est le nombre de bits insérés dans l'image par pixel.*
5. L'apprentissage online signifie que le perceptron assimile les données les unes aprés les autres.

$\mathbf{f}_{red}$. La taille du vecteur $\mathbf{f}_{red}$ et le nombre de classifieurs L sont des paramètres fixés à l'initalisation de l'ensemble classifieur. Chaque classifieur faible est représenté par une fonction h_i avec $i \in \{1..L\}$, telle que :

$$h_i(\mathbf{f}) : \begin{array}{ccc} \mathbf{f}_{red} & \rightarrow & \{stego, cover\} \\ \mathbb{R}^{N_{red}} & \rightarrow & \{-1, 1\} \end{array} \quad (5)$$

Lors de la prédiction d'image *cover* ou *stego*, un vote majoritaire est généralement appliqué. À partir de l'équation 5, nous déduisons la valeur du label prédit, noté z, dans l'équation 6.

$$z = sign(\sum_{i=0}^{L} (h_i(\mathbf{f}))) \quad (6)$$

Le classifieur simple, h_i (équation (5)), peut être de plusieurs types. Dans cet article, nous étudions l'ensemble average perceptron [7, 6] et l'ensemble de Kodovský. Dans l'approche Kodovský, chaque classifieur faible est obtenu via l'approche par déterminant linéaire de Fisher (FLD)[4].

3.3 Ensemble FLD avec sélection de caractéristiques

Dans [2], les auteurs émettent l'hypothèse que pour chaque classifieur simple, de type FLD, certaines caractéristiques du sous-échantillonnage du vecteur caractéristique sont moins importantes que d'autres. Dans la méthode ensemble FLD avec sélection de caractéristiques, noté EFLDFS, les auteurs déterminent cinq métriques permettant de sélectionner ces caractéristiques. Chaque métrique donne un score à chaque caractéristique du vecteur $\mathbf{f}$, noté c_i^n avec i le numéro de la métrique $i \in \{1..5\}$ et n le numéro de la caractéristique avec $n \in \{1..d_{red}\}$ avec d_{red} la taille du sous-échantillonnage du vecteur caractéristique $\mathbf{f}$. Pour chaque score, les auteurs suppriment les caractéristiques de façon à ne pas augmenter la probabilité d'erreurs. Les tests s'effectuent donc sur un espace réduit. Sur la base de données homogène BOSSv1 [6][1] les auteurs obtiennent un gain moyen sur le rappel de 1.7%.

4 Contributions

Nous allons maintenant décrire les deux prétraitements proposés. Le premier vise à amplifier la base d'apprentissage sans introduire plus d'images *cover*. Le second vise à lutter contre l'hétérogénéité contenue dans une base de données, en identifiant des groupes d'images homogènes et en y appliquant des classifieurs adaptés.

6. http ://exile.felk.cvut.cz/boss/BOSSFinal/

4.1 Génération de multiples images stego à partir d'une cover

La principale difficulté avec l'utilisation de l'ensemble perceptron est la taille de la base de données d'apprentissage nécessaire pour converger. Nous proposons de générer de multiples images *stego* à partir d'une seule image *cover*. Ainsi, nous obtenons plus de vecteurs caractéristiques et donc plus d'informations améliorant la vitesse de convergence des différentes méthodes. Généralement, à partir d'une image *cover,* nous obtenons via HUGO [9] une et une seule image *stego*. Désormais, à partir d'une image *cover*, nous effectuons S insertions avec des messages pseudo-aléatoires différents. Nous obtenons ainsi S versions différentes du vecteur caractéristique *stego*. Nous supposons ici que la base d'apprentissage obtenue est plus robuste contre le *cover-source mismatch*. En effet, même si une image est peu représentée dans la base d'apprentissage, S informations différentes sont générées pour la décrire.

4.2 Îlots

Dans le cas d'une analyse automatique de trafic, les images sont sélectionnées depuis internet. Celles-ci sont très hétérogènes de part l'appareil photo (traitement de l'appareil, focale, sensibilité ISO, ...), le lieu, le format (type, qualité, taux de compression). La variété des images implique que les descripteurs permettant de différencier une image *cover* d'une image *stego* ne se situent pas nécessairement dans les mêmes dimensions du vecteur $\mathbf{f}$. Il y a donc une réelle difficulté pour les classifieurs à déterminer les frontières entre les images *stego* et *covers*. Dans cette partie, nous proposons une méthode permettant de regrouper les images par catégories. Chaque catégorie forme alors ce que nous appelons des **îlots** homogènes d'images. La figure 1 représente un choix possible d'îlot, où nous avons regroupé les images en fonction de leur contenu. Nous associons à chaque îlot un classifieur et réalisons un apprentissage indépendant. Lors de la phase de tests, l'image d'entrée est testée par le classifieur associé à l'îlot dont le type est le plus proche. Ce classifieur s'est entrainé sur un groupe d'images similaires à celle testée, c'est à dire dont les vecteurs caractéristiques sont proches (en terme de distance euclidienne ou cosinus) à celui de l'image testée. Nous espérons lutter contre l'hétérogènéité et à fortiori contre le problème du *cover-source mismatch*.

La méthode se décompose en trois parties : la génération des îlots, l'apprentissage et les tests. Pour la génération des ilots, il est nécessaire de comparer les vecteurs caractéristiques des images. Pour cela, il faut définir une métrique notée $d(\mathbf{x}, \mathbf{y})$ permettant d'évaluer la distance entre des vecteurs $\mathbf{x}$ et $\mathbf{y}$. Nous appliquons l'algorithme

des k plus proches voisins ($K-means$) permettant de créer C *clusters*[7]. Chaque cluster est identifié par son centre que nous notons $\mathbf{v}_i$ avec $i \in \{1..C\}$. Chaque îlot correspond donc à un cluster d'images homogènes et à un classifieur apprenant sur celle-ci.

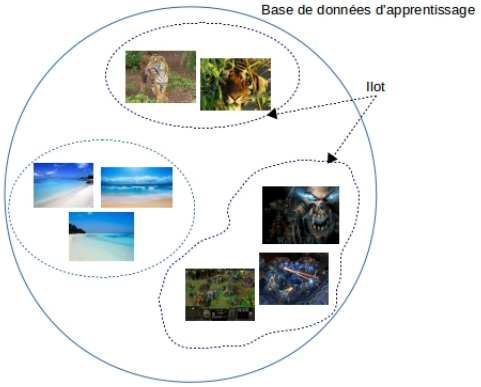

FIGURE 1 – Schéma simpliste des îlots en regroupant les images "similaires"

Après la génération des C îlots, nous effectuons l'apprentissage. Pour chaque nouvelle image, nous cherchons l'îlot dont le centre $\mathbf{v}_i$ est le plus proche et utilisons le classifieur associé. Comme la distance entre le vecteur caractéristique de l'image *cover* et *stego* est très faible, le classifieur utilisé pour $\mathbf{f}_x$ sera également utilisé pour $\mathbf{f}_y$. Le classifieur j (correspondant au classifieur associé à l'îlot j) utilisé pour l'apprentissage des vecteurs caractéristiques $\mathbf{f}_x$ et $\mathbf{f}_y$ est celui dont le centre $\mathbf{v}_j$ avec $j \in 1..C$, est le plus proche du vecteur caractéristique $\mathbf{f}_x$, équation (7) :

$$j = \underset{i}{arg}\ \underset{i\in\{1..C\}}{min}\ (d(\mathbf{v}_i, \mathbf{f}_x)) \tag{7}$$

L'apprentissage se fait alors de manière classique avec le classifieur j choisi. Ce processus est représenté dans la figure 2. Le vecteur centre $\mathbf{v}_1$ est le plus proche des vecteurs d'entrée $\mathbf{f}_x$ et $\mathbf{f}_y$, et c'est donc le classifieur associé à l'îlot 1 qui va effectuer l'apprentissage.

Lors de la phase de tests, quelque soit l'image en entrée, celle-ci sera testée sur un classifieur dont l'apprentissage a été effectué uniquement sur des images ayant des vecteurs caractéristiques similaires.

Pour appliquer cette méthode, deux paramètres importants sont à fixer. Le premier est la norme utilisée pour le calcul des distances entre vecteurs. Lors de nos expériences, nous avons choisi de tester trois types de normes : la norme 1, la norme euclidienne et la similarité cosinus. Le deuxième paramètre à fixer est le nombre C de clusters à générer. Augmenter le nombre de clusters est très bénéfique pour lutter contre le *cover-source mismatch*. En effet, plus le nombre d'îlots est important, plus l'image à classer sera proche du centre $\mathbf{v}_i$ du cluster et donc plus les images d'apprentissage dans cet îlot seront similaires. Plusieurs contraintes doivent être prises en compte. La première est une contrainte matérielle, à savoir la mémoire ram puisque pour chaque îlot, nous devons allouer un classifieur de type ensemble average perceptron ou ensemble FLD d'une taille conséquente[8]. L'utilisation du *HPC@LR*[9] et de leur 24Go de ram par noeud, a permis de résoudre ce problème. Le fait que les images soient réparties entre les îlots constitue la deuxième contrainte. En effet, en moyenne le nombre d'apprentissages de chaque îlot est donc $\frac{N}{C}$ avec N le nombre de couples d'images *stego* et *cover*. Ce problème est d'autant plus important que la convergence des perceptrons est très longue et demande une grande base de données.

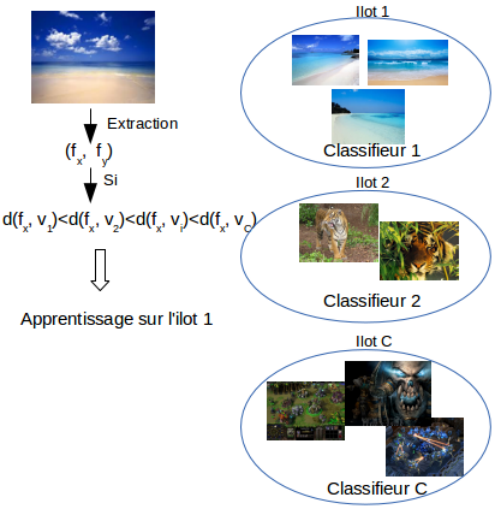

FIGURE 2 – Schéma représentant le fonctionnement de l'apprentissage sur les îlots.

5 Expériences et résultats

Deux hypothèses sont à vérifier dans ces expérimentations :

1. L'utilisation de grands ensembles de données n'est pas forcément la solution la plus performante pour une stéganalyse efficace.
2. Il est possible de regrouper les images d'apprentis-

7. Un *cluster* est un ensemble de données représentant des similarités. Dans notre cas, les *clusters* contiendront des vecteurs proches les uns des autres.

8. Par exemple, si chacun des C îlots contient un ensemble average perceptrons avec L classifieurs simples dont la taille de vecteur caractéristique est d_{red}, la consommation de mémoire ram est donc proportionnelle à $C.L.d_{red}$.

9. *HPC@LR : High Performance Computing@Languedoc-Roussillon : www.hpc-lr.univ-montp2.fr*

sage et de tests en *clusters* réduisant ainsi la diversité et améliorant les performances comme décrit dans la section 4.2.

Dans les sections 5.1 et 5.2, nous allons vérifier respectivement l'hypothèse 1 et 2.

5.1 Comparaisons avec et sans grands volumes de données

La base d'apprentissage que nous utilisons contient $N = 1.8$ million d'images différentes, dont 100 000, 185 000 et 100 000 sont stéganographiées respectivement 20, 5 et 3 fois sur le principe décrit dans la section 4.1. Au final, nous utilisons une base de 3.8 millions de couples d'images. Afin de stabiliser les perceptrons, une fois l'apprentissage terminé, nous le relançons sur les premières images utilisées pour l'apprentissage, ceci nous permet d'atteindre quasiment 7 millions de couples d'apprentissage avec des redondances.

Les résultats présentés sont issus de la moyenne de 3 simulations. Lors de chaque simulation, les éléments de la base d'apprentissage sont considérés dans un ordre différent. Pour chaque test, nous effectuons le calcul du taux de prédiction correcte (equation (8)) sur une base de données de 40 000 images ne faisant pas partie de la base d'apprentissage et changeant à chaque simulation.

$$Taux\ de\ prédiction = \frac{Nb\ labels\ correctement\ prédits}{Nb\ total\ de\ tests} \tag{8}$$

Sur cette grande base de données, nous réalisons un ensemble average perceptron (EAP). Afin de vérifier le gain obtenu par les multiples insertions, nous utiliserons également un ensemble average perceptron qui effectuera son apprentissage plusieurs fois sur les mêmes images *stego* (sans utiliser le concept de multiples insertions), nous le notons *EAP iteratif*. Nous comparons ces classifieurs de très faibles complexités avec le classique ensemble de FLD (EFLD) ainsi que son amélioration avec les sélections de caractéristiques (EFLDFS) (figure 3). La complexité de l'EFLD e l'EFLDFS est en $O(N.L.d_{red}^2)$ contre $O(N.L.d_{red})$. Ce problème calculatoire rend impossible l'apprentissage sur la totalité de la base de données pour l'EFLD et l'EFLDFS. Les paramètres L et d_{red} , pour l'EFLD [10] et l'EAP, sont fixés manuellement et nous utilisons les valeurs de [7, 6], c'est-à-dire $L = 100$ et $d_{red} = 2000$.

En observant l'allure des courbes EAP et EAP iteratif de la figure 3, on peut observer que l'amélioration présentée dans la section 4.1, ne présente qu'un intérêt minime. En effet, suite à la génération de multiples *covers*, nous obtenons des résultats légérement meilleurs pour moins de 3 millions d'apprentissage, ensuite les deux méthodes convergent vers les mêmes valeurs. On constate que le perceptron converge vers 0.93 de taux de prédiction correct. Ce résultat est largement supérieur à l'EFLD qui est à 0.83, confirmant l'hypothèse que l'utilisation de grandes bases de données est une solution contre le *cover-source mismatch* [7, 6]. Toutefois, la méthode EFLDFS avec peu d'images atteint des résultats très supérieurs à toutes les autres méthodes. Avec uniquement 150 000 images, EFLDFS est plus performant que l'EAP fonctionnant avec plus d'un million d'images. Ces observations appuient notre hypothèse 1, selon laquelle utiliser de grands volumes d'images est inutile.

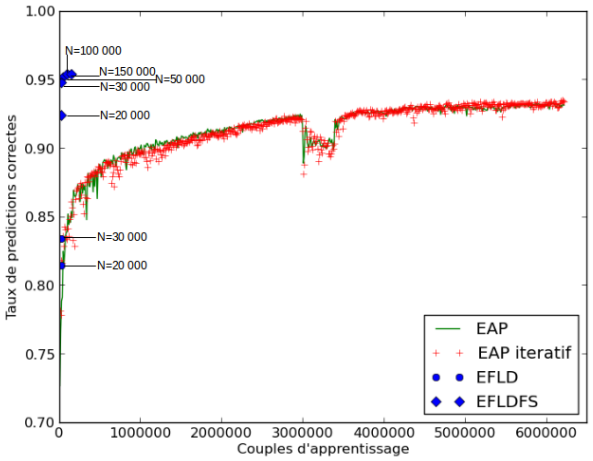

FIGURE 3 – Représentation de différents algorithmes de stéganalyse

5.2 Comparaisons avec îlots

Nous utilisons les îlots (section 4.2) dans différentes conditions : des îlots d'ensembles d'average perceptron et des îlots d'ensemble FLD avec sélection de caractéristiques en faisant varier le nombre C d'îlots et donc le nombre d'images d'apprentissage. Tous les îlots sont initialisés identiquement et seule leur base de données d'apprentissage varie en fonction de leur $\mathbf{v}_i$.

Nous avons effectué l'expérience en utilisant des îlots d'EAP, nous constatons alors que les îlots n'améliorent pas les résultats. Nous pouvons expliquer cela par deux phénomènes : - soit le nombre d'apprentissage, après division par le nombre de clusters C, est trop insuffisant pour faire converger les îlots, soit l'EAP, de part sa capacité à utiliser de grands volumes de données lutte déjà très efficacement contre le *cover-source mismatch*, et donc les îlots n'apportent aucune robustesse supplémentaire contre le *cover-source mismatch*.

Dans le tableau 1, nous présentons les résultats du classifieur EFLDFS que nous comparons à des îlots contenant des EFLDFS. Toutes les méthodes avec îlots

10. Pour l'EFLD, nous utilisons une implémentation C++ implémentée par les auteurs de [2].

confirment l'hypothèse 2, en procurant des résultats meilleurs de minimum 0.22% par rapport au EFLDFS. Bien que le gain soit faible, les résultats sont très encourageants. En effet, en se référant à la figure 3, on constate que l'EFLDFS obtient des résultats très similaires pour 50 000, 100 000 et 150 000 apprentissages montrant que l'EFLDFS a convergé. Toutefois, les îlots permettent de dépasser ce score de convergence. Dans le tableau 1, nous obtenons pour C=5 îlots un score 0.5% meilleur. Or, les EFLDFS contenus dans les îlots ont assimilé 5 fois moins d'apprentissages. On peut donc en déduire que le point de convergence des îlots et donc des EFLDFS apprenant sur une base de données plus homogène est plus élevé et n'est pas encore atteint.

Nb d'îlots	Nb d'apprentissages moyen par îlot	Norme cluster	Taux de prédiction correct
1	150 000		0.955
2	75 000	L2	0.958 (+0.3%)
5	30 000	cosinus	0.960 (+0.49%)
7	21 428	cosinus	0.957 (+0.21%)
10	15 000	cosinus	0.957 (+0.22%)

TABLE 1 – Résultats avec des îlots de EFLDFS pour 150 000 couples d'apprentissage

6 Conclusion

Dans cet article, nous avons montré que la méthode FLD avec sélection de caractéristiques est très efficace sur des données très hétérogènes. Celle-ci présente de bonnes performances pour lutter contre le problème du *cover-source mismatch* et cela avec très peu d'images. En effet, avec 150 000 images différentes, la méthode est en moyenne 2.5% plus efficace que l'ensemble average perceptron utilisant plus d'un million d'images. En combinant cette méthode très robuste à la diversité avec le concept îlot, nous obtenons un gain de 3% sur l'ensemble average perceptron.
La méthode utilisant les îlots ouvre une nouvelle voie pour lutter contre le *cover-source mismatch*. Dans cet article, nous avons restreint le nombre d'îlots à 10 mais il serait intéressant d'augmenter ce nombre ainsi que le nombre d'images d'apprentissage. De plus, nous nous sommes limités à comparer les vecteurs caractéristiques à l'aide de norme euclidienne et similarité cosinus, d'autres méthodes pouvant être utilisées formant ainsi des îlots d'une nature plus intéressante. Par ailleurs, les îlots avec d'autres classifieurs (plus complexes), peuvent être une piste prometteuse. En effet, en homogénéisant les bases de données de chaque îlot et en répartissant le nombre d'apprentissages, nous pouvons envisager l'utilisation d'un séparateur à vaste marge (SVM) qui donnerait très probablement de grandes performances sur les données homogènes.

Références

[1] P. Bas, T. Filler, and T. Pevný. "Break Our Steganographic System" : The Ins and Outs of Organizing BOSS. In *Information Hiding, 13th International Conference, IH'2011*, volume 6958 of *Lecture Notes in Computer Science*, pages 59–70, Prague, Czech Republic, may 2011. Springer.

[2] M. Chaumont and S. Kouider. Steganalysis by ensemble classifiers with boosting by regression, and post-selection of features. In *Image Processing (ICIP), 2012 19th IEEE International Conference on Image Processing*, pages 1133–1136, Lake Buena Vista (suburb of Orlando), Florida, USA, September 2012.

[3] J. Fridrich and J. Kodovský. Rich models for steganalysis of digital images. *Information Forensics and Security, IEEE Transactions on*, 7(3) :868–882, 2012.

[4] J. Kodovský and J. Fridrich. Steganalysis in high dimensions : fusing classifiers built on random subspaces. In *Media Watermarking, Security, and Forensics III, Part of IS&T/SPIE 21th Annual Symposium on Electronic Imaging, SPIE'2011*, volume 7880, San Francisco, California, USA, February 2011.

[5] J. Kodovský, J. Fridrich, and V. Holub. Ensemble classifiers for steganalysis of digital media. *Information Forensics and Security, IEEE Transactions on*, 7(2) :432–444, 2012.

[6] I. Lubenko and A. D. Ker. Going from small to large data in steganalysis. In *Media Watermarking, Security, and Forensics 2012*, volume 8303M of *Proc. SPIE*. SPIE, 2012.

[7] I. Lubenko and A. D. Ker. Steganalysis with mismatched covers : do simple classifiers help ? In *Proceedings of the on Multimedia and security*, MM&Sec '12, pages 11–18, New York, NY, USA, 2012. ACM.

[8] T. Pevný. Detecting messages of unknown length. In *Media Watermarking, Security, and Forensics III, Part of IS&T/SPIE 21th Annual Symposium on Electronic Imaging, SPIE'2011*, volume 7880, San Francisco, California, USA, February 2011.

[9] T. Pevný, T. Filler, and P. Bas. Using High-Dimensional Image Models to Perform Highly Undetectable Steganography. In *Information Hiding, 12th International Conference, IH'2010*, volume 6387 of *Lecture Notes in Computer Science*, pages 161–177, Calgary, Alberta, Canada, jun 2010. Springer.