Un Algorithme de Partitionnement d'Ontologies Orienté Alignement

Soumaya kasri[1,1], Fouzia Benchikha[1]
[1,1]Université de Skikda, département d'informatique, Algérie
[1]Université de Skikda, laboratoire LIRE Constantine, Algérie
{Kasri.soumaya, f_benchikha}@yahoo.fr

Résumé. Dans les applications réelles où les ontologies sont volumineuses, les exigences de l'exécution du temps et de l'espace de mémoire sont les deux facteurs significatifs qui influencent directement sur la performance d'un algorithme d'alignement. L'une des solutions du passage à l'échelle suppose la possibilité de partitionner les ontologies en blocs avant de réaliser l'alignement. Dans cet article, nous proposons un algorithme de partitionnement d'ontologie. Cet algorithme consiste à partitionner chaque ontologie en blocs autour des ancres en utilisant les algorithmes de clustering car les blocs qui contiennent les entités similaires (ancrées) devraient l'être aussi. Les résultats obtenus dans l'évaluation de notre algorithme montrent son efficacité.

Mot clé: ontologie, alignement, partitionnement, clustering.

1 Introduction

De nos jours, les ontologies sont devenues l'une des plus importantes orientations de recherche notamment avec l'avènement du Web Sémantique. Elles jouent un rôle primordial pour l'annotation de pages ou de services web puisqu'elles modélisent les concepts, attributs et relations utilisés pour annoter le contenu des ressources.

Dans de nombreux contextes applicatifs, plusieurs ontologies couvrant un même domaine ou des domaines connexes sont développées indépendamment les unes des autres par des communautés différentes. L'hétérogénéité entre les connaissances exprimées au sein de chacune d'entre elles doit être résolue. C'est la problématique de l'interopérabilité qui a pour objectif de permettre à des systèmes hétérogènes, qui s'appuient sur une de ces ontologies, de pouvoir communiquer et coopérer dans le but d'atteindre leurs objectifs. À cette fin, les liens sémantiques entre les entités appartenant à deux ontologies différentes doivent être établis d'où l'alignement d'ontologies.

L'alignement consiste à déterminer l'ensemble de correspondances entre deux ontologies en utilisant ou en mettant en œuvre des solutions aux différents problèmes d'hétérogénéité. L'ensemble de correspondances peut par la suite être utilisé notamment pour des applications telles que l'échange, l'intégration et la transformation des données.

De nombreuses méthodes d'alignement dédiées aux ontologies ont vu le jour cette dernière décennie [2,4,5]. Cependant, ces méthodes sont conçues pour aligner des

ontologies de petite taille. Dans la littérature actuelle, il existe très peu d'algorithmes d'alignement qui visent à traiter le problème du passage à l'échelle des méthodes d'alignement. L'une des solutions du passage à l'échelle suppose la possibilité de partitionner les ontologies en blocs avant de réaliser l'alignement [8,10]. En effet, pour diminuer l'espace de recherche des correspondances, il faut limiter la taille des ensembles de concepts en entrée de l'outil d'alignement.

L'objectif de notre travail est de relever le challenge du passage à l'échelle des méthodes d'alignement. En particulier, nous proposons un algorithme de partitionnement d'ontologie orienté alignement. Cet algorithme consiste à partitionner chaque ontologie en blocs autour des ancres[1] en utilisant les algorithmes de clustering. Les blocs générés peuvent être utilisés dans un algorithme d'alignement car les blocs qui contiennent les entités similaires (ancrées) devraient l'être aussi. Notre algorithme a été testé sur des ontologies de test disponibles pour le partitionnement notamment les paires Russia12 et TourimAB. Des résultats satisfaisants ont été obtenus.

La suite de l'article est organisée comme suit: la section 2 discute des travaux relatifs en présentant quelques méthodes de partitionnement qui sont appliquées dans différents domaines. Toutefois, nous nous sommes intéressés aux méthodes qui ont comme objectif l'alignement des ontologies. Dans la section 3, nous présentons notre algorithme de partitionnement. La section 4 présente une évaluation sur des ontologies de test et la section 5 conclut notre travail.

2 Etat de l'art

Avec l'apparition des nouveaux standards, outils et langages très expressifs, de grandes ontologies ont été développées dans plusieurs domaines (e.g biologie, géographie, médecine …etc.). Ces ontologies comportent plusieurs dizaines de milliers de concepts par exemples, l'ontologie Gene Ontology GO [14] et l'ontologie AGROVOC[2] [1] contiennent 26 057 et 28 439 concepts respectivement. Dans la littérature, on rencontre plusieurs algorithmes [8,10,12,13] qui visent le partitionnement des ontologies de façon à faciliter et à rendre plus performantes les opérations de maintenance, de visualisation, de raisonnement ou d'alignement.

Ainsi, les travaux de [12] proposent deux méthodes de décomposition d'ontologie en plusieurs sous-ontologies exprimées en logique de description. Cette décomposition est appliquée telle qu'elle préserve toujours la sémantique et les services d'inférence de l'ontologie originale. La première méthode est basée sur le séparateur minimal en utilisant l'algorithme récursive de Even [6] et la deuxième est basée sur la segmentation d'images en utilisant les vecteurs et les valeurs propres [11]. Les travaux dans [13] partitionnent une ontologie en blocs indépendants et cohérents à partir d'un graphe de dépendance. Le but de cette méthode est de faciliter

[1] Les ancres sont des entités jugées comme équivalentes en se basant sur une mesure de similarité.

[2] Une ontologie construite par le FAO (Food and Agriculture Organization).

les différentes opérations sur les ontologies (maintenance, validation et raisonnement).

Falcon-AO [10] et TaxoMap [8] sont deux méthodes de découverte de correspondance qui ont été préalablement développées dans le cadre d'alignement linguistique et structurel d'ontologies de petite taille. Mais, dans leurs versions actuelles Falcon-AO et TaxoMap intègrent des algorithmes de partitionnement adaptés au contexte d'alignement des ontologies volumineuses.

Dans le cadre de notre travail, nous nous intéressons plus particulièrement à ces deux méthodes que nous présentons brièvement dans ce que suit.

Falcon-AO. Cette méthode partitionne une ontologie en clusters $g_1, g_2, \ldots\ldots\ldots g_n$. L'algorithme proposé est agglomératif et inspiré de l'algorithme de clustering de ROCK [7]. On part de petits clusters nombreux et on les regroupe progressivement en clusters plus conséquents.

L'algorithme de partitionnement s'appuie sur deux propriétés essentielles: la cohésion au sein d'un cluster et le couplage entre deux clusters distincts. La cohésion mesure la similarité entre les entités appartenant à un même cluster et le couplage mesure la similarité entre les entités appartenant à deux clusters différents. Les deux propriétés sont calculées au sein d'une même fonction cut () qui mesure la distance entre deux clusters en reposant sur un critère d'agrégation. Ce dernier détermine la manière d'agglomérer deux clusters.

L'algorithme prend en entrée l'ensemble g des n clusters à partitionner, où chaque cluster est réduit au départ à une unique entité et ε le nombre des entités dans un cluster que l'on souhaite obtenir. Dans chaque itération, l'algorithme choisit tout d'abord le cluster qui a la cohésion maximale, puis le cluster qui a le couplage minimal avec ce premier cluster, et fusionne ces deux clusters.

Dans l'étape d'alignement, Falcon-AO aligne toutes les paires ayant une proximité supérieure à un seuil. Cette proximité s'effectue en s'appuyant sur des ancres. Soit b et b′ deux blocs, la proximité entre b et b′ est calculée comme suit :

$$\text{prox}(b, b') = \frac{2 \cdot \text{nombre d'ancres partagées par b et b'}}{\text{nombre d'ancres contenues par b} + \text{nombre d'ancres contenues par b'}} . \quad (1)$$

La proximité entre deux blocs est liée au nombre d'ancres partagées. Si le nombre d'ancres partagées est petit, la proximité sera inférieure au seuil et la paire n'est pas alignée c.-à-d. l'alignement de deux ancres partagées n'est pas retrouvé. D'autre part, plusieurs blocs pourront ne contenir aucune ancre. Dans ce cas ces blocs sont isolés[3] et on a risque de perdre plusieurs alignements.

TaxoMap. Deux méthodes ont été proposées. Soit O_S et O_T deux ontologies à aligner. La première méthode comprend trois étapes en plus du calcul des ancres. (1)Partitionner l'ontologie O_T en plusieurs blocs B_{Ti},. Le partitionnement est effectué

[3] Les blocs isolés sont des blocs qui ne contiennent aucune ancre ou dont la similarité avec les autres blocs ne dépasse pas le seuil choisi.

conformément à l'algorithme de Falcon-AO. (2)Identifier les centres CB_{Si} des futurs blocs de l'ontologie O_S. Les centres de O_S sont déterminés en se basant sur deux critères: les couples d'ancres identifiés entre O_S et O_T, et les blocs B_{Ti} construits à partir de l'ontologie O_T. (3)Partitionner l'ontologie source autour des centres CB_{Si} identifiés dans l'étape précédente.

La deuxième méthode comprend deux étapes. (1)Partitionner l'ontologie O_T en plusieurs blocs B_{Ti}. Le partitionnement est effectué conformément à l'algorithme de Falcon-AO mais en prenant en compte lors de la génération des blocs, l'ensemble des ancres. (2)Partitionner l'ontologie O_S de la même manière mais en se basant à chaque fois sur une partie des ancres qui appartiennent à un bloc de l'ontologie O_T .

Dans les méthodes de TaxoMap, le problème de l'alignement des ancres partagées est résolu mais le problème de perdre des alignements provenant des blocs isolés existe toujours.

Pour contribuer à résoudre ce problème, nous proposons un algorithme de partitionnement d'ontologie autour des ancres. Notre algorithme a l'avantage de n'avoir aucun bloc isolé en assurant la non perte des alignements et fournit en sortie des paires de blocs à ligner. Chaque bloc de la première ontologie ne s'aligne qu'avec un seul bloc de la deuxième ontologie.

3 Algorithme de partitionnement proposé

Dans une ontologie, les entités sont décrites en utilisant les noms, les étiquètes, et les commentaires. L'algorithme que nous proposons consiste à partitionner chaque ontologie en blocs autour des ancres, puis l'alignement s'effectue entre les paires des blocs fournies en sortie. Nous rappelons brièvement quelques définitions des mesures utilisées dans notre algorithme.

La similarité lexicale. Pour calculer la similarité lexicale entre les entités nous utilisons la mesure de Jaro-Winkler [15] qui prend simultanément en compte le nombre et la position des sous chaînes communes avec l'utilisation de la taille p du plus grand préfixe commun de deux chaines comparées. En OWL, les entités sont décrites en utilisant les constructeurs (rdf :id, rdfs : label, rdfs : comment) qui traduisent le triplet (nom, étiquète, commentaire) respectivement. Dans notre cas nous utilisons seulement les noms des entités (classe, propriété) pour calculer la similarité lexicale entre les entités, sans prétraitement, dans une mesure à moindre coût.

$$\mathrm{Sim_L}(e_1, e_2) = 1\text{-}\mathrm{DS_{Jaro-Winkler}}. \quad (2)$$

La similarité structurelle. Wu & Palmer [16] ont proposé de calculer la similarité entre deux concepts par la formule suivante :

$$\mathrm{Sim_S}(e_1, e_2) = \frac{2*\mathrm{depth}(e)}{\mathrm{depth}(e_1)+\mathrm{depth}(e_2)}. \quad (3)$$

e : le concept le plus spécifique qui subsume les deux concepts e_1, e_2 dans l'ontologie.

$depth(e)$: le nombre d'arcs qui séparent le concept e de la racine.

$depth(e_i)$: le nombre d'arcs qui séparent le concept e_i de la racine en passant par e .

La mesure de Wu & Palmer est intéressante, simple à implémenter et performante dans les évaluations. Cependant, pour une ontologie de grande taille, le calcul de similarité entre toutes les entités peut prendre beaucoup de temps. Pour cela, nous proposons d'effectuer le calcul de similarité seulement entre les entités qui se situent à un rayon r (r étant le nombre d'arcs entre les deux entités) dans un procédé itératif où r est défini selon la taille et la structure de l'ontologie.

Soient O_t et O_s deux ontologies à aligner, notre algorithme de partitionnement orienté alignement comprend les étapes suivantes :

1. Déterminer les couples d'ancres entre O_t et O_s en utilisant une mesure de similarité lexicale.
2. Classifier ces couples d'ancres en k clusters préliminaires.
3. Identifier les clusters d'ancres de chaque ontologie (l'ensemble des ancres appartenant à chaque ontologie) .
4. Classifier les concepts de chaque ontologie autour des clusters d'ancres.

Dans ce que suit, nous détaillons les différentes étapes de partitionnement.

3.1 Déterminer les ancres

Dans l'étape de classification, l'algorithme tentera de regrouper les entités autour des ancres. Nous disons que deux entités sont des ancres si et seulement si elles sont équivalentes. Pour cela, nous utilisons la similarité lexicale présentée ci-dessus dans la formule (2) pour les déterminer.

3.2 Partitionnement préliminaire

Le but de ce partitionnement est de regrouper les ancres dans K clusters. Après la détermination des ancres par la similarité lexicale, on regroupe chaque couple d'ancres dans un cluster comme montré en Fig. 1. Puis il s'agit d'exécuter un algorithme de clustering inspiré de celui de Falcon-AO[10].

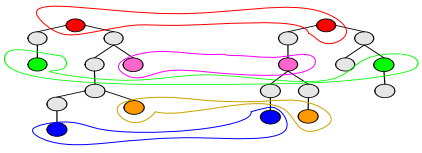

Fig. 1. L'entrée de l'algorithme (les couples d'ancres)

L'algorithme. La classification consiste à agréger progressivement les clusters d'ancres selon leur ressemblance. Cette agrégation nécessite un critère de classification. L'algorithme est réalisé grâce à l'utilisation d'un algorithme de partitionnement agglomératif hiérarchique inspiré de l'algorithme de Falcon-AO. La première différence entre notre algorithme et celui de Falcon-AO est que ce dernier prend en entrée toutes les entités de l'ontologie à aligner et le nombre k des blocs que l'on souhaite obtenir en sortie par contre notre algorithme prend en entrée les clusters initialisés par les paires d'ancres, le nombre de cluster k et le nombre maximum d'entités dans un cluster fusionnable m. Le but du partitionnement est de diminuer le nombre d'entités dans un bloc, car un système d'alignement n'est plus du tout efficace à partir d'un certain nombre d'entités.

La deuxième différence est que la méthode Falcon-AO utilise la fonction cut() comme un critère de classification. Celle-ci est définie comme suit :

$$\mathrm{cut}(g_i, g_j) = \frac{\sum_{d_i \in g_i} \sum_{d_j \in g_j} W(d_i, d_j)}{|g_i| \cdot |g_j|}. \quad (4)$$

$$\text{avec} \quad \begin{cases} \mathrm{cut}(g_i, g_i) = \mathrm{cohésion}(g_i) \\ \mathrm{cut}(g_i, g_j) = \mathrm{couplage}(g_i, g_j) \quad \text{avec } g_i \neq g_j \end{cases}.$$

Où g_i, g_j sont deux clusters et W est la matrice de la proximité (lexicale/structurelle) entre les entités. Pour calculer l'élément (i, j) dans W, Falcon-AO mesure le lien pondéré entre deux entités, $\mathrm{link}(e_i, e_j)$ comme suit :

$$\mathrm{link}(e_i, e_j) = \begin{cases} \mathrm{prox}(e_i, e_j) \text{ si } \mathrm{prox}(e_i, e_j) > \varepsilon \\ 0, \text{sinon} \end{cases}. \quad (5)$$

$$\text{avec } \mathrm{prox}(e_i, e_j) = \alpha\, \mathrm{prox}_s(e_i, e_j) + (1 - \alpha)\, \mathrm{sim}_l(e_i, e_j). \quad (6)$$

Où ε est un seuil donné tel que $\varepsilon \in [0, 1]$ et $\alpha \in [0, 1]$. Il permet à l'utilisateur de faire varier le poids relatif des mesures de similarité lexicale (sim_l) et structurelle (prox_s).

Dans notre cas, le calcul de similarité lexicale ne s'effectue que pour obtenir la cohésion au sein d'un cluster pour les raisons suivantes :

- le rapprochement de deux clusters se fait sur la base de similarité de leurs entités. Généralement, ces entités sont décrites différemment au sein d'une même ontologie. Pour cela, la similarité structurelle sera suffisante ;
- réduire la complexité du calcul ;
- obtenir un meilleur temps d'exécution.

Donc si $cut(g_i, g_j) = couplage(g_i, g_j)$, le calcul de lien pondéré s'effectue comme suit :

$$\mathrm{prox}(e_i, e_j) = \mathrm{Sim}_S(e_i, e_j). \quad (7)$$

Et si $\mathrm{cut}(g_i, g_j) = \mathrm{cohésion}(g_i)$, le calcul de lien pondéré s'effectue comme suit :

$$\mathrm{prox}(e_i, e_j) = \alpha\ \mathrm{Sim}_S(e_1, e_2) + \beta\ \mathrm{Sim}_L(e_1, e_2). \quad (8)$$

L'algorithme prend en entrée l'ensemble S de n clusters à partitionner, ou chaque cluster est réduit au départ à une paire d'ancres, le nombre k de clusters que l'on souhaite obtenir en sortie, et le nombre maximum m d'entités au sein d'un cluster fusionnable.

Comme Falcon-AO, dans chaque itération, l'algorithme choisit tout d'abord le cluster qui a la cohésion maximale (ordonner les clusters par leurs cohésions), puis le cluster qui a la valeur de couplage maximale. Si le nombre d'entités dans un cluster est égale au nombre maximum m, sa cohésion est réinitialisée par la valeur zéro, et les clusters sont réordonnés. Le pseudo-code de cet algorithme est présenté ci-dessous :

```
Algorithme(S,k,m)
Initialisercohesion(S);//initialiser chaque Si dans S
                       //par sa valeur de cohésion.
Ordonner(S);//ordonner les Si selon leurs cohésions .
Initialisercouplage(S);//dans une structure de type
                       //matrice
Tant que le nombre courant de cluster l > k  faire
   Prendre Si ;//le premier cluster dans S
   Choisir Sj ;//le cluster qui a la valeur de couplage
               //maximale avec Si et |Si|+|Sj| <= m*2
   Fusionner les deux cluster Si , Sj dans St;
   Supprimer Si et Sj ;
   Si |St| < m*2 faire// cluster des paires d'ancres
     Mettre à jour la cohésion de St ,
   Sinon
     Mettre la cohésion de St à zéro ,
   Finsi
   Ordonner(S),
   Mettre à jour la matrice de couplage,
   l:=l-1,
 Fin tant que
```

L'algorithme fournit en sortie un ensemble de k clusters d'ancres comme le montre la Fig. 2.

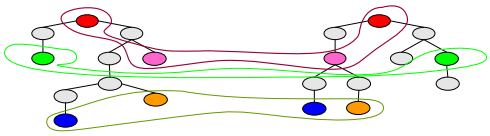

Fig. 2. La sortie de l'algorithme de partitionnement préliminaire (k clusters d'ancres)

3.3 Identifier les clusters d'ancres de chaque ontologie

Après la classification des ancres en K clusters d'ancres, nous partons de ces clusters et nous les divisons (voir Fig. 3) en des clusters qui ne contiennent que les ancres d'une même ontologie.

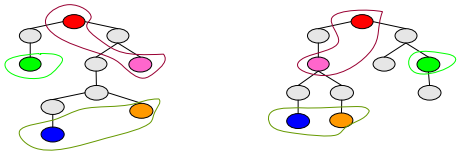

Fig. 3. Les clusters d'ancres de chaque ontologie.

3.4 Partitionnement autour des ancres

Dans cette étape, les concepts sont classifiés progressivement autour des ancres.

L'algorithme. La classification consiste à agréger progressivement dans chaque itération les concepts autour des ancres selon leur ressemblance en deux étapes :

- Etape d'affectation : pour chaque concept, on détermine le cluster d'ancres auquel on doit l'affecter (le cluster le plus proche c.à.d. où le couplage est le plus grand).
- Etape de représentation : pour chaque cluster défini, on recalcule les nouveaux couplages.

L'algorithme fournit en sortie un ensemble de blocs où chaque bloc contient une ou plusieurs ancres comme présenté en Fig. 4.

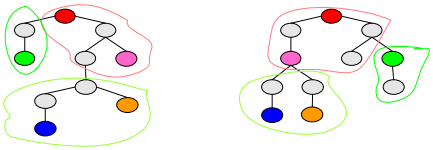

Fig. 4. Les blocs générés après la classification

Dans la phase d'alignement, chaque bloc de la première ontologie ne s'aligne qu'avec un seul bloc de la deuxième ontologie (voir la Fig. 5).

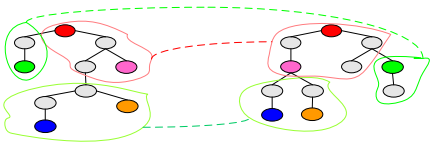

Fig. 5. Les blocs alignés après la classification

4 Evaluation de l'algorithme

Nous avons implémenté l'algorithme présenté ci-dessus sous forme d'un module java dans un système d'alignement. Dans l'évaluation de notre algorithme, nous avons choisi deux paires d'ontologies : Russia12 et TourismAB. Ce choix se justifie par :

- la taille des deux ontologies est modérée ;
- les fichiers de références, qui contiennent les entités classifiées en blocs, sont disponibles ;
- la comparaison des résultats de notre algorithme de partitionnement avec ceux des autres systèmes testés sur le même jeux de test.

Dans ce que suit, nous décrivons brièvement les deux paires de jeu de test.

Russia12. Les deux ontologies sont créées séparément par différentes personnes de deux contenus de sites web de voyages de la Russie. Russia1 contient 151 classes, 76 propriétés et 22 blocs dans son fichier de référence. Russia2 contient 162 classes, 81 propriétés et 22 blocs dans son fichier de référence.

TourismAB. Les deux ontologies sont créées séparément par différentes communautés décrivant le domaine du tourisme de MecKlenburg-Vorpommern (Allemagne). TourismA contient 340 classes, 97 propriétés et 18 blocs dans son fichier de référence. TourismB contient 447 classes, 100 propriétés et 23 blocs dans son fichier de référence.

Notre algorithme est testé sur une machine de technologie Intel Core 2 Duo CPU 2 GHz avec une mémoire de 3 GB DDR2, sur un système d'exploitation de Windows Vista, et un compilateur java 1.6.

4.1 Métrique d'évaluation

Nous utilisons la métrique d'entropie pour comparer les blocs générés automatiquement avec les blocs de référence [10].

Soit B l'ensemble de blocs générés automatiquement (|B|=n) et R l'ensemble de blocs de référence (|R|=m). b_i est un bloc dans B, tandis que r_j est un bloc dans R. $|b_i|$ est le nombre d'entités dans b_i , et $|r_j|$ est le nombre d'entités dans r_j .$b_i \cap r_j$ présente les entités communes dans les deux blocs b_i et r_j. Nous décrivons l'opération de base

appelée "prec" qui mesure la précision des blocs générés par rapport aux blocs de référence.

$$prec(b_i, r_j)=\frac{|b_i \cap r_j|}{b_i} \quad . \tag{9}$$

L'entropie mesure la distribution des entités dans les blocs et reflète la qualité de partitionnement. Un score faible de l'entropie indique une meilleure qualité de partitionnement. Le meilleur score est donc, égale à 0 et le mauvais est égal à 1. L'entropie de l'ensemble B est définie de la manière suivante :

$$\text{entropy (B)}=\frac{1}{\sum_{i=1}^{n}|b_i|}\sum_{i=1}^{n}\text{entropy}(b_i).|b_i|. \tag{10}$$

$$\text{entropy}(b_i) = \frac{1}{\log m}\sum_{j=1}^{n}\text{prec}(b_i, r_j).\log(\text{prec}(b_i, r_j)). \tag{11}$$

L'histogramme de la figure 6 présente les résultats d'évaluation de quelques systèmes (PBM, BMO,COMA) en utilisant cette métrique [10,9, 3]. Le critère d'arrêt du partitionnement est l'obtention du nombre de blocs de référence. L'histogramme présenté ci-dessous indique que les résultats d'expérimentation de PMB est dominant par rapport aux autres systèmes dans tous les tests.

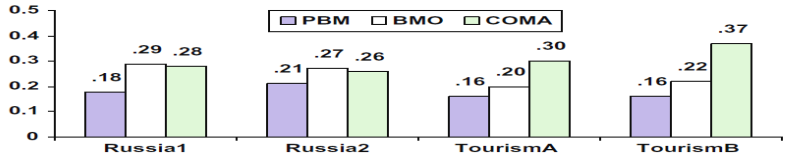

Fig. 6. L'entropie de PBM (Falcon-AO), BMO et COMA [10].

Notre algorithme de partitionnement est orienté alignement. La phase de partitionnement préliminaire des ancres est la plus importante car les blocs sont générés autour des ancres. Notre objectif est de partitionner les ontologies de façon à n'avoir aucun bloc isolé et diminuer le nombre de combinaisons à faire lors du processus d'alignement (un bloc de la première ontologie ne s'aligne qu'avec un seul bloc de la deuxième). En sortie, le nombre de blocs générés est exactement égal au nombre de bloc d'ancres généré lors de la phase de partitionnement préliminaire. Cependant, les blocs de référence disponibles sont construits manuellement sans prendre en compte l'objectif de l'alignement. Pour cela, nous adaptons les fichiers de référence aux fichiers de référence d'ancres pour évaluer notre algorithme de partitionnement préliminaire des ancres.

Dans notre expérimentation le nombre de blocs de référence après l'adaptation est comme suit : 13 blocs pour Russia1, 13 pour Russia2, 11 pour TourismA, et 14 pour TourismB. On a 55 ancres dans Russia12 et 144 dans TourismAB.

4.2 Démarche expérimentale

Les expérimentations que nous avons menées ont eu pour objectifs de calculer l'entropie de notre algorithme de partitionnement. Pour chaque paire d'ontologie, l'algorithme de partitionnement est exécuté plusieurs fois avec plusieurs valeurs présentant le nombre de blocs que l'on souhaite obtenir en sortie.

L'histogramme de la figure 7 présente les résultats d'évaluation de notre algorithme.

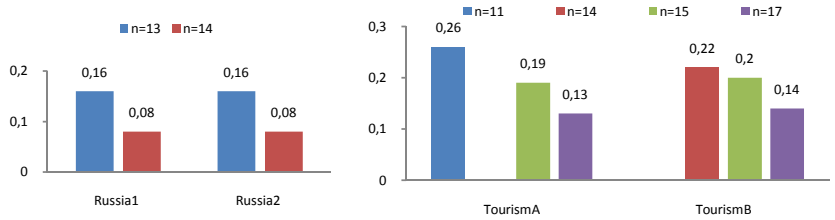

Fig. 7. L'entropie de l'algorithme de partitionnement d'ancres (les deux paires d'ontologies Russia12 et TourismAB)

Les résultats renvoyés par l'algorithme sont bons. Dans Russia12, si le nombre de blocs générés est égal au nombre de blocs de référence (n=m) l'entropie est égale à 0.16. Si nous augmentons le nombre de blocs générés, la valeur d'entropie diminue (0.08). Pour l'ontologie TourismAB, nous avons appliqué plusieurs valeurs du nombre de blocs générés notamment celui de référence (11 et 14) ainsi que les valeurs 15 et 17 (voir Fig. 7) et ceci afin de montrer l'influence de la valeur du nombre de blocs générés sur la qualité de partitionnement. Nous observons que l'entropie est changée en fonction du nombre de blocs générés choisi. Pour cela, le nombre de blocs a été choisi de façon à n'avoir aucun impact négatif sur l'alignement (i.e. la complexité de l'algorithme et le temps d'exécution). Les valeurs de l'entropie (0.26, 0.22) s'expliquent par le fait que le calcul de similarité structurelle se base seulement sur la taxonomie décrite par l'ontologiste et ignore les liens prédéfinis entre les classes et les propriétés OWL. Par exemple, les deux propriétés suivantes sont des ancres dans l'ontologie TourismA.

```
<rdf:Description rdf:about='http://meh/tourism1#DEFAULT_ROOT_RELATION'>
   <rdf:type rdf:resource='http://www.w3.org/2002/07/owl#ObjectProperty'/>
  </rdf:Description>
  <rdf:Description
rdf:about='http://meh/tourism1#DEFAULT_ROOT_DATATYPE_RELATION'>
   <rdf:type rdf:resource='http://www.w3.org/2002/07/owl#DatatypeProperty'/>
  </rdf:Description>
```

Elles sont classifiées dans deux clusters distincts car notre algorithme ne découvre pas le lien structurel à partir de leurs types (les deux propriétés DEFAULT_ROOT_RELATION et DEFAULT_ROOT_DATATYPE_RELATION' sont sous-propriétés de Property).

5 Conclusion

Dans le but de relever le challenge du passage à l'échelle des méthodes d'alignement, nous avons proposé une méthode de partitionnement des ontologies larges.

Notre algorithme de partitionnement est orienté alignement. Les blocs sont générés autour des ancres afin de n'avoir aucun bloc isolé. Nous poursuivons actuellement les expérimentations pour tester l'efficacité de l'algorithme avec notre méthode d'alignement structurelle à grande échelle. Cet algorithme sera appliqué sur les ontologies réelles, complexes, et de forte hétérogénéité.

Références

1. Agricultural Information Management Standards, http://www.fao.org/aims/
2. Bach, T.L.: Construction d'un Web Sémantique Multi-Points de Vue . These de doctorat Informatique, Ecole des Mines de Paris a Sophia Antipolis (2006)
3. Do, H.H. , Rahm, E.: Matching Large Schemas: Approaches and evaluation, Information Systems, vol.32 n.6, pp.857--885. (2007)
4. Ehrig, M., Staab, S.: QOM - Quick Ontology Mapping. In International Semantic Web Conference , pp. 683--697. (2004).
5. Euzenat, J., Valtchev, P.: Similarity-based Ontology Alignment in OWL-lite. In Proc. 15th ECAI, pp. 333--337. Valencia (ES), (2004)
6. Eyal, A., Sheila, M.: Partition-based Logical Reasoning for First-Order and Propositional theories, Artificial Intelligence, Vol. 162, pp. 49--88. (2005)
7. Guha, S., Rastogi, R., Shim, K.: ROCK: a Robust Clustering Algorithm for Categorical Attributes, in: Proceedings of the 15th International Conference on Data Engineering, pp. 512--521. (1999)
8. Hamdi, F., Zargayouna, H.,Safar, B., Reynaud, C.: TaxoMap in the OAEI Alignment Contest In Ontology Alignment Evaluation Initiative (OAEI) 2008 Campaign, 8p, Workshop ISWC'08, Karlsruhe, Germany, (2008)
9. Hu, W., Qu, Y.: Block Matching for Ontologies. In: LNCS, vol. 4273. Springer. pp. 300--313. (2006)
10. Hu, W., Qu, Y., Cheng, G.: Matching Large Ontologies: A Divide-and-Conquer Approach. Journal on Data and Knowledge Engineering, vol. 67, n°1, p.140--160. (2008)
11. Jianbo, S., Jitendra, M.: Normalized Cuts and Image Segmentation. IEEE Transactions on PAMI, Vol. 22, No. 8, pp. 888--905. (2000)
12. Pham, T.A.L., Thanh, N. L., Sander, P.: Some Approaches of Ontology Decomposition in Description Logics. 14th ISPE International Conference on Concurrent Engineering, CE2007, São José dos Campos, SP, Brazil, (2007)
13. Stuckenschmidt, H., Klein, M.: Structured-based Partitioning of Large Concept Hierarchies. In International Semantic Web Conference- ISWC, pp. 289--303. (2004).
14. The Gene Ontology, http://www.geneontology.org
15. Winkler W. E.: The State of Record Linkage and Current Research Problems. Statistics of Income Division, Internal Revenue Service Publication R99/04, (1999)
16. Wu, Z., Palmer, M.: Verb Semantics and Lexical Selection, Proceedings of the 32nd Annual Meetings of the Associations for Computational Linguistics, pp. 133--138. (1994).