Contributions au problème d'hétérogénéité sémantique dans les systèmes pair-à-pair : application à la recherche d'information

Thomas Cerqueus

UNIVERSITÉ DE NANTES
FACULTÉ DES SCIENCES ET DES TECHNIQUES

ÉCOLE DOCTORALE
SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET MATHÉMATIQUES

Année 2012

N° attribué par la bibliothèque

Contributions au problème d'hétérogénéité sémantique dans les systèmes pair-à-pair : application à la recherche d'information

---

THÈSE DE DOCTORAT
Discipline : Informatique
Spécialité : Informatique

Présentée
et soutenue publiquement par

Thomas CERQUEUS

le 15 novembre 2012 au LINA, devant le jury ci-dessous

Président	Gabriel ANTONIU, Directeur de recherche	INRIA Rennes
Rapporteurs	Sylvie CALABRETTO, Professeur	INSA de Lyon
	Abdelkader HAMEURLAIN, Professeur	Université Toulouse III
Examinateurs	Marc GELGON, Professeur	Université de Nantes
	Philippe LAMARRE, Professeur	INSA de Lyon
	Sylvie CAZALENS, Maître de Conférence	Université de Nantes

Directeur de thèse : Philippe LAMARRE
Co-encadrant de thèse : Sylvie CAZALENS

Contributions au problème d'hétérogénéité sémantique dans les systèmes pair-à-pair : application à la recherche d'information

Contributions to the problem of semantic heterogeneity in peer-to-peer systems : application to information retrieval

---

Thomas CERQUEUS

thomas.cerqueus@univ-nantes.fr
Équipe Gestion de Données Distribuées
Laboratoire d'Informatique de Nantes Atlantique, UMR 6241

Table des matières

Liste des figures

Liste des tableaux

Première partie

Introduction

Contexte général

Ces quarante dernières années l'informatique a connu deux grandes évolutions majeures. D'une part, la démocratisation de l'ordinateur personnel, dans les années 1970, a permis au grand public de consulter, de manipuler et de créer du contenu numérique. D'autre part, l'apparition du Web dans les années 1990, a permis de créer du contenu et de le partager avec les autres utilisateurs du Web. Ce mode de partage a rendu disponible une grande quantité de données sur le Web. Pour accéder à ces données, de nombreux outils ont été proposés, en particulier des moteurs de recherche tels que Google [PBMW99], Yahoo ! [MPR00], etc. Ces derniers permettent aux internautes de formuler des requêtes, souvent à l'aide de mots-clés, et de récupérer un ensemble de documents. Les outils de recherche à base de mots-clés souffrent de problèmes liés à la linguistique : synonymie, polysémie, etc. Pour palier ces problèmes, une solution consiste à considérer que les documents et les requêtes peuvent être représentés à l'aide de concepts issus d'une ontologie. Une ontologie est une structure permettant de représenter des connaissances en définissant des concepts et en liant les concepts par des relations [SS04, Pow03]. Bien qu'elle nécessite d'annoter ou d'indexer sémantiquement les documents et les requêtes, cette approche a l'avantage de considérer le sens des concepts plutôt que seulement leurs représentations, et les relations entre les différents concepts. Les ontologies permettent de raisonner sur les concepts qui les composent. Par exemple, il est possible de développer des méthodes d'expansion de requêtes : les requêtes sont enrichies par des concepts proches de ceux considérés initialement [Voo94, VCLV08].

Le Web fonctionne sur le modèle client/serveur : les données sont stockées sur des serveurs et les clients accèdent aux serveurs pour les consulter. Ce paradigme possède un certain nombre de limites. Premièrement il n'est pas tolérant aux pannes car la défaillance d'un serveur rend indisponible l'ensemble des données qui s'y trouvaient. Deuxièmement, lorsqu'un utilisateur publie une donnée sur un serveur, il en perd le contrôle : il est désormais tributaire du serveur. Troisièmement, le fait que de nombreux clients se connectent au même serveur peut conduire au phénomène de goulot d'étranglement. La réplication des données sur différents serveurs permet de palier partiellement les problèmes de tolérance aux pannes et de passage à l'échelle. Néanmoins cette approche oblige à maintenir la cohérence entre les données stockées sur les différents serveurs. De plus, les coûts d'achat, de maintenance et la consommation énergétique pour ces différents serveurs peuvent être importants. Les systèmes pair-à-pair (P2P) représentent une alternative au modèle client/serveur [SGG03, SMLN+03, Coh03, ZHS+04]. Dans ce modèle chaque participant, appelé *pair* ou *nœud*, joue à la fois le rôle de client et le rôle de serveur. Les systèmes P2P présentent des propriétés intéressantes : passage à l'échelle, tolérance aux pannes, autonomie des pairs, gestion de la dynamicité.

Dans des systèmes à grande échelle comme le Web, les utilisateurs ont souvent des objectifs, des points de vue, des niveaux d'expertise différents [ELBB+04, HPS04, SE08]. Même s'ils sont intéressés pour partager des documents sur un domaine précis, il est peu probable qu'ils ar-

rivent à s'entendre sur l'usage d'une ontologie universelle pour représenter leurs données. Dans le cas où plusieurs ontologies sont utilisées au sein d'un système, nous parlons d'*hétérogénéité sémantique*. L'hétérogénéité sémantique est un frein à l'interopérabilité car les requêtes émises par les pairs peuvent être incomprises par les autres pairs. Les méthodes d'alignements d'ontologies permettent d'identifier des correspondances entre les concepts de deux ontologies [ES07, ADMR05, MCH09]. Ces correspondances peuvent alors être utilisées pour permettre à deux participants de communiquer bien qu'ils utilisent des ontologies différentes. Néanmoins, dans les systèmes P2P, il se peut qu'un pair ne reçoive aucune réponse à une requête soit parce qu'il n'existe pas suffisamment de correspondances entre son ontologie et celles des pairs qui la reçoivent, ou soit parce que les pairs capables de la comprendre ne sont pas contactés pour la traiter.

Dans ce travail, nous visons des applications dans lesquelles des participants veulent partager des données. Par exemple, il peut s'agir de chercheurs de différents domaines (biologie, géologie, bio-informatique, chimie, informatique, etc.) qui souhaitent mettre à disposition des articles ou rapports scientifiques, des supports de présentation, des données ou des résultats d'expérimentations, tout en gardant un contrôle sur ce qu'ils partagent. Nous supposons que chaque participant annote ses documents par rapport à une ontologie qui permet de modéliser son domaine et qui correspond à ses objectifs et ses besoins. Comme les participants ont des profils différents, nous sommes obligés de considérer qu'ils utilisent des ontologies différentes. Dans ce contexte, nous souhaitons définir un système dans lequel ils peuvent rechercher de nouveaux documents bien que le système soit sémantiquement hétérogène.

L'objectif de ce travail est de concevoir un système permettant à des participants de partager des documents bien qu'ils puissent utiliser des ontologies qui leur sont propres. Nous nous intéressons plus précisément à la recherche d'information car elle représente une fonctionnalité centrale dans le partage de données. Nous voulons que le système proposé soit tolérant aux pannes (cela exclut donc toute centralisation de données dans le système), qu'il passe à l'échelle, qu'il garantisse l'autonomie des participants, et qu'il permette aux participants de rejoindre ou quitter le système à tout moment.

Problématiques

La problématique générale est de définir un système dans lequel les participants peuvent interopérer bien qu'ils utilisent différentes ontologies pour représenter leurs documents et leurs requêtes. Dans notre contexte, nous voulons permettre aux participants d'émettre des requêtes et d'obtenir les documents pertinents bien que le système soit sémantiquement hétérogène. Pour adresser ce problème, nous proposons de le décomposer en trois différentes sous-problématiques.

Les problématiques que nous abordons dans ce travail sont principalement liées à l'hétérogénéité sémantique des systèmes P2P non-structurés. Dans un premier temps, nous pensons qu'il faut étudier la notion d'hétérogénéité sémantique. Il nous semble crucial de pouvoir la caractériser finement afin de mieux l'appréhender. La première problématique est

donc la suivante :
Comment caractériser l'hétérogénéité sémantique d'un système P2P non-structuré ?

Nous pensons que plus l'hétérogénéité sémantique est importante, plus il est difficile d'assurer l'interopérabilité au sein du système. Il semble donc important de considérer des systèmes dans lesquels l'hétérogénéité sémantique est la plus faible possible. Par conséquent, il est naturel de vouloir réduire l'hétérogénéité d'un système. La deuxième problématique abordée est :
Comment réduire l'hétérogénéité sémantique d'un système P2P ?

Enfin la dernière problématique que nous abordons concerne l'interopérabilité. En effet, il faut disposer d'un mécanisme permettant de router les requêtes dans un système P2P non-structuré dans lequel tous les pairs n'utilisent pas la même ontologie. Ce mécanisme doit garantir au pair initiateur d'une requête qu'il récupère les documents pertinents stockés par les pairs qui l'entourent. Comme il ne semble pas raisonnable de renvoyer tous les documents pertinents à l'initiateur de la requête (car celui-ci peut se retrouver submergé par une grande quantité de documents) nous considérons des requêtes top-k, c.-à-d. des requêtes retournant à l'initiateur les k documents les plus pertinents [CW04, MTW05, APV06]. La principale difficulté liée au contexte hétérogène réside dans le fait que la requête doit être traduite par certains pairs pour la traiter et qu'il faut ensuite être capable de comparer les valeurs de pertinences attribuées aux documents même si ces derniers ne sont pas représentés avec la même ontologie. La troisième problématique est donc la suivante :
Comment assurer le routage et le traitement de requêtes top-k dans les systèmes P2P non-structurés sémantiquement hétérogènes ?

La section suivante résume nos contributions pour répondre à ces différentes problématiques.

Contributions

Nous proposons une approche originale pour résoudre le problème d'interopérabilité dans les systèmes P2P non-structurés sémantiquement hétérogènes.

Dans un premier temps, nous étudions l'hétérogénéité. Selon nous, elle dépend au moins du nombre d'ontologies utilisées, des disparités entre les pairs et de la topologie du système. Étant données ces différentes facettes, nous proposons un ensemble de mesures qui permettent de caractériser finement l'hétérogénéité. Ces mesures doivent servir à déterminer l'hétérogénéité d'un système à un instant précis. Elles doivent également permettre d'évaluer des méthodes qui visent à réduire l'hétérogénéité.

Dans un deuxième temps, nous proposons le protocole CorDis qui permet de réduire l'hétérogénéité liée aux disparités entre pairs. Il s'agit d'un protocole épidémique qui dissémine les correspondances connues par les pairs dans le système. Après un certain nombre d'échanges, les pairs apprennent de nouvelles correspondances et l'hétérogénéité sémantique est réduite. Nous étudions le cas où un pair apprend une correspondance qui est incohérente par rapport à l'ontologie qu'il utilise. Nous proposons un ensemble de stratégies qu'il peut appliquer pour gérer cette

situation. Nous montrons par une série d'expérimentations utilisant 149 ontologies du domaine biomédical, obtenues sur BioPortal, que le protocole CorDis permet de réduire l'hétérogénéité d'un système et qu'il reste efficace lorsque le nombre de pairs augmente. Nous montrons également qu'il est adapté aux systèmes dynamiques, c'est-à-dire lorsque les pairs rejoignent et quittent le système fréquemment.

Dans un troisième temps, nous définissons le protocole GoOD-TA qui permet de réduire l'hétérogénéité sémantique liée à la topologie du système. Comme CorDis, il s'agit d'un protocole épidémique. Il permet aux pairs de choisir comme voisins les pairs dont ils sont proches sémantiquement. Dans ce protocole, les pairs s'échangent régulièrement des descripteurs d'autres pairs et peuvent ainsi découvrir l'existence de pairs avec qui ils sont plus aptes à interopérer. Nous proposons deux versions du protocole. Dans la première version, les descripteurs des pairs sont relativement simples et permettent de rapprocher les pairs qui utilisent des ontologies identiques. Cette approche n'est pas adaptée lorsque de nombreuses ontologies différentes sont utilisées dans le système. Dans la deuxième version, les descripteurs prennent en compte le nombre de correspondances connues entre les ontologies des pairs. De cette manière, deux pairs ont tendance à se rapprocher s'ils utilisent la même ontologie ou s'il existe de nombreuses correspondances entre leurs ontologies. Nous proposons des mécanismes pour gérer l'évolution des connaissances des pairs (le fait qu'ils puissent changer d'ontologie ou découvrir de nouvelles correspondances), pour gérer la dynamicité du système et pour limiter le trafic réseau. Nous montrons expérimentalement que les deux versions du protocole GoOD-TA réduisent l'hétérogénéité sémantique liée à l'organisation du système. Encore une fois nous considérons des ontologies obtenues sur BioPortal. Les expérimentations montrent que GoOD-TA est adapté aux systèmes dynamiques, qu'il gère le cas où des pairs peuvent changer d'ontologie, et que le mécanisme de diminution du trafic réseau est efficace : il réduit le trafic et ne pénalise pas la réduction de l'hétérogénéité.

Enfin nous présentons l'algorithme DiQuESH qui permet de router et de traiter des requêtes top-k dans les systèmes P2P non-structurés sémantiquement hétérogènes. Dans DiQuESH, la requête originale est transmise à tous les pairs qui font partie du voisinage du pair initiateur. Ils sont alors responsables de traduire la requête localement si cela est nécessaire. Le calcul de pertinence d'un document par rapport à la requête prend en compte l'erreur commise lors de la traduction de la requête. De cette manière, toutes les valeurs de pertinences sont comparables et il est alors possible d'ordonner les documents suivant leurs pertinences.

L'ensemble des contributions ont été publiées dans des conférences nationales et internationales. La liste des publications est présentée dans l'annexe A.1 (page 161).

Organisation du document

Dans un premier temps nous présentons un état de l'art concernant les différents domaines sur lesquels portent ce travail :
Chapitre 1 : État de l'art concernant la sémantique : ontologies, alignements et mesures sémantiques.

Chapitre 2 : État de l'art concernant les systèmes P2P. Nous présentons les différents types d'architectures, et nous décrivons le principe général des protocoles épidémiques.
Chapitre 3 : Présentation des concepts relatifs à la recherche d'information (RI). Nous présentons différents modèles de RI, et les éléments clés permettant d'évaluer une méthode de RI (corpus de test et mesures).
Chapitre 4 : État de l'art concernant les systèmes distribués de RI et les systèmes distribués hétérogènes de gestion de données.

Dans un second temps nous présentons nos contributions. Chacun des quatre chapitres correspond à une contribution :
Chapitre 5 : Définition de mesures d'hétérogénéité sémantique de systèmes P2P non-structurés, et définition de mesures de disparité sémantique entre pairs.
Chapitre 6 : Définition du protocole CorDis permettant de réduire l'hétérogénéité sémantique liée aux disparités entre pairs.
Chapitre 7 : Définition du protocole GoOD-TA permettant de réduire l'hétérogénéité sémantique liée à l'organisation du système.
Chapitre 8 : Définition de l'algorithme de routage DiQuESH permettant l'évaluation de requêtes top-k dans les systèmes P2P non-structurés sémantiquement hétérogènes.

Naturellement la dernière partie présente un rappel des contributions et ouvre quelques perspectives de travail.

Deuxième partie

État de l'art

1 Sémantique : ontologies, alignements d'ontologies et mesures sémantiques

Dans ce chapitre nous présentons les notions qui concernent la sémantique et que nous utilisons dans notre travail. Dans un premier temps nous présentons les notions liées aux ontologies. Nous commençons par donner différentes définitions de ce qu'est une ontologie en science de l'information. Nous présentons ensuite les langages RDF(S) et OWL. Ces derniers sont des recommandations du World Wide Web Consortium (W3C) et sont centraux au Web sémantique. Ensuite nous montrons qu'il n'est pas envisageable de considérer que tous les utilisateurs d'une communauté, et à plus forte raison d'une communauté à l'échelle du Web, puisse s'accorder sur l'utilisation d'une seule ontologie.

Nous abordons ensuite la notion d'hétérogénéité. Nous listons les différents niveaux d'hétérogénéité et nous nous focalisons sur les sources de l'hétérogénéité sémantique, c'est-à-dire l'hétérogénéité qui apparaît au niveau conceptuel.

Nous présentons alors un certain nombre de travaux sur l'alignement d'ontologies dont l'objectif est de produire des correspondances entre entités de différentes ontologies. Un alignement entre deux ontologies permet aux participants qui les utilisent de communiquer dans le sens où ils peuvent échanger des requêtes exprimées avec leurs connaissances respectives.

Enfin nous présentons des travaux relatifs aux mesures sémantiques. D'abord nous nous focalisons sur des mesures donnant la similarité entre deux concepts d'une même ontologie. Ensuite nous présentons des mesures de distance entre ontologies. Certaines d'entre elles sont purement syntaxiques et permettent de déterminer s'il est raisonnable de calculer un alignement. D'autres considèrent des alignements et permettent de déterminer à quel point ces alignements permettent aux participants qui les utilisent de communiquer.

1.1 Des vocabulaires contrôlés aux ontologies

La richesse des langues naturelles fait qu'il est difficile, pour une machine, d'interpréter le contenu d'un texte. Pour faciliter la recherche d'information dans une collection de documents, il est possible de définir un vocabulaire précis et prédéfini pour caractériser ces données : c'est ce qu'on appelle un **vocabulaire contrôlé**. Par exemple, dans une librairie les livres peuvent être décrits suivant leur genre littéraire : roman, roman policier, pièce de théâtre, comédie, essai, etc. Ainsi *Les fourberies de Scapin* de Molière peuvent être décrites par les termes *pièce de théâtre* et *comédie*. Lorsque les termes de ce vocabulaire sont hiérarchisés, il s'agit d'une **taxonomie**. Une taxonomie permet de préciser qu'un roman policier est un type de roman, et qu'une comédie est une pièce de théâtre. De cette manière, annoter l'œuvre de Molière avec le terme comédie suffit pour expliciter le fait qu'il s'agit d'une pièce de théâtre. Dans une taxonomie, les termes sont hiérarchisés suivant la relation de subsomption (relation « est un »). Un **thésaurus** est un type de taxonomie dans lequel les termes sont organisés suivant divers types de relations (subsomption, synonymie, association, etc.). Les thésaurus permettent de décrire les concepts décrivant des domaines de connaissance. En ce sens ils se rapprochent de ce que sont les ontologies. Les **graphes conceptuels** permettent de représenter les connaissances sous forme de graphes étiquetés [CM08a] dans lesquels les nœuds sont des concepts, des propriétés de concepts, ou des instances de concepts, et les arcs sont des relations entre les nœuds. Les mécanismes de raisonnement sont basés sur l'homomorphisme des graphes. Les graphes conceptuels permettent d'interroger des bases de connaissances et de visualiser graphiquement les résultats. En science de l'information une **ontologie** est une structure permettant de représenter des connaissances.

1.2 Ontologies

En sciences de l'information, les ontologies ont initialement été utilisées en intelligence artificielle pour permettre à différents agents d'interopérer. L'une des premières définitions a été donnée par Thomas Gruber en 1993 : « Une ontologie est une spécification explicite d'une conceptualisation » [Gru93]. Une ontologie sert à décrire une partie du monde réel d'un point de vue particulier. En 1997, Borst définit une ontologie comme étant une « spécification formelle d'une conceptualisation partagée » [Bor97]. Cette définition suggère qu'une ontologie doit représenter le point de vue de plusieurs parties, et non d'un seul individu. Guarino et Giaretta proposent de définir une ontologie comme étant « une théorie logique qui définit explicitement une description partielle d'une conceptualisation » [GG95]. Ils insistent sur le fait qu'une ontologie ne peut pas être une représentation de la réalité dans le sens où celle-ci peut être perçue différemment selon les points de vue, les cultures, etc. Toutes ces définitions font apparaître la notion de *conceptualisation*. Selon Guarino, Oberle et Staab « une conceptualisation est une vue abstraite et simplifiée du monde que l'on souhaite représenter dans un certain but » [GOS09]. Dans [Hep08], Hepp insiste sur le fait qu'une ontologie n'est pas une base de connaissance. Une ontologie définit le vocabulaire et la spécification

formelle permettant de construire une base de connaissance : elle contient des concepts et des instances. Par exemple les concepts *homme* et *femme* peuvent être contenus dans une ontologie, mais les instances *Pierre*, *Paul* et *Jacques* (qui sont des instances d'*homme*) doivent être stockés dans une base de connaissance. En réalité, lorsqu'une ontologie contient des instances, nous parlons d'*ontologie peuplée* [AEPR11].

Les langages ontologiques doivent répondre à plusieurs exigences. Ils doivent :

- être bien définis syntaxiquement : c'est une condition nécessaire à tout langage devant être interprété par une machine,
- être bien définis sémantiquement : ils ne doivent pas être interprétables de différentes manières par différentes personnes, ou différents agents. La sémantique doit être sans ambiguïté,
- présenter un support de raisonnement efficace : ils doivent être construits de manière à faciliter le raisonnement, la déduction, etc.,
- être suffisamment expressifs : ils doivent permettre de définir des notions complexes,
- faciliter la définition d'ontologies : ils doivent être suffisamment simples pour que les concepteurs puissent définir des ontologies "facilement".

1.2.1 RDF(S) : Resource Description Framework (Schema)

Le ressource description framework (RDF) a été développé par le W3C pour permettre d'annoter sémantiquement le Web dit *syntaxique* [Pano9, Pow03]. L'objectif est de rendre le Web compréhensible non pas seulement par les humains, mais également par les machines (ou agents intelligents). Les ressources Web (pages, services, etc.) sont annotées à l'aide de triplets RDF. Un graphe RDF est un ensemble de triplets RDF, chaque triplet étant de la forme :

$$(\text{sujet} \;\; \text{propriété} \;\; \text{objet})$$

Le sujet et la propriété sont des ressources, c'est-à-dire qu'elles sont identifiées de manière unique (soit par un URI, soit par un URIref). L'objet peut être une ressource ou un littéral (une chaîne de caractères, un nombre entier, etc.).

Le framework RDF définit le langage RDF/XML permettant de représenter des triplets RDF. La figure 1.1 présente un ensemble de triplets RDF décrivant la ressource `http://example.org/People#John`. Le titre de cette ressource est *All about John Doe* et elle a été crée par *Jane Doe*. La figure 1.2 présente le même ensemble d'annotations dans la syntaxe RDF/XML.

```
@prefix rdf: <http://www.w3.org/1999/02/22-ref-syntax-ns#>
@prefix ex: <http://example.org/People#>

ex:John ex:titre "All about John Doe" .
ex:John ex:createur "Jane Doe" .
```

FIGURE 1.1 – *Annotations RDF.*

Le W3C a développé le langage RDFS (RDF Schema). Celui-ci prédéfinit des ressources permettant de créer des classes (`rdfs:Class`), des

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:ex="http://example.org/People#">
  <rdf:Description rdf:about="ex:John">
    <ex:titre>All about John Doe</ex:titre>
    <ex:createur>Jane Doe</ex:createur>
  </rdf:Description>
</rdf:RDF>
```

FIGURE 1.2 – *Annotations RDF exprimées en RDF/XML.*

propriétés (`rdfs:Property`) et des ressources (`rdfs:Resource`). Il définit également un ensemble de méta-propriétés :

- `rdf:type` : permet d'instancier une ressource,
- `rdfs:subClassOf` : correspond à la relation de subsomption entre deux classes,
- `rdfs:subPropertyOf` : correspond à la relation de subsomption entre deux propriétés,
- `rdfs:domain` : domaine d'une propriété,
- `rdfs:range` : codomaine d'une propriété.

La figure 1.3 présente une ontologie RDFS contenant trois classes : les classes *Homme* et *Femme* sont des sous-classes de *Personne*. Trois propriétés sont définies : *age* (entre une personne et un nombre entier), *mere* (entre une personne et une femme) et *enfant* (entre deux personnes).

Il est possible d'utiliser des schémas de métadonnées prédéfinis pour annoter des ressources. Par exemple le Dublin Core Metadata Element Set regroupe quinze propriétés permettant d'annoter des ressources bibliographiques. Il contient les propriétés `dc:Title`, `dc:Creator`, `dc:Date`, etc.

```
@prefix rdf: <http://www.w3.org/1999/02/22-ref-syntax-ns#>
@prefix ex: <http://example.org/People#>

ex:Personne rdf:type rdfs:Class .
ex:Homme rdf:type rdfs:Class ;
         rdfs:subClassOf ex:Personne .
ex:Femme rdf:type rdfs:Class ;
         rdfs:subClassOf ex:Personne .
ex:age rdf:type rdfs:Property ;
         rdfs:domain ex:Personne ;
         rdfs:range http://www.w3.org/2001/XLMSchema#Integer .
ex:mere rdf:type rdfs:Property ;
         rdfs:domain ex:Personne ;
         rdfs:range ex:Femme .
ex:enfant rdf:type rdfs:Property ;
         rdfs:domain ex:Personne ;
         rdfs:range ex:Personne .
```

FIGURE 1.3 – *Une ontologie RDFS.*

Le langage SPARQL permet d'interroger (ou de modifier) les bases de triplets RDF, aussi appelés *triplestores* [PS06, HBS09]. Depuis Janvier 2008, SPARQL est devenu la recommandation officielle du W3C. La figure 1.4 présente un exemple de requête SPARQL. Celle-ci permet d'extraire l'âge de tous les hommes contenus dans une base de triplets.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ex: <http://example.org/People#>
SELECT DISTINCT ?age
WHERE {
 ?personne rdf:type ex:Homme.
 ?personne ex:age ?age
}
```

FIGURE 1.4 – *Une requête SPARQL retournant les âges de tous les hommes présents dans une base de triplets RDF.*

1.2.2 OWL : Web Ontology Language

L'expressivité des langages RDF et RDFS est limitée. RDFS permet seulement de définir une hiérarchie de classes (via la relation `subClassOf`), une hiérarchie de propriétés (via la relation `subPropertyOf`) possédant un domaine (`domain`) et un codomaine (`range`).

L'une des exigences des langages ontologiques concerne le fait qu'ils doivent permettre de définir des notions complexes [AvH09]. C'est un des reproches qui est fait à RDF. En particulier celui-ci ne présente pas les notions de :

- disjonction de classes : elle permet de définir que deux classes sont disjointes (par ex. les classes *Homme* et *Femme*),
- combinaison booléenne de classes : elle permet de définir une classe à partir d'autres classes en utilisant les opérateurs d'union, d'intersection et de complément (par ex. la classe *Personne* est l'union des classes *Homme* et *Femme*),
- restriction de cardinalités : elle permet de contraindre le nombre de valeurs que peut avoir une propriété (par ex. elle doit permettre d'exprimer le fait qu'une personne a deux parents).

Le langage OWL est composé de trois espèces : OWL Full, OWL DL et OWL Lite. La figure 1.5 présente une ontologie décrite en OWL Lite.

OWL Full est la version qui présente le plus haut niveau d'expressivité. Comme RDF, il permet de redéfinir les primitives du langage. OWL Full est parfaitement compatible (syntaxiquement et sémantiquement) avec RDF : toute ontologie valide en OWL Full est également valide en RDF(S). L'avantage de ce langage est sa forte expressivité. Néanmoins OWL Full ne garantit ni la complétude des raisonnements (toutes les inférences ne sont pas calculables), ni leur décidabilité (le calcul ne se fait pas nécessairement en un temps fini).

OWL DL est un sous-langage de OWL Full. Il est fondé sur la logique de description. Cela entraîne une perte au niveau de l'expressivité, mais cela garantit la complétude des raisonnements et leur décidabilité. Toute ontologie valide en OWL DL est également valide en OWL Full. L'inverse n'est pas vrai : il faut y appliquer des extensions ou des restrictions.

OWL Lite est un sous-langage de OWL DL qui limite l'utilisation de certains constructeurs. Par exemple OWL Lite interdit l'usage d'énumé-

rations, de déclaration de disjonction, etc. L'expressivité est plus faible que pour les langages précédents, mais OWL Lite est plus simple à manipuler. Il est particulièrement bien adapté à la transformation de thésaurus (comme WordNet [Mil95]) en ontologie car il permet essentiellement de définir une hiérarchie de concepts. Toute ontologie valide en OWL Lite est une ontologie valide en OWL DL.

```
<rdf:RDF>
 <owl:Class rdf:ID="Personne" />
 <owl:Class rdf:ID="Homme">
  <rdfs:subClassOf rdf:resource="#Personne" />
  <owl:disjointWith rdf:resource="#Femme" />
 </owl:Class>
 <owl:Class rdf:ID="Femme">
  <rdfs:subClassOf rdf:resource="#Personne" />
  <owl:disjointWith rdf:resource="#Homme" />
 </owl:Class>

 <owl:DatatypeProperty rdf:ID="age">
  <rdfs:domain rdf:resource="#Personne" />
  <rdfs:range rdf:resource="http://www.w3.org/2001/XLMSchema#Integer" />
 </owl:DatatypeProperty>
 <owl:ObjectProperty rdf:ID="mere">
  <rdfs:domain rdf:resource="#Femme" />
  <rdfs:range rdf:resource="#Personne" />
 </owl:ObjectProperty>

 <rdf:Description rdf:ID="John">
  <rdf:type rdf:resource="#Homme" />
 </rdf:Description>
</rdf:RDF>
```

FIGURE 1.5 – *Une ontologie exprimée en OWL Lite.*

En 2009, le W3C a proposé le langage OWL 2. Ce dernier propose une structure très similaire à celle de OWL (aussi appelée OWL 1) [1]. OWL 2 définit un certain nombre de profils (équivalents aux espèces de OWL 1) : OWL 2 EL, OWL 2 QL et OWL 2 RL. Par rapport à OWL 1, OWL 2 propose de nouvelles fonctionnalités. En particulier il permet de définir des clés, d'utiliser des types de données plus riches, etc.

1.2.3 Logique de description

Les langages RDF(S) et OWL sont basées sur la logique de description. Les logiques de description sont une famille de langages de représentation de connaissances [BCM+03, BHS09]. En général les connaissances sont divisées en deux parties. Les connaissances liées à la terminologie sont stockées dans une TBox, et celles liées aux individus sont stockées dans une ABox. Les logiques de description manipulent des concepts, des individus (instances de concepts) et des rôles (relations entre concepts). Les concepts sont formés à partir des éléments suivants : le concept universel $\top$, le concept impossible $\bot$, les concepts atomiques (a et b), les concepts (c et d), les rôles (R et S), les constructeurs ($\sqcap$ et $\sqcup$), les quantificateurs ($\forall$, $\exists$). Un concept est défini comme un sous-ensemble d'un domaine, et un rôle est interprété comme une relation binaire sur le domaine. Une interprétation est définie par un couple $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ où $\Delta^{\mathcal{I}}$ est le domaine d'in-

1. Source : http ://www.w3.org/TR/owl2-overview/

terprétation et $\cdot^{\mathcal{I}}$ est la fonction d'interprétation. Le tableau 1.1 présente l'interprétation des différents concepts.

Notation	Interprétation
$\top$	$\Delta^{\mathcal{I}}$
$\bot$	$\emptyset$
c	$c^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
R	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
$\neg c$	$\Delta^{\mathcal{I}} \setminus c^{\mathcal{I}}$
$d \sqcap c$	$d^{\mathcal{I}} \cap c^{\mathcal{I}}$
$d \sqcup c$	$d^{\mathcal{I}} \cup c^{\mathcal{I}}$
$\exists R.c$	$\{e_i \in \Delta^{\mathcal{I}} \mid \exists e_j \in \Delta^{\mathcal{I}} : (e_i, e_j) \in R^{\mathcal{I}}$ et $e_j \in c^{\mathcal{I}}\}$
$\forall R.c$	$\{e_i \in \Delta^{\mathcal{I}} \mid \forall e_j \in \Delta^{\mathcal{I}} : (e_i, e_j) \in R^{\mathcal{I}} \Rightarrow e_j \in c^{\mathcal{I}}\}$

TABLE 1.1 – *Sémantique des concepts et des rôles en logique de description.*

Dans ce cadre, une TBox est définie formellement par un ensemble fini d'axiomes de rôles d'inclusion et de GCI (*General Concept Inclusion*). Une interprétation $\mathcal{I}$ est un modèle d'une TBox $\mathcal{T}$ s'il satisfait tous les axiomes de $\mathcal{I}$. C'est-à-dire que :

- $c^{\mathcal{I}} \subseteq d^{\mathcal{I}}$ est vérifié pour chaque $c \sqsubseteq d \in \mathcal{T}$, et
- $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$ est vérifié pour chaque $R \sqsubseteq S \in \mathcal{T}$.

Soient $a, b \in I$ des noms d'individus, c un concept, et R un rôle. L'expression $c(a)$ est appelée une assertion de concept et $R(a, b)$ est appelée une assertion de rôle. La figure 1.6 présente le contenu d'une TBox. Elle spécifie qu'une *Personne* est soit un *Homme*, soit une *Femme*, et qu'une *mère* est une *Femme* ayant un *enfant*.

```
Personne ⊑ Homme ⊔ Femme
mere ≡ Femme ⊓ ∃ enfant.⊤
```

FIGURE 1.6 – *Exemple d'une TBox.*

Une ABox est définie par un ensemble fini d'assertions de rôles et de concepts. Une fonction d'interprétation permet de lier chaque nom d'individu a à un élément $a^{\mathcal{I}}$ du domaine d'interprétation $\Delta^{\mathcal{I}}$. Une interprétation $\mathcal{I}$ satisfait :

- une assertion de concept $c(a)$ si $a^{\mathcal{I}} \in c^{\mathcal{I}}$, et
- un rôle d'assertion $R(a, b)$ si $\langle a^{\mathcal{I}}, b^{\mathcal{I}} \rangle \in R^{\mathcal{I}}$.

Une interprétation qui satisfait toutes les assertions de concept et de rôle dans une ABox $\mathcal{A}$ est appelé un modèle de $\mathcal{A}$. La figure 1.7 présente le contenu d'une ABox. Celle-ci spécifie que *John* est un *Homme*, que *Jane* est une *Femme*, et que *Jane* est la *mère* de *John*.

```
Homme(John)
Femme(Jane)
mere(Jane,John)
```

FIGURE 1.7 – *Exemple d'une ABox.*

Les raisonneurs permettent d'inférer des conséquences logiques à partir d'un ensemble d'axiomes. Ils permettent en particulier de traiter des requêtes intentionnelles (par exemple : vérifier qu'il n'y a pas de conflit

au niveau de la définition des concepts) et extensionnelles (par exemple : retourner toutes les instances d'un concept). La plupart des raisonneurs modernes sont basés sur des algorithmes de tableaux [MH09]. C'est le cas des raisonneurs FaCT++ [TH06] et Pellet [SPG+07]. D'autres raisonneurs considèrent des algorithmes de résolution [MH09].

1.3 Hétérogénéité : niveaux et sources

De manière générale, la conception d'une ontologie est une tâche complexe et critique car l'objectif est de « représenter le monde ». Il est raisonnable de penser que deux personnes peuvent avoir des points de vue différents sur le monde et sur la manière de le représenter.

1.3.1 Niveaux d'hétérogénéité

L'hétérogénéité entre deux ontologies peut apparaître à différents niveaux. Dans [BEE+04, ES07], Euzenat *et al.* en identifient quatre :

- Le niveau syntaxique concerne essentiellement le fait qu'une ontologie peut être modélisée dans différents langages comme OWL, KIF, DAML+OIL, etc. Les langages peuvent adopter des paradigmes différents (paradigme logique, paradigme objet, etc.) ou simplement utiliser des syntaxes différentes. Pour adresser ce problème, le W3C a proposé un certain nombre de standards.
- Le niveau terminologique concerne les différences de nommage des entités. Ces différences sont principalement liées à la complexité des langues naturelles. Par exemple, il peut y avoir des cas de polysémie (deux entités identiques nommées de manières différentes), de synonymie (deux entités différentes nommées de la même manière), de multilinguisme (deux entités identiques nommées de la même manière, mais dans des langues différentes), etc.
- Le niveau conceptuel/sémantique regroupe trois aspects : le point de vue duquel s'est placé le concepteur de l'ontologie (la perspective), le domaine du monde qu'il a cherché à représenter (la couverture), et le niveau de détail qu'il a voulu atteindre (la granularité). La perspective est issue de la difficulté à créer une ontologie avec un regard objectif.
- Le niveau sémiotique intervient lors de l'utilisation des ontologies : un même concept peut être utilisé de plusieurs manières.

Dans cette étude, nous nous concentrons essentiellement sur le niveau sémantique.

1.3.2 Sources de l'hétérogénéité sémantique

Une ontologie étant une représentation subjective de la réalité, il semble difficile (voire impossible) d'imaginer que les concepteurs d'ontologies puissent se mettre d'accord sur une représentation commune de la réalité, et que celle-ci soit adaptée aux besoins de tous les utilisateurs. Smart et Engelbrecht soulignent dans [SE08] que les ontologies sont un élément central du web sémantique et que sa nature ouverte, distribuée

et décentralisée tend à encourager la conception d'ontologies par des développeurs dont les objectifs sont variés. Hameed, Preece et Sleeman mettent en avant plusieurs raisons pour lesquelles il n'est pas envisageable de concevoir une ontologie universelle [HPS04]. Premièrement, un certain nombre d'ontologies ont déjà été développées, donc même s'il était possible de concevoir une ontologie universelle, il faudrait réconcilier cette ontologie avec celles déjà utilisées. Deuxièmement, la conception d'une ontologie est subjective et est guidée par l'utilisation qui en sera faite. Par exemple, une entreprise qui développe une ontologie n'a pas forcément pour but d'avoir une représentation fidèle du monde ; elle souhaite simplement représenter son domaine avec son propre point de vue. Enfin, dans l'hypothèse où un accord serait trouvé pour adopter une unique ontologie, il semble probable que celle-ci évolue au cours du temps car les connaissances dans certains domaines sont en perpétuelle évolution. Klein et Fenseil traitent ce problème spécifique dans [KF01].

Ces sources de disparité conduisent à la création d'ontologies pouvant différer à différents niveaux. Deux ontologies peuvent différer car elles modélisent différents domaines d'applications (couverture), à des niveaux de détail différents (granularité) et avec des points de vue différents (perspective) [ELBB+04].

1.4 Alignement d'ontologies

L'alignement d'ontologies (*ontology matching*) a pour objectif de combler le fossé sémantique entre deux ontologies [FN09, ES07, BBR11]. Le résultat de ce processus, appelé alignement (*alignment*), est un ensemble de correspondances entre ces deux ontologies. Euzenat et Shvaiko [ES07] définissent une correspondance entre deux ontologies o et o' comme étant un 5-uplet $\langle id, e, e', r, n \rangle$ dans lequel id représente l'identifiant de la correspondance, e et e' sont des entités de o et o', r est une relation entre e et e', et n est une valeur de confiance attribuée à la correspondance. La relation r est souvent un lien d'équivalence ($\equiv$) ou de subsomption (*plus-spécifique-que*, $\sqsubseteq$), mais peut également être une disjonction ($\not\equiv$), une généralisation (*plus-général-que*, $\sqsupseteq$), etc.

1.4.1 Processus d'alignement

En règle générale, un processus d'alignement fait intervenir plusieurs méthodes qui ont chacune pour objectif de trouver des correspondances entre deux ontologies. Chaque processus prend en entrée deux ontologies o et o', et un alignement a_i (éventuellement vide) et produit un alignement a_j en sortie (voir figure 1.8 [2]). Un ensemble de paramètres et de ressources sont également utilisées. Les méthodes d'alignement peuvent être locales ou globales [ELBB+04].

1.4.1.1 *Méthodes locales*

Méthodes terminologiques La technique de base pour calculer un alignement entre deux ontologies o et o' consiste à mesurer la similarité entre

2. La figure est tirée de [ES07].

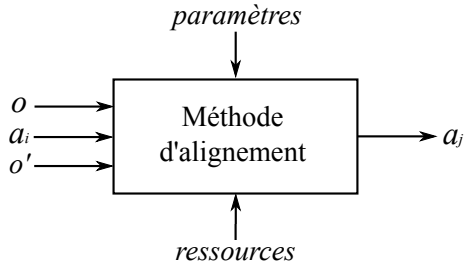


FIGURE 1.8 – *Processus d'alignement d'ontologies.*

leurs entités [ES07, DE08]. Les méthodes terminologiques comparent les chaînes de caractères pour calculer la similarité entre deux entités. Elles utilisent les noms des entités, les commentaires qui y sont attachés, etc. Dans la suite de cette section nous présentons une liste non-exhaustive de mesures de similarité (σ) ou de dissimilarité (δ).

Une première approche consiste simplement à déterminer si deux chaînes sont identiques. Cette méthode nécessite de normaliser les chaînes : normalisation de la casse, suppression de la ponctuation, etc. Elle ne permet pas de déterminer si deux chaînes différentes sont très similaires ou très dissimilaires (elle retourne un résultat booléen).

La mesure de Hamming compte le nombre de positions où les caractères de deux chaînes sont différents [Ham50]. Étant données deux chaînes s et s', la distance de Hamming est définie par :

$$\delta(s, s') = \sum_{i=1}^{\min(|s|,|s'|)} f(s, s', i) \tag{1.1}$$

où $f(s, s', i)$ vaut 1 si $s[i] \neq s'[i]$, et 0 sinon.

La mesure de Levenshtein (aussi appelée *distance d'édition*) compte le nombre minimal d'opérations (insertion, suppression ou remplacement de caractères) qui doivent être appliquées à une chaîne pour qu'elle devienne égale à une autre [Lev66]. Dans cette mesure, toutes les opérations ont le même coût. Les coûts de transformation sont pondérés différemment dans les mesures de Needleman-Wunch [NW70], de Monge-Elkan [ME96], etc.

D'autres mesures basées sur les chaînes ont été proposées : similarité de Jaro (basée sur le nombre et l'ordre des caractères communs à deux chaînes) [Jar89], la distance de Jaro-Winkler (variante de la distance de Jaro donnant plus d'importance aux sous-chaînes communes dont la taille est importante) [Win99], la distance n-gramme (considère et compare les sous-séquences de n caractères) [Kon05], etc.

Méthodes structurelles Plutôt que de simplement comparer les entités en considérant les chaînes de caractères, il est possible de comparer les entités en prenant en compte la structure des ontologies. Par exemple VALTCHEV et EUZENAT définissent la mesure de dissimilarité topologique entre deux entités [VE97]. Étant données deux ontologies o et o' partageant la même hiérarchie $H = \langle C, \leqslant \rangle$, la dissimilarité entre $e \in o$ et $e' \in o'$ appartenant à C est définie par :

$$\delta(e, e') = \min_{c \in C}[d(e, c) + d(e', c)] \tag{1.2}$$

où $d(a, b)$ est le nombre d'arcs intermédiaires entre a et b.

Dans le même contexte, Mäedche et Zacharias [MS02] définissent la distance *upward cotopic* entre deux concepts c et c' de la manière suivante :

$$\delta(c, c') = \frac{|UC(c, C) \cap UC(c', C)|}{|UC(c, C) \cup UC(c', C)|} \tag{1.3}$$

où $UC(c, C)$ représente l'ensemble des super-classes de c dans la hiérarchie H.

Méthodes extensionnelles Les méthodes extensionnelles utilisent les instances des entités pour mesurer la similarité entre ces entités [Rah11] : il s'agit donc clairement d'ontologies peuplées. Dans le cas où les deux ontologies partagent des instances, la similarité entre deux entités e et e' peut être mesurée en considérant l'indice de Jaccard [Jac12] :

$$\sigma(e, e') = \frac{|A \cap A'|}{|A \cup A'|} \tag{1.4}$$

où A (respectivement A') représente l'ensemble des instances de l'entité e (respectivement e').

Méthodes sémantiques Les méthodes sémantiques, qui utilisent le modèle théorique sémantique, sont des méthodes déductives. L'idée est de transformer le problème d'alignement en une formule propositionnelle et de vérifier sa validité [GS03]. Les techniques de satisfiabilité propositionnelle testent en réalité toutes les correspondances possibles. C'est pour cette raison qu'elles sont en général plus efficaces lorsqu'une autre méthode a été appliquée auparavant afin d'identifier des ancres (c'est-à-dire des correspondances fiables).

1.4.1.2 *Méthodes globales*

Agrégation de similarités locales L'agrégation de similarités locales a pour but de mesurer la similarité entre deux entités, en considérant la similarité des entités auxquelles elles sont liées. Par exemple la similarité entre deux concepts dépend de la similarité entre leurs super-classes, leurs sous-classes, leurs propriétés, etc. Soient e et e' deux entités pouvant être analysées suivant n dimensions. Ces dernières correspondent aux super-classes, aux sous-classes, aux attributs, etc. La distance de Minkowski permet de mesurer la dissimilarité entre e et e' :

$$\delta(e, e') = \sqrt[p]{\sum_{i=1}^{n} d(e_i, e'_i)^p} \tag{1.5}$$

où $d(e_i, e'_i)$ mesure la dissimilarité entre les dimensions e_i et e'_i des entités e et e'. La distance de Minkowski correspond à la distance de Manhattan lorsque $p = 1$, à la distance Euclidiénne lorsque $p = 2$ et à la distance de Chebichev lorsque $p = +\infty$. Cette mesure donne autant de poids à chaque dimension, alors que certaines dimensions peuvent être plus pertinentes que d'autres (certaines propriétés par exemple). Pour combler ce manque, des mesures pondérant les dimensions ont été proposées (cf. [ELBB+04]).

Similarité globale Les méthodes d'agrégation de similarités locales peuvent poser des problèmes car le calcul de similarité entre deux entités peut dépendre de la similarité entre deux autres entités. Si cette dernière dépend de la première, il devient impossible de faire les calculs car il y a un cycle dans le processus. Il est donc parfois préférable de considérer les ontologies dans leur entier pour mesurer la similarité entre entités.

Melnik, Garcia-Molina et Rahm ont proposé l'algorithme *similarity flooding* [MGMR02]. Il s'agit d'un algorithme générique d'alignement de graphes qui utilise un calcul de points fixes pour identifier les nœuds correspondants dans les graphes. Pour cela l'algorithme construit un nouveau graphe dont les nœuds sont des paires de nœuds du graphe initial.

Euzenat et Valtchev ont également proposé une mesure prenant en compte toutes les caractéristiques des entités (super-classes, propriétés, instances) [EV04]. Cette mesure permet de gérer les cycles et les dépendances entre les définitions des similarités.

Méthodes d'apprentissage Les méthodes d'apprentissage, à l'image du système GLUE [DMD+03], ont pour objectif de déterminer la similarité entre deux classes. L'idée de base est qu'il est probable que deux classes partageant les mêmes instances soient équivalentes. L'objectif est donc d'évaluer la probabilité qu'une instance quelconque soit une instance de deux classes c et c'. Cette probabilité est notée $P(c, c')$. Dans cette approche le système doit calculer les probabilités suivantes : $P(c, c')$, $P(c, \bar{c'})$, $P(\bar{c}, c')$ et $P(\bar{c}, \bar{c'})$. Cette méthode nécessite que les deux ontologies partagent des instances. Si ce n'est pas le cas l'algorithme apprend, de manière à assigner les instances à des classes de l'autre ontologie.

Composition de méthodes Il existe trois manières de combiner différentes méthodes. La première (la plus répandue) correspond au cas où les méthodes sont enchaînées quelles que soient les données en entrée. La seconde correspond au cas où la méthode à appliquer à chaque étape est choisie en fonction des données fournies en entrée. Enfin la troisième correspond au cas où un utilisateur interagit avec le système pour le guider.

Interaction avec l'utilisateur Dans le processus d'alignement, l'utilisateur peut interagir de plusieurs façons. Tout d'abord il peut fournir des valeurs de similarité entre entités (en déterminant par exemple un certain nombre de correspondances). Ces données servent alors à amorcer et à faciliter les méthodes automatiques d'alignement. Ensuite l'utilisateur peut intervenir dans le choix des méthodes à appliquer et de l'ordre dans lequel elles doivent être appliquées (voir paragraphe précédent). Enfin l'utilisateur peut porter un jugement sur les résultats produits par un processus d'alignement. Ce jugement sert à corriger le résultat, soit pour l'utiliser directement, soit pour le fournir en entrée d'une autre méthode.

Extraction de l'alignement Lorsque plusieurs méthodes d'alignements ont été appliquées, il faut agréger les résultats afin de fournir un résultat (c.-à-d. un alignement) optimal. En particulier cela nécessite de déterminer quelles sont les correspondances à partir des valeurs de similarité. La

méthode de base consiste à utiliser une valeur seuil. Cette valeur peut être choisie de manière brutale, ou plus finement [ES04].

1.4.2 Applications

Les alignements permettent à des agents utilisant différentes ontologies d'interopérer. Par exemple lorsque deux participants utilisent des ontologies différentes et qu'ils souhaitent s'échanger des requêtes, il est crucial qu'ils puissent traduire les requêtes qu'ils reçoivent ou qu'ils envoient. Les alignements d'ontologies aident également à la conception de nouvelles ontologies. Par exemple lors de la création d'une ontologie intégrant les ontologies Dublin Core (`dc`) et Friend-Of-A-Friend (`foaf`), il est important de pouvoir indiquer que la classe `dc:creator` est une spécialisation de la classe `foaf:Person`. Les alignements sont également centraux pour l'intégration de données lorsque les sources de données sont représentées suivant différentes ontologies. Dans ce contexte, les requêtes sont transmises à un *broker* (un courtier) qui se charge de transmettre les requêtes à des médiateurs. Ces derniers utilisent des alignements pour traduire les requêtes exprimées par rapport une ontologie globale, afin qu'elles soient exprimées par rapport aux ontologies locales (qui sont utilisées pour représenter les données). Un autre exemple d'application pour laquelle les alignements d'ontologies sont utiles concerne la composition de services. En effet lorsque ceux-ci sont décrits par des ontologies différentes, il est intéressant de pouvoir faire correspondre les descriptions des services afin de savoir si deux services sont composables même s'ils ne sont pas décrits suivant la même ontologie.

Les alignements sont également la base de processus complexes appliqués sur les ontologies :

- fusion d'ontologies : création d'une ontologie à partir de deux ontologies [BEF+06],
- intégration d'ontologies : intégration d'une ontologie dans une autre,
- la traduction/transformation d'ontologies : expression d'une ontologie dans le "vocabulaire" d'une autre ontologie,
- réconciliation d'ontologies : transformation de deux ontologies dans le but de les uniformiser [HPS04].

Ils sont également utilisés pour travailler sur les données :

- traduction de données : intégration des instances d'une ontologie dans une autre ontologie,
- intégration sémantique de données : possibilité d'interroger les données indépendamment de la manière dont elles sont représentées [Saï07].

1.4.3 Exemples de systèmes d'alignements existants

Dans cette section nous présentons succinctement quelques systèmes d'alignements d'ontologies. Des descriptions plus précises sont faites dans les ouvrages d'Euzenat et Svaiko [ES07], de Bellahsene, Bonifati et Rahm [BBR11], et d'Euzenat *et al.* [ELBB+04].

Le système COMA++ permet d'aligner des ontologies (ou des schémas) [ADMR05]. Ce système a été l'un des premiers à permettre la composition de méthodes pour l'alignement d'ontologies. COMA++ propose deux stratégies. La première consiste à identifier des fragments similaires dans les ontologies afin de traiter chaque fragment comme un sous-problème. La seconde stratégie consiste à réutiliser des résultats d'alignements précédents. Cela doit permettre de ne pas refaire plusieurs fois les mêmes calculs, et de guider l'alignement avec des correspondances pouvant éventuellement être calculées ou vérifiées manuellement.

Le système AgreementMaker, en développement depuis 2001, propose une interface graphique permettant à un utilisateur d'aligner des ontologies [CSC+11]. La version utilisée pour l'évaluation OAEI[3] 2011 propose de nouvelles approches. Les ontologies sont d'abord analysées afin de pouvoir adapter dynamiquement le processus d'alignement (ordre des méthodes à appliquer, choix des paramètres, etc.).

Le système RiMOM combine automatiquement et dynamiquement différentes stratégies pour produire un alignement [WZH+10]. Pour la campagne d'évaluation OAEI 2010, le système utilise trois stratégies. La première se base sur le label des entités. La deuxième considère les informations de chaque entité comme un document. Les entités sont alors représentées sous forme de vecteurs et la similarité entre deux entités est estimée par la mesure du cosinus entre les deux vecteurs correspondants (cf. section 3.2.2, page 47). La troisième stratégie est semblable à la précédente mais considère les instances des entités plutôt que les informations liées aux entités.

S-Match permet d'identifier des correspondances entre des structures de graphe (schémas XML, ontologies, etc.) [SGY09]. Dans S-Match, les entités sont transformées en formules propositionnelles qui expriment explicitement les descriptions des concepts. Le problème d'alignement est alors réduit à un problème de validation propositionnelle, qui peut être traité à l'aide de solveurs de satisfiabilité propositionnelle.

De nombreux autres systèmes ont été proposés : CUPID, GLUE, Prompt, SimilarityFlooding, FCA-Merge, Falcon, Asmov, etc.

1.5 Mesures sémantiques

Dans cette section nous présentons un état de l'art concernant les mesures de similarité sémantique entre entités d'une même ontologie (similarité intra-ontologie), et sur les mesures de (dis)similarité entre deux ontologies.

1.5.1 Similarités intra-ontologie

Mesurer la similarité entre deux concepts d'une même ontologie peut servir dans plusieurs contextes. Par exemple, cela permet d'étendre des requêtes sémantiques en recherche d'information [QF93, VCLV08] ou de désambiguïser des termes [BFCS12]. De nombreuses mesures ont été proposées dans la littérature. Elles permettent d'estimer la proximité séman-

3. OAEI : Ontology Alignment Evaluation Initiative.

tique entre concepts d'une même ontologie. Un certain nombre d'entre elles considèrent la structure hiérarchique des ontologies. C'est le cas de la mesure proposée par RADA [RMBB89]. Elle est définie par :

$$sim_{Ra}(c_1, c_2) = \frac{1}{1 + dist(c_1, c_2)} \tag{1.6}$$

où $dist(c_1, c_2)$ correspond au nombre d'arcs qu'il faut traverser dans l'ontologie pour relier les concepts c_1 et c_2. Quelles que soient leurs positions dans l'ontologie, plus les concepts sont éloignés dans la hiérarchie, moins la similarité est importante. WU et PALMER ont proposé une mesure prenant en compte la position des concepts, c'est-à-dire la profondeur à laquelle ils se trouvent dans l'ontologie [WP94] :

$$sim_{WP}(c_1, c_2) = \frac{2 \cdot prof(c)}{prof(c_1) + prof(c_2)} \tag{1.7}$$

Dans cette mesure, c désigne le plus petit ancêtre commun à c_1 et c_2 dans l'ontologie, et $prof(c)$ est une fonction donnant la profondeur du concept c dans la hiérarchie (la profondeur du concept racine vaut 0). La mesure de WU et PALMER ne prend pas en compte la profondeur de l'ontologie. Pourtant celle-ci peut avoir une importance. La profondeur d'une ontologie o est définie par : $Prof(o) = \max_{c \in o} prof(c)$. RESNIK a définit une mesure de similarité utilisant cette notion [Res95] :

$$sim_{Re}(c_1, c_2) = 2 \cdot Prof(o) - dist(c_1, c_2) \tag{1.8}$$

Des approches considèrent le contenu informationnel des concepts pour mesurer la similarité entre deux concepts. C'est le cas de la mesure proposée par RICHARDSON, SMEATON et MURPHY [RSM94] :

$$sim_{Ri}(c_1, c_2) = \max_{c \in C} ci(c) \tag{1.9}$$

où C correspond à l'ensemble des ancêtres communs aux concepts c_1 et c_2, et $ci(c)$ correspond au contenu informationnel associé au concept c. Plusieurs approches ont été proposées pour estimer le contenu informationnel d'un concept. RESNIK propose d'utiliser un corpus [Res95], tandis que SECO, VEALE et HAYES proposent de se limiter à l'ontologie en considérant que le contenu informationnel d'un concept décroît avec le nombre de concepts qui le spécialisent [SVH04].

JIANG et CONRATH [JC97] et Lin [Lin98] ont proposé des mesures combinant les deux approches (celle basée sur la structure de l'ontologie et celle basée sur le contenu informationnel des concepts). La mesure de LIN est définie par :

$$sim_{Lin}(c_1, c_2) = \frac{2 \cdot \log(P(c))}{\log(P(c_1)) + \log(P(c_2))} \tag{1.10}$$

où c est le plus petit ancêtre commun à c_1 et c_2, et $P(c)$ correspond à la probabilité que le concept c appartienne à un corpus donné.

1.5.2 Similarités entre ontologies

Il est souvent utile de pouvoir évaluer la distance entre deux ontologies. Pour cela il faut disposer de mesures de (dis)similarité. Certaines de ces mesures permettent de décider s'il est raisonnable d'aligner deux ontologies. Cela permet de ne pas lancer un processus d'alignement coûteux et produisant peu de correspondances dans le cas où les ontologies sont trop différentes. Le calcul de la distance doit être rapide (en tout cas plus rapide que l'alignement lui-même).

Maëdche et Staab définissent plusieurs mesures de similarité entre ontologies [MS02]. L'une d'entre elles se concentre sur le niveau lexical ($\overline{\text{SM}}$: String Matching). Elle considère les labels attribués aux concepts et est basée sur la distance d'édition (mesure de Levenshtein [Lev66]). Ils proposent également une mesure de similarité à partir des hiérarchies de deux ontologies ($\overline{\text{TO}}$: Taxonomie Overlap). Cette mesure permet de mesurer à quel point deux ontologies sont similaires au niveau de leurs structures. Enfin ils proposent une mesure prenant en considération les relations liant les concepts ($\overline{\text{RO}}$: Relation Overlap).

Dans [DE08], David et Euzenat présentent différentes mesures de (dis)similarité entre ontologies. Étant donnée une mesure de dissimilarité δ entre entités de deux ontologies o et o', les auteurs définissent la mesure de dissimilarité Average Linkage :

$$\Delta_{alo}(o,o') = \frac{1}{|o| \times |o'|} \sum_{(e,e') \in o \times o'} \delta(e,e') \tag{1.11}$$

où $|o|$ correspond au nombre d'entités contenues dans l'ontologie o. La mesure δ est une mesure de dissimilarité quelconque entre deux entités. Dans le même contexte, les auteurs présentent une mesure de Hausdorff qui mesure une distance entre deux ontologies :

$$\Delta_{Hauss}(o,o') = \max\left(\max_{e \in o} \min_{e' \in o'} \delta(e,e'), \max_{e' \in o'} \min_{e \in o} \delta(e,e')\right) \tag{1.12}$$

Enfin, les auteurs introduisent la notion de graphe d'alignement maximal de poids minimal (*minimum weight maximum graph matching*). Un tel graphe $G \subseteq o \times o'$ vérifie que quel que soit un autre graphe $G' \subseteq o \times o'$:

$$\sum_{\langle p,q' \rangle \in G} \delta(p,q) \leqslant \sum_{\langle p,q' \rangle \in G'} \delta(p,q') \tag{1.13}$$

La distance MWMGM (Minimum Weight Maximum Graph Matching) est définie par :

$$\Delta_{mwmgm}(o,o') = \frac{\sum_{\langle p,q' \rangle \in G} \delta(p,q') + \max(|o|,|o'|) - |G|}{\max(|o|,|o'|)} \tag{1.14}$$

Euzenat, Alloca, David *et al.* présentent un ensemble de mesures de distance entre ontologies dans [EAD+09]. Certaines considèrent uniquement les labels assignés aux entités : la distance d'Hamming sur les noms Δ_{hdcn}, la similarité Common Name σ_{cn}, etc. Une mesure similaire à la similarité Common Name est définie sur les axiomes des ontologies : σ_{cax}. En considérant la signification des ontologies, les auteurs présentent une mesure

sémantique : la distance Ideal Semantic. Elle prend en compte l'ensemble des conséquences d'une ontologie. Étant donné que cet ensemble peut être infini, les auteurs proposent de considérer l'ensemble des conséquences d'une ontologie par rapport à une autre. Ils définissent ainsi la similarité Common Consequence :

$$\sigma_{ccsq}(o, o') = \frac{|LCn(o, o') \cap LCn(o', o)|}{\max(|o|, |o'|)} \tag{1.15}$$

où $LCn(o, o') = o \cap Cn(o')$. Les auteurs proposent également une mesure qui permet de se focaliser sur les concepts les plus importants des ontologies. Pour cela les travaux de Peroni, Motta et d'Aquin peuvent être considérés [PMd08]. La fonction KC permet de retourner l'ensemble des n concepts clés (*key concepts*) d'une ontologie. La mesure de similarité est définie par :

$$\sigma_{ukc}(o, o') = \frac{|KC(o, n) \cap KC(o', n)|}{n} \tag{1.16}$$

Dans cette définition, $KC(o, n)$ désigne l'ensemble des n concepts les plus importants. L'expression $A \cap B$ désigne ici l'ensemble des concepts de A et de B ayant le même label. Cette mesure est un raffinement de la mesure σ_{cn}, et elle ne tient pas compte de l'ordre des concepts importants. Les auteurs proposent une mesure permettant de prendre en compte cet aspect :

$$\sigma_{rkc}(o, o') = \frac{1}{n} \sum_{c \in KC(o,n)} \sigma_{kc}(KC(o, n), KC(o', n), c) \tag{1.17}$$

où, étant donnés deux ensembles ordonnés de concepts S_1 et S_2 de cardinalité n, la mesure σ_{kc} est définie par :

$$\sigma_{kc}(S_1, S_2, c) = \begin{cases} 1 - \frac{1}{n}|rang(S_1, c) - rang(S_2, c)| & \text{si } c \in S_1 \cap S_2 \\ 0 & \text{sinon} \end{cases} \tag{1.18}$$

Contrairement à toutes les mesures présentées précédemment (qui sont définies dans l'espace des ontologies), certaines mesures ont été définies dans l'espace des alignements, c'est-à-dire qu'elles mesurent la distance entre deux ontologies en prenant en compte des alignements existants entre elles. Euzenat *et al.* proposent un ensemble de mesures [DEvZ10, EAD+09]. Ils définissent par exemple une mesure de similarité de manière très simple :

$$\sigma_{ap}(o, o') = \begin{cases} 1 & \text{si } o = o' \\ 2/3 & \text{si } o \neq o' \wedge \mathcal{A}(o, o') \neq \emptyset \\ 1/3 & \text{si } o \neq o' \wedge \mathcal{A}(o, o') = \emptyset \wedge \mathcal{A}^*(o, o') \neq \emptyset \\ 0 & \text{sinon} \end{cases} \tag{1.19}$$

où $\mathcal{A}(o, o')$ désigne l'ensemble des alignements existants entre o et o' dans l'espace d'alignement $\mathcal{A}$, et $\mathcal{A}^*(o, o')$ désigne l'ensemble des chemins d'alignements entre o et o' dans $\mathcal{A}$. Une mesure considérant le plus court chemin d'alignements a également été proposée (σ_{sap}). La mesure Alignment Coverage (cov) se focalise sur le nombre d'entités d'une ontologie o préservées par un alignement a. Elle est définie par :

$$cov(o, o', a) = \frac{|\{e \in o : \exists \langle id, e, e', r, n \rangle \in a\}|}{|o|} \tag{1.20}$$

Une mesure de distinguabilité est également définie :

$$sep(o,o',a) = \frac{|\{e' \in o' : \exists \langle id, e, e', r, n\rangle \in a\}|}{|\{e \in o : \exists \langle id, e, e', r, n\rangle \in a\}|} \tag{1.21}$$

La mesure *cov* détermine à quel point les concepts d'une ontologie sont préservés par un alignement, alors que la mesure *sep* détermine à quel point les concepts sont distincts lorsqu'un alignement est appliqué. Ces deux mesures peuvent être combinées pour définir la mesure *covdis*. Étant donné un ensemble de chemins d'alignement $\mathcal{A}^*(o,o')$ entre deux ontologies o et o', la mesure Largest Covering Preservation (σ_{lcp}) permet de déterminer à quel point les concepts d'une ontologie sont préservés sur ce chemin :

$$\sigma_{lcp}(o,o') = \max_{a \in \mathcal{A}^*(o,o')} covdis(o,o',a) \tag{1.22}$$

Enfin, la mesure de similarité Union Path Coverage (σ_{upc}) considère l'union de différents chemins d'alignements entre deux ontologies, plutôt qu'un seul chemin. L'idée sous-jacente est qu'une requête propagée entre deux individus au travers d'un chemin d'alignement peut être décomposée et propagée parallélement à travers différents chemins. La mesure de similarité est donc définie par :

$$\sigma_{upc}(o,o') = \frac{|\{e \in o : \exists a^* \in \mathcal{A}^*(o,o') \text{ tel que } e \text{ est préservé sur } a^*\}|}{|o|} \tag{1.23}$$

Dans [d'A09], d'Aquin propose des mesures d'accord et de désaccord (*agreement* et *disagreement*) entre deux ontologies. Pour cela il considère le degré d'accord entre toutes les déclarations des ontologies. Par exemple la déclaration $\langle c_1, r_a, c_2\rangle$ contenue dans l'ontologie o déclare que le concept c_1 est lié au concept c_2 via la relation r_a. Si les concepts c_1 et c_2 sont alignés avec les concepts c'_1 et c'_2 de l'ontologie o' et que o' contient la déclaration $\langle c'_1, r_b, c'_2\rangle$ alors la mesure d'accord (ou de désaccord) dépend de la proximité entre r_a et r_b :

$$agreement(o,o') = \frac{1}{|ST| + |ST'|} \sum_{st \in ST} agr(st,o') + \sum_{st' \in ST'} agr(st',o) \tag{1.24}$$

où ST est l'ensemble des déclarations contenues dans l'ontologie o, et $agr(st,o')$ mesure l'accord de la déclaration st dans l'ontologie o'

1.6 Bilan

Dans ce chapitre nous avons commencé par présenter différents langages permettant de définir des ontologies. Nous nous sommes concentrés sur les langages OWL, RDF(S) et sur la logique de description. Les langages OWL et RDF(S) sont des éléments centraux du Web sémantique et font parties des recommandations du W3C.

Nous avons ensuite montré pourquoi il n'est pas raisonnable d'envisager que tous les utilisateurs d'une système à large échelle, tels que les systèmes P2P ou le Web, s'accordent sur l'utilisation d'une unique ontologie. Lorsque les disparités entre ontologies dépendent de la manière dont

est perçu le domaine on parle alors d'hétérogénéité sémantique. Les méthodes d'alignement d'ontologies permettent de mettre en relation deux ontologies. Ils sont donc essentiels dès lors qu'il s'agit de faire interopérer des utilisateurs travaillant avec des ontologies différentes. Certaines de ces méthodes nécessitent d'estimer la distance entre deux ontologies. Cette estimation permet de déterminer s'il est raisonnable de procéder à un alignement. Des mesures permettent également de déterminer la distance entre deux ontologies une fois que celles-ci sont alignées. D'une certaine manière, elles mesurent à quel point un alignement (ou un ensemble d'alignements) permet à deux utilisateurs d'interopérer avec leurs ontologies respectives.

Systèmes P2P

2

Dans ce chapitre nous présentons un état de l'art sur les systèmes pair-à-pair (P2P). Nous commençons par présenter les caractéristiques des systèmes P2P : la décentralisation des données, la tolérance aux pannes, la dynamicité, etc. Nous présentons ensuite les différents types d'architectures de systèmes P2P : les systèmes à index centralisés, les systèmes non-structurés, les systèmes structurés et les systèmes hiérarchiques. Pour chacune, nous présentons l'architecture et les mécanismes de recherche qui permettent d'accéder à des données à partir d'une requête. Les avantages et inconvénients de chaque architecture sont également discutés.

Nous présentons ensuite une famille d'algorithmes permettant de diffuser des données dans les systèmes P2P : les protocoles épidémiques. Ces derniers peuvent être utilisés pour disséminer des informations dans un système distribué, pour modifier la topologie d'un système P2P, pour fournir un service d'échantillonnage de pairs, etc. Nous présentons les principes généraux des protocoles épidémiques, et nous détaillons les applications dans lesquelles ils peuvent être utilisés.

2.1 Paradigme P2P

Avant que les systèmes pair-à-pair (ou P2P) ne se développent, le mode de partage de données sur Internet était basé sur le modèle client/serveur. Dans ce modèle, les données sont hébergées sur des serveurs bien identifiés. Les utilisateurs peuvent alors accéder aux données en établissant une connexion vers ces serveurs (généralement par le biais d'un navigateur Web). Ce modèle est simple et puissant. Il permet aux fournisseurs de données de contrôler facilement la distribution du contenu. Néanmoins il présente plusieurs désavantages : (i) il n'est pas tolérant aux pannes : la défaillance d'un serveur empêche tous les utilisateurs d'accéder aux données ; (ii) il ne passe pas à l'échelle : un phénomène de saturation est observé lorsque le nombre de clients se connectant à un serveur pour accéder à une donnée devient important. En plus de ces désavantages techniques, le modèle client/serveur semble en décalage avec les usages (ou les besoins) des utilisateurs. En effet, ceux-ci peuvent souhaiter partager des données sans passer par l'intermédiaire d'un tiers.

En 1999, Fenning et Parker lancent le premier logiciel pair-à-pair grand public : Napster [SGG03]. Ce logiciel est conçu pour le partage de fichiers MP3 entre utilisateurs. Le développement du format MP3 et des supports de lecture sont probablement pour beaucoup dans son succès spectaculaire [1]. Le principe des systèmes P2P repose sur le fait que tout utilisateur est à la fois client (consommateur de données) et serveur (fournisseur de données). Les caractéristiques des systèmes P2P sont les suivantes :

- **distribution des données** : les données mises à disposition par les utilisateurs ne sont pas centralisées dans un serveur : elles sont réparties sur l'ensemble des pairs du système,
- **tolérance aux pannes** : le fait qu'un pair soit défaillant ne remet pas en cause le fonctionnement global du système,
- **passage à l'échelle** : le nombre de pairs dans le système et la quantité d'information qu'ils partagent n'empêche pas le système de fonctionner,
- **dynamicité** : les pairs sont libres de rejoindre ou de quitter le système à tout moment,
- **autonomie** : les pairs sont libres dans le choix des ressources qu'ils souhaitent partager, et de la manière dont ils veulent les partager.

La notion d'autonomie est fondamentale dans les systèmes P2P. En effet c'est l'autonomie laissée aux pairs qui leur permet de contrôler l'information qu'ils possèdent et qui leur permet de rejoindre ou de quitter librement le système. Dans la section suivante nous présentons les différentes architectures de systèmes P2P.

1. Les concepteurs revendiquent 35 millions d'utilisateurs en octobre 2000 (source : `http://wayback.archive.org/web/*/http://www.napster.com`).

2.2 Architectures P2P

2.2.1 Systèmes à index centralisés

Architecture Les premiers systèmes P2P (en particulier Napster) disposaient d'une architecture centralisée [SGG03]. Dans de tels systèmes, un serveur central répertorie tous les fichiers présents. Les fichiers eux-mêmes ne sont pas stockés sur ce serveur : seuls les index sont stockés. Ce serveur permet aux pairs de trouver les données qui les intéressent, ainsi que les pairs qui les stockent. La figure 2.1 représente un système P2P centralisé dans lequel six pairs sont connectés à un index central (I). Lorsqu'un pair entre dans le système, il se connecte au pair contenant l'index I et lui fournit la liste des données qu'il souhaite partager. Le serveur stocke alors l'information dans son index.

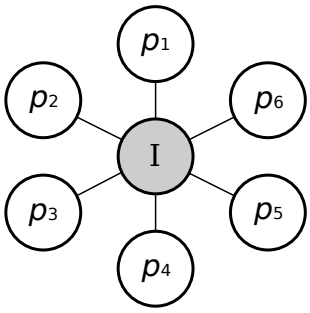


Figure 2.1 – *Système P2P à index centralisé.*

Méthode de recherche La recherche d'une donnée se fait via le serveur. Par exemple le pair p peut demander au serveur quels sont les pairs qui possèdent une donnée qui l'intéresse. Le serveur lui répond en lui fournissant la liste des pairs concernés. Le pair p peut alors se connecter directement à ces pairs pour accéder à la donnée. La manière dont est exprimé le besoin du pair p peut varier. Dans Napster, tous les fichiers partagés sont des fichiers musicaux au format MP3. Chaque donnée est alors décrite par un titre, un artiste, un genre, etc. Ces méta-données (exprimées au format ID3) sont utilisées pour effectuer la recherche [Abo01]. Dans ce type de système, la recherche est très efficace puisqu'il suffit d'un seul échange de message pour localiser les données pertinentes (le coût ne dépend pas du nombre de pairs dans le système).

Avantages et inconvénients L'avantage incontestable des systèmes P2P à index centralisé concerne l'efficacité de la recherche et de l'accès aux données. Chaque pair est capable de trouver une donnée en contactant simplement le serveur central. De ce point de vue, les performances sont comparables à celles qu'on trouve dans les architectures client/serveur classiques. En revanche la centralisation des index rend ce type de système très peu tolérant aux pannes et sensible aux attaques. Elle peut également poser des problèmes de passage à l'échelle.

2.2.2 Systèmes décentralisés

Hormis le fait qu'ils ne permettent pas de garantir la tolérance aux pannes et qu'ils sont sensibles aux attaques, les systèmes P2P à index centralisé ont rapidement été remplacés par les systèmes décentralisés pour des raisons légales. En effet dès décembre 1999, la société Napster est attaquée en justice par la RIAA (Recording Industry Association America) car elle enfreint la loi en stockant les index sur ses serveurs. Les systèmes P2P décentralisés ont été conçus pour palier les problèmes de tolérance aux pannes, et les problèmes législatifs. Notons que depuis la fin des années 2000, les lois ont évolué et les systèmes décentralisés présentent désormais les mêmes défauts que les systèmes centralisés au niveau législatif [2].

2.2.2.1 *Systèmes non-structurés*

Architecture Les systèmes décentralisés non-structurés, à l'image de Gnutella développé par Pepper et Frankel en 2000 (Nullsoft), sont conçus de telle sorte qu'aucune donnée ne soit centralisée [RFI02, SGG03]. Chaque pair stocke ses propres données, ainsi que les index correspondants. La figure 2.2 présente un système P2P non-structuré dans lequel dix pairs sont interconnectés. Pour maintenir la connexion avec d'autres pairs, chaque pair maintient une vue locale du système, contenant les identifiants de certains autres pairs. Par exemple le pair p_3 dans la figure 2.2 stocke les identifiants des pairs p_1, p_2, p_4 et p_5 dans sa vue locale. L'identifiant d'un pair permet de le contacter, et peut être simplement défini à partir de son adresse IP et d'un numéro de port. Dans l'exemple de la figure 2.2, les connexions sont bidirectionnelles (ce qui n'est pas toujours le cas).

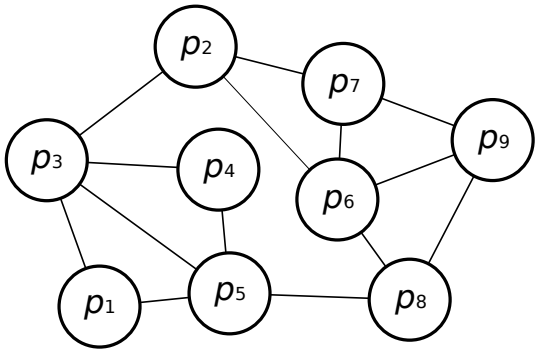


Figure 2.2 – *Système P2P non-structuré.*

Méthodes de recherche Pour accéder aux données, les pairs émettent des requêtes qu'ils envoient à leur voisinage. Plusieurs méthodes de routage ont été proposées.

L'algorithme Breadth-First Search (BFS, algorithme de parcours en largeur) considère une valeur de TTL (*Time-To-Live*) afin d'inonder le système dans un certain rayon autour du pair initiateur de la requête. Chaque pair

2. Loi Hadopi ou loi Création et Internet : « loi n°2009-669 du 12 juin 2009 favorisant la diffusion et la protection de la création sur internet ».

recevant la requête la transmet à ses propres voisins en prenant le soin de décrémenter la valeur du TTL de 1. Lorsque le TTL vaut 0, les pairs cessent de transmettre la requête. Tous les pairs ayant reçu la requête la traitent localement et retournent les résultats directement au pair initiateur. Cette approche était initialement utilisée dans le système Gnutella. Les principaux désavantages de cette méthode sont liés au nombre de messages échangés, et au risque de surcharger le pair initiateur avec un grand nombre de réponses.

Les algorithmes Modified BFS et Intelligent BFS [KGZY02] ont été proposés pour limiter le nombre de messages échangés. La réduction du nombre de messages échangés impacte la qualité des résultats (certaines données pertinentes peuvent ne pas être retournées), et le coût de maintenance de statistiques sur les requêtes.

Dans l'algorithme Random Walks [LCC+02], le pair initiateur transmet sa requête à seulement k de ses voisins : il initie k marcheurs. Les pairs qui reçoivent une requête la transmettent à un seul de leurs voisins (le choix du voisin est aléatoire). Encore une fois, une valeur de TTL est utilisée. L'avantage de cette méthode est qu'elle génère au maximum $k \times TTL$ messages (indépendamment de la topologie du système). Néanmoins elle a le désavantage d'être irrégulière au niveau des performances : elles dépendent de la topologie du système et des choix aléatoires.

D'autres méthodes de routage ont été proposées, comme par exemple Iterative Deepening [YGM02], Adaptative Probabilistic Search [TR03], Local Indices [CGM02], Bloom Filter-based Indices [RK02].

L'algorithme Fully Distributed (FD) [APV06] est un algorithme top-k qui vise à obtenir les k données les plus pertinentes (selon une fonction de score) accessibles dans un certain rayon autour du pair initiateur. L'objectif de FD est de réduire le temps de réponse et le nombre de messages échangés. L'algorithme est composé de quatre phases :

- Transmission de la requête : si le TTL de la requête est strictement positif, alors la requête est transmise suivant l'une des méthodes existantes (par exemple BFS) et le TTL est décrémenté de 1.
- Exécution locale de la requête : elle consiste à sélectionner les k données locales les plus pertinentes. La notion de pertinence est donnée par une fonction de score $sc(d, q)$ qui attribue à chaque document d un score par rapport à la requête q. Une fois les documents sélectionnés, le pair attend les résultats de ceux à qui il avait précédemment transmis la requête.
- Fusion et remontée des résultats : elle consiste à fusionner les k meilleures données trouvées localement avec les $n \times k$ données retournées par les n pairs à qui la requête avait été envoyée. Le résultat de cette fusion est un ensemble de k éléments. Chaque élément est de la forme $\langle id_p, s \rangle$ où id_p est l'identifiant du pair possédant la donnée dont le score par rapport à la requête est égal à s.
- Récupération des données : lorsque le pair initiateur de la requête reçoit les résultats de ses voisins, il les fusionne (comme dans la phase précédente) et peut donc accéder aux données elles-mêmes en contactant les pairs : il connaît les identifiants des pairs et les scores associés aux données pertinentes.

Cette approche permet de réduire le trafic réseau tout en évitant de sub-

merger le pair initiateur avec un grand nombre de résultats. Le temps d'exécution de cet algorithme dépend des pairs les plus lents (car avant de transmettre ses résultats, un pair doit attendre les résultats de ses voisins). La famille d'algorithmes As Soon As Possible, basée sur FD, adresse ce problème en estimant la qualité des résultats en court d'exécution [DLAV10]. Cela permet de fournir des réponses de bonne qualité avant la fin de l'exécution de la requête.

Avantages et inconvénients Le principal désavantage des systèmes non-structurés est que la recherche de données est complexe. Pour trouver et accéder à une donnée (ou à un ensemble de données), un pair doit émettre une requête et la transmettre aux autres pairs. Cela a comme conséquence de générer un nombre de messages important et donc d'encombrer le réseau.

Les avantages de cette architecture sont nombreux. Tout d'abord elle est tolérante aux pannes et n'est pas sensible aux attaques. En effet la défaillance d'un pair entraîne la perte des données qu'il stockait mais ne remet pas en cause le fonctionnement du système dans sa globalité. Ensuite, le fait que les pairs stockent leurs index permet de garantir leur autonomie, en particulier dans la manière de construire l'index. Enfin la décentralisation totale des données permet de passer à l'échelle. Il n'y a pas de risque de surcharge dans la mesure où les pairs sont "égaux".

2.2.2.2 *Systèmes structurés*

Architecture Les systèmes P2P structurés ont été proposés pour palier au problème de performances connus dans les systèmes non-structurés. L'idée est de répartir les données de sorte qu'elles soient accessibles "rapidement". Les systèmes structurés utilisent des tables de hashage distribuées (DHT, *Distributed Hash Table*) pour répartir les données. Les pairs sont identifiés de manière unique dans le système. L'identifiant d'un pair est obtenu en appliquant une fonction de hashage (par exemple la fonction SHA-1) sur son adresse IP. Chaque pair du système stocke dans une table de routage les identifiants d'autres pairs. Dans le système Chord, proposé en 2001 par Stoica, Morris, Karger *et al.*, les pairs sont organisés en anneau (*ring*) [SMK+01]. Chaque pair p stocke m identifiants d'autres pairs (appelés successeurs) dans sa table : le i-ème identifiant correspond au successeur $((p + 2^{i-1}) \mod 2^m)$. La figure 2.3 présente un système structuré de 8 pairs. Le pair p_1 stocke dans sa table les identifiants de $m = 3$ pairs : p_2, p_3 et p_5.

Étant donné un espace de clés, chaque pair du système se trouve responsable d'un intervalle de clés de cet espace. Lorsqu'un pair souhaite partager une donnée *data*, identifiée par le nom *name*, il utilise une fonction de hashage pour générer une clé *key*. Il peut alors utiliser la primitive *put* pour publier la donnée dans la DHT : $put(key, data)$. Le pair responsable de l'intervalle auquel appartient la clé est alors en charge de stocker la donnée localement. Supposons que l'espace des clés est l'intervalle $[1, 80]$, et que le pair p_i est responsable de l'intervalle $[10 \cdot (i-1) + 1, 10 \cdot i]$ pour $i \in [1, 8]$. Lorsqu'un pair du système

présenté sur la figure 2.3 publie une donnée dont la clé est 68, celle-ci doit être prise en charge par le pair p_7 car $68 \in [61, 70]$.

D'autres topologies que l'anneau existent : *tree* (comme Tapestry [ZHS+04]), *hypercube* (comme CAN [RFH+01]), *butterfly* (comme Viceroy [MNR02]), XOR (comme Kademlia [MM02]), hybride (comme Pastry [RD01]).

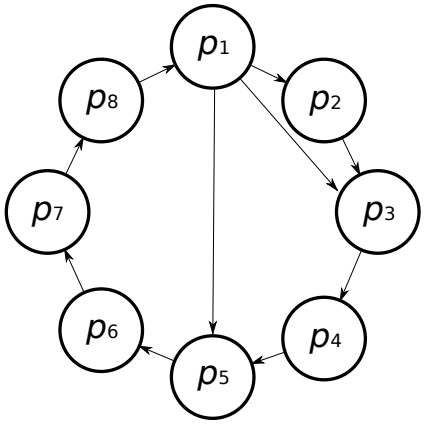


FIGURE 2.3 – *Système P2P structuré en anneau. Une seule connexion apparaît pour chaque pair, sauf pour le pair p_1 pour lequel les 3 connexions sont matérialisées.*

Méthode de routage Dans les systèmes structurés, l'accès aux données se fait en utilisant le système de clé. Lorsqu'un pair souhaite accéder à une donnée, il doit générer sa clé (à partir du nom du fichier par exemple) et utiliser la primitive *get* pour récupérer la donnée chez le pair qui la stocke. Le routage de la requête utilise le fait que les pairs sont organisés. Par exemple si le pair p_1 du système présenté sur la figure 2.3 souhaite accéder à la donnée *data* dont la clé est 68, il suffit qu'il choisisse dans sa table le pair dont l'intervalle de clé est le plus proche de la clé qu'il recherche. Dans ce cas il s'agit du pair p_5. Ce dernier route à son tour la requête vers le pair le plus proche de la clé. La requête atteint finalement le pair p_7 qui peut retourner la donnée au pair p_1. Dans les DHT en anneau, comme Chord, n'importe quelle donnée est accessible en, au plus, $O(\log N)$ sauts où N est le nombre de pairs dans le système.

Avantages et inconvénients Le principal avantage des systèmes P2P structurés est leur capacité à accéder rapidement aux données : $O(\log N)$ sauts pour accéder à n'importe quelle donnée du système. Néanmoins cette approche est conçue pour les requêtes *exact match*. Elle est donc peu adaptée aux requêtes complexes telles que les requêtes top-k (même si des solutions ont été proposées [APV07]). De plus la maintenance de la topologie a un coût important car l'entrée ou la sortie d'un pair nécessite la réattribution des intervalles de clés.

2.2.3 Systèmes hiérarchiques

Architecture Les systèmes P2P hiérarchiques ont été proposés pour tenter de tirer partie des avantages des systèmes centralisés et décentrali-

sés. Dans de tels systèmes, certains pairs sont choisis/élus pour devenir des super-pairs. Ceux-ci jouent le rôle de serveurs pour d'autres pairs. Ils peuvent être choisis parce qu'ils ont une capacité importante de calcul, de mémoire, etc. Au sein du système, les super-pairs sont organisés soit de manière structurée, soit de manière non-structurée. La figure 2.4 présente un système hiérarchique dans lequel trois super-pairs (p_1, p_3 et p_4) sont organisés de manière non-structurée et jouent le rôle de serveurs pour les autres pairs. Edutella est un exemple de système P2P hiérarchique [NWQ+02]. Le système Napster peut également être vu comme un système hiérarchique dans lequel il n'y a qu'un seul super-pair (le serveur).

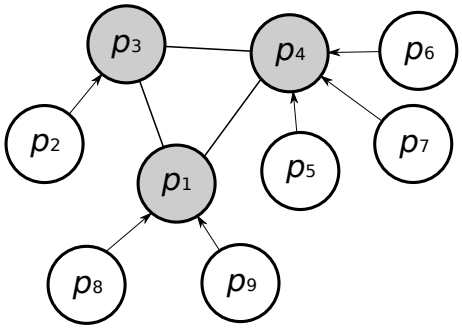


FIGURE 2.4 – *Système P2P hiérarchique.*

Méthodes de recherche Dans ces systèmes, les super-pairs accèdent aux données de la même manière que dans les systèmes non-structurés ou structurés (suivant la manière dont ils sont organisés). Les autres pairs contactent simplement le super-pair auquel ils sont rattachés. C'est ce dernier qui est chargé de leur fournir les données qu'ils recherchent.

Avantages et inconvénients Les systèmes hiérarchiques permettent de réduire les coûts de recherche de données de manière importante. En particulier le nombre de messages échangés lors d'une recherche est bien inférieur à celui correspondant à une méthode telle que la BFS. Néanmoins cette architecture est moins tolérante aux pannes car la défaillance d'un super-pair peut être problématique pour les pairs qui y sont rattachés. De plus cette approche nécessite de maintenir un réseau de super-pairs et de mettre en place une stratégie pour les choisir.

2.2.4 Comparaison

Dans cette section nous présentons une comparaison succincte des différentes architectures de systèmes P2P. Les critères sur lesquels nous nous focalisons sont le passage à l'échelle, la dynamicité, la tolérance aux pannes ou aux attaques, l'autonomie des pairs (en particulier dans le choix de la représentation des données), l'expressivité des requêtes et l'efficacité des méthodes de routage de requêtes.

Dans le tableau 2.1, nous voyons que les systèmes P2P à index centralisé ne permettent pas le passage à l'échelle car le fait qu'un seul pair (ou

serveur) stocke l'index de toutes les données stockées dans le système peut conduire à l'effet de goulot d'étranglement. De plus, l'aspect centralisé rend le système sensible aux pannes et empêche les pairs de choisir la manière dont ils représentent leurs données (ils sont contraints par le "schéma" de l'index central). En contrepartie de ces faiblesses, l'architecture centralisée fournit un service de recherche efficace, c'est-à-dire qui ne dépend pas du nombre de pairs du système.

Les systèmes P2P non-structurés répondent à toutes les exigences : passage à l'échelle, tolérance aux pannes, dynamicité, autonomie. Néanmoins, le routage de requêtes dans de tels systèmes est coûteux, car celles-ci doivent être envoyées à un grand nombre de pairs afin de localiser des données pertinentes.

Les systèmes P2P structurés ont l'avantage de fournir un mécanisme de recherche efficace. Le désavantage est qu'il faut maintenir une topologie particulière et que les requêtes correspondent à des clés (ce ne sont pas des requêtes riches/complexes). D'une certaine manière cela limite l'autonomie des pairs.

Les systèmes hiérarchiques présentent certains avantages et inconvénients des systèmes non-structurés et structurés. Le choix de les utiliser dépend en réalité de l'usage que l'on souhaite en faire.

	Central.	Non-struct.	Struct.	Hiérarch.
Passage à l'échelle	-	+	+	+
Dynamicité	+	+	-/+	-/+
Tolérance aux pannes	-	+	+	+
Autonomie	-	+	-	-/+
Expr. des requêtes	+	+	-	-/+
Rout. de requêtes	+	-	+	-/+

TABLE 2.1 – *Tableau comparatif des différentes architectures de systèmes P2P.*

2.3 Protocoles épidémiques

Les algorithmes épidémiques ont d'abord été proposés pour la maintenance de base de données répliquées sur différents sites [DGH+88]. Ce principe a été largement utilisé dans les systèmes distribués en général, et dans les systèmes P2P en particulier [KvS07].

2.3.1 Principes

Dans les protocoles épidémiques (ou protocoles de *gossiping*), chaque pair est composé de deux processus (*threads*) : un actif et un passif. Le processus actif est utilisé pour initier régulièrement des communications avec d'autres pairs. Ces derniers sont sollicités au travers de leurs processus passifs. Les algorithmes 1 et 2 présentent les instructions exécutées par chacun des processus.

Dans le processus actif, le pair p doit commencer par sélectionner de manière aléatoire d'autres pairs du système. Pour simplifier la présentation nous ne considérons qu'un seul autre pair p'. Il sélectionne ensuite un ensemble de données à envoyer, et les envoie. Il attend alors que le pair p'

lui réponde. Lorsque que ce dernier a répondu, le pair p traite les données reçues. Quand un pair p' est contacté par un autre au travers du processus passif, il doit lui répondre en sélectionnant et en envoyant des données.

Algorithme 1 : Processus actif d'un pair p.

```
1 while true do
2 |   p' ← selectPeer() ;
3 |   data ← selectToSend() ;
4 |   sendTo(p', data) ;
5 |   data' ← receiveFrom(p') ;
6 |   processData(data') ;
7 |   wait(θ) ;              // θ : délai entre deux cycles.
```

Algorithme 2 : Processus passif d'un pair p'

```
1 data ← receiveFromAny() ;
2 data' ← selectToSend() ;
3 sendTo(p, data') ;
4 processData(data) ;
```

Dans les protocoles épidémiques, les pairs doivent réaliser trois tâches principales : la sélection aléatoire d'un pair du système : selectPeer(), la sélection de données à envoyer : selectToSend(), et le traitement de données reçues : processData(*data*).

Sélection d'un pair Le choix du pair avec qui initier un échange doit être aléatoire et uniforme : tous les pairs du système ont la même probabilité d'être choisis (probabilité égale à $1/(N-1)$ s'il y a N pairs dans le système). Évidemment cette contrainte est forte car cela nécessite que chaque pair ait la connaissance de l'ensemble des pairs du système. Pour réaliser cette tâche, il est possible d'utiliser un service d'échantillonnage de pairs (*peer sampling service*) [JGKvS04]. Ce service permet à un pair de sélectionner de manière aléatoire et uniforme un autre pair du système.

Sélection de données La sélection de données est complétement dépendante de l'application concernée et des objectifs visés. De manière générale les pairs sélectionnent des données (soit aléatoirement, soit en utilisant une mesure de pertinence) qu'ils possèdent localement. Par exemple dans le cas du service d'échantillonnage de pairs, qui est lui-même un protocole épidémique, les données échangées sont des identifiants de pairs. Dans d'autres cas, il peut s'agir d'informations plus complexes sur les pairs, ou même d'informations qui ne sont pas directement liées aux pairs (des index, des données, etc.).

Traitement des données Le traitement des données est tout autant dépendant de l'application que la sélection. Dans le cas du service d'échantillonnage de pairs, le traitement des données consiste simplement à stocker localement les nouveaux identifiants des pairs.

2.3.2 Applications

Les protocoles épidémiques permettent de disséminer des informations ou des données au sein d'un système distribué. Ils permettent par exemple de maintenir des bases de données répliquées [DGH+88]. Leur efficacité repose sur le fait que les pairs échangent régulièrement des messages, et que le choix des pairs avec qui échanger des messages est uniformément aléatoire. Sous ces conditions, les protocoles épidémiques permettent de propager une donnée dans le système en $O(\log N)$ cycles. Le problème de redondance (c.-à-d. le fait qu'un pair reçoive de nombreuses fois la même donnée) est présent, en particulier pour les pairs dont le nombre de connexions entrantes est important.

Les protocoles épidémiques permettent également de manipuler la topologie d'un système P2P. Jelasity, Montresor et Babaoglu ont proposés le protocole T-Man [JB05, JMB09]. Celui-ci permet de construire une topologie particulière (par exemple un torus). Pour cela les pairs échangent des informations aux sujets d'autres pairs. Une mesure de distance entre pairs est utilisée pour les ordonner. Ce type d'approche a été utilisé pour modifier la topologie d'un système afin d'améliorer les performances d'accès aux données. Par exemple, dans [BK07, BFG+10], des protocoles épidémiques sont utilisés pour propager les profils d'utilisateurs. Cela permet de rapprocher les pairs qui ont des intérêts en communs. Dans [BBG+10], les auteurs présentent un protocole épidémique appelé P3Q. Ce dernier permet de rapprocher les pairs ayant des comportements sociaux similaires. Dans ce cas le comportement des pairs est associé à une activité de tagging.

Les protocoles épidémiques permettent également de gérer des ressources, de fournir un service d'échantillonnage de pairs, etc. Des descriptions plus précises sont données dans [KvS07].

2.4 Bilan

Dans ce chapitre nous avons commencé par présenter les principes et caractéristiques des systèmes P2P. Nous avons décrit les principales architectures et nous les avons comparées. Nous avons vu que les systèmes P2P à index centralisé ne garantissent pas l'autonomie des pairs. Nous avons également constaté que les systèmes structurés sont performants pour le routage de requêtes mais qu'ils sont seulement adaptés pour les requêtes *exact match*. Les systèmes hiérarchiques peuvent souffrir d'un problème de tolérance aux pannes. Si les participants du système souhaitent émettre des requêtes riches/complexes (des requêtes en langage naturel, des requêtes à base de mots-clés, ou des requêtes top-k), les systèmes non-structurés sont les plus adaptés. De plus, ces derniers garantissent l'autonomie des pairs car ceux-ci sont libres de représenter leurs documents comme ils le souhaitent.

Dans la seconde section de ce chapitre nous avons introduit les protocoles épidémiques. Ils permettent de réaliser de nombreuses tâches dans les systèmes P2P : maintenance de la topologie, dissémination de données, etc.

Recherche d'information

3

Dans ce chapitre nous présentons les concepts relatifs à la recherche d'information (RI). Après avoir introduit la notion de recherche d'information, nous présentons différents modèles de RI. Un modèle de RI définit la manière dont les documents et les requêtes sont représentés, et la manière dont est évaluée la pertinence d'un document par rapport à une requête. Nous présentons entre autres le modèle booléen, le modèle vectoriel, et le modèle probabilistique de pertinence.

Nous présentons ensuite la méthodologie générale qui permet d'évaluer une méthode de RI. Pour cela nous introduisons la notion de corpus de test. Un corpus contient généralement un ensemble de documents, de requêtes et un jugement de pertinence. Ce dernier détermine l'ensemble des documents pertinents pour chaque requête du corpus. Parmi les corpus connus, nous présentons brièvement Cranfield, TREC, et CLEF. Nous présentons également différentes mesures qui permettent de caractériser l'efficacité d'une méthode : la précision, le rappel, la F-mesure, etc. Ces mesures sont applicables dans un contexte d'évaluation, c'est-à-dire lorsque les réponses attendues sont connues : il est donc nécessaire d'utiliser un corpus.

3.1 Introduction à la recherche d'information

Le domaine de la Recherche d'Information (RI) a commencé à faire l'objet de nombreuses études lorsque les bibliothèques ont entrepris d'automatiser le rangement, le classement et la recherche d'ouvrages. La RI est une branche de l'informatique qui s'intéresse à l'acquisition, l'organisation, le stockage, la recherche et la sélection d'information. Selon Boughanem et Savoy, son but est retourner des documents pertinents pour le besoin en information [BS08]. Manning, Raghavan et Schütze définissent la recherche d'information comme étant la tâche qui consiste à « trouver des données (généralement des documents) d'une nature non structurée (généralement des textes) qui satisfont un besoin en information au sein de grandes collections (généralement stockées sur des ordinateurs) » [MRS08a]. De manière plus générale, les documents peuvent être des textes, des ouvrages, des tuples d'une base de données, etc. Les trois éléments centraux à la RI sont le **besoin en information** (exprimé sous forme de requête), les **données**, et la **pertinence** entre une donnée et un besoin en information. Ces trois éléments se retrouvent dans les différents modèles de RI que nous présentons dans la section suivante.

3.2 Modèles de recherche d'information

Un modèle de recherche d'information définit de quelle manière les documents et les requêtes doivent être représentés ainsi que la manière d'estimer la pertinence entre un document et une requête. La représentation des données (documents et requêtes) sert à (i) stocker les informations de manière synthétique, (ii) faciliter la comparaison entre deux données (par ex. un document et une requête), (iii) unifier les données (l'information contenue dans une page web ou une vidéo peut être représentée de la même façon). Cette représentation est obtenue par indexation. L'indexation est le processus qui consiste à extraire les informations importantes d'un document ou d'une requête. L'objectif est d'obtenir, pour chaque document et chaque requête, une description reflétant leur contenu. L'indexation peut être réalisée manuellement ou bien de manière automatique. Dans le cas de l'indexation manuelle, il faut que celle-ci soit réalisée par des spécialistes du domaine traité dans les documents. En effet, pour obtenir une indexation de qualité, il faut que les termes de l'index soient choisis avec précision. Le coût d'une telle indexation est important, d'autant plus lorsque la collection de documents à indexer est grande. L'indexation automatique est bien moins coûteuse, et fournit des résultats similaires à ceux obtenus manuellement. Baeza-Yates et Ribeiro-Neto assimilent l'indexation à un pré-traitement des documents [BYRN99]. Les phases clés du pré-traitement sont :

- L'élimination des mots vides de sens (*stop words*) : *le*, *la*, *du*, *pour*, etc. Ces derniers sont tellement communs qu'il ne sert à rien de les considérer. Ils sont généralement spécifiques à une langue et/ou à une collection de documents.
- La lemmatisation (ou racinisation) consiste à ramener les termes à une racine commune. Par exemple, les termes *précieuse*, *précieuses*, ou *préciosité*, sont transformés en une racine simple : *précieux*.

- La sélection des termes de l'index consiste à choisir les termes porteurs de sens et significatifs. En général ils sont choisis parmi les noms car ceux-ci sont plus significatifs que les verbes ou les adjectifs.
- La pondération des termes se base souvent sur les facteurs *tf* (la fréquence d'apparition d'un terme dans un document) et *idf* (la fréquence d'apparition de ce même terme dans toute la collection considérée) qui permettent de considérer les pondérations locale et globale d'un terme.

Le calcul de pertinence (ou calcul de similarité) entre un document et une requête sert à déterminer si un document est pertinent pour une requête donnée. Le résultat de ce calcul peut être booléen (le document est pertinent ou il ne l'est pas), ou il peut être numérique (par ex. une valeur comprise entre 0 et 1). Dans ce dernier cas, la mesure de pertinence permet de classer les documents en fonction de leur pertinence.

3.2.1 Modèle booléen

Représentation des documents et des requêtes Dans le modèle booléen, les documents sont représentés par des ensembles de termes issus d'un vocabulaire $V = t_1, ..., t_n$. La représentation d'un document d est l'ensemble des termes de V qui apparaissent dans d. Par exemple un document peut être représenté par l'ensemble $\{t_1, t_3, t_8\}$.

Les requêtes sont exprimées suivant le formalisme de l'algèbre de Boole. Une requête est une expression booléenne dans laquelle les variables logiques sont les termes du vocabulaire V, et les opérateurs usuels sont considérés : conjonction ($\wedge$), disjonction ($\vee$) et négation ($\neg$). La requête t_2 permet d'obtenir les documents dont la représentation contient le terme t_2. La requête $t_2 \vee (t_3 \wedge t_8)$ permet d'obtenir les documents dont la représentation contient soit le terme t_2, soit à la fois les termes t_3 et t_8.

Calcul de pertinence La mesure de pertinence d'un document d par rapport à une requête q est donnée par :

$$pert(d, q) = \begin{cases} 1 & \text{si } d \text{ satisfait } q \\ 0 & \text{sinon} \end{cases} \tag{3.1}$$

On dit alors simplement que d est pertinent pour q si $pert(d, q) = 1$. Par exemple le document $d = \{t_1, t_3, t_8\}$ n'est pas pertinent pour la requête t_2 (car $t_2 \notin d$). Par contre ce même document est pertinent pour la requête $t_2 \vee (t_3 \wedge t_8)$ (car $t_3 \in d \wedge t_8 \in d$). Dans ce modèle, le résultat du calcul de pertinence est une valeur booléenne. Il ne permet donc pas de classer les documents en fonction de leur pertinence par rapport à une requête.

Avantages et inconvénients Le modèle booléen a l'avantage d'être simple à mettre en œuvre. En effet, un document est seulement représenté par l'ensemble des termes qui le composent. L'indexation est donc relativement simple à effectuer. Ce modèle est également très simple du point de vue des utilisateurs car les requêtes sont simples à écrire. Néanmoins il

a l'inconvénient de ne pas permettre le tri des documents pertinents. Dans le cas d'une requête renvoyant de nombreux documents cela est handicapant pour les utilisateurs qui doivent parcourir la liste en intégralité.

3.2.2 Modèle vectoriel

Représentation des documents et des requêtes Dans le modèle vectoriel, les documents et les requêtes sont représentés par des vecteurs à n dimensions. Chaque dimension correspond à un terme d'un vocabulaire $V = t_1, ..., t_n$. Chaque dimension du vecteur d'un document (ou d'une requête) est pondérée selon l'importance du terme dans le document. La valeur de pondération peut être normalisée dans l'intervalle $[0, 1]$. La figure 3.1 présente le vecteur d'un document dont toutes les dimensions sont pondérées à 0, sauf les dimensions relatives aux termes t_1, t_3 et t_8. Cela signifie que seuls ces concepts sont représentatifs du contenu du document.

t_1	t_2	t_3	...	t_8	...	t_n
0,3	0	1	...	0,5	...	0

FIGURE 3.1 – *Exemple d'un vecteur représentant un document.*

La pondération des dimensions reflète l'importance des termes dans le document. Elle peut être calculée en considérant la fréquence d'apparition de chaque terme dans le document. La fréquence du terme t_i dans un document d est notée tf (pour *term frequency*) et est donnée par :

$$tf(t_i, d) = \frac{\text{nombre d'apparitions de } t_i \text{ dans } d}{\text{nombre de termes dans } d} \tag{3.2}$$

Plus un terme apparaît dans un document, plus sa pondération dans le vecteur est élevée. La fréquence des termes d'un document doit être recalculée à chaque fois que le document est modifié. La fréquence d'un terme dans un document peut être élevée pour deux raisons : soit parce que le terme est très représentatif du document, soit parce que le terme est très général. Pour capturer ce phénomène, il faut considérer la fréquence du terme dans l'ensemble des documents de la base $\mathcal{B}$ (ou du corpus). La fréquence inverse du document (notée idf pour *inverse document frequency*) est donnée par :

$$idf(t_i, \mathcal{B}) = \log\left(\frac{\text{nombre de documents dans la base } \mathcal{B}}{\text{nombre de documents contenant le terme } t_i}\right) \tag{3.3}$$

Cette mesure détermine à quel point un terme est général. Elle doit être recalculée à chaque fois que la base $\mathcal{B}$ est modifiée (ajout ou suppression de documents). La pondération du terme t_i dans document d (de la base $\mathcal{B}$) est généralement donnée par le $tf \cdot idf$ définit par le produit $tf(t_i, d) \times idf(t_i, \mathcal{B})$.

Calcul de pertinence Dans ce modèle, la pertinence entre un document et une requête se calcule en mesurant la similarité entre vecteurs. La me-

sure la plus utilisée est le cosinus :

$$pert(d,q) = \cos(\overrightarrow{d}, \overrightarrow{q}) = \frac{\overrightarrow{d} \cdot \overrightarrow{q}}{\| \overrightarrow{d} \| \times \| \overrightarrow{q} \|} \tag{3.4}$$

Dans cette mesure, $\overrightarrow{d} \cdot \overrightarrow{q}$ représente le produit scalaire entre le vecteur du document et le vecteur de la requête ; et $\| \overrightarrow{v} \|$ représente la norme du vecteur $\overrightarrow{v}$.

Avantages et inconvénients Le premier avantage de ce modèle est que les documents et les requêtes sont représentés de manière simple. Le deuxième avantage est qu'il permet de pondérer les termes dans les index. Au delà de la recherche d'information, cela peut être utile pour caractériser un document (en présentant ses termes les plus représentatifs), ou pour catégoriser les documents. Enfin le dernier avantage est qu'il permet de trier les documents en fonction de leur pertinence. Cela représente évidemment un confort supplémentaire pour l'utilisateur final. Le principal inconvénient de ce modèle est que les dimensions qui forment l'espace vectoriel sont indépendantes. Ainsi les dimensions correspondant aux termes "voiture" et "véhicule" sont aussi indépendantes que les dimensions "voiture" et "arbre". Cela est un problème car une requête ne contenant que le terme "véhicule" n'est pas satisfaite par un document ne contenant que le terme "voiture" (alors qu'une voiture est un véhicule). Notons que ce problème est également présent dans le modèle booléen.

Extensions Plusieurs extensions de ce modèle ont été proposées. Le modèle vectoriel généralisé a été proposé pour tenter de limiter le problème d'indépendance des dimensions [WZW85]. Chaque dimension correspond à une combinaison de termes (co-occurrences). Cette approche a le désavantage d'augmenter le nombre de dimensions des vecteurs. Le modèle vectoriel thématisé est une extension du modèle vectoriel généralisé [BK03]. Dans ce modèle les dimensions sont orthogonales (c.-à-d. indépendantes). Chacune d'entre elles correspond à une thématique pouvant regrouper plusieurs termes. Cette approche est intéressante mais est relativement complexe à mettre en place. Le modèle vectoriel sémantique est similaire au modèle vectoriel classique sauf que les dimensions ne correspondent pas à des termes mais à des concepts issus d'une ontologie (ou d'une taxonomie). Ce modèle n'adresse pas le problème lié à l'indépendance des dimensions. Il permet néanmoins d'étendre les requêtes en utilisant la notion de similarité entre concepts (voir section 1.5.1, page 25) [Voo94, VCLV08], et/ou en exploitant la structure de l'ontologie. Ce modèle nécessite d'être capable d'indexer les documents sémantiquement. Harrathi *et al.* proposent une approche statistique pour l'indexation sémantique de documents multilingues [HRMC10]. Le processus est composé de deux étapes : l'extraction des termes (simples ou complexes), et la détection des concepts d'une ressource sémantique (thésaurus ou ontologie). Le framework Mysins intègre également un module d'indexation sémantique qui produit des vecteurs sémantiques à partir de documents textuels [VCC+10].

3.2.3 Modèle booléen étendu

Représentation des documents et des requêtes Dans le modèle booléen étendu proposé par Salton, Fox et Wu [SFW83] les documents sont représentés par des vecteurs à n dimensions (comme dans le modèle vectoriel) et les requêtes sont des expressions booléennes (comme dans le modèle booléen). L'idée est de profiter des avantages du modèle vectoriel (le fait que les termes sont pondérés en fonction de leur importance dans le document) et des avantages du modèle booléen (la simplicité du point de vue de l'utilisateur).

Calcul de pertinence Pour expliquer le calcul de la pertinence d'un document par rapport à une requête, nous nous limitons à un espace à deux dimensions : t_1 et t_2. Dans un souci de simplicité, nous présentons la mesure de pertinence sur deux requêtes particulières. Soient deux requêtes $q_\wedge$ et $q_\vee$ définies par $t_1 \wedge t_2$ et $t_1 \vee t_2$. Pour qu'un document soit totalement pertinent pour $q_\wedge$, il faut que les termes t_1 et t_2 soient pondérés à 1 : il doit se trouver à la coordonnée $(1,1)$. De manière générale, plus le document est proche de la position $(1,1)$, plus il est pertinent pour la requête q_1. Ainsi le document d_1 de la figure 3.2.a est plus pertinent que le document d_2. Dans ce cas, le degré de pertinence d'un document par rapport à une requête est donné par :

$$pert(\overrightarrow{d}, q_\wedge) = 1 - \sqrt{\frac{(1 - \overrightarrow{d}[t_1])^2 + (1 - \overrightarrow{d}[t_2])^2}{2}} \tag{3.5}$$

où $\overrightarrow{d}[t_i]$ représente la pondération du terme t_i dans le vecteur $\overrightarrow{d}$.

Pour qu'un document soit totalement pertinent pour $q_\vee$, il faut que le terme t_1 ou le terme t_2 soit pondéré à 1 : il doit se trouver à la coordonnée $(1,0)$, à la coordonnée $(0,1)$ ou à la coordonnée $(1,1)$. Ainsi le document d_1 de la figure 3.2.b est plus pertinent que le document d_2. Dans ce cas, le degré de pertinence d'un document par rapport à une requête est donné par :

$$pert(\overrightarrow{d}, q_\vee) = \sqrt{\frac{\overrightarrow{d}[t_1]^2 + \overrightarrow{d}[t_2]^2}{2}} \tag{3.6}$$

où $\overrightarrow{d}[t_i]$ représente la pondération du terme t_i dans le vecteur $\overrightarrow{d}$.

Les mesures de pertinence peuvent être généralisées au cas des requêtes composées d'un nombre quelconque de termes. Une mesure permet également de mesurer la pertinence d'un document par rapport à une requête combinant disjonctions et conjonctions. Les détails sont donnés dans [SFW83].

Avantages et inconvénients Le premier avantage de ce modèle est que les requêtes sont simples à formuler pour les utilisateurs. Le deuxième concerne la représentation des documents et la mesure de pertinence. En effet, ceux-ci permettent de trier les documents en fonction de leur pertinence (contrairement au modèle booléen classique). Les désavantages sont les mêmes que pour les modèles vectoriels.

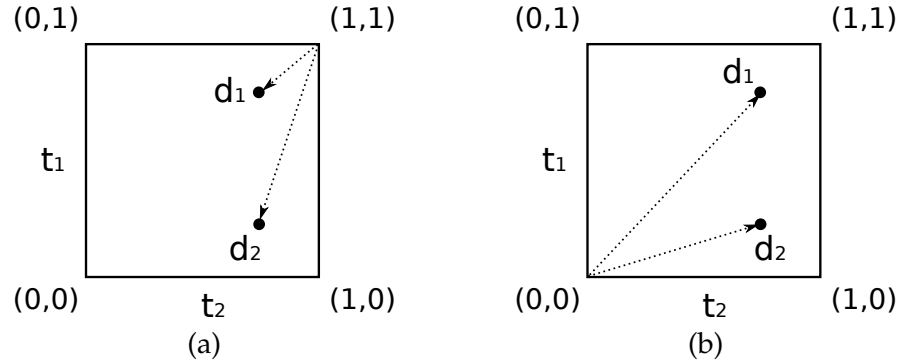


FIGURE 3.2 – *Exemple de deux documents d_1 et d_2 représentés dans un espace à deux dimensions. Dans le modèle booléen étendu, la pertinence est mesurée par la distance entre les documents et les coordonnées $(0,0)$ pour les requêtes disjonctives (a), et $(1,1)$ pour les requêtes conjonctives (b).*

3.2.4 Modèle probabiliste de pertinence

Représentation des documents et des requêtes Le modèle probabiliste a été proposé par ROBERTSON et SPÄRCK JONES en 1976 [RSJ76]. Dans ce modèle les documents et les requêtes sont représentés par des ensembles de termes d'un vocabulaire V (comme les documents dans le modèle booléen, voir section 3.2.1, page 46). On note $d[t_i]$ la pondération du terme t_i dans le document d. Elle vaut 0 si le terme t_i est absent de d, et 1 s'il est présent.

Calcul de pertinence La mesure de pertinence considère la probabilité $P(R|d)$ que le document d soit pertinent pour une requête q, et la probabilité $P(\overline{R}|d)$ que le document d ne soit pas pertinent pour la requête q. La mesure de pertinence est définie par :

$$pert(d,q) = \frac{P(R|d)}{P(\overline{R}|d)} \tag{3.7}$$

En utilisant le théorème de BAYES [Bay63], on obtient :

$$pert(d,q) = \sum_{i=1}^{|V|} d[t_i] \times w_i \tag{3.8}$$

où w_i dépend de la probabilité de présence du terme t_i dans l'ensemble des documents pertinents et des documents non pertinents. Au départ ces probabilités sont estimées de manière triviale puis sont raffinées au cours des itérations durant lesquelles les utilisateurs donnent un avis (un *feedback*).

Avantages et inconvénients Les systèmes utilisant le modèle probabilistique obtiennent de bons résultats, à l'image du système OKAPI [RWHB00]. Néanmoins ce modèle est assez complexe à mettre en œuvre car il nécessite d'avoir une estimation des probabilités initiales.

3.3 Évaluation d'une méthode de RI

Pour évaluer une méthode de RI il faut être capable de déterminer à quel point cette méthode retourne les résultats qu'elle aurait théoriquement dû renvoyer. Pour cela, il faut disposer d'un corpus de test, contenant à la fois des documents, des requêtes et les résultats théoriques attendus, et de mesures permettant de mesurer la qualité entre ce que retourne la méthode et ce qui aurait dû être retourné.

Dans cette partie nous ne prenons en compte que la notion d'*efficacité* (qualité des résultats de la méthode), et pas celle d'*efficience* (coûts engagés pour traiter le problème : temps, mémoire, etc.).

3.3.1 Corpus de test

Un corpus de test contient une collection de documents, une collection de requêtes et un jugement de pertinence [MRS08b].

3.3.1.1 Documents et requêtes

La collection de documents regroupe généralement des documents de même type (des textes, des images, des vidéos, etc.). Les documents sont stockés dans leur forme originale : l'indexation doit être faite par le système de RI. Les requêtes expriment un besoin en information en langage naturel (par exemple : *Est-il plus efficace de boire du vin rouge que du vin blanc pour réduire les risques d'attaque cardiaque ?*) ou peuvent simplement être composées de mots clés (par exemple : *vin rouge blanc attaque cardiaque*) [1]. Les documents et les requêtes sont identifiés de manière unique afin de pouvoir les distinguer aisément. Les documents doivent être en nombre suffisant pour garantir que l'obtention des documents pertinents pour une requête n'est pas une tâche triviale. De la même manière le nombre de requêtes doit être raisonnable : 50 semble être un nombre minimal suffisant [MRS08c].

3.3.1.2 Jugement de pertinence

La construction du jugement de pertinence nécessite l'intervention d'évaluateurs. Ceux-ci sont chargés de déterminer l'ensemble des pertinents documents pour chaque requête. Cela représente évidemment un travail important et délicat car il faut que le jugement soit le plus objectif possible. Dans sa forme la plus simple, le jugement de pertinence est booléen : un document est pertinent pour une requête ou ne l'est pas. Dans certains corpus, une valeur de pertinence est attribuée aux documents. Ces valeurs permettent d'ordonner les documents en fonction de leur pertinence par rapport à une requête.

3.3.1.3 Corpus existants

Cranfield est le premier corpus a avoir permis d'utiliser des mesures précises pour qualifier un système de RI. Il est composé de 1400 docu-

1. Les exemples sont tirés de [MRS08c].

ments collectés au Royaume-Uni dans les années 50 : ils correspondent à des résumés d'articles publiés dans des journaux d'aéronautique. Il contient également 225 requêtes et un jugement de pertinence : pour chaque requête la liste complète des documents pertinents est fournie.

TREC (*Text REtrieval Conference*) est une conférence organisée annuellement depuis 1992. À chaque édition, le NIST (*National Institute of Standards and Technology*) fournit des collections de documents et de requêtes et un jugement de pertinence [2]. Ils sont utilisés pour tester différents systèmes de RI. Différentes pistes ont été proposées. L'édition de 2012 propose les pistes Contextual Suggestion Track, Crowdsourcing Track, Web Track, etc. Cette dernière piste se focalise sur des tâches spécifiques à la recherche sur le Web. Elle considère la collection ClueWeb09 qui contient environ 1 milliard de pages dans dix langues (anglais, français, chinois, arabe, etc.).

CLEF (*Cross Language Evaluation Forum*) vise à fournir une plateforme pour faciliter le développement, le test et l'évaluation de systèmes de recherche d'information multilingue [BP01]. Les documents qui composent le corpus sont issus de journaux : LA Times, Le Monde, Frankfurter Rundschau, Der Spiegel et La Stampa. Au total cela représente 360000 articles et quarante requêtes, traduites dans quatre langues (français, anglais, italien et allemand).

Le système SMART (System for the Mechanical Analysis and Retrieval of Text) est un système de RI développé dans les années 60 à l'université de Cornell [Sal90]. Ce système inclut un certain nombre de corpus : **ADI**, **Time**, **CACM**, **CISI**, **NPL**, etc. Ils sont accessibles en ligne [3].

Le tableau 3.1 présente les caractéristiques des quelques corpus que nous venons de présenter.

	Nbre de documents	Nbre de requêtes
Cranfield	1 400	225
TREC Web Track 2009	1 040 809 705	50
CLEF	360 000	40
ADI	82	35
Time	425	83
CACM	3204	64
CISI	1 460	112
NPL	11 429	93

TABLE 3.1 – *Corpus de test pour la recherche d'information.*

3.3.2 Mesures

Pour caractériser une méthode de RI, et plus particulièrement pour comparer deux méthodes, il faut être capable de déterminer à quel point une méthode est performante : à quel point elle est efficace et à quel point elle est efficiente. L'efficacité mesure la capacité d'une méthode à atteindre un objectif. L'efficience mesure la capacité d'une méthode à effectuer une tâche avec un coût minimal. Elle peut être mesurée en terme de temps

2. `http://trec.nist.gov/`
3. `ftp://ftp.cs.cornell.edu/pub/smart/`

de calcul, de ressources, de coût énergétique, etc. En RI on s'intéresse généralement davantage à l'efficacité qu'à l'efficience. Dans la suite de cette section nous présentons des mesures permettant de caractériser l'efficacité d'une méthode de RI.

3.3.2.1 *Précision et rappel*

Les mesures de précision et de rappel sont très utilisées pour évaluer la qualité d'une méthode de recherche d'information. Étant donnée une requête q, la précision détermine à quel point les résultats retournés par la méthode sont pertinents. Soient $\mathcal{A}$ l'ensemble des données retournées par la méthode et $\mathcal{R}$ l'ensemble des données pertinentes. La précision est donnée par :

$$P_q = \frac{\text{nombre de documents pertinents retournés}}{\text{nombre de documents retournés}} = \frac{|\mathcal{A} \cap \mathcal{R}|}{|\mathcal{A}|} \tag{3.9}$$

Si la précision est égale à $0,75$ cela signifie que 75% des documents retournés sont effectivement pertinents. Le rappel quant à lui détermine à quel point les résultats retournés couvrent l'ensemble des données pertinentes. Le rappel est défini par :

$$R_q = \frac{\text{nombre de documents pertinents retournés}}{\text{nombre de documents pertinents}} = \frac{|\mathcal{A} \cap \mathcal{R}|}{|\mathcal{R}|} \tag{3.10}$$

Si le rappel est égal à $0,75$ cela signifie que 75% des documents pertinents ont été retournés par la méthode.

La figure 3.3 présente l'ensemble des documents retournés pour une requête q, et l'ensemble des documents pertinents pour q. La zone hachurée correspond aux documents pertinents retournés. L'objectif d'une méthode de RI est de maximiser cette surface, tout en réduisant le nombre de faux-négatif (zone grise) et le nombre de faux-positifs (zone blanche).

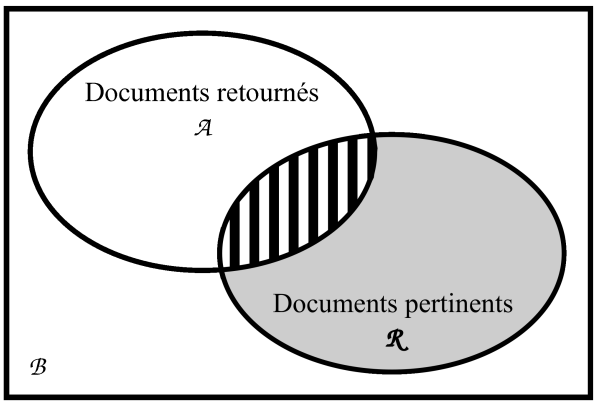


FIGURE 3.3 – *Ensembles des documents retournés ($\mathcal{A}$) et des documents pertinents ($\mathcal{R}$) pour une requête donnée parmi l'ensemble des documents de la collection ($\mathcal{B}$).*

3.3.2.2 *F-mesure*

La F_β-mesure est une mesure qui permet d'agréger les mesures de précision et de rappel. Étant donnée une requête q, la définition générale

de la F_β-mesure, pour $\beta \in \mathbb{R}^+$, est la suivante :

$$F_{q,\,\beta} = (1+\beta^2) \cdot \frac{P_q \cdot R_q}{\beta^2 \cdot P_q + R_q} \tag{3.11}$$

Le coefficient β sert à donner plus ou moins d'importance à la précision ou au rappel. Lorsque $\beta = 1$, la précision et le rappel sont considérés équitablement. Lorsque $\beta < 1$, l'accent est mis sur le rappel. La F_1-mesure est la plus utilisée (elle est d'ailleurs souvent appelée F-mesure) :

$$F_{q,\,1} = 2 \cdot \frac{P_q \cdot R_q}{P_q + R_q} \tag{3.12}$$

Dans ce cas, la F_1-mesure est la moyenne harmonique de la précision et du rappel.

3.3.2.3 *Précision et rappel à n*

Quand le modèle de RI le permet, il est possible de classer les documents selon leurs pertinences pour une requête donnée. Il est alors intéressant de mesurer l'ordre dans lequel les documents sont triés. Nous pouvons alors caractériser l'efficacité d'une méthode en considérant seulement les n meilleurs documents retournés. Dans ce cas, nous pouvons utiliser les mesures de précision à n et de rappel à n respectivement notées *P@n* et *R@n*. Plus n augmente, plus il devient difficile de ne pas sélectionner des documents non-pertinents (la précision décroît), et plus il est probable que les documents effectivement pertinents soient sélectionnés (le rappel croît). La figure 3.4 illustre ce phénomène.

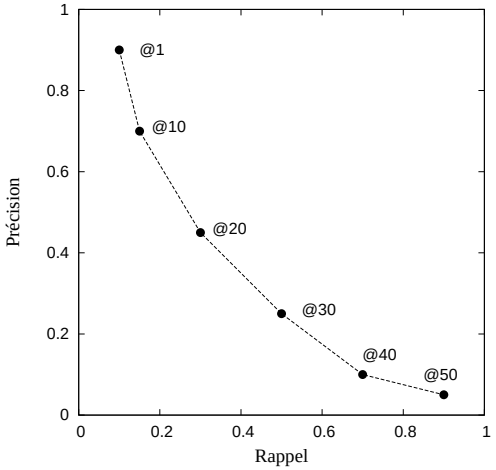


FIGURE 3.4 – *Valeurs de précision/rappel à n obtenues par une méthode donnée pour différentes valeurs de n.*

3.3.2.4 *Précision moyenne*

La précision moyenne (*average precision*) sert à mesurer la précision d'une méthode en tenant compte de l'ordre dans lequel les documents sont triés. Cette observation ne peut être faite qu'avec les modèles de RI

qui attribuent un score numérique pour la pertinence des documents. La précision moyenne est définie par :

$$\widetilde{P}_q = \frac{1}{|\mathcal{R}|} \sum_{i=1}^{|\mathcal{A}|} P_q[i] \cdot pert(i) \tag{3.13}$$

où $pert(i)$ vaut 1 si le i-ième document de l'ensemble des documents retournés $\mathcal{A}$ est pertinent, et 0 sinon ; et $P_q[i]$ est la précision limitée aux i premiers éléments de $\mathcal{A}$.

3.3.2.5 *MAP : Mean Average Precision*

Étant donné un ensemble de requêtes Q, la moyenne des précisions moyennes (*Mean Average Precision*, MAP) correspond à la moyenne des précisions moyennes pour chaque requête. Elle est définie par :

$$MAP_Q = \frac{1}{|Q|} \sum_{q \in Q} \widetilde{P}_q \tag{3.14}$$

Cette mesure est utilisée dans les campagnes d'évaluation telles que TREC ou CLEF.

3.4 Bilan

Dans ce chapitre nous avons présenté différents modèles de recherche d'information. Le modèle vectoriel sémantique semble le plus intéressant car il permet de trier les documents en fonction de leur pertinence par rapport à une requête. En particulier il permet de répondre à des requêtes de type top-k. De plus ce modèle permet d'étendre les requêtes en se basant sur une base de connaissance, un thésaurus ou une ontologie. Dans ce cas l'expansion de requêtes est guidée par des connaissances formalisées. Le modèle vectoriel classique permet également d'étendre les requêtes mais l'expansion se base généralement sur des données statistiques (cooccurrence de termes, etc.). Le dernier avantage du modèle vectoriel sémantique est qu'il est simple à mettre en place (contrairement au modèle probabiliste de pertinence).

Dans la seconde partie de ce chapitre nous avons présenté différents corpus et différentes mesures permettant de mesurer la qualité d'une méthode de RI. Ces deux éléments permettent d'évaluer l'efficacité d'une méthode de RI, c'est-à-dire à quel point elle permet de retourner les documents pertinents pour les requêtes considérées. Les mesures les plus utilisées sont la précision et le rappel.

Systèmes distribués de RI et de gestion de données

4

Dans ce chapitre nous présentons un état de l'art des systèmes distribués de recherche d'information, et des systèmes distribués de gestion de données dans lesquels les pairs utilisent des représentations (schémas, ontologies) différentes pour représenter leurs données. Nous présentons ensuite quelques méthodes permettant de réduire, dans une certaine mesure, l'hétérogénéité sémantique de systèmes P2P. Nous les classons en deux catégories : celles qui tentent de faire émerger une ontologie partagée et/ou d'identifier de nouvelles correspondances, et celles qui (ré)organisent les systèmes P2P de manière à faciliter le routage de requêtes, et à limiter le trafic réseau.

4.1 Systèmes distribués homogènes de recherche d'information

Dans cette section, nous présentons différents systèmes de recherche d'information distribués dans lesquels le problème d'hétérogénéité ne se pose pas : soit parce que les pairs n'utilisent pas de représentation sémantique, soit parce qu'ils utilisent tous la même.

FASD [1] est un moteur de recherche distribué [Kroo2]. Il repose sur un système P2P non-structuré (Freenet). L'idée de FASD est que les pairs partagent de l'espace disque pour stocker des méta-données (des vecteurs de termes), et participent au routage de requêtes dans le système. Le système permet de retourner les k documents les plus pertinents pour une requête (il s'agit de requêtes top-k).

Le système **iCluster** est un réseau superposé (*overlay network*) supportant la recherche d'information dans un environnement dynamique [RP08]. Dans ce travail les pairs partagent des documents qui sont représentés par des vecteurs de termes. Les documents sont organisés dans des groupes, et chaque groupe est représenté par son barycentre qui correspond à un intérêt. Chaque pair est représenté par la liste de ses intérêts. Les auteurs définissent plusieurs mesures de similarité entre pairs en se basant sur leurs intérêts. L'idée présentée dans ce travail consiste à organiser le système en *clusters* de pairs présentant des intérêts similaires. Des liens de courte portée (*short-range links*) sont créés entre des pairs ayant les mêmes intérêts et des liens de longue portée (*long-range links*) sont créés entre les clusters. Les liens de courte portée sont utilisés lors du routage des requêtes émises par les pairs, alors que les liens de longue portée sont utilisées pour maintenir le système connexe. Afin de construire et maintenir les différents clusters de pairs, chaque pair calcule régulièrement (pour chacun de ses intérêts) la cohésion de son cluster. Si la cohésion n'est pas suffisante (c.-à-d. inférieure à une valeur seuil), le pair initie une procédure permettant de découvrir de nouveaux liens. Pour cela il émet un message qui est alors transmis à certains pairs du système (un sous-ensemble de ceux qui sont accessibles avec un nombre limité de sauts).

Le moteur de recherche **JXTA Search** (distribué sous licence BSD) permet à un ensemble de pairs de partager des ressources, et d'émettre des requêtes [WDKF02]. Dans JXTA Search certains pairs jouent le rôle de pivot (*hub*). Il s'agit clairement d'une architecture hiérarchique. Les fournisseurs de données décrivent explicitement les données qu'ils partagent en XML. Par exemple un pair peut déclarer qu'il stockent des données issues du site `http://music.org` et qu'il peut traiter les requêtes portant sur l'article *Robert Johnson*. Ces informations sont transmises à un pair pivot qui les utilise ensuite pour router les requêtes dans le système. Les requêtes sont également définies en XML. Une requête émise par un pair peut exprimer le besoin suivant : quelles sont toutes les données issues du site `http://music.org` concernant un artiste dont le nom est *Johnson* ? Dans JXTA Search tous les pairs utilisent le même schéma XML.

DART [2] est un projet qui vise à développer, tester et intégrer des outils

1. FASD : Fault-tolerant, Adaptative, Scalable and Distributed search engine.
2. DART : Distributed Agent-based Retrieval Toolkit

pour construire un moteur de recherche distribué, sémantique et sensible à la géographie [AD+07]. Les participants d'un tel système sont appelés des nœuds de la communauté DART (*DART Community Node*). Certains d'entre eux sont des fournisseurs de données : ils parcourent le Web pour collecter des données. Ces données sont ensuite réparties sur l'ensemble des nœuds qui sont organisés dans une DHT. Les données sont indexées sémantiquement (par le biais d'ontologies et de processus de traitement automatique des langues). Cela permet de satisfaire les requêtes des utilisateurs en retournant les données qui présentent les mêmes concepts sémantiques que ceux contenus dans les requêtes.

Le système **AlvisP2P** est basé sur une DHT [LSK+08]. Son but principal est d'adresser le problème rencontré par les systèmes structurés lorsque les requêtes sont composées de plusieurs mots-clés. Un prototype a été développé [3]. AlvisP2P est composée de plusieurs couches comme cela est suggéré dans [AKRW04]. La première couche concerne la communication de bas niveau (protocoles TCP/IP, UDP/IP, etc.). La deuxième couche est liée aux fonctionnalités des systèmes P2P. Pour les DHT, cela correspond aux primitives (*put*, *get*, etc.). La troisième couche regroupe les fonctionnalités liées à la gestion des documents (indexation, clustering, etc.). La quatrième couche concerne les fonctionnalités liées à la recherche d'information (par exemple le tri des documents pertinents par rapport à une requête). Dans AlvisP2P, une cinquième couche est présentée (ce n'est pas le cas dans [LSK+08]). Elle correspond à un raffinement de la couche inférieure.

Les systèmes que nous venons de présenter tentent d'adresser les problèmes liés à l'accès aux données lorsque celles-ci sont distribuées dans un systèmes. Par contre, ils ne considèrent pas que les données sont représentées par rapport à différents schémas (schémas de bases de données, schémas XML, ontologies) : soit les pairs n'utilisent pas de schéma, soit ils utilisent un schéma commun. En cela, ces systèmes sont relativement éloignés de notre problématique. Dans la section suivante nous présentons des systèmes considérant que les pairs utilisent différents schémas pour représenter leurs données.

4.2 Systèmes distribués hétérogènes de gestion de données

Un système distribué de gestion de données (*Peer Data Management System*, PDMS) est un système d'intégration fournissant un accès à des bases de données hétérogènes sans utiliser un schéma centralisé [CM08b]. Plus globalement, nous considérons qu'un PDMS est un système distribué dans lequel les participants utilisent différents formalismes pour représenter leurs données : il peut s'agir de différents schémas de base de données, de différents schémas XML, de différentes ontologies. Dans cette section nous présentons différents PDMS. Nous commençons par présenter des systèmes dans lesquels les pairs partagent le contenu de leurs bases de données, puis des systèmes dans lesquels les pairs partagent des données

3. Il est disponible à l'adresse suivante : `http://globalcomputing.epfl.ch/alvis`.

semi-structurées représentées par rapport à des schémas XML. Enfin nous présentons des systèmes dans lesquels les pairs stockent des triplets RDF.

4.2.1 Bases de données relationnelles

Le système **Orchestra**, basé sur une DHT, se focalise sur la gestion des désaccords qui peuvent se produire au niveau des schémas utilisés pour représenter les données ou au niveau des instances [IKKC05]. Orchestra fournit une infrastructure permettant à des participants de partager de manière collaborative des données scientifiques ou académiques. Il s'intéresse au cas où les schémas utilisés par les pairs peuvent évoluer rapidement. L'approche proposée consiste à modifier les schémas (ou les données) hors ligne, puis de propager ces modifications aux autres pairs qui sont ensuite chargés de la réconciliation.

KRAFT est un système multi-agent dans lequel les participants (agents) partagent des données depuis de multiples sources hétérogènes [PHG+00]. Chaque participant stocke des données dans une base de données, et est libre d'utiliser le schéma de son choix. Une ontologie locale est associée à chaque source de données. Celle-ci définit l'ensemble des termes considérés dans le schéma. Pour adresser le problème d'hétérogénéité sémantique, une ontologie partagée est utilisée. La traduction d'un message, exprimé à l'aide des concepts d'une ontologie locale, est assurée par des agents médiateurs capables de calculer les correspondances entre deux ontologies.

Dans [AK10, AKHM08], Al King *et al.* considèrent des systèmes P2P structurés dans lesquels les pairs utilisent différents schémas de bases de données. Ils proposent de résoudre le problème d'hétérogénéité sémantique en utilisant une ontologie de domaine répliquée chez tous les pairs. Pour interopérer, les pairs "traduisent" leurs requêtes à l'aide du vocabulaire commun à tous les pairs défini par l'ontologie. Afin de pouvoir rejoindre le système, les pairs doivent définir des règles de correspondances entre leurs schémas et l'ontologie. En réalité les pairs doivent fournir une liste de synonymes de chaque relation (table) et pour chaque attribut. Des mesures de similarité sont proposées pour aider les pairs à définir les correspondances entre les tables/attributs et les classes de l'ontologie. Les correspondances sont alors utilisées pour reformuler les requêtes SQL avec les concepts de l'ontologie et les pairs pertinents sont contactés au travers de la DHT pour renvoyer les données adéquates.

4.2.2 Données XML

Le système **Piazza** a pour principal objectif de permettre aux utilisateurs d'échanger des données hétérogènes, c.-à-d. des données décrites avec différents schémas XML [TIM+03, HIM+04]. Comme tous les pairs n'utilisent pas nécessairement le même schéma, ils sont supposés calculer des correspondances. Le système utilise ces correspondances pour traduire les requêtes, et retourner les réponses pertinentes. Ce travail se focalise principalement sur la réécriture de requêtes. Lorsqu'une requête q est émise par un pair p (sur un schéma s), la requête est reformulée par le pair p' utilisant le schéma s'. Pour cela le pair p' utilise les correspon-

dances qui existent entre s et s'. Celles-ci sont accessibles au sein d'un catalogue centralisé. Pour améliorer les performances, les correspondances sont mises en cache chez tous les pairs. Hormis le problème lié à la centralisation des correspondances, qui est un frein au passage à l'échelle dans les systèmes P2P, cette solution ne traite pas le cas où des correspondances peuvent être incohérentes les unes par rapport aux autres.

Le système **PARIS** est un système P2P hybride [CPT05]. Étant donné un pair p, l'ensemble du système est divisé en quatre groupes : (i) le *local group* : il regroupe tous les pairs utilisant le même schéma que p, (ii) le *direct semantic group* : il regroupe les pairs n'utilisant pas le même schéma que p mais pour lesquels il existe des correspondances entre leurs schémas et celui de p, (iii) le *semantic transitive group* : il regroupe les pairs pour lesquels il existe un chemin d'alignements entre leurs schémas et celui de p, et (iv) le *foreign group* : il regroupe tous les pairs qui n'appartiennent à aucun autre groupe. Les pairs formant un *local group* sont organisés en système non-structurés. Certains d'entre eux participent à un système structuré (DHT) avec des pairs d'autres groupes. Cette architecture nécessite la mise en place de mécanismes spécifiques pour la gestion de la dynamicité, en particulier pour gérer l'arrivée et le placement de nouveaux pairs dans le système. L'avantage de cette architecture est qu'elle facilite le routage de requêtes : au sein des *local groups* les requêtes n'ont pas à être traduites, et les super-pairs (les pairs qui participent à la DHT) sont en charge de traduire les requêtes avant de les envoyer aux autres groupes. Dans PARIS les pairs faisant partie du même groupe partagent l'ensemble des correspondances qu'ils connaissent en les diffusant à tous les pairs (*broadcast*).

Dans l'approche **Chatty Web**, les auteurs considèrent des systèmes P2P dans lesquels les pairs utilisent des schémas XML pour représenter leurs données, et émettent des requêtes XQuery [ACMH03a, ACMH03b]. Les pairs, qui n'utilisent pas tous le même schéma, connaissent éventuellement des correspondances qu'ils stockent localement. Chaque correspondance est exprimée par une requête XQuery. Dans ce contexte, les auteurs font les constatations suivantes : (i) les pairs peuvent s'échanger des requêtes même s'ils ne possèdent de correspondances directes entres leurs schémas (par transitivité des correspondances), (ii) une requête peut effectuer un cycle dans le système et donc revenir au pair initiateur (après avoir éventuellement été traduites). La présence de cycle est utilisée pour tenter de détecter des correspondances incorrectes et de faire converger les pairs vers un accord sémantique global. Les auteurs proposent une mesure de similarité syntaxique pour déterminer s'il est nécessaire de transmettre la requête après traduction. En effet si la traduction a provoqué trop de perte d'information il n'est pas nécessaire de l'envoyer à d'autres pairs. Lorsqu'une requête atteint un pair pour la seconde fois, celui-ci analyse la requête et les résultats pour déterminer si des mauvaises traductions ont été appliquées. L'intérêt de cette analyse est double. Premièrement il permet de guider le routage des requêtes afin d'éviter les pairs dont les correspondances sont erronées. Deuxièmement il permet de corriger (ou supprimer) les correspondances connues par les pairs.

4.2.3 Dépôts de triplets RDF

Edutella est un système P2P dans lequel les pairs annotent leurs ressources à l'aide d'ontologies (décrites en RDFS) [NWQ+02, NSS03]. Le système est basé sur une architecture hiérarchique. Les super-pairs sont organisés en hypercube, et les pairs "ordinaires" sont connectés en étoile (*star-like fashion*) autour des super-pairs. L'intérêt d'une telle topologie repose sur le routage des requêtes : les requêtes sont envoyées aux super-pairs qui sont chargés de produire les plans d'exécution. Les pairs "ordinaires" n'ont plus qu'à exécuter localement les requêtes.

GridVine est un système P2P composé de deux couches : une couche physique et une couche logique [ACMHvP04, CMAA07]. Dans ce système, les données publiées sont des triplets RDF et les schémas sont des schémas RDFS. Au niveau physique, les pairs sont organisés de manière structurée au sein d'une DHT. Les opérations de base permettent aux pairs de faire correspondre des données ou des méta-données à des clés de routage. Cette approche permet de tirer parti des avantages des systèmes structurés. Le niveau logique est dédié aux tâches relatives à la gestion, à l'alignement et à l'héritage de schéma. L'héritage de schéma permet aux utilisateurs de créer de nouveaux schémas à partir de schémas existants (par dérivation) pour coller aux besoins des utilisateurs. Le niveau logique supporte également une méthode de réconciliation de schémas. Cette méthode est appelée *Semantic gossiping* [ACMH02, ACMH06], mais elle ne se rapporte pas à la notion de *gossiping* présentée dans la section 2.3 (page 40).

Le système **RDFPeers** est basé sur une DHT [CF04]. Dans ce système les pairs stockent des triplets RDF. Chaque triplet $\langle o, p, v\rangle$ est stocké trois fois en fonction de l'objet o, du prédicat p et de la valeur v. Les pairs responsables des clés correspondant à o, p et v stockent tous le triplet $\langle o, p, v\rangle$. Les requêtes sont également des triplets. Par exemple, la requête $\langle ?o, dc : creator, John\rangle$ demande l'ensemble des ressources ayant été créées par l'auteur John. L'obtention des triplets répondant à une requête se fait par l'utilisation des fonctionnalités de base de la DHT.

4.3 Méthodes de réconciliation sémantique et méthodes de réorganisation

Dans la littérature des méthodes ont été proposées pour réduire dans une certaine mesure l'hétérogénéité sémantique de systèmes P2P. Nous les classons en deux catégories principales : celles qui tentent de faire émerger une ontologie partagée ou qui identifient de nouvelles correspondances, et celles qui organisent le système de manière à faciliter le routage de requêtes. Dans les deux cas, les méthodes proposées sont principalement conçues pour améliorer l'interopérabilité.

4.3.1 Méthodes de réduction des disparités entre pairs

De Souza *et al.* proposent la méthode **Emergent Ontology** qui a pour objectif de générer une ontologie globale permettant de représenter toutes

les ressources du système [DDC10]. Pour cela les auteurs proposent un ensemble d'heuristiques permettant d'identifier les concepts dans les ontologies des pairs devant être pris en considération dans l'ontologie globale. Ils se basent sur un ensemble de correspondances entre ontologies pouvant être obtenu avec une méthode d'alignement classique [ES07]. L'ontologie globale doit permettre aux pairs de communiquer sans perdre d'information. Néanmoins, les auteurs ne discutent pas du cas où les ontologies peuvent évoluer. Dans un tel contexte, il faudrait que l'ontologie globale soit mise à jour régulièrement. Par ailleurs, il se peut que des pairs (ou des communautés de pairs) aient des points de vue très divergents sur certaines connaissances. Dans ce cas, le fait de vouloir créer une ontologie unique à partir d'ontologies très différentes peut être difficile et générer des incohérences ou des contradictions au sein de l'ontologie globale.

Nedos *et al.* considèrent des systèmes mobiles ad-hoc (MANET) dans lesquels chaque pair utilise une ontologie pour représenter les services qu'il propose [NSCC07]. Les auteurs proposent un protocole épidémique (proche de ceux présentés en section 2.3, page 40) dans lequel les pairs s'envoient des concepts sélectionnés aléatoirement depuis leurs ontologies. Lorsqu'un pair reçoit un message (c.-à-d. un ensemble de concepts d'une ontologie o), il tente d'identifier des correspondances entre ces concepts et ceux de son ontologie o'. De cette manière il identifie de nouvelles correspondances entre o et o'. L'idée de ce protocole est de faciliter l'émergence d'une ontologie partagée. Ce protocole peut avoir l'inconvénient de générer un trafic réseau important pour identifier des correspondances, en particulier lorsque les ontologies sont très volumineuses (c.-à-d. lorsqu'elles contiennent de nombreux concepts).

Pires *et al.* proposent une méthode d'alignement d'ontologies dans le cadre des PDMS. Dans ce travail, les auteurs considèrent des ontologies de domaine comme base [PSPS09]. L'idée de la méthode proposée est d'identifier des correspondances de type *closeness* ou *disjointness* à partir de correspondances d'équivalences. Pour cela elle s'appuie sur une ontologie de domaine qui sert de pivot. Les deux ontologies à aligner sont d'abord alignées à l'ontologie de domaine avec des relation d'équivalence. Les relations existantes au sein de l'ontologie de domaine sont alors utilisées pour identifier des correspondances "moins précises" entre les ontologies. Par exemple si $c \equiv c'_1$, $c'_1 \sqsubseteq c'_2$ et $c'_2 \equiv c''$ alors la correspondance $c \sqsubseteq c''$ est déduite ($c \in o$, $c'_1, c'_2 \in o'$ et $c'' \in o''$).

4.3.2 Méthodes de réorganisation

Des méthodes ont été proposées pour tenter d'améliorer l'interopérabilité dans les systèmes P2P hétérogènes. Il s'agit principalement de méthodes facilitant le routage de requêtes. Dans [NWS+04], Nejdl, Löser, Sibersky *et al.* suggèrent qu'il est intéressant de considérer des systèmes hiérarchiques pour faciliter le routage de requêtes. Les pairs sont alors organisés en fonction de l'ontologie qu'ils utilisent. Les pairs utilisant la même ontologie sont connectés en étoile à un super-pair (ils ne sont pas connectés entre eux). Cette architecture peut être problématique dans le cas où le système est très peu hétérogène : si seulement deux ontologies sont utilisées, le système est alors constitué de deux super-pairs servant de

serveur à la moitié de système. On retombe alors sur les problématiques rencontrées dans les architectures clients/serveurs (passage à l'échelle, tolérance aux pannes).

Dans [LNS+03], les mêmes auteurs considèrent des systèmes P2P hiérarchiques dans lesquels les schémas de bases de données relationnelles peuvent être hétérogènes. Ils proposent une approche décentralisée pour associer les pairs à des super-pairs. L'objectif est de limiter les coûts engendrés par la propagation de requêtes dans le système. Il s'agit clairement de considérations liées à l'activité des pairs (c.-à-d. à l'interopérabilité), mais dans une certaine mesure, cela permet de réduire l'hétérogénéité car les pairs utilisant des ontologies similaires sont regroupés dans le système au sein de *semantic overlays clusters*. Les super-pairs sont choisis en fonction de critères prédéfinis fournis par des experts humains. Ces critères sont exprimés sous forme de règles décrivant quels pairs sont autorisés à rejoindre un cluster.

Penzo *et al.* proposent une méthode permettant la création et la maintenance d'un réseau recouvrant sémantique (*Semantic Overlay Network*, SON) pour limiter le trafic réseau dans les systèmes P2P hétérogènes. La méthode proposée est itérative et permet de regrouper les pairs en fonction de leurs centres d'intérêt. Dans ce travail les centres d'intérêt d'un pair sont définis par un ensemble de concepts issus de son ontologie. Lorsqu'un pair rejoint le système, il détermine d'abord les clusters dans lesquels il doit s'intégrer, puis il choisit les pairs dont il est le plus proche sémantiquement au sein du cluster.

4.4 Bilan

Dans ce chapitre nous avons vu qu'il existe un certain nombre de systèmes de RI distribués, homogènes ou hétérogènes. Les travaux qui se rapprochent le plus de notre contexte sont ceux présentés dans les section 4.2 et 4.3. Les critiques que nous pouvons émettre sur ces solutions sont les suivantes. Premièrement, certaines considèrent des données centralisées (par ex. le catalogue centralisé utilisé dans Piazza). Cela limite grandement le passage à l'échelle et la tolérance aux pannes. Nous pensons donc qu'il faut éviter ce type de centralisation. Deuxièmement, certains systèmes considèrent qu'une ontologie globale est partagée par l'ensemble des pairs du système pour interopérer avec les ontologies locales. Nous pensons que ce type d'approche n'est pas adapté au cadre hétérogène car il semble difficile de faire émerger une ontologie sur laquelle tous les pairs sont en accord. Nous pensons que cette proposition limite l'autonomie des pairs dans le choix de leurs ontologies. Troisièmement, certaines méthodes, telles que [LNS+03] requièrent l'intervention d'experts humains. Nous pensons que cela ne permet pas de considérer des ontologies volumineuses, qui peuvent éventuellement évoluer dans le temps. Quatrièmement, la plupart des méthodes ont tendance à mélanger les différentes notions clés : l'hétérogénéité sémantique et l'interopérabilité. C'est le cas des méthodes proposées dans la section 4.3. Nous pensons que ces deux notions méritent d'être étudiées séparément même si elles sont liées : la séparation des préoccupations permet d'adresser les différentes problématiques de manière indépendante.

L'étude des travaux existants faite dans ce chapitre doit nous guider dans les choix à faire, et dans les contraintes à respecter dans nos contributions. En clair, nous voulons adresser le problème d'hétérogénéité sémantique des systèmes P2P pour garantir l'interopérabilité. Plus précisément, nous souhaitons permettre aux participants d'un système hétérogène de rechercher des documents mis à disposition par d'autres participants. Pour cela, nous voulons proposer une ou plusieurs solutions complètement distribuées (c.-à-d. sans aucune centralisation), ne considérant pas d'ontologie globale partagée, et ne nécessitant pas l'intervention d'acteurs humains.

Troisième partie

Contributions

Mesures d'hétérogénéité sémantique

5

Dans ce chapitre, nous nous intéressons à la notion d'hétérogénéité sémantique des systèmes P2P. Nous commençons par expliquer pour quelles raisons il nous semble primordial de pouvoir la caractériser. Ensuite nous montrons que l'hétérogénéité sémantique est une notion complexe qui présente différentes facettes. Nous montrons en quoi ces facettes nous semblent pertinentes dans ce contexte. Nous définissons ensuite un ensemble de mesures d'hétérogénéité sémantique. Chacune d'entre elles permet de caractériser l'hétérogénéité suivant l'une de ses facettes. Comme certaines de ces mesures considèrent la notion de disparité sémantique entre pairs, nous finissons par présenter différentes nouvelles mesures. Les travaux décrits dans ce chapitre ont été présentés aux conférences EGC et WI-IAT en 2011 [CCL11b, CCL11c].

5.1 Problématique et objectifs

Dans ce travail nous considérons des systèmes P2P non-structurés dans lesquels chaque pair utilise une ontologie pour représenter ses données (documents et requêtes). Comme nous l'avons vu dans la section 1.3 (page 19), il est peu probable que tous les pairs s'accordent sur l'utilisation d'une unique ontologie, car ils ont différents objectifs, différents points de vue, etc. Cette situation génère ce qu'on appelle de l'hétérogénéité sémantique. Comme nous l'avons vu dans le chapitre 4 (page 57), de nombreux travaux ont été menés dans des contextes similaires. Néanmoins très peu ont tenté de dissocier les deux notions clées que sont l'hétérogénéité sémantique et l'interopérabilité. En particulier très peu d'efforts ont été portés sur la source du problème : l'hétérogénéité. Il nous semble primordial de comprendre et d'être capable de caractériser cette notion pour pouvoir proposer des solutions permettant en définitive d'améliorer l'interopérabilité. La problématique que nous abordons dans ce chapitre est la suivante : *Comment caractériser l'hétérogénéité sémantique d'un système P2P non-structuré ?*

Cudré-Mauroux et Aberer ont défini un ensemble de conditions nécessaires à l'interopérabilité [CMA04]. En réalité il s'agit de critères permettant de déterminer s'il est possible pour les pairs d'échanger des requêtes dans le système. Ce travail propose une analyse intéressante. Néanmoins il ne semble pas suffisant pour caractériser l'hétérogénéité d'un système. Nous pensons que pour caractériser l'hétérogénéité il faut définir une mesure. Cette mesure doit permettre de comparer les performances (en terme d'interopérabilité) de deux méthodes dans une situation d'hétérogénéité particulière. Par exemple la figure 5.1 présente le cas où deux méthodes (m_1 et m_2) sont comparées dans un cas particulier d'hétérogénéité. La mesure d'hétérogénéité doit également faciliter l'évaluation d'une méthode particulière dans différentes situations d'hétérogénéité. La figure 5.2 présente les performances de la méthode m_1 dans différentes situations d'hétérogénéité. L'utilisation d'une mesure permet de tirer des conclusions du type : les performances de m_1 sont dégradées lorsque l'hétérogénéité augmente et elles deviennent "chaotiques" lorsque l'hétérogénéité est supérieure à $0,7$. Enfin la mesure peut être utilisée pour mesurer à quel point un algorithme (ou plusieurs algorithmes) permet de réduire l'hétérogénéité. La figure 5.3 présente les valeurs d'hétérogénéité mesurées après exécution de différents algorithmes (la référence correspond au cas initial, c.-à-d. lorsqu'aucun algorithme n'est appliqué). Il est ainsi possible de comparer leurs capacités à réduire l'hétérogénéité.

L'hétérogénéité est une notion relativement complexe. Nous pensons qu'il est intéressant de proposer différentes mesures permettant de caractériser l'hétérogénéité selon différentes perspectives. L'objectif est donc de proposer de telles mesures. Chaque mesure doit permettre de caractériser l'hétérogénéité avec une perspective particulière. Un système peut être très hétérogène suivant une facette mais très peu suivant une autre. Nous pensons qu'il n'est pas souhaitable d'agréger les différentes mesures pour n'obtenir qu'une valeur numérique car dans ce cas l'hétérogénéité est seulement considérée de manière globale et il n'est plus possible de faire la distinction entre ses différentes facettes.

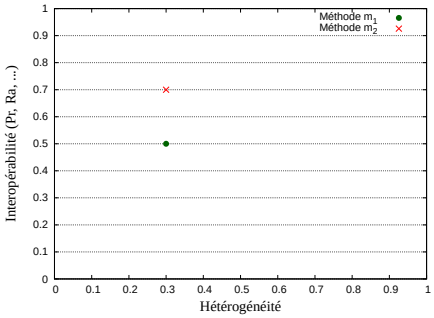


FIGURE 5.1 – *Exemple de comparaison de deux méthodes de RI dans une situation d'hétérogénéité précise.*

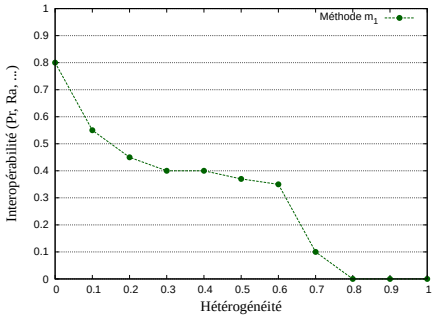


FIGURE 5.2 – *Exemple de l'évaluation d'une méthode de RI dans différentes situations d'hétérogénéité.*

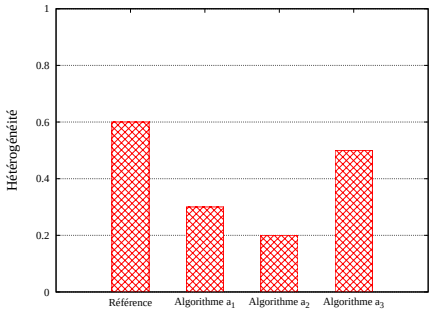


FIGURE 5.3 – *Exemple de comparaison de différents algorithmes dont l'objectif est de réduire l'hétérogénéité sémantique.*

Dans la suite de ce chapitre, nous commençons par définir le modèle que nous considérons : système P2P, ontologies, alignements, etc. (section 5.2). Nous présentons ensuite des intuitions qui justifient le fait de définir plusieurs mesures d'hétérogénéité (section 5.3), et nous proposons une typologie des mesures. La section 5.4 présente différentes mesures d'hétérogénéité sémantique. Au moins une mesure est proposée pour chaque classe de la typologie. Certaines d'entre elles considèrent les disparités entre pairs, et/ou la topologie des systèmes. Nous proposons un ensemble de mesures de disparité sémantique entre deux pairs dans la section 5.5. Nous concluons dans la section 5.6.

5.2 Modèle de système P2P sémantique

Système P2P

Définition 5.1 Un système P2P non-structuré $s \in \mathcal{S}$ est un couple $\langle \mathcal{P}, \mathcal{N} \rangle$ tel que $\mathcal{P}$ est un ensemble de pairs, et $\mathcal{N} \subseteq \mathcal{P} \times \mathcal{P}$ est une relation binaire : $(p, p') \in \mathcal{N}$ si le pair p stocke l'adresse de p' dans sa table de routage.

Définition 5.2 Le voisinage d'un pair p, aussi appelé vue locale d'un pair, correspond à l'ensemble des pairs contenus dans sa table de routage. Il est défini par : $\mathcal{N}_p = \{p' \in \mathcal{P} : (p, p') \in \mathcal{N}\}$.

Définition 5.3 Le voisinage d'un pair p dans un rayon $r \in \mathbb{N}$ correspond à l'ensemble des pairs que p peut atteindre avec au plus r sauts. Il est noté $\mathcal{N}_p^r$. Par

définition p n'appartient pas à $\mathcal{N}_p^r$. On remarque que $\mathcal{N}_p^0 = \emptyset$, $\mathcal{N}_p^1 = \mathcal{N}_p$ et $\forall r \in \mathbb{N}^+, \mathcal{N}_p^{r-1} \subseteq \mathcal{N}_p^r$.

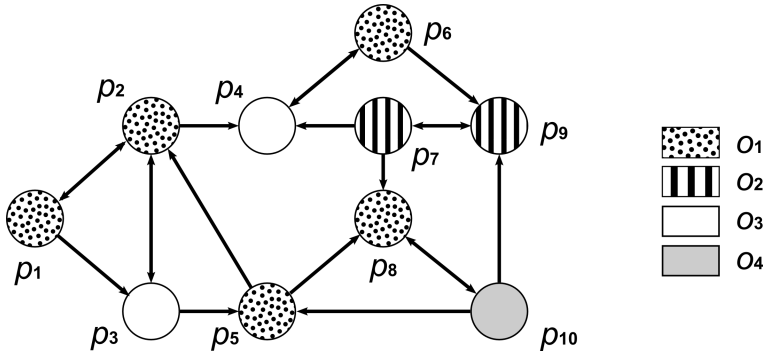


FIGURE 5.4 – *Exemple d'un système P2P non-structuré dans lequel chaque pair utilise une ontologie.*

Exemple 5.1 Sur le système P2P présenté sur la figure 5.4, le voisinage du pair p_5 est : $\mathcal{N}_{p_5} = \{p_2, p_8\}$. Le voisinage de p_5 dans un rayon de 2 est : $\mathcal{N}_{p_5}^2 = \{p_1, p_2, p_3, p_4, p_8, p_{10}\}$.

5.2.1 Ontologies et correspondances

Définition 5.4 Une ontologie $o \in \mathcal{O}$ est constituée de concepts, de relations (subsomption, méronymie, etc.) et de propriétés. Elle ne contient pas d'individus. À chaque concept peuvent être attribuées des propriétés, et les concepts sont reliés via des relations. L'ensemble des concepts, des relations et des propriétés de l'ontologie o sont respectivement notés C_o, R_o et P_o, et l'union de ces trois ensembles d'entités est noté E_o.

Définition 5.5 Une correspondance $c \in \mathcal{C}$ est un 5-uplet $\langle id, e, e', r, n \rangle$ tel que id est un identifiant, e et e' sont des entités de deux ontologies o et o', r est une relation entre e et e', et n est une valeur de confiance (comprise entre 0 et 1) attribuée à la correspondance. La relation r peut être une équivalence ($\equiv$), une spécialisation ($\sqsubseteq$), une généralisation ($\sqsupseteq$), etc.

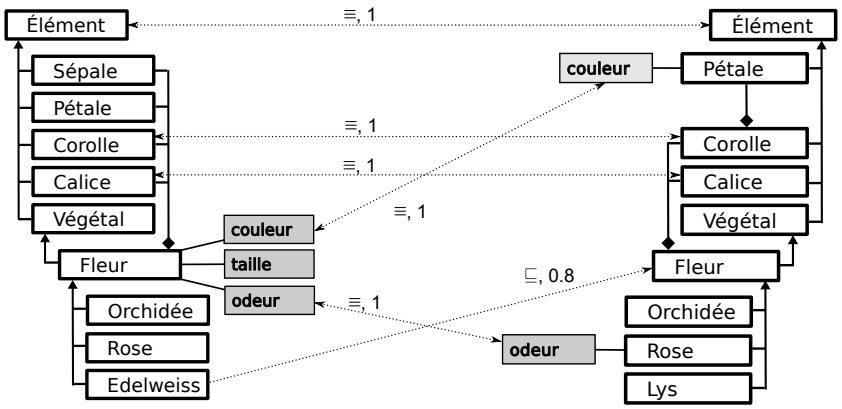


FIGURE 5.5 – *Exemple de deux ontologies (o_1 et o_2) et d'un ensemble de correspondances.*

Exemple 5.2 La figure 5.5 présente deux ontologies (o_1 et o_2) et un ensemble de correspondances entre ces deux ontologies (les identifiants des correspondances n'apparaissent pas sur la figure). Ces dernières définissent que : le concept $Element_1$ est équivalent à $Element_2$ avec une valeur de confiance maximale, le concept $Edelweiss_1$ est une spécialisation de $Fleur_2$ avec une valeur de confiance égale à $0,8$, la propriété $odeur_1$ est équivalente à $odeur_2$ avec une valeur de confiance maximale, etc.

5.2.2 Fonctions d'appariement

Nous considérons que chaque pair d'un système P2P utilise une (et une seule) ontologie pour représenter ses données. Pour définir la relation entre un pair et l'ontologie qu'il utilise nous introduisons la notion de fonction d'appariement pair-ontologie.

Définition 5.6 Une fonction d'appariement pair-ontologie est une fonction $\mu : \mathcal{P} \to \mathcal{O}$ qui associe une ontologie à chaque pair.

De même, nous considérons que chaque pair connaît un ensemble de correspondances. Sans faire d'hypothèse sur la manière dont elles sont obtenues (méthode d'alignement automatique, identification par des experts, etc.), nous proposons de considérer une fonction d'appariement pair-correspondances.

Définition 5.7 Une fonction d'appariement pair-correspondances est une fonction $\kappa : \mathcal{P} \to 2^{\mathcal{C}}$ qui associe un ensemble de correspondances à chaque pair.

Exemple 5.3 Le pair p_5 de la figure 5.4 utilise l'ontologie o_1 de la figure 5.5. On a donc : $\mu(p_5) = o_1$. Si p_5 connaît l'ensemble de correspondances présentées sur la figure 5.5 alors on a :
$\kappa(p_5) = \{\langle Element_1, Element_2, \equiv, 1\rangle, \langle Edelweiss_1, Fleur_2, \sqsubseteq, 0.8\rangle, ...\}$.

5.2.3 Mesure de disparité entre pairs

L'hétérogénéité sémantique provient du fait que les pairs n'utilisent pas tous la même ontologie pour représenter leurs données. Les pairs sont plus ou moins disparates les uns des autres selon qu'ils utilisent des ontologies relativement similaires ou complètement différentes, qu'ils connaissent peu ou beaucoup de correspondances, etc. Nous introduisons la notion de mesure de disparité entre pairs.

Définition 5.8 Une mesure de disparité entre pairs $d : \mathcal{P} \times \mathcal{P} \to [0,1]$ permet de déterminer le degré de disparité entre deux pairs au niveau sémantique. Toute mesure de disparité respecte les propriétés suivantes :
- minimalité : $\forall p \in \mathcal{P},\ d(p,p) = 0$;
- positivité : $\forall p, p' \in \mathcal{P},\ d(p,p') \geqslant 0$.

Une mesure de disparité n'est pas nécessairement symétrique. On peut avoir : $d(p,p') \neq d(p',p)$. Il ne s'agit donc pas nécessairement d'une distance au sens mathématique.

Lorsque nous définissons les mesures d'hétérogénéité sémantique (section 5.4), nous considérons une mesure de disparité générique. Nous ne

précisons pas la manière dont celle-ci est instanciée car elle est indépendante de la manière dont nous abordons le problème d'hétérogénéité sémantique d'un système P2P. Nous proposons des mesures de disparité entre pairs dans la section 5.5.

5.2.4 Modèle de système P2P sémantique

Étant donnés tous ces éléments (ontologies, correspondances, fonctions d'appariement, etc.), nous définissons ce qu'est un modèle de système P2P sémantique.

Définition 5.9 Soient $s = \langle \mathcal{P}, \mathcal{N} \rangle$ un système P2P non-structuré, $\mathcal{O}$ un ensemble d'ontologies, μ une fonction d'appariement pair-ontologie, $\mathcal{D}$ un ensemble de fonctions de disparité entre pairs. Un modèle $m \in \mathcal{M}$ est défini par $m = \langle s, \mathcal{O}, \mu, \mathcal{D} \rangle$.

La fonction d'appariement pair-correspondances κ est éventuellement prise en compte dans une des mesures de disparité entre pairs de l'ensemble $\mathcal{D}$.

5.2.5 Mesure d'hétérogénéité sémantique

Définition 5.10 Une mesure d'hétérogénéité sémantique est une fonction $\mathcal{H} : \mathcal{M} \rightarrow [0, 1]$ telle que pour tout modèle m de l'espace des modèles $\mathcal{M}$, elle vérifie les propriétés suivantes :

1. $\mathcal{H}(m) = 0$ si $\forall p, p' \in \mathcal{P},\ \mu(p) = \mu(p')$ (minimalité) ;
2. $\mathcal{H}(m) = 1$ si $\forall p \neq p' \in \mathcal{P},\ \forall d \in \mathcal{D},\ d(p, p') = 1$ (maximalité).

Ces conditions signifient que (1.) l'homogénéité est atteinte lorsque tous les pairs utilisent la même ontologie, et que (2.) l'hétérogénéité est maximale lorsque les disparités entre pairs sont maximales, quelque soit la mesure de disparité considérée.

5.3 Typologie de mesures d'hétérogénéité sémantique

5.3.1 Intuitions

Dans cette section nous présentons trois scénarios qui visent à mettre en évidence les différentes facettes de l'hétérogénéité.

Scénario 1 Considérons un système P2P dans lequel les pairs utilisent différentes ontologies. Si les données sont distribuées aléatoirement dans le système, chaque pair peut potentiellement répondre aux requêtes. Dans cette situation, la capacité d'un pair d'être compris quand il émet une requête dépend du nombre d'ontologies utilisées dans le système, et du nombre de pairs utilisant la même ontologie que lui. Si seulement deux ontologies sont utilisées, un pair a plus de chance d'être compris que si dix ontologies sont utilisées. Ce scénario montre qu'il est important de considérer le nombre d'ontologies utilisées dans le système, et le nombre de pairs y participant.

Scénario 2 Considérons un système dans lequel les pairs utilisent seulement deux ontologies : soit o_1, soit o_2. Dans ce cas l'hétérogénéité dépend de la disparité entre o_1 et o_2. Plus généralement elle dépend de la disparité entre les pairs qui utilisent des ontologies différentes. Si les ontologies o_1 et o_2 sont très différentes alors le système risque d'être très hétérogène car les pairs auront beaucoup de difficulté à communiquer. Nous pensons donc qu'il est important de considérer la disparité entre les pairs pour capturer l'hétérogénéité globale du système.

Scénario 3 Nous considérons le même cas que dans les scénarios précédents : un système P2P dans lequel les pairs utilisent différentes ontologies. Cette fois nous considérons le fait qu'en général, dans les systèmes P2P non-structurés, les requêtes ne sont pas transmises à tous les pairs du système. Seulement un sous-ensemble des pairs du système est amené à traiter une requête lorsqu'elle est émise (cf. section 2.2.2.1, page 35). Le routage est généralement effectué en fonction des liens entre les pairs. Il est donc important que les pairs entourant un émetteur de requêtes soient en mesure de comprendre ses requêtes afin de pouvoir les traiter. Par conséquent, nous pensons qu'il est important de considérer la topologie du système (c.-à-d. l'organisation des pairs) pour capturer l'hétérogénéité.

Ces trois scénarios mettent en évidence plusieurs facettes complémentaires de l'hétérogénéité que nous devons prendre en compte dans notre analyse. Selon nous au moins trois aspects doivent être étudiés :

- le nombre et la distribution des ontologies utilisées dans le système,
- la disparité entre les pairs,
- la topologie du système.

Nous pensons qu'il n'est pas possible de caractériser l'hétérogénéité avec une seule mesure. C'est pour cette raison que nous proposons différentes mesures d'hétérogénéité. Chacune se focalise sur une facette particulière.

5.3.2 Typologie des mesures d'hétérogénéité sémantique

Étant donné le modèle considéré et les différentes facettes de l'hétérogénéité, nous proposons une typologie des mesures. L'idée est de les distinguer en fonction des éléments du modèle qu'elles considèrent dans leurs définitions. Nous pensons que chaque mesure doit au moins considérer $\mathcal{P}$, $\mathcal{O}$ et μ qui sont les composants de base du modèle. Nous différencions alors les mesures d'hétérogénéité selon deux critères : prise en compte la topologie du système ($\mathcal{N}$), prise en compte de la disparité entre les pairs ($\mathcal{D}$). Ces deux critères pouvant être combinés, nous obtenons quatre classes de mesures. Pour chacune d'entre elles, le tableau 5.1 énumère les éléments du modèle qui sont utilisés dans leurs définitions.

	Sans topologie	Avec topologie
Sans disparité	$\mathcal{P}$, $\mathcal{O}$, μ	$\mathcal{P}$, $\mathcal{O}$, μ, $\mathcal{N}$
Avec disparité	$\mathcal{P}$, $\mathcal{O}$, μ, $\mathcal{D}$	$\mathcal{P}$, $\mathcal{O}$, μ, $\mathcal{N}$, $\mathcal{D}$

TABLE 5.1 – *Quatres classes de mesures d'hétérogénéité.*

5.4 Mesures d'hétérogénéité sémantique

Dans cette section, nous proposons des mesures d'hétérogénéité sémantique suffisamment générales pour être utilisées dans différents domaines d'application, tout en restant informatives. Nous les présentons en fonction de la typologie définie précédemment.

5.4.1 Mesures ne tenant pas compte de la topologie

Mesures ne tenant pas compte de la disparité

La notion de diversité est communément utilisée pour mesurer l'hétérogénéité d'une population (par exemple en biologie). Dans notre contexte, la diversité dépend du nombre d'ontologies différentes utilisées dans le système. Si tous les pairs utilisent la même ontologie, le système est homogène. À l'inverse, plus il y a d'ontologies différentes, plus le système est hétérogène. Nous proposons la mesure suivante :

$$\mathcal{H}_{Div}(m) = \frac{|o_s| - 1}{|\mathcal{P}| - 1} \tag{5.1}$$

où o_s est l'ensemble des ontologies utilisées dans le système s, et $\mathcal{P}$ est l'ensemble des pairs de s.

Exemple 5.4 Dans le système de la figure 5.4 (page 73), quatre ontologies sont utilisées par les dix pairs :

$$\mathcal{H}_{Div}(m) = \frac{4-1}{10-1} = 0,33$$

Mesurer la diversité permet de tirer des conclusions préliminaires. En particulier, cela donne des informations sur la nécessité de disposer d'alignements pour assurer l'interopérabilité. Si la diversité est nulle, aucun alignement n'est nécessaire pour communiquer. Par contre, si la diversité est maximale (c.-à-d. $\mathcal{H}_{Div}$ égale à 1), cela signifie que des alignements doivent être lancés pour permettre aux pairs de communiquer.

La mesure de diversité ne donne aucune indication sur la manière dont les ontologies sont distribuées. Idéalement la distribution devrait être uniforme (ou équitable) : chaque ontologie devrait être utilisée par $\frac{|\mathcal{P}|}{|o_s|}$ pairs. De telles mesures ont été proposées pour mesurer la biodiversité d'un environnement : mesure de Shannon [Sha48], mesure de Simpson [Sim49], etc. Par exemple nous pouvons adapter la mesure de Simpson (appelée index de diversité et notée D dans la littérature) :

$$D(m) = \frac{1}{|\mathcal{P}|^2} \sum_{o \in o_s} |\{p \in \mathcal{P} : \mu(p) = o\}|^2 \tag{5.2}$$

où o_s est l'ensemble des ontologies utilisée dans le système s. D est maximale (égale à 1) si $|o_s| = 1$, c'est-à-dire si tous les pairs utilisent la même ontologie. D est minimal (égale à $\frac{1}{|\mathcal{P}|}$) si $|o_s| = |\mathcal{P}|$, c'est-à-dire si tous les pairs utilisent des ontologies différentes. Afin de vérifier les propriétés de minimalité et de maximalité énoncées dans la définition 5.10 (page 75), nous définissons la mesure d'équité $\mathcal{H}_{Equi}$ par :

$$\mathcal{H}_{Equi}(m) = \frac{|\mathcal{P}| \cdot (1 - D(m))}{|\mathcal{P}| - 1} \tag{5.3}$$

Exemple 5.5 Sur le système présenté à la figure 5.4 (page 73), l'index de Simpson est :

$$D(m) = \frac{5^2 + 2^2 + 2^2 + 1^2}{10^2} = 0,34$$

On trouve que :

$$\mathcal{H}_{Equi}(m) = \frac{10 \cdot (1 - 0,34)}{10 - 1} \simeq 0,73$$

Si la mesure $\mathcal{H}_{Equi}$ donne une valeur proche de 0, cela signifie que tous les pairs ont potentiellement le même nombre d'interlocuteurs dans le système. Si la valeur obtenue est proche de 1, cela signifie que certains pairs ne partagent leur ontologie avec aucun autre pair, alors que d'autres la partage avec de nombreux autres pairs.

Les mesures de diversité et d'équité ($\mathcal{H}_{Div}$ et $\mathcal{H}_{Equi}$) sont complémentaires car elles capturent des aspects différents de l'hétérogénéité. En effet un système peut être diversifié et équitable, ou peu diversifié et pas équitable, etc.

Mesure tenant compte de la disparité

Plutôt que de ne considérer que les ontologies utilisées par les pairs, nous proposons ici de prendre en compte l'importance des différences (ou disparités) entre ces pairs. Si la disparité entre les pairs est globalement élevée, alors les pairs ont d'importantes différences au niveau de leur connaissances. Plus leurs connaissances diffèrent, plus il est difficile de communiquer (c.-à-d. répondre aux requêtes). En effet une perte importante d'information peut se produire pendant la traduction des requêtes. Dans un premier temps nous considérons la disparité entre chaque couple de pairs :

$$\mathcal{H}_{Disp}(m) = \frac{1}{|\mathcal{P}|^2 - |\mathcal{P}|} \sum_{p_i \neq p_j \in \mathcal{P}} d(p_i, p_j) \tag{5.4}$$

La fonction d est une fonction de disparité entre pairs. La mesure $\mathcal{H}_{Disp}$ détermine si les pairs du système sont globalement disparates les uns des autres. Si la disparité entre les pairs est toujours nulle, alors nous pouvons dire que le système est homogène. De manière générale, plus ils sont disparates, plus le système est hétérogène.

Exemple 5.6 Dans le système de la figure 5.4 (page 73) nous avons :

$$\mathcal{H}_{Disp}(m) = \frac{d(p_1, p_2) + d(p_1, p_3) + \ldots + d(p_{10}, p_9)}{10^2 - 10}$$

5.4.2 Mesures tenant compte de la topologie

Dans les systèmes P2P hétérogènes, il est important de considérer le voisinage des pairs car il représente l'ensemble des pairs avec qui il est susceptible d'échanger des informations. Pour cela nous utilisons la notion de voisinage des pairs (cf. définition 5.3, page 72).

Mesure ne tenant pas compte de la disparité

Dans un premier temps, nous étudions le voisinage de chaque pair en comptabilisant le nombre de pairs qui ne peuvent pas le comprendre directement, c.-à-d. ceux qui n'utilisent pas la même ontologie :

$$\mathcal{H}^r_{DivAP}(m,p) = \frac{|\{p_i \in \mathcal{N}^r_p : \mu(p_i) \neq \mu(p)\}|}{|\mathcal{N}^r_p|} \tag{5.5}$$

Cette mesure permet de caractériser le voisinage d'un pair simplement. Elle peut d'ailleurs être calculée par le pair p lui-même car elle nécessite seulement de savoir quelles ontologies sont utilisées par ses voisins.

Exemple 5.7 Dans le voisinage du pair p_1 à un rayon de 2, deux pairs utilisent des ontologies différentes de celle de p_1 (cf. figure 5.4, page 73). Nous obtenons donc :

$$\mathcal{H}^2_{DivAP}(m,p_1) = \frac{2}{4} = 0,5$$

Cette mesure n'est pas une mesure d'hétérogénéité au sens de la définition 5.10 (page 75) car elle est définie autour d'un pair particulier. Une mesure globale du système peut-être obtenue de la manière suivante :

$$\mathcal{H}^r_{DivAPMoy}(m) = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} \mathcal{H}^r_{DivAP}(m,p) \tag{5.6}$$

Il s'agit simplement d'une moyenne de $\mathcal{H}_{DivAP}$ sur chacun des pairs. Si la valeur de la mesure $\mathcal{H}^r_{DivAPMoy}$ est proche de 1, cela signifie qu'il est nécessaire que les pairs aient des alignements pour interopérer avec leurs voisins.

Mesure tenant compte de la disparité

Le fait que deux pairs n'utilisent pas la même ontologie ne signifie pas qu'ils ne peuvent pas communiquer ensemble (au moins partiellement). Nous affinons la mesure précédente en considérant la disparité entre pairs, et nous étudions à quel point un pair est incompris par ses voisins :

$$\mathcal{H}^r_{DispAP}(m,p) = \frac{1}{|\mathcal{N}^r_p|} \sum_{p_i \in \mathcal{N}^r_p} d(p,p_i) \tag{5.7}$$

Exemple 5.8 Dans le voisinage du pair p_1 à un rayon de 2, les pairs p_2 et p_5 utilisent la même ontologie que p_1 (cf. figure 5.4, page 73). L'hétérogénéité centrée sur p_1 est donc égale à :

$$\mathcal{H}^2_{DispAP}(m,p_1) = \frac{1}{4} \cdot d(p_1,p_3) + d(p_1,p_4)$$

De la même manière que pour la mesure ne tenant pas compte de la disparité, nous pouvons définir une mesure globale $\mathcal{H}^r_{DispAPMoy}$:

$$\mathcal{H}^r_{DispAPMoy}(m) = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} \mathcal{H}^r_{DispAP}(m,p) \tag{5.8}$$

Si $\mathcal{H}_{DispAPMoy}$ est faible, cela signifie que les pairs sont globalement proches de leurs voisinages : chaque pair est entouré de pairs capables de le comprendre.

Remarque Toutes les mesures présentées dans cette section satisfont les propriétés de minimalité et de maximalité énoncées dans la définition 5.10 (page 75). Nous ne présentons pas les preuves car celles-ci sont évidentes.

5.5 Mesures de disparité sémantique entre pairs

Dans la section précédente, nous avons proposé des mesures d'hétérogénéité sémantique. Certaines d'entre elles considérent une mesure de disparité entre pairs. Nous n'avions jusqu'à présent pas déterminé de quelle manière ce type de mesure peut être défini. C'est l'objet de cette section. Dans la section 1.4.1.2 (page 22) nous avons listé un certain nombre de mesures de distance entre ontologies. Celles-ci peuvent être utilisées pour définir des mesures de disparité entre pairs. Nous proposons dans cette section de nouvelles mesures de disparité sémantique entre pairs. Nous considérons trois classes de mesures : (i) des mesures basées sur les entités des ontologies : soit les concepts, soit les propriétés assignées aux concepts, (ii) des mesures qui considèrent que les pairs utilisent une mesure de similarité intra-ontologie (c.-à-d. une mesure qui détermine la similarité entre deux concepts d'une même ontologie) et (iii) des mesures qui prennent en compte les intérêts des pairs. Les trois sections suivantes sont consacrées à la présentation de ces mesures.

5.5.1 Mesures basées sur les entités des ontologies

Dans les mesures de disparités que nous définissons, nous utilisons une mesure de similarité inter-ontologies.

Définition 5.11 Une mesure de similarité inter-ontologies $\text{Sim}_{o\to o'} : E_o \times E_{o'} \to [0,1]$ est une mesure qui détermine la proximité sémantique entre deux entités e et e' de deux ontologies o et o'.

Il est important de noter qu'une mesure de similarité inter-ontologies n'est pas nécessairement symétrique dans le sens où on peut avoir $\text{Sim}_{o\to o'}(e,e') \neq \text{Sim}_{o'\to o}(e',e)$. Les valeurs de similarités peuvent être calculées comme c'est le cas dans les méthodes d'alignement d'ontologies (cf. section 1.4.1.1, page 20). Une autre solution consiste à considérer que $\text{Sim}_{o\to o'}(e,e') = 1$ s'il existe une correspondance entre e et e', et 0 sinon. Dans ce cas nous utilisons l'ensemble des correspondances connues par un pair : $\kappa(p)$. Nous pouvons aussi considérer que la mesure analyse les labels des entités pour en déterminer la proximité. Par exemple, nous pouvons la définir par : $\text{Sim}_{o\to o'}(e,e') = \mathit{distanceEdition}(l,l')$ où l et l' sont respectivement les labels des entités e et e'. Dans la section 1.5.2 (page 27) nous avons présenté des mesures de distance Δ entre ontologies. Ces dernières sont souvent le résultat d'une agrégation des distances δ entre les entités des deux ontologies. Dans ce cas nous pouvons définir $\text{Sim}_{o\to o'}$ en fonction d'une distance δ : $\text{Sim}_{o\to o'}(e,e') = \delta(e,e')$.

5.5.1.1 *Mesure basée sur les concepts*

Les concepts définis dans une ontologie reflètent le domaine de connaissances modélisé ainsi que le niveau d'expertise (car un même domaine peut être représenté avec plus ou moins de concepts). Pour mesurer la disparité entre deux pairs p et p', nous considérons la similarité de chaque concept c de o avec chacun de ceux de o'. Nous choisissons de ne considérer que la valeur maximale de $\text{SIM}_{o\to o'}(c,\cdot)$. Si celle-ci est égale à 1, le concept ne génère pas de disparité. Dans le cas contraire, la disparité est d'autant plus forte que la similarité est petite. Pour traduire cette intuition, nous introduisons la fonction $dispC$ monotone décroissante définie sur $[0,1]$ telle que $dispC(1) = 0$ et $dispC(0) = 1$. Par exemple, en prenant $dispC(x) = 1 - x$ quel que soit x, nous considérons que la valeur de disparité est le complément à 1 de la valeur de similarité maximale. Nous définissons alors la disparité par :

$$d_{concepts}(p,p') = \frac{1}{|C_o|}\sum_{c\in C_o} dispC(\max_{c'\in C_{o'}} \text{SIM}_{o\to o'}(c,c')) \tag{5.9}$$

où o est l'ontologie de p (c.-à-d. $\mu(p) = o$) et o' est l'ontologie de p'. Intuitivement, un concept $c \in C_o$ ne présente pas de disparité avec o' s'il y possède un équivalent. Si ce n'est pas le cas, la disparité dépend de la similarité maximale entre c et un concept de l'ontologie o'. On remarque que la mesure $d_{concepts}$ vérifie les conditions de minimalité et de positivité (cf. définition 5.8, page 74).

Exemple 5.9 Soient o_1 et o_2 les deux ontologies présentées sur la figure 5.5 (page 73), et p_1 et p_7 deux pairs utilisant respectivement les ontologies o_1 et o_2 (comme c'est le cas sur la figure 5.4, page 73). En considérant que $\text{SIM}_{o\to o'}(e,e')$ est égale à 1 si les labels de e et e' sont similaires et 0 sinon, et que $\forall x \in [0,1], dispC(x) = 1 - x$ on obtient :

$$d_{concepts}(p_1,p_7) = \frac{1}{10}(8\times 0 + 2\times 1) = 0,2$$

En considérant une mesure inter-ontologies syntaxique, nous trouvons que les pairs p_1 et p_7 sont peu disparates.

5.5.1.2 *Mesure basée sur les propriétés des concepts*

Les propriétés attribuées aux concepts permettent de les caractériser. Pour qu'une propriété pr, attribuée à un concept c de l'ontologie o, ne soit pas source de disparité entre deux ontologies o et o', il faut que le concept c ait un équivalent dans o' et que cet équivalent possède une propriété équivalente à pr. Si ce n'est pas le cas, cela signifie que les ontologies ne représentent pas le concept c avec le même point de vue ou avec le même niveau de détail. Notons que la différence entre les ontologies au niveau de la propriété pr n'est pas la même si pr n'est pas représentée dans l'ontologie o' que si elle est présente dans la plupart des concepts hyponymes de l'équivalent de c. Pour capturer ce phénomène, nous proposons de compter le nombre d'hyponymes de c' (équivalent de c) qui possèdent la propriété pr' (équivalente de pr). Nous notons cette valeur $hypoCard(c',pr')$

et nous la normalisons pour assurer que la mesure renvoie une valeur comprise entre 0 et 1 :

$$hypoCard(c', pr') = \frac{|\{c'' \in hypo_{o'}^{c'} \cup \{c'\} : pr' \in c''\}|}{|hypo_{o'}^{c'}| + 1} \tag{5.10}$$

Exemple 5.10 Soit l'ontologie o_2 présentée à la figure 5.5 (page 73). Les concepts hyponymes du concept $Fleur_2$ sont : $Orchidée_2$, $Rose_2$ et Lys_2. Comme seul le concept $Rose_2$ possède la propriété $odeur_2$, on trouve que :

$$hypoCard(Fleur_2, odeur_2) = \frac{1}{3+1} = 0,25$$

Nous introduisons la fonction $dispP$ définie de la manière suivante :

$$dispP_{o'}(c, pr) = \begin{cases} 1 - hypoCard(c', pr') & \text{si } \exists c' \in C_{o'} : \textsc{Sim}_{o \to o'}(c, c') = 1 \wedge \\ & \exists pr' \in P_{o'} : \textsc{Sim}_{o \to o'}(pr, pr') = 1 \\ 1 & \text{sinon} \end{cases} \tag{5.11}$$

Exemple 5.11 En considérant les ontologies o_1 et o_2 de la figure 5.5 (page 73) et que la similarité inter-ontologies est définie comme dans l'exemple 5.9, on trouve que :

- $dispP_{o_2}(Fleur_1, taille_1) = 1$ car la propriété $taille_1$ n'a pas d'équivalent dans l'ontologie o_2 ;
- $dispP_{o_2}(Fleur_1, couleur_1) = 1$ car aucun des hyponymes du concept $Fleur_2$ ne possède la propriété $couleur_2$ (qui est l'équivalent de $couleur_1$) ;
- $dispP_{o_2}(Fleur_1, odeur_1) = 1 - 0,25 = 0,75$ car un seul des hyponymes du concept $Fleur_2$ possède la propriété $odeur_2$ (cf. exemple 5.10).

En notant C_o^{pr}, l'ensemble des concepts de l'ontologie o possédant la propriété pr, nous définissons la disparité entre p et p' (qui utilisent respectivement o et o') par :

$$d_{proprietes}(p, p') = \frac{1}{|P_o|} \sum_{pr \in P_o} \frac{1}{|C_o^{pr}|} \sum_{c \in C_o^{pr}} dispP(c, pr) \tag{5.12}$$

Si la valeur de $d_{proprietes}(p, p')$ est faible, cela signifie que les propriétés définies dans l'ontologie o sont également définies dans l'ontologie o', et qu'elles sont attribuées à des concepts équivalents. On peut en conclure que l'ontologie o' est globalement modélisée avec le même niveau de détails que o, et qu'elle respecte la même perspective. À l'inverse si la valeur de $d_{proprietes}(p, p')$ est forte, alors on peut en conclure que les ontologies utilisées par p et p' sont modélisées avec différents niveaux de détail et différentes perspectives.

5.5.2 Mesures considérant la notion de similarité intra-ontologie

Dans cette section, nous considérons que les pairs utilisent une mesure similarité intra-ontologie :

Définition 5.12 Une mesure de similarité intra-ontologie $sim_o : C_o \times C_o \rightarrow [0,1]$ est une mesure qui détermine la proximité sémantique entre deux concepts d'une même ontologie o. Elle vérifie la propriété de séparation : $sim_o(c,c') = 1 \Leftrightarrow c = c'$. Elle ne vérifie pas nécessairement les propriétés de symétrie (on peut avoir $sim_o(c,c') \neq sim_o(c',c)$) et d'inégalité triangulaire (on peut avoir $sim_o(c,c') + sim_o(c',c'') < sim_o(c,c'')$).

Un certain nombre de mesures de similarité intra-ontologie ont été proposées dans la littérature (cf. section 1.5.1, page 25). Elles peuvent être utilisées pour l'expansion de requêtes [VCLV08], pour la désambiguïsation [Tch12], etc. Nous pensons que la disparité entre deux pairs dépend en partie des différences au niveau de leurs ontologies, mais également des mesures de similarité intra-ontologie qu'ils utilisent. En effet même deux pairs utilisant la même ontologie peuvent avoir des points de vue différents sur le domaine et exprimer ces différences par l'adoption de mesures de similarité différentes. Nous introduisons la notion de contexte sémantique d'un pair :

Définition 5.13 Le contexte sémantique d'un pair ϕ est composé d'une ontologie et d'une mesure de similarité intra-ontologie : $\phi = \langle o, sim_o \rangle$.

Dans la suite de cette section, nous proposons deux nouvelles mesures de disparité entre pairs qui prennent en compte les contextes sémantiques des pairs.

5.5.2.1 *Mesure basée sur le désordre*

Ce qui peut poser dans le fait que deux pairs utilisent des mesures de similarité différentes concerne l'ordre dans lequel sont triés les concepts de l'ontologie par rapport à un concept donné. Nous introduisons la notion de rang d'un concept c_1 par rapport à un concept c :

$$rang_{\phi}^{c_1}(c) = |\{s \in S_{\phi}^{c} : s \geqslant sim_o(c_1, c)\}| \tag{5.13}$$

Étant donné un concept $c \in C_o$, l'ensemble S_{ϕ}^{c} représente l'ensemble des valeurs de similarité entre c et les autres concepts de o. Formellement il est défini par :

$$S_{\phi}^{c} = \{s \in [0,1] : \exists c' \in C_o \text{ tel que } sim_o(c,c') = s\} \tag{5.14}$$

Exemple 5.12 Sur les exemples présentés sur la figure 5.6 (page 84), le concept $Rose_1$ est classé au quatrième rang selon la fonction de similarité sim_{o_1} par rapport au concept $Fleur_1$ (cf. figure 5.6.a) alors que $Fleur_2$ (qui est l'équivalent de $Fleur_1$ dans o_2) est classé au deuxième rang selon sim_{o_2} (cf. figure 5.6.b).

Le fait que le classement soit différent peut avoir de l'importance. En effet, certaines méthodes de recherche d'information utilisent les valeurs de similarité pour étendre les requêtes. Par exemple une requête constituée du concept *Fleur* pourrait être étendue avec le concept *Végétal* dans le contexte ϕ_1 (car il est le concept le plus proche de *Fleur*) alors que dans le contexte ϕ_2, l'expansion porterait sur les concepts *Rose*, *Lys* et *Orchidée*. Dans ce cas, on voit que le désordre engendré par l'utilisation de différentes fonctions de similarité reflète le fait que les perspectives ne

sont pas les mêmes. Pour mesurer cela, nous proposons de mesurer le désordre autour de chaque concept. Pour un concept $c \in C_o$ ayant un équivalent dans o', le désordre $des_{\phi,\phi'}(c)$ est défini par la distance *Rank Distance* [Dino3] où les différences de rangs sont normalisées dans $[0,1]$:

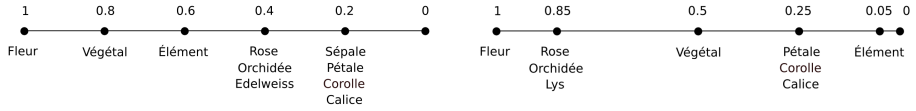
$$des_{\phi,\phi'}(c) = \frac{1}{|C_{o\equiv o'}| - 1} \sum_{c_1 \in C_{o\equiv o'} \setminus \{c\}} \frac{|rang^{c}_{\phi}(c_1) - rang^{c'}_{\phi'}(c'_1)|}{\max(|S^{c}_{\phi}|, |S^{c'}_{\phi'}|) - 1} \tag{5.15}$$

où $C_{o\equiv o'}$ désigne l'ensemble des concepts de o qui ont un équivalent dans o', c' et c'_1 sont respectivement les équivalents de c et c_1 dans o'.

Exemple 5.13 Étant données les ontologies o_1 et o_2 présentées sur la figure 5.5 (page 73) et les valeurs de similarités présentées sur la figure 5.6, le désordre autour du concept $Fleur_1$ est :

$$des_{\phi,\phi'}(Fleur_1) = \frac{1}{8-1} \cdot \frac{1+2+2+2+1+1+1}{5-1} \simeq 0,36$$

si on considère que deux concepts sont équivalents si et seulement leurs labels sont identiques.



(a) (b)

FIGURE 5.6 – *Valeurs de similarités intra-ontologies (a) des concepts de l'ontologie o_1 par rapport au concept* Fleur_1 *et (b) des concepts de l'ontologie o_2 par rapport au* Fleur_2.

En appliquant la fonction $des_{\phi,\phi'}$ sur chacun des concepts de l'ontologie o ayant des équivalents dans o', nous définissons la mesure de disparité suivante :

$$d_{desordre}(p,p') = \frac{1}{|C_{o\equiv o'}|} \sum_{c \in C_{o\equiv o'}} des_{\phi,\phi'}(c) \tag{5.16}$$

où $\phi = \langle o, sim_o \rangle$ (resp. $\phi' = \langle o', sim_{o'} \rangle$) est le contexte sémantique du pair p (resp. du pair p'). Si la valeur de $d_{desordre}$ est faible, cela signifie que les concepts communs aux ontologies o et o' sont rangés de la même façon dans les deux contextes sémantiques. On peut conclure que les perspectives des deux participants sont proches (pour les concepts qu'ils ont en commun). Par contre, si la valeur de $d_{desordre}$ est forte, alors on peut dire que les perspectives des deux participants divergent fortement.

5.5.2.2 *Mesure basée sur la densité autour des concepts*

Un moyen de capturer les disparités liées à l'utilisation de différentes mesures de similarité est d'analyser l'environnement des concepts. L'environnement d'un concept c est composé de tous les concepts qui en sont proches. Formellement tous les concepts dont la similarité par rapport à c est supérieure à un certain seuil τ font partie de son environnement :

$$env^{c}_{\phi,\tau} = \{c_1 \in C_o : sim_o(c, c_1) \geqslant \tau\} \setminus \{c\} \tag{5.17}$$

Pour qu'un concept c de l'ontologie o, ayant un équivalent dans o', ne soit pas source de disparité entre o et o', il faut que son environnement soit "identique" (à l'équivalence près) à l'environnement de son équivalent. La densité est définie par :

$$densite^{\tau}_{\phi,\phi'}(c) = \frac{|\{c_1 \in (env^{c}_{\phi,\tau} \cap C_{o\equiv o'}) : c'_1 \in env^{c'}_{\phi',\tau}\}|}{|env^{c}_{\phi,\tau}|} \tag{5.18}$$

où $C_{o\equiv o'}$ désigne l'ensemble des concepts de o ayant un équivalent dans o', et c' et c'_1 sont les équivalents de c et c_1 dans o'. La densité autour du concept c est maximale lorsque tous les concepts de son environnement ont un équivalent dans l'environnement du concept c' (s'il existe), et elle est minimale lorsqu'aucun n'en a.

Exemple 5.14 En considérant les similarités de la figure 5.6 (page 84) et en posant $\tau = 0,5$, on obtient :

$$env^{Fleur_1}_{\phi_1,\tau} = \{Vegetal_1, Element_1\}$$

$$env^{Fleur_2}_{\phi_2,\tau} = \{Rose_2, Orchidee_2, Lys_2, Vegetal_2\}$$

On a donc :

$$densite^{\tau}_{\phi_1,\phi_2}(Fleur_1) = \frac{1}{3}$$

car seulement un concept de $env^{Fleur_1}_{\phi_1,\tau}$ a un équivalent dans $env^{Fleur_2}_{\phi_2,\tau}$.

Finalement nous définissons la disparité entre deux pairs par :

$$d_{densite}(p, p') = \frac{1}{|C_{o\equiv o'}|} \sum_{c \in C_{o\equiv o'}} 1 - densite^{\tau}_{\phi,\phi'}(c) \tag{5.19}$$

où $C_{o\equiv o'}$ désigne l'ensemble des concepts de o ayant un équivalent dans o'. Lorsque le résultat de cette mesure est faible, cela signifie que les participants ont globalement la même perception du domaine qu'ils ont en commun. La valeur peut être élevée soit parce que les concepts de l'environnement de c dans o n'ont pas d'équivalents (différences de couvertures), soit parce que les concepts de l'environnement de c dans o ne sont pas considérés comme étant proche de c dans o' (différences de perspectives).

5.5.3 Mesures prenant en compte les intérêts des pairs

La mesure de disparité $d_{concept}$ fait implicitement l'hypothèse que tous les concepts de l'ontologie sont utilisés et considérés de la même manière pour le pair. Nous pensons qu'en réalité certains concepts sont plus importants que d'autres pour les pairs. Plusieurs approches peuvent permettre d'identifier de tels concepts : ceux fréquemment utilisés dans les requêtes, ceux utilisés pour décrire les données, etc. Nous pouvons même imaginer que les pairs décrivent explicitement l'ensemble des concepts qu'ils considèrent importants. L'idée de caractériser les pairs par un ensemble de concepts a été exploitée dans [PLM+08]. Dans ces travaux les auteurs considèrent que les pairs peuvent être représentés par un ensemble de concepts reflétant leurs intérêts.

Étant donné un ensemble de concepts $C_{o \triangleright p}$ représentant les concepts importants pour le pair p, nous définissons la disparité de p' par rapport à p par :

$$d_{interets}(p, p') = \frac{1}{|C_{o \triangleright p}|} \sum_{c \in C_{o \triangleright p}} dispC(\max_{c' \in C_{o'}} \text{SIM}_{o \to o'}(c, c')) \tag{5.20}$$

où o et o' sont les ontologies utilisées par les pairs p et p'. Si tous les concepts importants pour le pair p ont des équivalents dans l'ontologie utilisée par p' alors la disparité est nulle. La mesure $d_{interets}$ est identique à la mesure $d_{concepts}$ sauf qu'elle se focalise sur certains concepts : un sous-ensemble des concepts de l'ontologie o. C'est pour cette raison que nous avons les caractéristiques suivantes :

- $d_{concepts}(p, p') = 0 \Rightarrow d_{interets}(p, p') = 0$,
- $d_{concepts}(p, p') = 1 \Rightarrow d_{interets}(p, p') = 1$.

Comme la mesure focalise seulement sur certains concepts, nous pouvons avoir $d_{concepts}(p, p')$ proche de 1 (car p' comprend mal p de manière générale) et $d_{interets}(p, p')$ proche de 0 car p' est capable de traduire les concepts ayant de l'importance pour p. La mesure $d_{interets}$ permet d'avoir une approche plus fine prenant en compte les intérêts des pairs.

5.6 Discussions

La principale contribution de ce chapitre est la définition de plusieurs mesures d'hétérogénéité sémantique de systèmes P2P non-structurés. Ces mesures sont complémentaires et permettent de caractériser finement l'hétérogénéité d'un système suivant différentes facettes. Elle doivent permettre d'évaluer des méthodes particulières (par exemple de RI) dans des systèmes hétérogènes. Elles doivent également permettre de mesurer à quel point un algorithme particulier permet de réduire l'hétérogénéité d'un système. Nous pensons que cette étape de caractérisation de l'hétérogénéité est un premier pas important vers la définition d'algorithmes de réduction de l'hétérogénéité. C'est d'ailleurs à partir de cette analyse que nous avons conçu les algorithmes proposés dans les chapitres suivants (chapitres 6 et 7). Les mesures que nous avons proposées sont évidemment conçues en fonction du modèle que nous avons défini dans la section 5.2 (page 72) mais nous pensons qu'elles sont suffisamment génériques pour être adaptées à d'autres contextes de travail. La définition des mesures d'hétérogénéité ont fait l'objet de publications aux conférences EGC et WI-IAT en 2011 [CCL11b, CCL11c].

Nous insistons sur le fait que les mesures d'hétérogénéité définies dans ce chapitre sont applicables dans un cadre d'évaluation : il faut avoir un regard extérieur et objectif sur le système étudié. De ce point de vue, elles sont similaires aux mesures de précision et de rappel (utilisées pour l'évaluation de méthodes de RI) qui nécessitent également d'avoir un regard extérieur sur le système étudié. Dans le cas de la RI il faut avoir connaissance des résultats attendus, c.-à-d. avoir accès au jugement de pertinence généralement contenu dans un corpus de test. Certaines mesures d'hétérogénéité pourraient être calculées (ou estimées) par les pairs eux-mêmes.

Par exemple, la mesure $\mathcal{H}_{DivAP}$ pourrait être calculée par un pair particulier (cf. équation 5.5, page 79). Pour cela, il lui suffit de contacter les pairs de son voisinage (dans un certain rayon) et de collecter les identifiants des ontologies utilisées. De cette manière le pair détermine à quel point son environnement dans le système est hétérogéne (par rapport à lui-même).

À notre connaissance, il n'existe pas d'autres mesures permettant de déterminer le degré d'hétérogénéité sémantique d'un système P2P. Nous n'avons donc pas de référentiel auquel nous comparer pour évaluer la qualité de nos mesures. D'ailleurs cela pose la question de savoir ce qu'est une bonne mesure. Par exemple comment pouvons nous déterminer si la précision (ou le rappel) est une bonne mesure ? Cela ne semble pas évident et en réalité nous pensons qu'il est suffisant de savoir ce qu'elle mesure précisémment pour pouvoir tirer des conclusions.

Les mesures que nous avons proposées dans ce chapitre devraient faciliter la création de bancs d'essais. Par exemple, il serait intéressant de créer une collection contenant différentes configurations de systèmes P2P dont les caractéristiques varient : le nombre de pairs, la connectivité, la dynamicité, mais également le nombre d'ontologies utilisées dans le système, la distance moyenne entre les ontologies, le nombre de correspondances connues par les pairs, etc. Cette collection devrait permettre d'étudier et de comparer différentes méthodes qui visent à assurer l'interopérabilité ou à réduire l'hétérogénéité, dans des situations similaires.

6 Diminution de l'hétérogénéité liée aux disparités

Dans ce chapitre, nous présentons le protocole CorDis qui vise à réduire l'hétérogénéité sémantique liée aux disparités entre pairs. CorDis est un protocole épidémique qui permet de disséminer les correspondances connues par les pairs dans le système. De cette manière certains pairs apprennent de nouvelles correspondances, qui étaient auparavant connues par d'autres pairs. Nous étudions également le cas où une correspondance apprise par un pair peut être incohérente par rapport à son ontologie. Nous proposons différentes stratégies que les pairs peuvent adopter pour faire face à cette situation. Nous consacrons également une partie importante de ce chapitre à l'évaluation de CorDis. Pour cela nous considérons des ontologies réelles utilisées dans le domaine biomédical. Nous montrons ainsi que notre solution permet de réduire certaines facettes de l'hétérogénéité, qu'il reste efficace lorsque le nombre de pairs ou le nombre d'ontologies augmente, qu'il est adapté aux systèmes dynamiques. Nous étudions également le coût qu'il engendre en termes d'espace de stockage et de trafic réseau. Les travaux décrits dans ce chapitre ont été présentés à la conférence Globe en 2011 [CCL11a], et ont été publiés dans la revue Transactions on Large-Scale Data- and Knowledge-Centered Systems VII 9en 2012 [CCL12c].

6.1 Problématique et objectifs

Dans ce chapitre, nous considérons le modèle défini dans la section 5.2 (page 72). Comme nous l'avons vu dans le chapitre 5, l'hétérogénéité sémantique d'un système P2P dépend en partie des disparités entre les pairs. Nous pensons donc qu'une manière de réduire l'hétérogénéité d'un système consiste à réduire les disparités entre les pairs. La problématique que nous abordons dans ce chapitre est la suivante : *Comment réduire l'hétérogénéité sémantique liée à la disparité entre pairs d'un système P2P ?*

Nous faisons l'hypothèse que les pairs utilisent les correspondances qu'ils connaissent localement pour traduire les requêtes qu'ils reçoivent, et qu'ils les transmettent sans les modifier. Chaque pair est donc responsable de la traduction de la requête initiale. Pour qu'une requête soit traitée sans perte d'information, il faut que le pair qui la reçoit utilise la même ontologie que le pair initiateur de la requête, ou qu'il connaisse suffisamment de correspondances pour la traduire correctement. Dans ce cas, la disparité entre deux pairs dépend de l'ensemble des correspondances connues par le pair. Dans ce contexte, nous pouvons réduire la disparité entre pairs (et plus généralement l'hétérogénéité du système) en augmentant le nombre de correspondances connues entre les ontologies des pairs. Une première approche consiste à faire en sorte que les pairs se contactent et lancent un processus d'alignement entre leurs ontologies. Cela permet en effet d'augmenter le nombre de correspondances connues. Néanmoins cela présente au moins deux inconvénients. Premièrement le processus d'alignement est généralement coûteux en termes de temps de calcul, et en ressources. De plus, comme le processus peut être long, il ne semble pas très adapté aux systèmes P2P qui sont dynamiques par nature. Deuxièmement, comme plusieurs pairs sont susceptibles d'utiliser la même ontologie, il se peut que des correspondances entre deux ontologies aient déjà été calculées. Nous pensons qu'à partir du moment où les pairs connaissent un certain nombre de correspondances, il est préférable de faire en sorte qu'ils les partagent plutôt qu'ils en calculent de nouvelles (qui ont potentiellement déjà été calculées par d'autres pairs). C'est l'idée du protocole CorDis [1]. Dans ce protocole nous faisons deux hypothèses :

- Certains pairs du système possèdent des correspondances : soit elles ont été calculées après que les pairs soient entrés dans le système, soit les pairs les connaissaient en rejoignant le système.
- Les correspondances connues par les pairs sont correctes, c'est-à-dire que toutes les correspondances sont cohérentes avec les ontologies utilisées par les pairs.

Dans la seconde partie de ce chapitre, nous relaxons cette deuxième hypothèse et nous discutons de différentes stratégies permettant de gérer les cas où les correspondances peuvent être incohérentes par rapport aux ontologies des pairs. Chaque stratégie décrit le comportement local et le comportement social que doit adopter chaque pair.

Ce chapitre est organisé comme suit. La section 6.2 décrit le protocole CorDis. Puis la section 6.3 traite de la gestion des incohérences qui peuvent apparaître entre l'ontologie d'un pair et les correspondances qu'il connaît. Nous proposons différentes stratégies pour réagir face à cette si-

1. CorDis : **Cor**respondences **Dis**semination.

tuation et nous redéfinissons des mesures d'hétérogénéité sémantique permettant de prendre en compte la présence d'incohérence. La section 6.4 présente une analyse théorique des coûts engendrés par le protocole en termes d'espace de stockage et de trafic réseau. La section 6.5 présente les expérimentations que nous avons menées pour évaluer le protocole CorDis. Nous concluons dans la section 6.6.

6.2 Le protocole CorDis

L'idée du protocole CorDis est de disséminer des correspondances dans le système afin de partager les correspondances connues par certains pairs mais ignorées par d'autres. Le but est de diminuer l'hétérogénéité du système liée aux disparités entre les pairs.

Lorsque le processus démarre, chaque pair p connaît des correspondances. Il les conserve durant toute sa présence dans le système. Parmi les correspondances connues par un pair, un certain nombre impliquent sa propre ontologie. Nous introduisons la notion d'o_p-correspondances.

Définition 6.1 Étant donné un pair p, les o_p-correspondances sont l'ensemble des correspondances connues par p et impliquant son ontologie :
$\{\langle id, e, e', r, n\rangle \in \kappa(p) : e \in \mu(p) \vee e' \in \mu(p)\}$

Le but de disséminer des correspondances est de permettre à des pairs d'apprendre de nouvelles correspondances éventuellement utiles pour traduire les requêtes qu'ils reçoivent. Dans ce travail, nous disséminons les correspondances à l'aide d'un protocole épidémique (cf. section 2.3, page 40) : chaque pair p initie de manière régulière un échange avec un autre pair p'. Chacun d'eux sélectionne et envoie des correspondances à l'autre. Les pairs doivent être capables d'accomplir trois tâches :

1. la sélection aléatoire d'un pair dans le système,
2. la sélection de correspondances à envoyer,
3. le traitement de correspondances reçues.

Nous considérons que la première tâche est réalisée par un service externe, comme par exemple le service d'échantillonnage de pairs (*peer sampling service*) présenté dans [JGKvS04]. Les algorithmes 3 et 4 présentent les instructions effectuées par le thread actif et le thread passif. Ils sont identiques aux algorithmes des protocoles épidémiques classiques présentés en page 41. Les fonctions de sélection de correspondances (selectCorrespondencesToSend) et de traitement des correspondances (storeCorrespondences) sont décrites en détail dans les sections 6.2.2 et 6.2.3. Avant de les présenter nous montrons comment les correspondances sont stockées localement par les pairs.

6.2.1 Stockage des correspondances

Chaque pair doit stocker les correspondances dont il a connaissance dans un cache de taille limitée afin de ne pas stocker toutes les correspondances. Le choix des correspondances à conserver est fait à l'aide d'une fonction de score qui permet d'ordonner les correspondances : seules les

Algorithme 3 : Thread actif d'un pair p.

```
1 while true do
2     p' ← selectPeer() ;
      // Sélection des correspondances à envoyer :
3     corrs ← selectCorrespondencesToSend() ;
4     sendTo(p', corrs) ;
5     corrs' ← receiveFrom(p') ;
      // Traitement des correspondances reçues :
6     storeCorrespondences(corrs') ;
7     wait(θ) ;              // θ : délai entre deux cycles.
```

Algorithme 4 : Thread passif d'un pair p'

```
1 corrs ← receiveFromAny() ;
  // Sélection des correspondances à envoyer :
2 corrs' ← selectCorrespondencesToSend() ;
3 sendTo(p, corrs') ;
  // Traitement des correspondances reçues :
4 storeCorrespondences(corrs) ;
```

meilleures sont gardées. En théorie, la fonction de score peut être spécifique à chaque pair. Dans ce travail, nous considérons qu'ils utilisent tous un historique des requêtes reçues pour déterminer le score de chaque correspondance.

Un historique de requêtes est composé de deux listes : $\mathcal{L}_1$ et $\mathcal{L}_2$. La liste $\mathcal{L}_1$ contient les entités utilisées dans les k dernières requêtes reçues. La liste $\mathcal{L}_2$ contient quant à elle les ontologies utilisées pour exprimer les k' dernières requêtes. Notons qu'un élément peut apparaître plusieurs fois dans une liste s'il a été utilisé plusieurs fois dans les dernières requêtes.

Exemple 6.1 Après qu'un pair p a successivement reçu les requêtes q_1, q_2, q_3 et q_4 respectivement exprimées avec des entités des ontologies o_1, o_3, o_1 et o_2 telles que :

- q_1 est composée de e_1,
- q_2 est composée de e_3,
- q_3 est composée de e'_1 et e''_1,
- q_4 est composée de e_2,

son historique est composé des listes $\mathcal{L}_1 = [e_1, e_3, e'_1, e''_1, e_2]$
et $\mathcal{L}_2 = [o_1, o_3, o_1, o_1, o_2]$.

Intuitivement, la fonction de score permet aux pairs de favoriser les correspondances pouvant être utiles (localement ou pour d'autres pairs) pour traduire des requêtes. Par ailleurs l'historique de requêtes reflète dans une certaine mesure les intérêts des pairs. Il est donc préférable de favoriser les correspondances fréquemment utilisées.

Définition 6.2 Étant donné un ensemble de correspondances $\mathcal{C}$, nous définissons la fonction de score $sc : \mathcal{C} \to [0, 1]$ par :

$$sc(c) = \omega \cdot [f(e, \mathcal{L}_1) + f(e', \mathcal{L}_1)] + (1 - \omega) \cdot [f(o, \mathcal{L}_2) + f(o', \mathcal{L}_2)] \quad (6.1)$$

où $c = \langle id, e, e', r, n \rangle$ (e et e' sont des entités de o et o'), et f donne la fréquence d'apparition d'un élément dans une liste.

Le coefficient $\omega \in [0, 1]$ est utilisé pour donner plus ou moins d'importance à une correspondance impliquant des entités qui n'apparaissent pas dans des requêtes récentes mais qui appartiennent à des ontologies utilisées récemment. Par défaut il peut être fixé à $0,5$.

Exemple 6.2 En considérant l'historique de requêtes présenté dans l'exemple 6.1, et en posant $\omega = 0,5$, le score de la correspondance $\langle id, e_1, e_2''', r, n \rangle$ (telle que $e_1 \in o_1$ et $e_2''' \in o_2$) est :

$$\begin{aligned} sc(\langle id, e_1, e_2''', r, n \rangle) &= 0,5 \cdot [f(e_1, \mathcal{L}_1) + f(e_2''', \mathcal{L}_1) + f(o_1, \mathcal{L}_2) + f(o_2, \mathcal{L}_2)] \\ &= 0,5 \cdot [0,2 + 0 + 0,6 + 0,2] = 0,5 \end{aligned}$$

Si de nouvelles requêtes sont reçues, alors les scores des correspondances changent, donnant plus d'importance aux correspondances pertinentes. Les scores des correspondances sont calculés régulièrement afin de prendre en compte la dynamicité.

Les correspondances impliquant l'ontologie d'un pair p (c.-à-d. les o_p-correspondances) sont d'une importance cruciale pour ce pair car elles vont lui permettre de traduire les requêtes qu'il reçoit. Nous proposons d'en stocker autant que possible (toutes si possible) dans un dépôt. Le cache, distinct de ce dépôt, est dédié au stockage des autres correspondances. Si le dépôt (ou le cache) est trop petit pour stocker toutes les correspondances, le pair peut utiliser la fonction de score pour en éliminer certaines. Nous utilisons les notations $repository(p)$ et $cache(p)$ pour faire référence au dépôt et au cache du pair p. Ils sont respectivement limités à r_{max} et c_{max} éléments.

6.2.2 Sélection des correspondances

Quand un pair doit envoyer des correspondances, il les sélectionne depuis le dépôt et le cache. Pour éviter de surcharger le réseau, nous limitons le volume d'information échangé : au plus m_{max} correspondances sont sélectionnées et envoyées. Par ailleurs nous introduisons la variable $\pi \in [0, 1]$ pour représenter le ratio de correspondances sélectionnées dans chaque ensemble (par défaut elle peut être fixé à $0,5$). Ainsi, un pair sélectionne $[\pi \cdot m_{max}]$ correspondances dans son dépôt et $[(1 - \pi) \cdot m_{max}]$ dans son cache. L'algorithme 5 explicite ce processus. Une sélection aléatoire et uniforme est effectuée afin d'assurer que deux correspondances du dépôt (ou du cache) ont la même probabilité d'être envoyées (cf. algorithme 5, lignes 2 et 4).

6.2.3 Traitement des correspondances

Quand un pair p reçoit un message, il exécute deux tâches. Premièrement il stocke localement les correspondances contenues dans le message : les o_p-correspondances sont placées dans $repository(p)$ et les autres dans $cache(p)$ (cf. algorithme 6, lignes 1 à 5). Deuxièmement il élimine certaines correspondances de son dépôt (si besoin) en conservant les r_{max} meilleures (cf. algorithme 6, ligne 6). Il procède de la même manière pour

Algorithme 5 : Fonction selectCorrespondencesToSend()

```
// Sélection des correspondances par un pair p
Output : ensemble des correspondances à envoyer à un pair p'.
1 r ← π · m_max ;
2 msg ← selectRandomly(repository(p), r) ;
3 c ← (1 − π) · m_max ;
4 msg ← msg ∪ selectRandomly(cache(p), c) ;
5 return msg ;
```

le cache (cf. algorithme 6, ligne 7). Pour cela la fonction de score est utilisée.

Algorithme 6 : Fonction storeCorrespondences(*corrs*)

```
Input : Un ensemble corrs de correspondances.
// Les correspondances sont ajoutées dans le dépôt
   ou dans le cache.
1 for c = ⟨id, e, e', r, n⟩ ∈ corrs do
2     if e ∈ μ(p) ∨ e' ∈ μ(p) then
          // Si c est une o_p-correspondance
3         repository(p) ← repository(p) ∪ {c} ;
4     else
5         cache(p) ← cache(p) ∪ {c} ;
// Limitation de la taille du dépôt et du cache.
// r_max et c_max sont des paramètres d'application.
6 if |repository(p)| > r_max then
7     sort(repository(p), sc) ;   // sc est la fonction de score
8     repository(p) ← keepBest(repository(p), r_max) ;
9 if |cache(p)| > c_max then
10    sort(cache(p), sc) ;
11    cache(p) ← keepBest(cache(p), c_max) ;
```

6.3 Gestion des incohérences

Jusqu'à présent, nous avons supposé que les correspondances étaient correctes : certaines peuvent manquer mais celle qui sont utilisées sont exactes. Dans la suite de ce chapitre, nous enlevons cette hypothèse et nous supposons que certaines correspondances peuvent être inexactes. L'inexactitude ne provient pas d'une mauvaise intention d'un pair. Il peut simplement s'agir d'une erreur liée à la méthode d'alignement utilisée. De plus, sans parler de correspondance inexacte, nous pensons que les pairs peuvent ne pas être d'accord sur l'existance d'une correspondance. En effet un pair p peut considérer que la correspondance $\langle id, e, e', r, n\rangle$ est correcte dans son contexte, alors que le pair p' la juge incorrecte. En réalité, cette correspondance peut être incohérente pour p'. Cette section vise à (i) définir ce qu'est une incohérence dans notre contexte, (ii) définir dif-

férentes stratégies possibles pour gérer ces incohérences, (iii) discuter sur la manière de prendre en compte à la fois l'incohérence et l'hétérogénéité.

6.3.1 Contexte

Dans cette partie de notre travail nous considérons que les pairs peuvent faire évoluer leurs ontologies au cours du temps. Néanmoins nous faisons l'hypothèse que les ontologies restent toujours cohérentes. Par contre le fait de considérer certaines correspondances peut mener à une situation d'incohérence avec l'ontologie. Par exemple cette situation peut se produire si l'ontologie d'un pair définit que les concepts c et c' sont disjoints, alors que l'ensemble des correspondances définit que le concept c est inclus dans le concept c'. Dans ce cas, est-ce que le pair doit participer au partage de cette correspondance ? Doit-il continuer à utiliser cette correspondance pour traduire les requêtes qu'il reçoit ? Doit-il essayer de "réparer" l'ensemble des correspondances lorsque celui-ci est incohérent par rapport à son ontologie ? Le pair pourrait considérer la valeur de confiance attribuée à chaque correspondance (cf. définition 5.5, page 73), ou attribuer des valeurs de confiance aux autres pairs pour déterminer si une correspondance peut être considérée comme fiable. Mais cette stratégie ne permet pas de prendre de décision lorsque les valeurs de confiance sont égales. Dans ce cas le problème général à adresser concerne la détection et la réparation. Ce problème a fait l'objet de nombreux travaux dans le domaine de la logique en général et de la logique de description en particulier. Notre objectif est d'utiliser les résultats des travaux récents pour proposer différentes stratégies à adopter en cas d'incohérence.

6.3.2 Logique de description

Dans cette partie nous considérons la logique de description $\mathcal{ALC}$ (cf. section 1.2.3, page 17). Le tableau 1.1 (page 18) présente la sémantique d'$\mathcal{ALC}$. Bien qu'elle ne soit pas très expressive, elle permet de définir la notion d'incohérence qui peut être généralisée à des logiques plus expressives.

Rappelons qu'une base de connaissances est souvent présentée comme l'union d'une TBox et d'une ABox. Une TBox contient les axiomes décrivant le domaine d'application alors que la ABox contient les assertions concernant les individus. Comme nous considérons que les ontologies ne contiennent pas d'individus, nous nous focalisons seulement sur les TBox.

Définition 6.3 Soit $(\Delta^{\mathcal{I}}, .^{\mathcal{I}})$ une interprétation.
- $(\Delta^{\mathcal{I}}, .^{\mathcal{I}})$ est un modèle d'axiomes $c \sqsubseteq d$ si $c^{\mathcal{I}} \subseteq d^{\mathcal{I}}$.
- $(\Delta^{\mathcal{I}}, .^{\mathcal{I}})$ est un modèle de $c \equiv d$ si $c^{\mathcal{I}} \subseteq d^{\mathcal{I}}$ et $d^{\mathcal{I}} \subseteq c^{\mathcal{I}}$.
- $(\Delta^{\mathcal{I}}, .^{\mathcal{I}})$ est un modèle d'une TBox si c'est un modèle de tous les axiomes de la TBox.

Définition 6.4 Un concept c est insatisfiable si et seulement si il n'existe aucun modèle $\mathcal{I}$ d'une TBox, $c^{\mathcal{I}} = \emptyset$.

Définition 6.5 Une TBox est cohérente s'il existe au moins un modèle. Elle est incohérente s'il n'existe aucun modèle.

6.3.3 Incohérence de la base de connaissance d'un pair

Sans modifier la définition de l'incohérence, il est intéressant d'analyser d'où peuvent venir les incohérences. Soit o l'ontologie d'un pair. Nous notons $\mathcal{A}x(o)$ l'ensemble des axiomes décrivant l'ontologie o, et $\mathcal{SC}$ l'ensemble des axiomes représentants les correspondances stockées (celles contenues dans le dépôt ou dans le cache). La base de connaissance du pair est alors $\mathcal{A}x(o) \cup \mathcal{SC}$.

Définition 6.6 La base de connaissance d'un pair est incohérente si $\mathcal{A}x(o) \cup \mathcal{SC}$ n'a aucun modèle.

Néanmoins rappelons que par hypothèse l'ensemble $\mathcal{A}x(o)$ est maintenu cohérent par le pair. Comme pour $\mathcal{SC}$, sa cohérence intrinsèque dépend des types de correspondances qui sont partagées. En effet si des relations comme la disjonction ou la non-inclusion sont autorisées, alors $\mathcal{SC}$ peut être lui-même incohérent. Néanmoins nous pensons que ce type de correspondances n'aide pas directement lors du processus de traduction de requêtes et que le fait de le calculer systématiquement peut surcharger le réseau avec des informations non utilisables. Quand seules des correspondances *positives* sont partagées (c.-à-d. où l'opérateur de négation et le concept impossible ne sont pas utilisés) l'ensemble $\mathcal{SC}$ est aussi cohérent. Ainsi l'incohérence peut seulement apparaître entre l'ensemble des correspondances $\mathcal{SC}$ et les axiomes de l'ontologie $\mathcal{A}x(o)$. Cela n'aide pas à vérifier la cohérence comme le pair considère $\mathcal{A}x(o) \cup \mathcal{SC}$ mais cela peut être utile pour restreindre l'ensemble des axiomes candidats à retirer lors de la réparation.

Exemple 6.3 Le pair p fait une distinction claire entre les concepts *Petale* et *Feuille* c'est-à-dire que $Petale \sqcap Feuille \sqsubseteq \bot$ appartient à l'ensemble $\mathcal{A}x(o)$. Le pair p' a envoyé une correspondance déclarant que le concept *Petale* est un type de *Feuille*, c'est-à-dire que $Petale \sqsubseteq Feuille$ appartient à l'ensemble des correspondances stockées par p ($\mathcal{SC}$). Dans ce cas, il n'y a pas de modèle possible pour $\mathcal{A}x(o) \cup \mathcal{SC}$. Une stratégie de réparation devrait éliminer $Petale \sqsubseteq Feuille$.

Le fait que ni $\mathcal{A}x(o)$ ni $\mathcal{SC}$ puissent être incohérents restreint les possibilités. Sans vouloir être exhaustif, nous listons des cas typiques pour lesquels il y a incohérence (dans un souci de simplicité, nous supposons que les ensembles d'axiomes sont clôts par déduction) :

- $\mathcal{A}x(o)$ déclare que les entités e et e' sont différentes et $\mathcal{SC}$ déclare qu'elles sont équivalentes ;
- $\mathcal{A}x(o)$ déclare que les entités e et e' sont différentes et que e' est moins général que e alors que $\mathcal{SC}$ déclare que e est moins général que e' ;
- $\mathcal{A}x(o)$ déclare que les entités e et e' sont disjointes et $\mathcal{SC}$ déclare qu'elles sont équivalentes ;
- $\mathcal{A}x(o)$ déclare que les entités e et e' sont disjointes et $\mathcal{SC}$ déclare que e est moins général que e'.

La vérification de cohérence, dépend de la logique considérée et peut être une tâche complexe. Pour plusieurs logiques, dont $\mathcal{ALC}$, les méthodes par tableaux ont été définies et de nombreux raisonneurs peuvent détecter les incohérences. C'est le cas de FaCT++ [TH06] ou Pellet [SPG+07] qui sont activement utilisés. Nous supposons que chaque pair utilise un raisonneur pour vérifier la cohérence de $\mathcal{A}x(o) \cup \mathcal{SC}$ (à l'arrivée de nouvelles correspondances ou moins fréquemment). Lorsqu'une incohérence est détectée, chaque pair met en œuvre une stratégie pour y faire face.

6.3.4 Stratégies en cas d'incohérence

Nous distinguons le *comportement local* d'un pair de son *comportement social*. Son comportement local comprend la manière dont il vérifie l'incohérence, la manière dont il modifie les correspondances qu'il stocke (dû au fait qu'il utilise une stratégie de réparation), et le choix des correspondances qu'il utilise pour traduire les requêtes qu'il reçoit. Son comportement social définit la manière dont il transmet les correspondances. Il est indépendant de son comportement local. Nous proposons trois stratégies possibles.

6.3.4.1 Stratégie "insouciante"

Un pair peut considérer que l'incohérence de $\mathcal{A}x(o) \cup \mathcal{SC}$ n'est pas véritablement un problème parce que $\mathcal{A}x(o)$ demeure cohérent (c'est l'une de nos hypothèses). Un pair utilise une stratégie "insouciante" si :

- il ne vérifie même pas si $\mathcal{A}x(o) \cup \mathcal{SC}$ est cohérent ;
- il ne retire aucune correspondance de $\mathcal{SC}$ et continue de traduire les requêtes qu'il reçoit en utilisant toutes les correspondances de $\mathcal{SC}$;
- il transmet toutes les correspondances de $\mathcal{SC}$.

Intuitivement, pour disséminer les correspondances, les pairs étant égaux, aucun d'entre eux ne fait prévaloir son point de vue. L'avantage de cette approche est que les pairs économisent des ressources de calcul car ils n'ont pas à vérifier la cohérence à chaque fois qu'une correspondance est reçue (dans le pire cas). Bien sûr, elle a le désavantage de disséminer des correspondances douteuses dans le système entier si un pair en émet (volontairement ou involontairement).

6.3.4.2 Stratégie "prudente"

Une autre stratégie consiste à essayer d'utiliser uniquement les correspondances qui semblent n'être impliquées dans aucune incohérence. Nous utilisons la notion d'ensemble minimal incohérent d'axiomes pour décrire cette stratégie [HS08].

Définition 6.7 Un ensemble minimal incohérent $\mathcal{MIS}$ de $\mathcal{A}x(o) \cup \mathcal{SC}$ est un ensemble d'axiomes $\mathcal{MIS} \subseteq \mathcal{A}x(o) \cup \mathcal{SC}$ tel que $\mathcal{MIS}$ est incohérent et que tout autre ensemble $\mathcal{MIS}' \subset \mathcal{MIS}$ est cohérent.

Notons qu'un ensemble d'axiomes incohérent peut avoir plusieurs sous-ensembles minimaux incohérents. Nous utilisons la notation $S_{\mathcal{MIS}}$ pour décrire l'ensemble des sous-ensembles minimaux incohérents

de $\mathcal{A}x(o) \cup \mathcal{SC}$. En utilisant les travaux de Reiter [Rei87], il est possible de définir des algorithmes permettant de calculer un *diagnostique*, c'est-à-dire le plus petit ensemble d'axiomes qui doivent être supprimés ou modifiés pour rendre l'ensemble initial cohérent. Dans notre cas, l'ensemble des candidats à éliminer est limité à un sous-ensemble de $\mathcal{SC}$.

Définition 6.8 Une correspondance douteuse est une correspondance de $\mathcal{SC}$ qui appartient à au moins un ensemble $\mathcal{MIS} \in S_{\mathcal{MIS}}$.

Étant données ces définitions, un pair met en œuvre une stratégie "prudente" si :

- il vérifie la cohérence et calcule régulièrement $S_{\mathcal{MIS}}$;
- il n'utilise pas de correspondance douteuse pour traduire les requêtes qu'il reçoit, et utilise une stratégie de réparation qui conduit à l'élimination d'une ou plusieurs correspondances de $\mathcal{SC}$;
- il ne transmet aucune correspondance douteuse.

Il y a en réalité de nombreuses variantes à cette stratégie car nous pourrions ajouter la notion de degré de doute, en considérant par exemple le nombre d'ensembles minimaux incohérents dans lesquels apparaît une correspondance.

Le pair doit alors choisir entre différentes stratégies de réparation, ou choisir de ne pas en effectuer. S'il choisit de réparer l'ensemble, il peut alors considérer que toutes les correspondances restantes sont sûres.

L'avantage de cette stratégie est que les pairs utilisent seulement les correspondances qui sont cohérentes avec ses connaissances lors de la traduction de requêtes. Le comportement social est davantage discutable si nous considérons qu'un pair n'ayant pas de droits particuliers par rapport aux autres pairs, n'est pas autorisé à empêcher la dissémination de certaines correspondances. Néanmoins s'il n'y a pas de pairs malveillants et si leurs points de vue sont compatibles cela semble être le meilleur moyen de ne pas propager des erreurs "honnêtes".

6.3.4.3 *Stratégie centrée sur les concepts clés*

Cette stratégie est basée sur deux observations. Premièrement, certains concepts semblent plus importants que d'autres dans l'ontologie des pairs. Deuxièmement, il peut être coûteux de calculer tous les ensembles minimaux incohérents, même si les pairs ne le font pas à chaque fois qu'ils reçoivent une nouvelle correspondance. L'idée de cette stratégie est d'identifier les correspondances qui rendent insatisfiables les concepts importants et d'utiliser une stratégie de réparation qui assure qu'ils restent satisfaisables. Pour définir la notion de concept important nous proposons de considérer les caractéristiques suivantes.

Concepts importants Comme les ontologies peuvent être volumineuses, il peut être utile de considérer ses concepts importants comme dans [PMd08]. Les auteurs proposent un algorithme pour calculer les concepts importants en se basant sur différents critères liés à des principes cognitifs, la structure de l'ontologie (couverture, densité, etc.) et à la popularité par rapport à certaines ressources (par exemple le moteur

de recherche Yahoo). Leur objectif est d'obtenir un genre de résumé de l'ontologie.

Concepts représentatifs Comme les ontologies sont utilisées pour annoter les ressources des pairs, certaines parties peuvent être plus représentatives que d'autres. Par exemple si le concept *Phénotype* est utilisé pour annoter 50% des documents avec un poids supérieur à $0,7$, alors ce concept est plus représentatif que le concept *Neige* qui est seulement utilisé pour annoter 5% des documents avec un poids inférieur à $0,3$. La représentativité des concepts par rapport aux documents peut être obtenue par une analyse des index.

Concepts populaires Un concept peut être défini comme populaire s'il est souvent utilisé dans les requêtes, directement ou après traduction. Cela nécessite que les pairs maintiennent une version plus riche de leurs historiques de requêtes (cf. section 6.2.1, page 92). Chaque pair doit compter le nombre de fois que chaque concept de son ontologie apparaît dans une requête traduite. La mesure devrait être normalisée en divisant par le nombre total d'occurrences de concepts dans les requêtes traduites. Ainsi un concept est populaire si le nombre de fois qu'il apparaît est supérieur à un certain seuil.

Chaque pair peut définir ses concepts importants, en considérant une combinaison des différents critères. Une fois identifiés, l'idée est d'empêcher les traductions problématiques de concepts importants. Dans [SHCVH07], les auteurs définissent la notion de plus petite sous-TBox préservant l'insatifiabilité d'un concept a d'une certaine TBox. Ils fournissent également un algorithme pour effectuer le calcul. Nous adaptons leur définition. Notons qu'elle est définie pour un concept atomique (comme dans le papier original).

Définition 6.9 Un plus petit sous-ensemble préservant l'insatisfiabilité d'un concept a par rapport à $\mathcal{A}x(o) \cup \mathcal{SC}$, noté $\mathcal{MUS}$ est un ensemble d'axiomes $\mathcal{MUS} \subset \mathcal{A}x(o) \cup \mathcal{SC}$ tel que le concept a est instatisfaisable dans $\mathcal{MUS}$ et est satisfaisable pour tout ensemble $\mathcal{MUS}' \subset \mathcal{MUS}$.

Une fois un ensemble $\mathcal{MUS}$ obtenu, il est possible de calculer un diagnostique, c'est-à-dire un plus petit ensemble d'axiomes qui doit être supprimé ou corrigé pour qu'un concept donné devienne satisfaisable. En d'autres termes, au lieu de calculer un diagnostique qui rend tous les concepts satisfaisables, nous proposons de limiter le calcul du diagnostique aux concepts importants, et de réparer l'ensemble des correspondances en conséquence. Notons que pour des raisons de traçabilité pour de grosses ontologies, Schlobach *et al.* ont proposé une heuristique qui permet de guider l'algorithme vers l'ensemble approprié de diagnostiques [SHCVH07].

Finalement, un pair met en œuvre une stratégie centrée sur les concepts clés s'il maintient une liste de concepts clés de son ontologie et si :

- il calcule l'ensemble $\mathcal{MUS}$ pour chaque concept important de son ontologie et répare l'ensemble $\mathcal{SC}$;

– il utilise les correspondances de $\mathcal{SC}$ pour traduire les requêtes entrantes ;
– il transmet les correspondances de $\mathcal{SC}$.

L'avantage de cette stratégie est qu'elle assure que les pairs sont cohérents lorsqu'ils traduisent les requêtes concernant les concepts clés. La discussion à propos du comportement social (c.-à-d. ce qui doit être transmis) est la même que précédemment.

Pour conclure nous pouvons dire qu'il n'existe pas de stratégie parfaite et que la meilleure stratégie à choisir dépend de l'application. En effet si le risque d'incohérence est limité, il n'est pas nécessaire que chaque pair utilise un raisonneur pour vérifier la cohérence, qu'il calcule les ensembles minimaux, et qu'il adopte une stratégie de réparation.

6.3.5 Mesures d'hétérogénéité et incohérence

Dans cette section nous discutons brièvement du lien qui existe entre l'incohérence et différentes facettes de l'hétérogénéité. Nous montrons également comment des mesures appropriées peuvent reflèter ce lien.

Intuitivement la diversité, qui dépend du nombre d'ontologies différentes utilisées dans le système, n'est pas affectée par l'incohérence. À l'inverse, les mesures de disparité entre pairs peuvent être impactées par l'existence de correspondances douteuses. Il en est de même pour l'hétérogénéité. Il y a deux options. La première consiste à conserver les mesures d'hétérogénéité telles qu'elles sont définies dans la section 5.4 (page 77) et à introduire en complément des mesures d'incohérence. Dans ce cas, une mesure d'hétérogénéité faible doit être considérée à la lumière de l'incohérence globale du système (car il peut y avoir beaucoup de correspondances douteuses). La deuxième option consiste à modifier la définition de certaines mesures d'hétérogénéité pour englober la notion d'incohérence.

6.3.5.1 Introduction d'une mesure d'incohérence

Plusieurs travaux ont proposé des mesures d'incohérence d'une base de connaissance, aussi bien pour la logique classique que pour la logique de description. Certaines sont basées sur une logique paraconsistente [MQHL07, MQH11]. Nous adaptons les définitions utilisées dans [HS08] pour donner des définitions simples, utilisant les ensembles incohérents minimaux $\mathcal{MIS}$. Nous renvoyons aux travaux cités ci-dessus pour plus de détails.

L'approche consiste à évaluer l'incohérence introduite par des correspondances douteuses, puis à calculer l'incohérence globale de la base de connaissances du pair. Finalement l'incohérence du système doit être définie en considérant l'incohérence de chaque pair.

Définition 6.10 Soit *corr* une correspondance douteuse. La valeur d'incohérence de *corr*, notée $IncV(corr)$, par rapport à une base de connaissance $\mathcal{A}x(o) \cup \mathcal{SC}$ est donnée par :

$$IncV(corr) = \sum_{\mathcal{MIS} \in \mathcal{MIS}(corr)} \frac{1}{|\mathcal{MIS}|} \tag{6.2}$$

où $\mathcal{MIS}(corr) = \{\mathcal{MIS} \in S_{\mathcal{MIS}} : corr \in \mathcal{MIS}\}$.

Au lieu de simplement compter le nombre d'ensembles minimaux incohérents auxquels la formule appartient, la définition ci-dessus considère également la cardinalité de l'ensemble minimal incohérent. Intuitivement, une correspondance conflictuelle a plus d'impact si le nombre de correspondances dans l'ensemble minimal incohérent est petit.

Exemple 6.4 Soient deux correspondances $corr_1$ et $corr_2$ telles que :
- $corr_1$ appartient à deux ensembles minimaux incohérents dont les cardinalités sont respectivement 10 et 5,
- $corr_2$ appartient à deux ensembles minimaux incohérents dont les cardinalités sont respectivement 2 et 4.

En utilisant l'équation 6.2, on trouve que $IncV(corr_1) = \frac{1}{10} + \frac{1}{5} = 0,33$ et que $IncV(corr_2) = \frac{1}{2} + \frac{1}{4} = 0,75$. Cela reflète le fait que $corr_2$, qui appartient à deux ensembles minimaux incohérents (tout comme $corr_1$) peut être plus conflictuelle parce que leurs cardinalités sont faibles.

Étant données les valeurs d'incohérences de toutes les correspondances, il est possible de définir la valeur d'incohérence pour la base de connaissance d'un pair $\mathcal{A}x(o) \cup \mathcal{SC}$. Plusieurs choix sont possibles pour agréger les valeurs d'incohérence. Selon nous il est important de considérer le pire cas, c'est-à-dire la valeur maximale.

Définition 6.11 Soit $\mathcal{A}x(o) \cup \mathcal{SC}$ la base de connaissance d'un pair p et soit $\mathcal{D}$ un ensemble de correspondances douteuses. La valeur d'incohérence du pair p, notée $IncV(p)$, est donnée par :

$$IncV(p) = \max_{corr \in \mathcal{D}}(IncV(corr)) \tag{6.3}$$

Exemple 6.5 Soit un pair p connaissant un ensemble de correspondances douteuses $\mathcal{D}$ contenant les correspondances $corr_1$ et $corr_2$ de l'exemple 6.4. Alors la valeur d'incohérence du pair p est donnée par :
$IncV(p) = \max(incV(corr_1), incV(corr_2)) = 0,75$

Ainsi pour un système P2P donné $s = \langle \mathcal{P}, \mathcal{N} \rangle$, nous considérons la moyenne des valeurs d'incohérence des pairs, pour refléter dans une certaine mesure à quel point le système est globalement incohérent. Nous ne considérons pas la relation de voisinage $\mathcal{N}$ parce que selon nous le fait que certains pairs soient liés n'augmente ni ne diminue l'incohérence.

Dans cette approche, le système est toujours caractérisé par ses mesures d'hétérogénéité (cf. section 5.4, page 77) mais étant données ses valeurs d'incohérence.

6.3.5.2 *Adaptation des mesures d'hétérogénéité*

Intuitivement nous voulons prendre en compte le fait que certaines correspondances connues par un pair entre son ontologie et l'ontologie d'un autre pair sont douteuses. Une idée simple est de modifier la définition de disparité entre deux pairs. Par exemple, considérons que la disparité d'un pair p' par rapport à un pair p prend en compte le nombre

d'entités connues par p ayant une correspondance dans l'ontologie de p' :

$$d(p,p') = \begin{cases} 0 & \text{si } \mu(p) = \mu(p') \\ \frac{|\{e \in \mu(p) : \nexists \langle id,e,e',r,n\rangle \in \kappa(p') \text{ tel que } e' \in \mu(p')\}|}{|\{e \in \mu(p)\}|} & \text{sinon} \end{cases} \quad (6.4)$$

Si nous considérons l'ensemble $\kappa(p')$ comme étant l'ensemble des correspondances connues par p', la mesure de disparité ne prend pas en compte le fait que certaines correspondances peuvent être douteuses. Il suffit de modifier cette définition pour ne considérer que les correspondances qui ne sont pas douteuses pour obtenir une mesure de disparité qui englobe la notion d'incohérence :

$$d(p,p') = \begin{cases} 0 & \text{si } \mu(p) = \mu(p') \\ \frac{|\{e \in \mu(p) : \nexists \langle id,e,e',r,n\rangle \in (\kappa(p') \setminus DC) \text{ tel que } e' \in \mu(p')\}|}{|\{e \in \mu(p)\}|} & \text{sinon} \end{cases} \quad (6.5)$$

où DC est l'ensemble des correspondances douteuses contenues dans $\kappa(p')$. Par conséquence les mesures d'hétérogénéité dans lesquelles elle est utilisée (par exemple $\mathcal{H}_{Disp}$ ou $\mathcal{H}_{DapAvg}$) prennent également en compte l'incohérence.

6.4 Analyse théorique des coûts

Dans cette section nous étudions le coût théorique du protocole CorDis en termes de stockage local et de trafic réseau. Dans ce protocole les pairs échangent des correspondances de la forme $\langle id, e, e', r, n\rangle$. Nous supposons que :

- id est un nombre entier long codé sur x octets,
- e et e' sont des URIs codés sur y octets,
- r est représenté sous la forme d'un nombre entier court codé sur w octets (par ex. $0 : \equiv$, $1 : \sqsubseteq$, etc.),
- n est un nombre réel codé sur z octets.

Dans ce cas, le volume ψ d'une correspondance est de $[x + 2y + w + z]$ octets.

6.4.1 Stockage local

Les pairs stockent au plus r_{max} correspondances dans leurs dépôts et c_{max} correspondances dans leurs caches. Dans le pire des cas, les correspondances occupent $[(r_{max} + c_{max}) \cdot \psi]$ octets dans l'espace de stockage de chaque pair.

6.4.2 Trafic réseau

Comme à chaque échange les pairs se transmettent au plus m_{max} correspondances, le trafic réseau généré à chaque cycle est de $[2 \cdot |\mathcal{P}| \cdot m_{max} \cdot \psi]$ octets. Le facteur 2 vient du fait que les échanges sont bidirectionnels.

6.5 Évaluations

6.5.1 Objectifs

Dans cette section, nous avons pour objectif d'évaluer les performances du protocole CorDis présenté à la section 6.2 (page 92). Dans un premier temps nous étudions l'évolution de l'hétérogénéité sémantique pour des systèmes P2P dont le nombre d'ontologies utilisées varie. Dans un deuxième temps nous nous intéressons à l'évolution de l'hétérogénéité dans des systèmes de différentes tailles. L'objectif est de vérifier que le protocole est toujours efficace lorsque le nombre de pairs croît. Enfin, dans un troisième temps, nous étudions les performances de CorDis dans différentes situations de dynamicité. L'objectif est de montrer qu'il est toujours efficace même lorsque les pairs rejoignent et quittent le système fréquemment.

6.5.2 Matériel, paramétrage de CorDis et méthodologie

6.5.2.1 Ontologies et correspondances

Nous avons utilisé les services fournis par BioPortal pour obtenir un ensemble d'ontologies activement utilisées dans le domaine biomédical. BioPortal est un dépôt ouvert de ressources biomédicales qui fournit en particulier un accès à des ontologies développées en OWL, RDF ou OBO [FNSW+09]. Nous avons transformé les ontologies OBO en ontologies OWL à l'aide de l'API ONTO-PERL [2] [AED+08]. En définitive nous disposons de 149 ontologies exploitables avec l'API OWL [3] [BVL03]. En moyenne, les ontologies contiennent 1100 concepts. La plus petite ontologie contient seulement 5 concepts et la plus importante en contient 23141.

Nous avons également obtenu un ensemble de 1442 alignements accessibles via des services de BioPortal. Ces alignements lient 93 ontologies : 58 ontologies ne sont concernées par aucun alignement. Globalement les alignements contiennent 60944 correspondances d'équivalence entre concepts pour lesquelles la valeur de confiance est maximale : toutes les correspondances sont de la forme $\langle id, c, c', \equiv, 1\rangle$. Cela signifie qu'en moyenne, chacune des 93 ontologies est concernée par 655 correspondances. Le tableau 6.1 montre que les ontologies sont très peu liées par les correspondances. La colonne "Nombre théorique" correspond au cas où il existe un alignement entre tous les couples d'ontologies, et où chaque alignement est complet (c.-à-d. qu'il contient une équivalence pour chaque concept de l'ontologie). La colonne "Nombre réel" correspond aux données que nous avons obtenu sur BioPortal. Du point de vue de l'hétérogénéité, ce jeu de données est très critique puisque les ontologies ont très peu de concepts en commun : les ontologies se recouvrent très peu. Il est donc probable que l'hétérogénéité liée aux disparités entre pairs soit difficile à réduire car le nombre de correspondances est faible.

2. `http://search.cpan.org/dist/ONTO-PERL`.
3. `http://owlapi.sourceforge.net`.

	Nombre théorique	Nombre réel	Ratio (%)
Ontologies	-	93	-
Alignements	8556 (93×92)	1442	16,8
Correspondances	8958132 (1047×8556)	60944	0,68
Corresp./ontologie	96324 (1047×92)	655	0,68

TABLE 6.1 – *Comparaison des valeurs théoriques et réelles concernant les ontologies et correspondances obtenues sur BioPortal.*

6.5.2.2 *Système P2P sémantique non-structuré*

Au lieu de mettre en place un système P2P réel, nous avons choisi d'utiliser un outil de simulation pour mener nos expérimentations. Cette approche a les avantages suivants :

- elle permet de déployer facilement un système sur une seule machine (la collecte des résultats est facilitée),
- elle permet de maîtriser tous les paramètres du système (nombre de pairs, dynamicité, etc.),
- elle permet de répéter les expérimentations dans des conditions parfaitement identiques.

Nous avons utilisé le simulateur PeerSim qui est largement utilisé dans le milieu académique [MJ09]. PeerSim, qui est développé en Java, permet de générer des graphes aléatoires de pairs : chaque pair est lié à d'autres pairs. Il génère des systèmes P2P non-structurés dont la taille et la connectivité de chaque nœud peuvent être paramétrés.

Nous avons implémenté le protocole CORDIS en Java et nous l'avons intégré à PeerSim. Ce dernier propose un modèle cyclique : cela nous permet de faire exécuter régulièrement le processus du thread actif de CORDIS sur chaque pair (cf. algorithme 3, page 93).

Par ailleurs, les ontologies sont assignées aléatoirement aux pairs suivant une distribution de Poisson où $\lambda = 5$. Cela représente une situation dans laquelle certaines ontologies sont utilisées par de nombreux pairs, et d'autres sont utilisées par peu de pairs. Nous pensons que cela représente une situation réaliste. Enfin chaque pair est muni d'un ensemble de correspondances qui concerne sa propre ontologie. Comme il est peu réaliste que les pairs aient connaissance de toutes les correspondances concernant leurs ontologies, nous avons limité leur nombre. Ainsi, en moyenne, chaque pair connaît 20% des correspondances qui existent (et qui concernent son ontologie).

Le protocole CORDIS utilise l'historique de requêtes des pairs pour attribuer des scores aux correspondances. Dans les expérimentations, nous considérons que ces historiques évoluent constamment au cours du temps. Par conséquent, les valeurs des fonctions de score varient continuellement. Cette situation est considérée comme critique.

6.5.2.3 *Métriques*

Hétérogénéité sémantique Pour mesurer les variations des différentes facettes de l'hétérogénéité, nous utilisons des mesures définies dans la section 5.4 du chapitre 5 (page 77). Nous utilisons la mesure de diversité sémantique, et les mesures qui considèrent la disparité entre les pairs :

	Type	Volume
w	Entier court	4
x	Entier long	20
y	URI	50
z	Réel	4

TABLE 6.2 – *Volumes correspondant aux différents types considérés. La première colonne correspond aux symboles utilisés dans les analyses théoriques de coûts (section 6.4, page 103 et section 7.3, page 128).*

- $\mathcal{H}_{Div}$ mesure la diversité sémantique d'un système (cf. équation 5.1, page 77) ;
- $\mathcal{H}_{Disp}$ mesure la disparité moyenne du système sans prendre en compte sa topologie (cf. équation 5.4, page 78) ;
- $\mathcal{H}_{DispAPMoy}$ mesure la disparité moyenne du système en considérant sa topologie, c.-à-d. les liens de voisinage entre les pairs (cf. équation 5.8, page 79).

Nous focalisons notre analyse sur ces mesures car CORDIS vise à diminuer l'hétérogénéité liée aux disparités entre pairs. La mesure tenant seulement compte de la topologie du système ($\mathcal{H}_{DivAPMoy}$) ne varie pas car le protocole ne modifie pas son organisation. La mesure de diversité $\mathcal{H}_{Div}$ est utilisée pour caractériser la situation du système : plus le système est dans une situation critique du point de vue de l'hétérogénéité, plus sa valeur est élevée. Elle n'évolue pas au cours du temps, même lorsque CORDIS est en cours d'exécution.

Pour mesurer la disparité entre un pair p et un autre pair p' nous avons utilisé la mesure $d_{concepts}$ définie par l'équation 5.9 (page 81) en posant :

$$\textsc{Sim}_{o\to o'}(c,c') = \begin{cases} 0 & \text{si } \exists\langle id, c, c', \equiv, 1\rangle \in \kappa(p') \\ 1 & \text{sinon} \end{cases} \tag{6.6}$$

où p et p' utilisent respectivement les ontologies o et o', et où $\kappa(p')$ représente l'ensemble des correspondances connues par p'.

Volume de données Pour mesurer le volume de données qui transitte sur le réseau ou qui est stocké chez les pairs, nous considérons les volumes donnés dans le tableau 6.2.

D'après l'analyse théorique (section 6.4, page 103) et le tableau 6.2, le volume ψ (en octet) d'une correspondance est égal à :

$$\begin{aligned} \psi &= x + 2y + w + z \\ &= 20 + (2 \cdot 50) + 4 + 4 \\ &= 128 \end{aligned} \tag{6.7}$$

Nous mesurons le trafic réseau en additionnant le volume de données envoyé par un pair à un autre pair à chaque cycle. Pour un cycle donné, le trafic réseau TR est donné par :

$$TR = \sum_{p\in\mathcal{P}} emission(p) \tag{6.8}$$

où $emission(p)$ correspond au volume (en octet) de données émises par p. De la même manière, le volume de données stockées (DS) est donné par :

$$DS = \sum_{p \in \mathcal{P}} stocke(p) \tag{6.9}$$

où $stocke(p)$ correspond au volume (en octet) de données stockées par p. Ces deux mesures sont globales : pour obtenir une moyenne par pair, il suffit de diviser TR ou DS par le nombre de pairs du système (c.-à-d. $|\mathcal{P}|$).

6.5.2.4 *Paramétrage de* CorDis

Le protocole CorDis considère un certain nombre de paramètres : la taille du dépôt r_{max}, la taille du cache c_{max}, le nombre maximal de correspondances échangées à chaque échange m_{max}, et la proportion de correspondances sélectionnées depuis le dépôt et depuis le cache π. Dans ces expérimentations nous avons choisi de ne pas limiter la taille du dépôt (c.-à-d. $r_{max} = \infty$) en considérant que les pairs sont prêts à stocker toutes les correspondances impliquant leurs ontologies. Cela suppose évidemment qu'ils disposent de suffisamment d'espace de stockage. Par ailleurs nous avons limité la taille maximale du cache à 200 correspondances (c.-à-d. $c_{max} = 200$). De cette manière nous assurons que les pairs ne stockent pas une trop grosse quantité de correspondances qui ne les concernent pas, tout en faisant en sorte que la dissémination soit rapide. Le nombre de correspondances échangées à chaque cycle est limité à 75 entrées (c.-à-d. $m_{max} = 75$). Comme nous considérons que $\pi = 2/3$, cela signifie que 50 correspondances sont sélectionnées depuis le dépôt et 25 depuis le cache. Nous avons montré (expérimentalement) dans [CCL11a] que le fait de réduire le nombre de correspondances échangées à chaque cycle n'empêche pas de réduire l'hétérogénéité au même niveau. Par contre la vitesse à laquelle elle est réduite est moins importante. Nous avons également montré que la taille du cache n'a pas d'impact sur la réduction de l'hétérogénéité. C'est pour cette raison que nous considérons un cache de taille relativement importante (200 correspondances). Enfin nous avons montré que lorsque la taille du dépôt est limitée, la réduction de l'hétérogénéité est également limitée. Le tableau 6.3 présente l'ensemble des paramètres de CorDis choisis pour les expérimentations.

Paramètre	Valeur
Taille maximale du dépôt r_{max}	∞
Taille maximale du cache c_{max}	200
Correspondances par message m_{max}	75
Proportion π	2/3
Stratégie de gestion des incohérences	"insouciante"

Table 6.3 – *Paramètres de* CorDis.

En ce qui concerne la stratégie de gestion des incohérences, nous avons choisi de considérer la stratégie "insouciante" car nous n'avons pas d'information sur la qualité ou l'exactitude des correspondances que nous avons obtenues sur BioPortal. Cette stratégie (décrite dans la section 6.3.4.1, page 98) ne vérifie pas la présence d'incohérence dans la base de connaissances et dissémine toutes les correspondances.

6.5.2.5 *Méthodologie d'évaluation*

Le simulateur PeerSim permet de simuler le fonctionnement d'un système P2P. La plupart des événements sont effectués de manière aléatoire : génération des liens entre pairs, attribution des ontologies aux pairs, arrivée/départ de pairs, etc. Pour garantir la reproductibilité des simulations dans conditions parfaitement similaires, PeerSim utilise une *graine* pour simuler les événements aléatoires. Pour éviter de présenter des résultats issus de cas particuliers, nous avons lancé chaque expérimentation trois fois avec des graines différentes. Les résultats présentés correspondent à la moyenne des résultats obtenus par chaque simulation.

6.5.3 Évaluation de CorDis dans différentes situations de diversité (variation du nombre d'ontologies)

Paramètres de simulation

Dans ces expérimentations nous étudions le comportement du protocole CorDis en fonction du nombre d'ontologies utilisées dans le système. Cela représente différentes situations de diversité sémantique (qui peut être mesurée avec $\mathcal{H}_{Div}$). Nous considérons des systèmes P2P de 500 pairs dans lesquels nous faisons varier le nombre d'ontologies différentes utilisées : 23, 46 ou 93. De cette manière la diversité sémantique a pour valeur $0,045$, $0,09$ et $0,18$. Ces valeurs semblent peu élevées, mais en réalité elles représentent des situations relativement critiques. Par exemple, lorsque 93 ontologies sont utilisées, cela signifie qu'en moyenne chaque ontologie est partagée par seulement $5,3$ pairs. La connectivité du système est fixée à 4 : chaque pair connaît l'adresse de quatre autres pairs du système. Le tableau 6.4 présente l'ensemble des paramètres considérés. Dans ces expérimentations les systèmes ne sont pas dynamiques : aucun pair ne rejoint ni ne quitte le système au cours de la simulation.

Paramètre	Valeur(s)
Nombre de pairs	500
Nombre d'ontologies	23, 46, 93
Connectivité	4

Table 6.4 – *Paramètres considérés pour étudier les performances de* CorDis *en faisant varier le nombre d'ontologies utilisées.*

Résultats et observations

Les figures 6.1 et 6.2 présentent l'évolution des valeurs de $\mathcal{H}_{Disp}$ et $\mathcal{H}_{DispAPMoy}$ au cours du temps lorsque le protocole CorDis est en cours d'exécution. Les lignes horizontales correspondent aux valeurs optimales théoriques qui ne peuvent être obtenues que lorsque tous les pairs connaissent toutes les correspondances impliquant leurs ontologies. *Il est important de noter que les valeurs d'hétérogénéité ne peuvent pas descendre en dessous de ces valeurs limites.* Celles-ci sont d'ailleurs directement liées aux données (ontologies et correspondances) que nous considérons.

Nous pouvons voir que $\mathcal{H}_{Disp}$ et $\mathcal{H}_{DispAPMoy}$ évoluent de la même manière. Cela est dû au fait que l'organisation du système est aléatoire : les pairs ne sont pas organisés en fonction de l'ontologie qu'ils utilisent.

Nous constatons que plus la diversité sémantique ($\mathcal{H}_{Div}$) est élevée, moins le protocole CorDis permet de réduire l'hétérogénéité liée aux disparités entre pairs ($\mathcal{H}_{Disp}$ et $\mathcal{H}_{DispAPMoy}$). Néanmoins, même lorsque $\mathcal{H}_{Div} = 0,18$, la valeur de $\mathcal{H}_{Disp}$ est réduite de $0,94$ à $0,78$. Dans tous les cas, elle converge vers la valeur optimale après 250, 300 et 350 cycles lorsque $\mathcal{H}_{Div}$ vaut respectivement $0,045$, $0,09$ et $0,18$. Les valeurs de $\mathcal{H}_{DispAPMoy}$ sont légèrement différentes mais les observations sont identiques (cf. figure 6.2).

L'analyse théorique des coûts faite dans la section 6.4 (page 103) montre que chaque pair du système stocke au plus $[(r_{max} + c_{max}) \cdot \psi]$ octets. Comme nous n'avons pas limité la taille du dépôt ($r_{max} = \infty$), le volume de données stocké peut en théorie tendre vers l'infini. La figure 6.3 montre que le volume stocké DS augmente de manière logarithmique. Lorsque 46 ou 93 ontologies sont utilisées, le volume stocké DS tend vers 75 Mo, et lorsque 23 ontologies sont utilisées, le volume DS tend vers 150 Mo. On peut déduire que moins le nombre d'ontologies utilisées dans le système est élevé, plus les pairs stockent de correspodances localement. Comme le volume du cache est limité à $25,6$ Mo ($128 \times c_{max}$), nous pensons que lorsque trop peu de pairs partagent la même ontologie (c'est le cas lorsque 93 ontologies sont utilisées par 500 pairs), il se peut que le nombre de correspondances connues initialement soit trop faible pour remplir le dépôt de la même manière que lorsque peu d'ontologies sont utilisées (c'est le cas lorsque 23 ontologies sont utilisées par 500 pairs). Pour confirmer ce résultat, il faudrait mener des expérimentions complémentaires, en choisissant éventuellement une autre distribution que la loi de Poisson pour distribuer les ontologies

Au niveau du trafic réseau, nous nous attendons à ce qu'il soit égal à $(2 \cdot |\mathcal{P}| \cdot m_{max} \cdot \psi)$ octets où $|\mathcal{P}| = 500$, $m_{max} = 75$ et $\psi = 128$ (cf. équation 6.7, page 106). En définitive le trafic réseau TR doit tendre vers 9600 Mo. C'est effectivement ce que nous observons sur la figure 6.4. Nous pensons que le trafic réseau n'atteint pas cette valeur lorsque 46 ou 93 ontologies sont utilisées car les pairs envoient moins de m_{max} correspondances à chaque échange. La raison provient probablement du fait qu'ils ne stockent pas suffisamment de correspondances dans leur dépôt, c'est-à-dire que : $|repository(p)| < \pi \times m_{max}$.

6.5.4 Évaluation de CorDis dans différentes situations de diversité (variation du nombre de pairs)

Paramètres de simulation

Dans cette section nous étudions le comportement de CorDis dans des systèmes de tailles différentes. L'objectif est de vérifier que le protocole reste efficace lorsqu'on fait croître le nombre de pairs. Nous étudions trois systèmes : le premier contient 250 pairs, le deuxième contient 500 pairs et le troisième en contient 1000. La connectivité des systèmes est

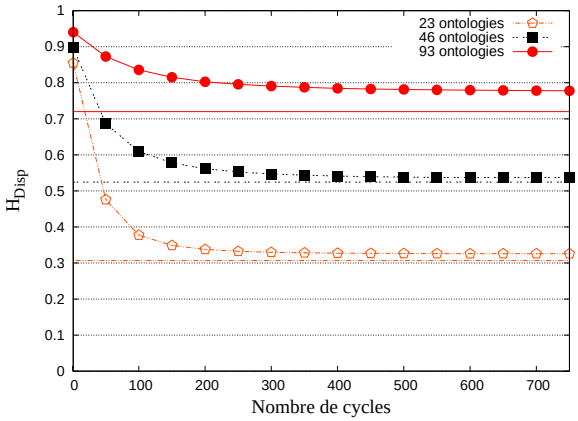


FIGURE 6.1 – *Évolution de $\mathcal{H}_{Disp}$ dans des systèmes de* 500 *pairs.*

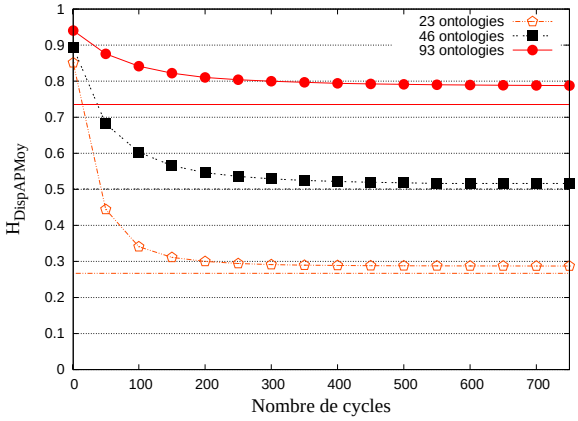


FIGURE 6.2 – *Évolution de $\mathcal{H}_{DapAPMoy}$ dans des systèmes de* 500 *pairs.*

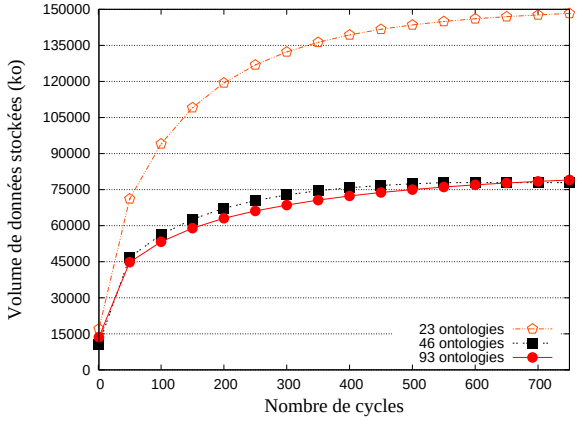


FIGURE 6.3 – *Évolution du volume des correspondances stockées dans des systèmes de* 500 *pairs.*

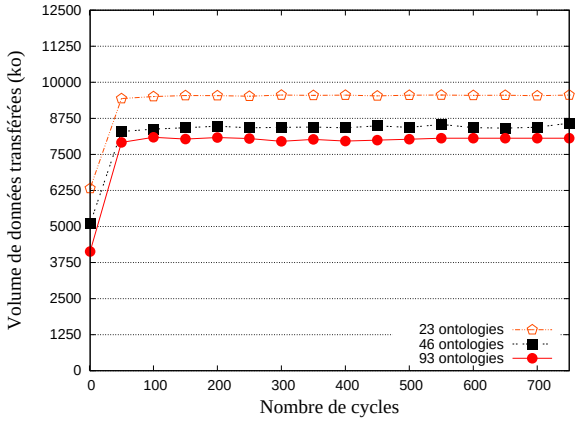


FIGURE 6.4 – *Évolution du trafic réseau dans des systèmes de* 500 *pairs.*

toujours fixée à 4. Dans tous les cas, les pairs utilisent 93 ontologies différentes. Par conséquent, les valeurs de diversité sémantique valent respectivement 0,36, 0,18 et 0,09. Cette fois encore, les systèmes ne sont pas dynamiques. Le tableau 6.5 présente les paramètres utilisés.

Nous avons considéré des systèmes P2P de petites tailles car nous disposons de seulement 93 ontologies. Pour un nombre donné d'ontologies, le fait de considérer des système de 10000 ou 100000 pairs ne fait que rendre la situation d'hétérogénéité sémantique plus simple à gérer. En effet, dans ce cas la diversité $\mathcal{H}_{Div}$ est égal à 0,0093 ou 0,00093. C'est pour cette raison que nous n'avons pas considéré des systèmes de tailles importantes.

Paramètre	Valeur(s)
Nombre de pairs	250, 500, 1000
Nombre d'ontologies	93
Connectivité	4

TABLE 6.5 – *Paramètres considérés pour étudier les performances de* CORDIS *en faisant varier le nombre de pairs.*

Résultats et observations

Les figures 6.5 et 6.6 présentent respectivement l'évolution des valeurs d'hétérogénéité $\mathcal{H}_{Disp}$ et $\mathcal{H}_{DispAPMoy}$ au cours du temps. Les valeurs optimales théoriques sont matérialisées par les lignes horizontales : l'hétérogénéité ne peut pas être réduite en dessous de ces limites (qui dépendent des données considérées).

Pour les deux mesures considérées ($\mathcal{H}_{Disp}$ et $\mathcal{H}_{DispAPMoy}$) nous pouvons voir que la valeur d'hétérogénéité est davantage réduite lorsque le nombre de pairs croît. Cela montre clairement que, pour un nombre donné d'ontologies, le fait que les pairs soient nombreux favorise la diminution de l'hétérogénéité. Cela s'explique par le fait que le taux de réplication des correspondances dans le système est plus élevé : chaque correspondance est stockée par un nombre plus important de pairs. Par conséquent elles sont plus facilement "accessibles" aux pairs pour qui elles sont utiles.

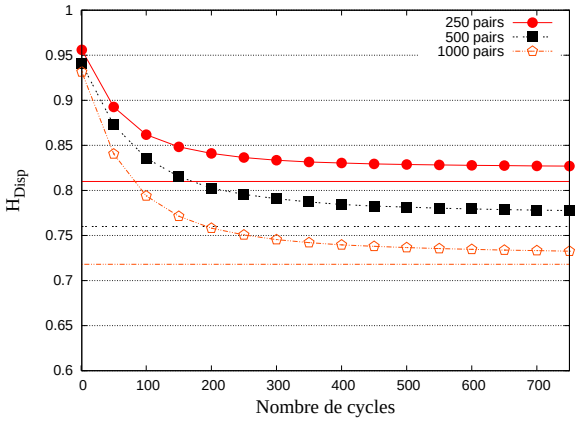


FIGURE 6.5 – *Évolution de $\mathcal{H}_{Disp}$ dans des systèmes dans lesquels 93 ontologies sont utilisées.*

On peut donc conclure que le protocole CORDIS reste efficace lorsque le nombre de pairs augmente.

Le temps de convergence est plus important pour les systèmes de 1000 pairs que pour les systèmes de 250 pairs. De manière générale, plus le système contient de pairs et plus le temps de convergence est important. Par contre, nous voyons qu'après 100 cycles, la valeur d'hétérogénéité obtenue dans le système de 1000 pairs est inférieure à celle obtenue après convergence dans le système de 250 pairs : il faut donc nuancer la constatation que le temps de convergence est plus important car cela correspond au fait que l'hétérogénéité est davantage réduite.

La figure 6.7 montre que plus les pairs sont nombreux dans le système, plus le volume total de données stockées est important. Lorsque le nombre de pairs double, le volume fait plus que doubler. Cela montre que les correspondances sont davantages répliquées lorsque les pairs sont nombreux. Comme dans la section précédente (section 6.5.3, page 108), on remarque que l'évolution du volume stocké est logarithmique.

La figure 6.8 montre que le trafic réseau mesuré est celui attendu d'après l'analyse théorique faite en section 6.4 (page 103). En effet le trafic réseau est proche de :

- 19200 Mo pour les systèmes de 1000 pairs ($128 \times 75 \times 2 \times 1000$),
- 9600 Mo pour les systèmes de 500 pairs ($128 \times 75 \times 2 \times 500$),
- 4800 Mo pour les systèmes de 250 pairs ($128 \times 75 \times 2 \times 250$),

Comme dans la section précédente, nous remarquons que les valeurs mesurées sont inférieures aux valeurs théoriques car les pairs ne connaissent probablement pas suffisamment de correspondances pour en sélectionner $\pi \cdot m_{max}$ depuis leurs dépôts. Des expérimentations complémentaires devraient être menées pour vérifier cette hypothèse.

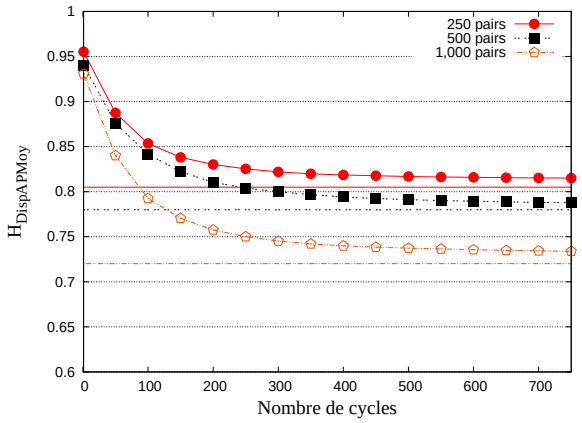


FIGURE 6.6 – *Évolution de $\mathcal{H}_{DapAPMoy}$ dans des systèmes dans lesquels 93 ontologies sont utilisées.*

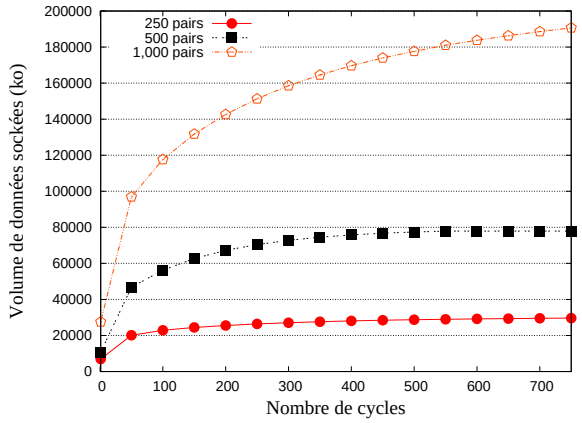


FIGURE 6.7 – *Évolution du volume des correspondances stockées dans des systèmes dans lesquels 93 ontologies sont utilisées.*

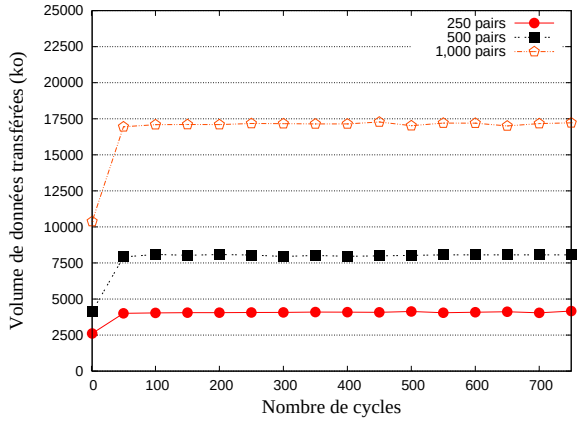


FIGURE 6.8 – *Évolution du trafic réseau dans des systèmes dans lesquels 93 ontologies sont utilisées.*

6.5.5 Évaluation de CorDis dans différentes situations de dynamicité

Paramètres de simulation

Dans ces expérimentations nous étudions le comportement du protocole CorDis en fonction de la dynamicité des systèmes. La dynamicité est une propriété importante des systèmes P2P. Elle est liée au fait que les pairs peuvent rejoindre ou quitter un système à tout moment. Ce phénomène est également appelé l'effet de *churn* (ou d'attrition). Nous considérons des systèmes de 500 pairs utilisant 93 ontologies différentes. Par conséquent la diversité $\mathcal{H}_{Div}$ vaut $0,09$. Encore une fois la connectivité est fixée à 4. Les paramètres utilisés sont présentés dans le tableau 6.6.

Paramètre	Valeur(s)
Nombre de pairs	500
Nombre d'ontologies	93
Connectivité	4

TABLE 6.6 – *Paramètres considérés pour étudier les performances de* CorDis *dans différentes situations de dynamicité.*

Afin de simuler la dynamicité, nous faisons en sorte qu'un certain nombre de pairs rejoignent et quittent le système à chaque cycle. Le nombre de pairs entrant dans le système et le nombre de pairs sortant sont égaux : à tout moment le système contient le même nombre de pairs (500). Dans [SGG03], les auteurs soulignent que dans les systèmes de partage de données (comme Gnutella et Napster) la moitié des participants restent dans le système plus d'une heure. À partir de cette analyse, nous étudions le comportement de CorDis lorsque la durée de session moyenne varie entre 1 et 30 minutes : il s'agit très clairement de cas critiques. La durée de session d'un pair correspond au temps durant lequel il reste dans le système. Les pairs qui rejoignent le système utilisent des ontologies qui sont potentiellement déjà utilisées par des pairs déjà présents dans le système. Nous considérons qu'un cycle correspond à une période de 5 secondes (c.-à-d. 12 cycles par minutes). Nos expérimentations consistent à observer les valeurs d'hétérogénéité d'un système pendant 750 cycles (c.-à-d. $62,5$ minutes) alors que CorDis est en cours d'exécution.

Si la durée de session est fixée à m minutes (c.-à-d. $(m \times 12)$ cycles), cela signifie que $\frac{|\mathcal{P}|}{m \times 12}$ pairs rejoignent/quittent le système à chaque cycle. Dans ce cas, le taux de churn vaut $(\frac{100}{m \times 12})\%$. Dans ce contexte, un taux de churn supérieur à 1% représente un cas critique et peu probable dans lequel les pairs restent dans le système en moyenne seulement 8 minutes. Ce délai ne semble pas suffisant pour partager ou obtenir des données. Les différentes configurations que nous étudions sont présentées dans le tableau 6.7.

Résultats et observations

Les figures 6.9 et 6.10 présentent l'évolution des valeurs d'hétérogénéité $\mathcal{H}_{Disp}$ et $\mathcal{H}_{DispAPMoy}$ au cours du temps. Le cas où le taux de churn vaut 0 sert de référence. Il faut noter que la dynamicité ne peut que rendre

Durée de session (minute)	(cycle)	Taux de churn (%)
1	12	8,33
10	120	0,83
30	360	0,27
∞	∞	0 (pas de dynamicité)

TABLE 6.7 – *Configurations considérées pour l'étude de* CORDIS *dans des systèmes dynamiques.*

plus difficile la diminution de l'hétérogénéité car elle peut empêcher les pairs de partager toutes leurs correspondances avant de quitter le système, et parce que certaines correspondances peuvent être perdues si un pair quitte le système sans les avoir partager.

Les résultats montrent que la diminution de l'hétérogénéité n'est quasiment pas impactée lorsque les sessions durent entre 10 et 30 minutes. Lorsque la durée de session moyenne n'est que d'une minute, l'hétérogénéité est moins réduite. Cela est dû au fait que les pairs n'ont pas le temps d'apprendre de correspondances impliquant leur ontologie. L'hétérogénéité $\mathcal{H}_{Disp}$ est réduite de 0,94 à 0,85, et $\mathcal{H}_{DispAPMoy}$ de 0,94 à 0,83. Dans ce cas l'hétérogénéité ne converge pas : elle continue d'osciller car certaines correspondances peuvent avoir complètement disparu du système à un moment donné, puis être réintégrées.

Pour conclure nous pouvons dire que CORDIS est adapté aux systèmes P2P dynamiques : même si la durée de session moyenne est très courte (une minute) l'hétérogénéité est réduite.

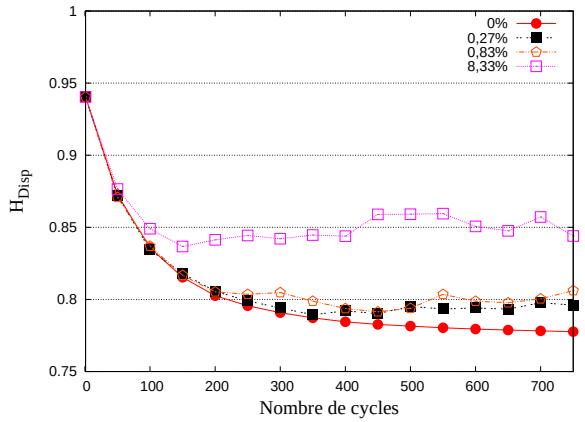


FIGURE 6.9 – *Évolution de* $\mathcal{H}_{Disp}$ *dans des systèmes dynamiques de* 500 *pairs utilisant* 93 *ontologies.*

6.6 DISCUSSIONS

Dans ce chapitre nous avons présenté le protocole CORDIS qui permet de réduire l'hétérogénéité sémantique liée aux disparités entre pairs. Ce travail a été présenté à la conférence Globe en 2011 [CCL11a]. Nous avons aussi proposé un certain nombre de stratégies que peuvent utiliser les pairs dans le cas où ils apprennent une nouvelle correspondance incohé-

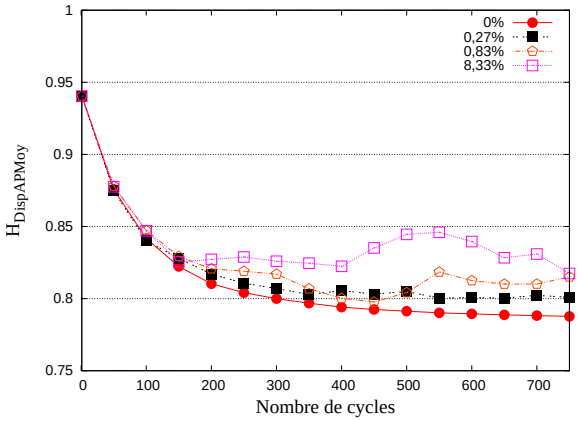


FIGURE 6.10 – *Évolution de $\mathcal{H}_{DapAPMoy}$ dans des systèmes dynamiques de* 500 *pairs utilisant* 93 *ontologies.*

rente avec leurs ontologies. Cette partie a été acceptée dans la revue Transactions on Large-Scale Data- and Knowledge-Centered Systems (TLDKS) en 2012 [CCL12c].

Dans l'article publié à Globe 2011, nous avons utilisé un autre jeu de données : les ontologies et les alignements sont issus de la collection OntoFarm [vSB+05]. Cette collection est utilisée dans la campagne d'évaluation de méthodes d'alignement d'ontologies OAEI (Ontology Alignment Evaluation Initiative [4]). Elle contient seulement 15 ontologies. C'est pour cette raison que nous avons considéré un autre ensemble d'ontologies pour confirmer les résultats obtenus. Nous ne présentons pas les résultats obtenu avec la collection OntoFarm car ils sont moins convaiquants (du fait du nombre d'ontologies) et les observations sont similaires.

La partie relative à la présence d'incohérence n'est pas réellement évaluée puisque nous n'étudions pas et ne comparons pas les différentes stratégies proposées. Hormis le fait que ce type d'expérimentation est long à mettre en place, nous sommes confronté au problème lié au manque de données. En effet nous ne possédons pas d'ensemble de correspondances dont certaines sont correctes pour tous les pairs, certaines sont sources de conflit pour tous les pairs, et certaines sont correctes pour certains pairs et sources de conflit pour d'autres. Nous pourrions introduire ce type de correspondances dans l'ensemble que nous possédons déjà mais cela n'est pas trivial. En effet, dans quelle proportion doit-on introduire des correspondances "douteuses" ? Combien doivent impliquer des concepts qui sont contenus dans des correspondances correctes ?

Dans l'ensemble des expérimentations que nous avons menées, nous avons utilisé les mesures que nous avons définies dans le chapitre 5 (page 69). Elle nous permettent d'évaluer notre protocole indépendamment d'un jeu de requêtes, et sans le comparer à d'autres méthodes. Il serait néanmoins intéressant de comparer notre approche à d'autres approches qui visent également à réduire l'hétérogénéité liée aux disparités entre pairs, mais à notre connaissance il n'existe pas de telles méthodes. Nous pourrions éventuellement comparer la version actuelle de CorDis avec une autre version utilisant un autre mode de dissémina-

4. http ://oaei.ontologymatching.org/

tion des correspondances (par ex. inondation du système avec toutes les correspondances). Dans ce cas, nous verrions s'il est préférable d'utiliser un protocole épidémique pour disséminer les correspondances, ou s'il vaut mieux utiliser une autre approche. Cette étude fait partie de nos perspectives de travail.

Nous pensons qu'il serait également intéressant d'optimiser le protocole afin de réduire le volume de correspondances qui transitent sur le réseau. Il est sans doute envisageable de mettre en place un mécanisme permettant de ne pas envoyer plusieurs fois la même correspondance au même pair. Du point de vue de l'hétérogénéité, cela ne devrait pas être pénalisant. Il faut évidemment étudier quels sont les coûts d'un tel mécanisme (par ex. les coûts liés au stockage et au maintient d'un historique), et quels sont les bénéfices.

Par ailleurs nous envisageons d'étudier le cas où certains pairs du système sont *malveillants*, c'est-à-dire qu'ils transmettent volontairement des correspondances erronées. Dans ce cas, nous envisageons de proposer un mécanisme permettant, de manière décentralisée, d'identifier de tels pairs et de faire en sorte de les exclure du système. Encore une fois des expérimentations devront être menées pour tester l'efficacité de ce mécanisme et également tester sa tolérance : c'est-à-dire déterminer jusqu'à quelle proportion de pairs malveillants il reste efficace.

7 Diminution de l'hétérogénéité liée à la topologie du système

Dans ce chapitre nous présentons le protocole GoOD-TA qui a pour objectif de réduire l'hétérogénéité sémantique liée à la topologie du système. GoOD-TA ré-organise le système de manière à rapprocher les pairs proches sémantiquement, c'est-à-dire ceux qui utilisent une même ontologie, ou ceux pour lesquels il existe de nombreuses correspondances entre leurs ontologies. Nous proposons deux versions de ce protocole. La première est assez simple et a l'avantage de considérer peu de données : les coûts en espace de stockage et en trafic réseau sont faibles. La seconde version est plus complexe mais permet de gérer des cas d'hétérogénéité plus difficiles. Nous proposons des mécanismes permettant de gérer la dynamicité du système (c.-à-d. le fait que des pairs peuvent rejoindre ou quitter le système à tout moment), de gérer l'évolution des connaissances sémantiques des pairs : changement d'ontologie, évolution de leur ontologies, découverte de nouvelles correspondances, et de réduire de manière importante le trafic réseau généré par le protocole. Nous avons mené des expérimentations pour vérifier que GoOD-TA permet effectivement de réduire de manière importante certaines facettes de l'hétérogénéité. Les expérimentations montrent également qu'il est efficace dans des systèmes dynamiques, et dans des systèmes dans lesquels les pairs peuvent être amenés à changer d'ontologie. Nous portons également une attention particulière au volume de données qui transitent sur le réseau. Ces travaux ont été présentés à la conférence P2P Computing en 2012 [CCL12a].

7.1 Problématique et objectifs

Dans ce chapitre, nous considérons le même contexte de travail que dans le chapitre précédent : différents pairs interagissent au sein d'un système P2P, et utilisent des ontologies (une chacun) pour représenter leurs données. Dans la section 5.3 (page 75), nous avons vu que l'hétérogénéité sémantique dépend en partie de l'organisation des pairs du système. Cette facette de l'hétérogénéité est un frein important à l'interopérabilité, et plus précisément à la recherche d'information. En effet, lorsqu'un pair émet une requête, il la transmet à un certain nombre de pairs du système. Typiquement, la requête est envoyée à ses voisins, qui eux-mêmes la transmettent à leurs voisins, etc. La requête est donc envoyée autour du pair initiateur dans un certain "rayon" (aussi appelé TTL). Le voisinage d'un pair représente un ensemble d'interlocuteurs privilégiés. Il est donc crucial que celui-ci soit composé de pairs capables de traiter ses requêtes. Nous nous focalisons sur le fait qu'une requête peut être incomprise lors de la réception car les deux pairs (l'émetteur et le récepteur) n'utilisent pas la même ontologie. La présence de correspondances entre les ontologies des pairs peut permettre de traduire (au moins partiellement) les requêtes. Néanmoins il semble préférable que si des pairs utilisent la même ontologie que l'émetteur, ceux-ci soient en position de la recevoir, c.-à-d. qu'ils se trouvent dans son voisinage. La problématique que nous abordons dans ce chapitre est la suivante : *Comment réduire l'hétérogénéité sémantique liée à la topologie d'un système P2P non-structuré ?*

Nous avons pour objectif de proposer une solution permettant de réduire l'hétérogénéité sémantique liée à la topologie en respectant un certain nombre de contraintes :

- elle doit être adaptée à la dynamicité des systèmes P2P, c'est-à-dire qu'elle doit permettre aux pairs de rejoindre ou quitter le système ;
- elle doit supporter l'évolution des connaissances sémantiques des pairs : le fait qu'un pair change d'ontologie ou découvre de nouvelles correspondances doit être considéré ;
- elle ne doit pas générer un trafic réseau déraisonnable.

Pour répondre à la problématique formulée ci-dessus, nous proposons le protocole GoOD-TA [1].

Ce chapitre est organisé comme suit. Dans la section 7.2 nous présentons le protocole GoOD-TA qui intègre des mécanismes pour gérer la dynamicité des systèmes, l'évolution des connaissances sémantiques des pairs et pour limiter le trafic réseau. Ensuite, la section 7.3 présente une analyse théorique des coûts engendrés par GoOD-TA en termes de volume de stockage et de trafic réseau. Enfin la section 7.4 présente les résultats d'expérimentations menées avec un ensemble d'ontologies utilisées dans domaine biomédical.

1. GoOD-TA : **Go**ssip-based **O**ntology-**D**riven **To**pology **A**daptation.

7.2 Le protocole GoOD-TA

7.2.1 Principe général

Dans un système P2P non-structuré, chaque pair p maintient une vue partielle du système. Celle-ci est notée $view(p)$, et sa taille est limitée à v_{max} entrées. L'objectif du protocole GoOD-TA est d'organiser le système de manière à ce que la vue de chaque pair contienne des pairs dont il est proche sémantiquement. Les n pairs les plus proches de p contenus dans sa vue sont appelés ses voisins. Cet ensemble est noté $\mathcal{N}_p$. GoOD-TA est un protocole épidémique dans lequel les pairs échangent des informations au sujet d'autres pairs. Nous appelons ces informations des descripteurs. Le descripteur d'un pair p', noté λ', contient au moins les informations nécessaires pour le contacter. Chaque pair du système fournit son propre descripteur. Le descripteur d'un pair reflète dans une certaine mesure ses connaissances de manière synthétique. Nous faisons l'hypothèse que les pairs sont honnêtes et qu'ils fournissent un descripteur correct. Comme dans tous les protocoles épidémiques (définis au sens de [KvS07]), chaque pair p doit réaliser trois tâches : la sélection d'un pair, la sélection de données et le traitement des données.

Sélection d'un pair

Lorsque le pair p souhaite initier un échange avec un autre pair, il invoque un service de sélection de pair. Celui-ci consiste à sélectionner un pair aléatoire p' dans sa vue locale $view(p)$. De cette manière, p connaît le descripteur de p'. Si la vue de p ne contient pas suffisamment de pairs pour découvrir de nouveaux voisins, un service d'échantillonnage de pairs peut être invoqué pour rafraîchir ou réinitialiser sa vue [JGKvS04].

Sélection de données

Lorsque le pair p doit envoyer des données au pair p', il trie les descripteurs de sa vue par rapport à leurs proximités avec p'. De cette manière p envoie les meilleurs (c.-à-d. les plus proches) descripteurs à p'. Trier la vue par rapport à p' permet de réduire le temps de convergence du protocole. À chaque échange, seulement m_{max} descripteurs sont considérés afin de ne pas surcharger le réseau.

Traitement des données

Lorsque le pair p reçoit un ensemble de descripteurs du pair p', il doit fusionner sa vue avec les descripteurs reçus. Les n pairs de la vue les plus proches (c.-à-d. ceux dont la similarité à p est la plus grande) deviennent ses voisins. Un descripteur n'est pas ajouté à la vue s'il est déjà présent. Si l'espace de stockage de p est limité, seules v_{max} descripteurs sont conservés : les n descripteurs les plus proches et $(v_{max} - n)$ autres descripteurs sélectionnés aléatoirement.

Afin de trier les descripteurs contenus dans la vue d'un pair, nous utilisons la notion de proximité entre descripteurs.

Définition 7.1 Une fonction de proximité $prox_\lambda(\lambda')$ retourne une valeur comprise dans $[0,1]$ qui correspond à la proximité d'un descripteur λ' par rapport à un autre descripteur λ.

Si $prox_\lambda(\lambda')$ vaut 1, cela signifie que le pair p' comprend tous les concepts de l'ontologie de p. Une fonction de proximité vérifie la propriété suivante : $\lambda = \lambda' \Rightarrow prox_\lambda(\lambda') = 1$. L'inverse n'est pas nécessairement vrai : la proximité peut être égale à 1 alors que les descripteurs sont différents.

Jusqu'à présent, nous n'avons pas explicité ce qu'est un descripteur et comment la fonction de proximité peut être définie. Nous proposons deux approches. La première est relativement simple et requiert très peu d'information. La seconde est plus fine mais nécessite davantage d'information. Rappelons que le descripteur d'un pair est défini par le pair lui même (sauf dans un cas, qui est décrit dans la section 7.2.4, page 126). Ce dernier est en charge de modifier son descripteur lorsque ses connaissances changent. Nous présentons les deux approches dans les sections suivantes.

7.2.1.1 *Approche basique*

Dans cette approche le descripteur d'un pair p, noté λ est défini par l'identifiant de p (noté *id*) et par l'identifiant de l'ontologie utilisée par p, c.-à-d. l'identifiant de $\mu(p)$. Ce dernier est noté *onto_id*. Le tableau 7.1 présente un exemple de la vue d'un pair.

	id	*onto_id*
λ_2	$id(p_2)$	$uri(o_1)$
λ_9	$id(p_9)$	$uri(o_2)$
λ_3	$id(p_3)$	$uri(o_3)$
λ_7	$id(p_7)$	$uri(o_2)$

TABLE 7.1 – *Descripteurs de pairs dans une vue (version basique).*

Nous définissons la fonction de proximité par :

$$prox_\lambda(\lambda') = \begin{cases} 1 & \text{si } \lambda.onto\_id = \lambda'.onto\_id \\ 0 & \text{sinon} \end{cases} \tag{7.1}$$

Cette fonction est symétrique. En effet, on a : $prox_\lambda(\lambda') = prox_{\lambda'}(\lambda)$.

Exemple 7.1 Étant donné un pair p_4 utilisant l'ontologie o_2 et ayant la vue présentée dans le tableau 7.1, nous trouvons que : $prox_{\lambda_4}(\lambda_2) = 0$, $prox_{\lambda_4}(\lambda_9) = 1$, $prox_{\lambda_4}(\lambda_3) = 0$, et $prox_{\lambda_4}(\lambda_7) = 1$. On peut alors dire que les pairs les plus pertinents pour p_4 sont p_7 et p_9.

Cette approche permet de trier les descripteurs en fonction des ontologies utilisées par les pairs. Ainsi le protocole GoOD-TA permet de rapprocher les pairs qui utilisent la même ontologie. L'avantage de cette approche est qu'elle requiert très peu d'information. Mais elle pourrait être inefficace lorsque certaines ontologies sont partagées par très peu de pairs. En effet il serait difficile pour eux de se rencontrer. De plus la définition de la fonction de proximité ne permet pas de distinguer le cas où

deux ontologies ont 99% de leurs concepts en commun, du cas où seulement 5% d'entre eux sont en commun. En effet elle ne considère pas les correspondances connues par les pairs. Pour palier à cette faiblesse, nous proposons une approche plus fine.

7.2.1.2 *Approche raffinée*

Comme la version basique considère seulement les ontologies utilisées par les pairs, elle a le désavantage de définir la fonction proximité de manière manichéenne. Pour palier à cela, nous proposons une autre approche qui considère les correspondances connues par les pairs. Même si cela permettrait de calculer précisément la disparité entre pairs, il est irréaliste d'inclure l'ensemble des correspondances connues dans le descripteur des pairs. Nous proposons donc de ne considérer que le nombre de correspondances connues par le pair entre son ontologie et d'autres ontologies. Le descripteur d'un pair p est alors défini par (i) l'identifiant du pair : id, (ii) l'identifiant de l'ontologie utilisée par le pair : $onto\_id$, (iii) le nombre de concepts contenus dans l'ontologie : $onto\_size$, (iv) un ensemble $corr\_info$ de triplets $\langle onto\_id, onto\_id', nb\rangle$ où nb est le nombre de correspondances que p connaît entre les ontologies o et o' (c'est-à-dire un sous-ensemble des correspondances de $\kappa(p)$). Dans ce chapitre nous ne considérons que les correspondances d'équivalence ($\equiv$), mais la solution proposée pourrait être généralisée à d'autres types de correspondances. Chaque triplet $\langle onto\_id, onto\_id', nb\rangle$ concerne p : p utilise soit o soit o'. Le tableau 7.2 présente un exemple de la vue d'un pair. La deuxième ligne montre que le pair p_9 utilise l'ontologie o_2 (qui contient 95 concepts) et qu'il connait 72 correspondances entre o_2 et o_1, et 36 entre o_2 et o_4.

	id	$onto\_id$	$onto\_size$	$corr\_info$
λ_2	$id(p_2)$	$uri(o_1)$	110	$\{\langle uri(o_1), uri(o_2), 85\rangle\}$
λ_9	$id(p_9)$	$uri(o_2)$	95	$\{\langle uri(o_2), uri(o_1), 72\rangle,$ $\langle uri(o_2), uri(o_4), 36\rangle\}$
λ_3	$id(p_3)$	$uri(o_3)$	1417	$\emptyset$
λ_7	$id(p_7)$	$uri(o_2)$	95	$\{\langle uri(o_2), uri(o_3), 58\rangle\}$

TABLE 7.2 – *Descripteurs de pairs dans une vue (version raffinée).*

En définissant les descripteurs de cette manière, nous voulons que la proximité de λ' par rapport à λ reflètent à quel point p' est capable de comprendre les concepts de l'ontologie de p. La fonction de proximité est définie par :

$$prox_\lambda(\lambda') = \begin{cases} 1 & \text{si } \lambda.onto\_id = \lambda'.onto\_id \\ \frac{nb}{\lambda.onto\_size} & \text{si } \exists\langle \lambda.onto\_id, \lambda'.onto\_id, nb\rangle \in \lambda'.corr \\ 0 & \text{sinon} \end{cases} \quad (7.2)$$

Cette fonction ne satisfait pas la propriété de symétrie : le fait que p connaisse nb correspondances entre $\mu(p')$ et $\mu(p)$ n'implique pas que p' connaisse également ces correspondances.

Exemple 7.2 Pour le pair p_4 utilisant o_2 (qui contient 95 concepts) et ayant la vue décrite dans le tableau 7.2, nous trouvons que :

- $prox_{\lambda_4}(\lambda_2) = \frac{85}{95}$ car p_2 connait 85 correspondances entre o_1 et o_2 ;
- $prox_{\lambda_4}(\lambda_9) = 1$ car p_9 utilise également o_2 ;
- $prox_{\lambda_4}(\lambda_3) = 0$ car p_3 n'a aucune correspondance entre o_3 et o_1 ;
- etc.

7.2.2 Limitation du trafic réseau

Le protocole GoOD-TA génère un certain trafic réseau. Si la fréquence des échanges est élevée, le système peut devenir surchargé. Une solution naïve consiste à réduire la fréquence des échanges. Elle permet de brider le trafic réseau, mais elle ralentit la reconfiguration du système. Nous proposons un mécanisme pour limiter le trafic. Nous considérons qu'un pair p (dont le descripteur est λ) peut éviter d'initier des échanges si la proximité avec chacun de ses voisins est maximale :

$$\forall p' \in \mathcal{N}_p : prox_\lambda(\lambda') = 1 \tag{7.3}$$

Lorsque cette condition est vérifiée pour p, ce dernier modifie la fréquence de ses échanges. Typiquement si p ne souhaite plus exécuter le processus actif, il fixe le délai entre deux échanges θ à ∞. Si p veut réduire la fréquence des échanges par deux, il lui suffit de changer le délai θ comme suit : $\theta \leftarrow 2 \times \theta$. Un pair qui vérifie la condition continue à participer aux échanges au travers du thread passif. Cela garantit que les autres pairs peuvent continuer d'apprendre de nouveaux descripteurs.

Cette optimisation est possible parce que la fonction de proximité est normalisée dans $[0, 1]$. Lorsque la condition (7.3) est vérifiée, la probabilité de trouver un meilleur voisinage est égale à 0. Ce raisonnement n'est pas possible si la fonction de proximité renvoie une valeur dans $\mathbb{R}$. Il est toujours possible d'utiliser un seuil dans (7.3), mais dans ce cas, il n'y a pas de garantie que les voisins courants soient les plus proches de p. Notons que si tous les pairs vérifient la condition (7.3), le protocole ne génère plus aucun trafic réseau.

7.2.3 Gestion de l'évolution des connaissances des pairs

Dans cette section, nous nous intéressons au fait que le descripteur d'un pair p peut changer. Cela peut arriver car :

- p décide d'utiliser une autre ontologie (une nouvelle),
- p fait évoluer son ontologie,
- p découvre de nouvelles correspondances.

Pour supporter ces évolutions, nous ajoutons un **numéro de version** v à chaque descripteur. Une horloge locale peut être utilisée pour initialiser le numéro de version. Par exemple, ce dernier peut être un nombre entier représentant le nombre de secondes écoulées depuis le 1er Janvier 1970. Le numéro de version est mis à jour lorsque le descripteur du pair change. Le numéro de version d'un descripteur est utilisé pour comparer deux versions du descripteur d'un même pair : il n'est jamais utilisé pour comparer les descripteurs de deux pairs différents. Lorsqu'un pair p reçoit un ensemble de descripteurs (c.-à-d. lors du traitement des données), il doit vérifier pour chaque descripteur s'il existe une autre version du même descripteur dans sa vue. Si c'est le cas, il doit conserver la version la plus récente. Nous illustrons la solution proposée sur deux scénarios.

Scénario 1 Soit un pair p_2 utilisant une ontologie o_2 de 95 concepts, et ayant 50 correspondances entre o_2 et o_1. Si nous considérons la version raffinée de GoOD-TA, le descripteur de p_2 est :

$$[id(p_2), uri(o_2), 95, \{\langle o_2, o_1, 50\rangle\}, v_1]$$

Si par la suite, p_2 découvre 45 nouvelles correspondances, son descripteur change et le numéro de version est mis à jour :

$$[id(p_2), uri(o_2), 95, \{\langle o_2, o_1, 95\rangle\}, v_2]$$

Si p_2 réalise qu'il pourrait utiliser l'ontologie o_1 (qui contient 110 concepts) plutôt que o_2 parce que tous les concepts de o_2 ont des équivalents dans o_1, alors il obtient o_1 et commence à l'utiliser. Cette situation est réaliste, en particulier si o_1 et o_2 sont issues d'une ontologie commune. Par conséquent, un nouveau descripteur est créé :

$$[id(p_2), uri(o_1), 110, \{\langle o_1, o_2, 95\rangle\}, v_3]$$

Les correspondances connues sont conservées car elles impliquent toujours l'ontologie de p_2.

Scénario 2 Soit un pair p_2 utilisant une ontologie o_2 de 95 concepts, et ayant connaissance de 55 correspondances entre o_2 et o_1. Son descripteur est :

$$[id(p_2), uri(o_2), 95, \{\langle o_2, o_1, 55\rangle\}, v_1]$$

Le pair p_2 peut alors choisir d'ajouter des concepts à l'ontologie o_2 pour l'adapter à ses besoins. Une nouvelle ontologie o_2' est créée : elle contient tous les concepts de o_2 et 6 autres concepts. L'ontologie o_2' contient alors 101 concepts parmi lesquels 95 sont communs avec o_2. Il y a donc 95 correspondances entre o_2 et o_2'. Par ailleurs le triplet $\langle o_2, o_1, 55\rangle$ n'est plus pertinent pour p_2 car il n'utilise plus o_2. Comme tous les concepts de o_2 sont alignés avec ceux de o_2', le triplet $\langle o_2', o_1, 55\rangle$ doit être considéré. Ainsi le descripteur de p_2 est :

$$[id(p_2), uri(o_2'), 101, \{\langle o_2', o_1, 55\rangle, \langle o_2', o_2, 95\rangle\}, v_2]$$

L'identifiant de o_2' ($uri(o_2')$) doit être unique. Il peut être construit à partir de l'identifiant de o_2, l'identifiant de p_2 et éventuellement la valeur courante de l'horloge locale de p_2. Cela assure que plusieurs pairs peuvent créer différentes ontologies à partir d'une ontologie commune en même temps, et qu'un pair peut créer différentes ontologies à partir d'une ontologie commune à différents moments.

7.2.4 Gestion de la dynamicité du système

Dans cette section nous nous intéressons au fait que les systèmes P2P sont dynamiques : les pairs peuvent rejoindre ou quitter le système à tout moment. Pour prendre cela en compte, nous introduisons une nouvelle notion : le **statut d'un descripteur**. Il peut être représenté par une valeur booléenne (`0` ou `1`). Pendant sa présence dans le système, le pair p communique son descripteur (au travers du protocole classique) avec un statut

égal à 1. Cette valeur reste inchangée tant que p est dans le système. Si p quitte le système (volontairement ou parce qu'il est défaillant), son descripteur devient obsolète parce qu'il n'est plus accessible : son statut doit être fixé à 0. La proximité d'un descripteur λ' par rapport à un autre descripteur λ doit prendre en compte le fait qu'un des deux descripteurs (ou les deux) peut être obsolète. La mesure de proximité $prox_\lambda(\lambda')$ doit vérifier :

$$(\lambda.statut = 0 \ \vee \ \lambda'.statut = 0) \Rightarrow prox_\lambda(\lambda') = 0 \tag{7.4}$$

La redéfinition des mesures de proximité définies dans les sections précédentes pour vérifier cette condition est simple.

Nous distinguons deux cas : lorsque le pair quitte volontairement le système, et lorsqu'il est défaillant. Nous présentons ensuite le cas où un pair rejoint le système.

Départ d'un pair Lorsque le pair p choisit de quitter le système, il change son descripteur : il modifie le statut en le fixant à 0. Dans la version basique du protocole, le pair p devrait modifier son descripteur $[id(p), uri(o), v, 1]$ en $[id(p), uri(o), v', 0]$ où $v' > v$. Avant de quitter le système, p envoie son descripteur à ses voisins. De cette manière, ces derniers apprennent que p est parti, et ils sont alors capables de transmettre cette information aux autres pairs du système en partageant ce descripteur.

Défaillance d'un pair Si le pair p tombe en panne, le descripteur ne peut pas être mis à jour par p lui-même. Quand un autre pair p' remarque que p n'est plus accessible, il modifie le descripteur λ : il modifie simplement la valeur du statut en attribuant la valeur 0, et en incrémentant le numéro de version de un. Les horloges des pairs ne sont a priori pas synchronisées, donc p' n'utilise pas sa propre horloge pour faire la mise à jour. Dans la version basique du protocole, le descripteur $[id(p), uri(o), v, 1]$ serait transformé en $[id(p), uri(o), v', 0]$ où $v' = v + 1$. Notons que c'est la seule situation dans laquelle un pair est autorisé à modifier le descripteur d'un autre pair. Après avoir mis à jour le descripteur de p, le pair p' continue à le propager afin d'informer les autres pairs que p n'est plus accessible. La détection de panne est assurée par un mécanisme indépendant.

Arrivée d'un pair Lorsqu'un pair p rejoint le système, le numéro de version est initialisé en utilisant son horloge locale, et le statut est fixé à 1. Si ce n'est pas la première fois que p rejoint le système, le nouveau numéro de version est plus grand que le numéro de version propagé lors des précédentes sessions.

7.2.5 Algorithmes

Les algorithmes 7 et 8 présentent les instructions contenues dans le thread actif et le thread passif. Contrairement aux protocoles épidémiques classiques (comme CorDis), la sélection des données se fait en fonction du pair à qui elles sont destinées (cf. algorithme 7, lignes 4 et algorithme 8,

ligne 2). La ligne 2 de l'algorithme 7 correspond à la condition présentée dans le mécanisme de limitation du trafic réseau (cf. équation 7.3, page 125).

Algorithme 7 : Thread actif d'un pair p.

```
1 while true do
2     if ∃ p_i ∈ N_p : prox_λ(λ_i) ≠ 1 then
          // Si la condition 7.3 n'est pas vérifiée :
3         p' ← selectPeer() ;
4         entries ← selectEntriesToSend(p') ;
5         sendTo(p', corrs) ;
6         entries' ← receiveFrom(p') ;
7         storeEntries(entries') ;
8     wait(θ) ;              // θ : délai entre deux cycles.
```

Algorithme 8 : Thread passif d'un pair p'

```
1 entries ← receiveFromAny() ;
2 entries' ← selectEntriesToSend(p) ;
3 sendTo(p, entries') ;
4 storeEntries(entries) ;
```

L'algorithme 9 décrit la manière dont un pair p sélectionne les descripteurs à envoyer à un autre pair p'. Nous voyons à la ligne 1 que la vue du pair p est triée en fonction du descripteur de p'. Ainsi les meilleurs descripteurs pour p' sont sélectionnés.

Algorithme 9 : Fonction selectEntriesToSend(p')

```
   // Sélection des entrées par un pair p pour un
      pair p'
   Input : Un pair p' à qui envoyer les entrées.
   Output : ensemble des entrées à envoyer à un pair p'.
   // λ' est le descripteur du pair p'.
1  sortView(view(p), λ') ;
2  entries ← bestEntries(view(p), m_max) ;
3  return entries ;
```

Enfin l'algorithme 10 montre comment l'évolution des descripteurs est prise en compte lors de la réception de descripteurs par un pair (lignes 2 à 5).

7.3 Analyse théorique des coûts

Dans cette section nous étudions les coûts des deux versions du protocole GoOD-TA en termes de stockage local et de trafic réseau. Ces derniers dépendent principalement de la taille des descripteurs des pairs. Nous étudions donc d'abord cet aspect. Nous considérons que :

Algorithme 10 : Fonction storeEntries(*entries*)

```
// Stockage des entrées entries par un pair p
Input : Un ensemble d'entrées d'une vue.
for λ1 ∈ entries do
    if ∄ λ2 ∈ view(p) : λ2.id = λ1.id then
        // λ1 n'est pas déjà dans view(p)
        view(p) ← view(p) ∪ {λ1} ;
    else if λ1.version > λ2.version then
        // λ1 est plus récent que λ2
        view(p) ← view(p) \ {λ2} ∪ {λ1} ;
if |view(p)| > v_max then
    // Si la vue contient trop de descripteurs,
    sortView(view(p), λ) ;
    // on conserve les n meilleurs (pour p)
    B ← bestEntries(view(p), n) ;
    // et v_max − n autres descripteurs.
    R ← selectRandom(view(p) \ B, v_max − n) ;
    view(p) ← B ∪ R ;
```

- la taille d'un entier court est de w octets,
- l'identifiant d'un pair est représenté par un entier long dont la taille est de x octets,
- la taille d'un URI est de y octets.

Comme la taille d'un booléen est de seulement un bit, nous considérons qu'elle est négligeable par rapport aux autres valeurs.

7.3.1 Taille des descripteurs

Dans la version basique du protocole, un descripteur est constitué de l'identifiant d'un pair, d'un URI et d'un numéro de version (le statut est ignoré car il s'agit d'un booléen et sa taille est négligeable). Par conséquent, la taille d'un descripteur est de $[x + y + w]$ octets.

Dans la version raffinée, un descripteur est composé de l'identifiant d'un pair, d'un URI, d'un entier, d'un ensemble $corr\_info$ et d'un numéro de version. Chaque élément de $corr\_info$ est constitué de deux URIs et d'un entier. Sa taille est donc de $[2y + w]$ octets. La taille maximale de $corr\_info$ dépend du nombre d'ontologies utilisées dans le système. En général la taille d'un descripteur est $[x + y + 2w + |corr\_info| \cdot (2y + w)]$ octets. Dans le meilleur des cas, la taille d'un descripteur est de $[x + y + 2w]$ octets (lorsque le pair ne connaît aucune correspondance). Dans le pire des cas, chaque pair connaît un alignement entre son ontologie et chacune des autres ontologies du système. Par conséquent, un pair peut connaître $(|o_s| - 1)$ alignement (où $|o_s|$ est le nombre d'ontologies du système). Dans cette situation extrême, la taille d'un descripteur est $[x + y + 2w + (|o_s| - 1)(2y + w)]$ octets. Comme il n'est pas possible d'estimer la taille d'un descripteur dans un cas réaliste, nous étudions cet aspect expérimentalement dans la section 7.4.5 (page 134). Dans la suite de cette analyse, nous notons φ la taille d'un descripteur.

7.3.2 Stockage local et trafic réseau

Pendant sa présence dans le système, un pair est amené à stocker au plus v_{max} descripteurs. Par conséquent, l'espace maximal occupé par les descripteurs dans la vue est de $[v_{max} \cdot \varphi]$ octets. Chaque pair p initie régulièrement un autre pair p' pour échanger des descripteurs. Un échange (de p à p' ou de p' à p) concerne au plus m_{max} descripteurs. Lorsque tous les pairs initient un échange, $[2 \cdot |\mathcal{P}| \cdot m_{max}]$ descripteurs passent sur le réseau. Le facteur 2 vient du fait que les échanges sont bidirectionnels. En définitive cela représente $[2 \cdot |\mathcal{P}| \cdot m_{max} \cdot \varphi]$ octets à chaque cycle. Cette valeur peut être importante. C'est pour cette raison que nous avons proposé le mécanisme de limitation du trafic réseau (cf. section 7.2.2, page 125). Ce mécanisme est étudié expérimentalement dans la section 7.4.5 (page 134).

7.4 Évaluations

7.4.1 Objectifs

L'objectif de ces expérimentations est d'étudier les performances du protocole GoOD-TA. Premièrement nous voulons vérifier que les deux versions du protocole permettent de réduire l'hétérogénéité sémantique liée à la topologie du système. Deuxièmement nous voulons comparer leurs temps de stabilisation. Troisièmement nous étudions le trafic réseau et le bénéfice du mécanisme de limitation du trafic présenté en section 7.2.2 (page 125). Quatrièmement nous étudions l'évolution de l'hétérogénéité lorsque certains pairs changent leur descripteur. Enfin nous étudions l'impact de la dynamicité sur l'évolution de l'hétérogénéité sémantique.

7.4.2 Matériel, paramétrage de GoOD-TA et méthodologie

7.4.2.1 Ontologies et correspondances

Pour ces expérimentations, nous avons considéré les mêmes ontologies et les mêmes correspondances que dans la section 6.5 (page 104). Nous avons utilisé 149 ontologies OWL et 60944 correspondances d'équivalence entre concepts. Les correspondances relient 93 ontologies entre elles : 56 ontologies sont totalement déconnectées des autres.

Nous avons réparti les ontologies de manière aléatoire dans le système suivant une loi de distribution uniforme. Chaque pair a connaissance de toutes les correspondances qui impliquent l'ontologie qu'il utilise.

7.4.2.2 Métriques

Hétérogénéité sémantique Pour mesurer l'hétérogénéité sémantique, nous utilisons des mesures définies dans la section 5.4 du chapitre 5 (page 77). Nous utilisons la mesure de diversité sémantique, et les mesures qui considèrent la topologie du système :

- $\mathcal{H}_{Div}$ mesure la diversité sémantique d'un système (cf. équation 5.1, page 77) ;

- $\mathcal{H}_{DivAPMoy}$ mesure la diversité moyenne du système en considérant sa topologie, c.-à-d. les liens de voisinage entre les pairs (cf. équation 5.6, page 79) ;
- $\mathcal{H}_{DispAPMoy}$ mesure la disparité moyenne du système en considérant sa topologie (cf. équation 5.8, page 79).

Nous focalisons notre analyse sur ces mesures car GoOD-TA vise à diminuer l'hétérogénéité liée à la topologie. Pour l'évaluation, nous considérons que la valeur du rayon r (utilisé dans les mesures) est fixée à 3. La mesure tenant seulement compte de la disparité entre les pairs ($\mathcal{H}_{Disp}$) ne varie pas. La mesure de diversité $\mathcal{H}_{Div}$ est utilisée pour caractériser la situation du système : plus le système est dans une situation critique du point de vue de l'hétérogénéité, plus sa valeur est élevée. Elle n'évolue pas au cours du temps, même lorsque GoOD-TA est en cours d'exécution. Pour mesurer la disparité entre un pair p et un autre pair p' nous avons utilisé la même mesure que dans la section 6.5 (page 104).

Volume de données Pour mesurer le volume de données stockées par les pairs et le trafic réseau, nous utilisons les mesures définies dans la section 6.5.2.3 (cf. équations 6.8 et 6.9, page 106).

7.4.2.3 *Paramétrage de GoOD-TA*

Le protocole GoOD-TA considère un certain nombre de paramètres : la taille de la vue v_{max} et le nombre maximal de descripteurs échangés à chaque échange m_{max}. Dans ces expérimentations nous avons choisi de limiter la vue à vingt descripteurs (c.-à-d. $v_{max} = 20$) et le nombre de descripteurs échangés à cinq (c.-à-d. $m_{max} = 5$).

7.4.2.4 *Méthodologie d'évaluation*

Nous avons considéré la même méthode que pour les expérimentations du chapitre précédent : section 6.5.2.5, page 108.

7.4.3 Réduction de l'hétérogénéité sémantique en fonction de la diversité sémantique

Paramètres de simulation

Dans cette section nous étudions les performances de GoOD-TA dans différentes situations d'hétérogénéité sémantique. Nous évaluons sa capacité à réduire l'hétérogénéité liée à la topologie du système (mesurable avec $\mathcal{H}_{DivAPMoy}$ et $\mathcal{H}_{DispAPMoy}$) en fonction de la diversité sémantique du système (mesurable avec $\mathcal{H}_{Div}$). Nous faisons donc varier le degré de diversité entre 0 et 1. Comme nous disposons de 149 ontologies, nous considérons des systèmes de 149 pairs. Comme le cas où une seule ontologie est utilisée est trivial (car le système est sémantiquement homogène), nous ne le considérons pas. Nous faisons donc varier le nombre d'ontologies de 16 à 149. Par conséquent l'hétérogénéité $\mathcal{H}_{Div}$ varie de 0,1 à 1. Après initialisation du système (cf. tableau 7.3), le protocole GoOD-TA s'exécute pendant 300 cycles. À chaque cycle, $\mathcal{H}_{DivAPMoy}$ and $\mathcal{H}_{DispAPMoy}$ sont calculées. Pour ces expérimentations, le mécanisme de réduction du trafic est

inactif, et les pairs ne quittent/rejoignent pas le système et ne changent pas leurs descripteurs. Comme nous considérons des systèmes P2P de petite taille (149 pairs), nous fixons le degré de connectivité à 3. Ainsi chaque pair est directement lié à 3 autres pairs. Comme la valeur du rayon r est fixée à 3, chaque pair peut communiquer avec au plus 39 pairs (c.-à-d. $3^3 + 3^2 + 3^1$). Comme nous considérons des systèmes de 149 pairs, cela représente 26% du système.

Paramètre	Valeur(s)
Nombre de pairs	149
Nombre d'ontologies	16, 31, ..., 134, 149
Connectivité n	3

TABLE 7.3 – *Paramètres utilisés pour étudier les performances de GoOD-TA lorsque le nombre d'ontologies varie.*

Résultats et observations

Les résultats de ces expérimentations sont présentés sur les figures 7.1 et 7.2. Elles présentent les valeurs de $\mathcal{H}_{DivAPMoy}$ et de $\mathcal{H}_{DispAPMoy}$ après 300 cycles. À ce stade, ces valeurs n'évoluent plus de manière significative, même si le protocole est toujours en cours d'exécution. Nous pouvons voir que $\mathcal{H}_{DivAPMoy}$ et $\mathcal{H}_{DispAPMoy}$ ont des comportements similaires. Cela est dû au fait que même s'il y a des correspondances entre certaines ontologies, celles-ci sont trop peu nombreuses, et la disparité entre les pairs est donc souvent proche de 1. Avec d'autres données la différence pourrait être plus importante. La référence correspond au cas où GoOD-TA n'est pas en cours d'exécution. Nous pouvons voir que les deux versions du protocole sont efficaces lors que la diversité est faible. Plus la diversité du système augmente et moins le protocole est efficace pour réduire l'hétérogénéité $\mathcal{H}_{DivAPMoy}$ et $\mathcal{H}_{DispAPMoy}$. Dans tous les cas, la version raffinée est plus efficace que la version basique : l'hétérogénéité est davantage réduite. Lorsque tous les pairs utilisent des ontologies différentes ($\mathcal{H}_{Div}$ vaut 1), la version basique du protocole est inefficace, alors que la version raffinée permet de réduire l'hétérogénéité $\mathcal{H}_{DispAPMoy}$.

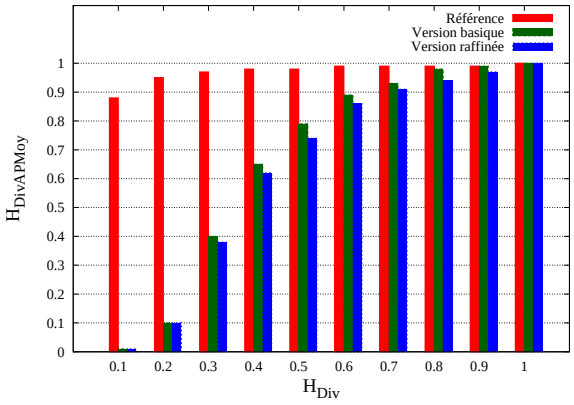


FIGURE 7.1 – *Comparaison des deux versions de GoOD-TA dans différentes situations de diversité. L'hétérogénéité est mesurée avec $\mathcal{H}_{DivAPMoy}$ après 300 cycles.*

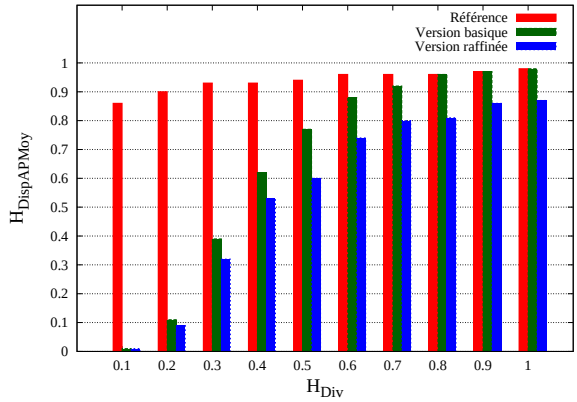


FIGURE 7.2 – *Comparaison des deux versions de GoOD-TA dans différentes situations de diversité. L'hétérogénéité est mesurée avec $\mathcal{H}_{DispAPMoy}$ après 300 cycles.*

7.4.4 Temps de stabilisation

Paramètres de simulation

Dans cette section, nous comparons les temps de stabilisation des deux versions de GoOD-TA. Les paramètres utilisés pour ces expérimentations sont ceux décrits dans la section précédente (cf. tableau 7.3). Soit *min* la valeur minimale de l'hétérogénéité atteinte pendant la simulation (ici elle dure 300 cycles) :

$$min = \min_{t \in [1,300]} \mathcal{H}[t](m) \tag{7.5}$$

où $\mathcal{H}[t](m)$ correspond à la valeur d'hétérogénéité du système à la fin du cycle t. Nous définissons le **temps de stabilisation** st comme le temps minimal après lequel toutes les valeurs d'hétérogénéité sont proches de *min* :

$$st = \min_{t \in [1,300]} t, \text{ tel que } \forall t' \in [st, 300], \mathcal{H}[t'](m) \leqslant min + \varepsilon \tag{7.6}$$

Dans ces expérimentations, nous avons fixé ε à $0,05$.

Résultats et observations

La figure 7.3 présente les temps de converge des deux versions de GoOD-TA. Elle présente seulement le temps de stabilisation pour la mesure $\mathcal{H}_{DivAPMoy}$ mais les observations sont similaires pour $\mathcal{H}_{DispAPMoy}$. La première constatation est que le temps de stabilisation dépend de la diversité sémantique du système. Ensuite nous pouvons remarquer que les courbes correspondant aux deux versions ont la même forme. Lorsque la diversité $\mathcal{H}_{Div}$ est comprise entre 0 et $0,3$ les temps de stabilisation augmentent. Ils diminuent lorsque la diversité est supérieure à $0,4$. Nous pensons que la stabilisation est rapidement obtenue lorsque la diversité est faible car la situation est relativement simple : l'hétérogénéité est rapidement réduite. Elle est également rapidement obtenue lorsque la diversité est très élevée car la situation est trop complexe et par conséquent l'hétérogénéité est très peu réduite. Dans les cas intermédiaires, le temps de stabilisation est plus important car de nombreux descripteurs

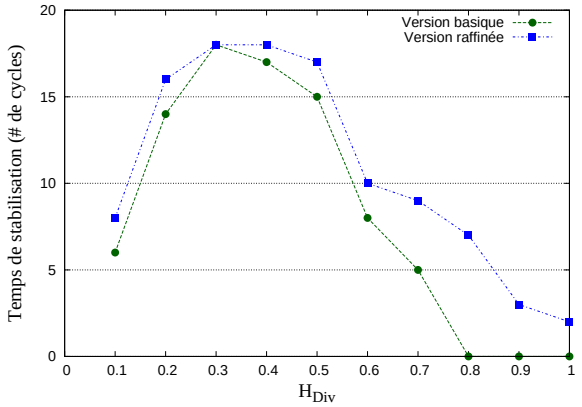


FIGURE 7.3 – *Vitesse de convergence des deux versions de GoOD-TA en fonction de la diversité sémantique.*

doivent être échangés. Le temps de convergence est égal à 0 lorsque l'hétérogénéité $\mathcal{H}_{Div}$ est supérieure à 0,8 parce que dans ce cas l'hétérogénéité $\mathcal{H}_{DispAPMoy}$ n'est quasiment pas réduite (cf. figure 7.1).

7.4.5 Trafic réseau et espace de stockage

Paramètres de simulation

Dans cette section nous comparons les deux versions de GoOD-TA en terme de :

- volume occupé par les descripteurs chez les pairs,
- trafic réseau.

Nous étudions également le bénéfice apporté par le mécanisme de limitation de trafic réseau, et son impact sur la diminution de l'hétérogénéité. Pour mesurer le volume occupé par les descripteurs et le trafic réseau nous utilisons les mesures définies dans le chapitre précédent : équations 6.8 et 6.9 (page 106). Et nous considérons que les types manipulés (entiers longs, entiers courts, URIs, etc.) ont les volumes présentés dans le tableau 6.2 (page 106).

Nous considérons des systèmes de 1000 pairs utilisant 149 ontologies différentes. Ainsi la diversité sémantique $\mathcal{H}_{Div}$ est égale à 0,15. Comme les systèmes considérés contiennent plus de pairs que dans les expérimentations précédentes, nous fixons le degré de connectivité n à 4. Lorsque le mécanisme de limitation du trafic réseau est actif, les pairs vérifiant la condition 7.3 (page 125) cessent d'initier des échanges (θ est fixé à ∞). Après chaque cycle de simulation, les valeurs de $\mathcal{H}_{DivAPMoy}$, $\mathcal{H}_{DispAPMoy}$ et le volume de données transférées sont calculées. Les paramètres de configurations considérés sont présentés dans le tableau 7.4. Nous considérons cinq configurations : la première sert de référence (GoOD-TA n'est pas exécuté), les quatre suivantes correspondent aux versions basique (VB-1 et VB-∞) et raffinées (VR-1 et VR-∞) de GoOD-TA. Le mécanisme de limitation de trafic réseau (LTR) est actif (✓) ou inactif.

	Protocole	LTR
Référence		
VB-1	Version basique	
VB-∞	Version basique	✓
VR-1	Version raffinée	
VR-∞	Version raffinée	✓

TABLE 7.4 – *Configurations considérées pour étudier le trafic réseau généré par GoOD-TA (avec $v_{max} = 20$, $m_{max} = 5$, $n = 4$, et $r = 3$).*

	Volume d'un descripteur		Volume d'une vue	
	VB	VR	VB	VR
Meilleur cas attendu	74	78	1480	1560
Pire cas attendu	74	15470	1480	309400
Observé	74	150	1480	3000

TABLE 7.5 – *Volume (en Ko) d'un descripteur et volume (en Ko) occupé par les descripteurs d'une vue pour chacune des versions de GoOD-TA.*

Résultats et observations

Espace de stockage Le tableau 7.5 présente le volume d'un descripteur (colonne "Volume d'un descripteur") et le volume occupé par l'ensemble des descripteurs d'une vue pour un pair (colonne "Volume d'une vue") dans chacune des versions de GoOD-TA. Les deux premières lignes correspondent aux calculs issus de l'analyse des coûts présentée en section 7.3 (page 128). Évidemment les valeurs obtenues pour la version basique sont celles attendues car la taille d'un descripteur ne dépend pas du nombre de pairs dans le système ou du nombre d'ontologies utilisées dans le système. Par contre dans la version raffinée, nous pouvons voir que la différence entre le meilleur cas et le pire cas est importante : entre 78 Ko et 15,47 Mo. Nous voyons que sur nos expérimentations, le volume d'un descripteur est plus proche du meilleur cas. Cela montre que sur un cas réel, l'ensemble $corr\_info$ contenu dans les descripteurs contient peu d'éléments. En définitive, nous pouvons dire que les deux versions du protocole occupent très peu d'espace de stockage chez les pairs : 1,4 Mo pour la version basique et 3 Mo pour la version raffinée.

Trafic réseau Comme nous pouvons le voir sur la figure 7.4, la version basique de GoOD-TA génère moins de trafic réseau que la version raffinée. Cela est évidemment dû à la taille des descripteurs. Nous pouvons également voir que le mécanisme de limitation du trafic (LTR) réduit le volume de données transférées. Ce dernier est divisé par 2 après 50 cycles pour la version basique, et après 100 cycles pour la version raffinée. À partir du 175^{e} cycle, le volume de données transférées se stabilise à 120 Ko par cycle pour la version basique au lieu de 750 Ko. Pour la version raffinée, le volume est de 150 Ko au lieu de 1550 à partir du 200^{e} cycle.

La figure 7.5 montre qu'avec la configuration LV-∞ la valeur de $\mathcal{H}_{DivAPMoy}$ converge vers la valeur obtenue avec LV-1. Cela montre que le mécanisme LTR ne nuit pas à la qualité du protocole GoOD-TA (en terme de réduction de l'hétérogénéité). La différence se manifeste au niveau de la vitesse de convergence. LV-1 converge en 200 cycles alors

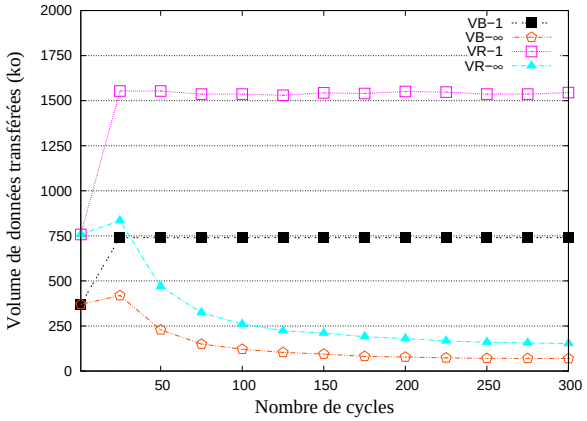


FIGURE 7.4 – *Évolution du volume de données transférées pendant l'exécution des deux versions de GoOD-TA avec ou sans le mécanisme de limitation du trafic réseau.*

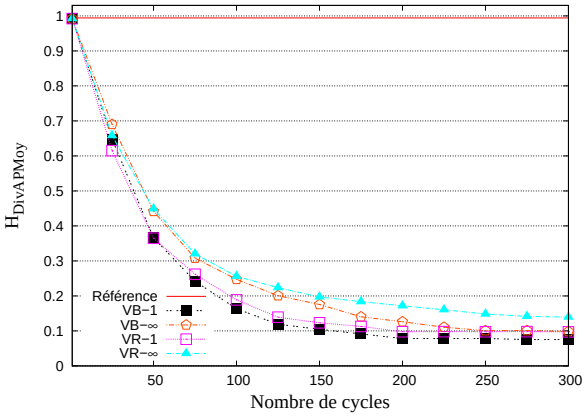


FIGURE 7.5 – *Évolution de $\mathcal{H}_{DivAPMoy}$ pendant l'exécution des deux versions de GoOD-TA avec ou sans le mécanisme de limitation du trafic réseau.*

que LV-∞ converge en 250 cycles. Les observations relatives à la version raffinée sont identiques. Les observations que nous faisons concernent la mesure $\mathcal{H}_{DivAPMoy}$. Celles concernant $\mathcal{H}_{DispAPMoy}$ sont similaires.

Pour conclure, nous pouvons dire que le mécanisme LTR réduit drastiquement le volume de données transférées sans pénaliser la réduction de l'hétérogénéité sémantique.

7.4.6 Évolution des descripteurs des pairs

Paramètres de simulation

Dans cette section nous étudions le protocole GoOD-TA lorsque certains pairs changent leur descripteur. Nous considérons des systèmes de 1000 pairs utilisant 149 ontologies différentes. Le degré de connectivité est fixé à 4. Dans ces expérimentations nous lançons le protocole GoOD-TA pendant 500 cycles. Pendant les 250 premiers cycles, les pairs ne changent pas leur ontologie. Au 250^{e} cycle, un certain nombre de pairs (1, 10 ou 100) changent leur ontologie. Leurs descripteurs sont alors mis à jour. Les nouvelles ontologies choisies par les pairs sont éventuellement déjà utilisées par d'autres pairs du système. Le défi pour ces pairs est de trouver de nouveaux voisins pertinents.

Nous étudions l'impact de ces changements pour les pairs (localement) et pour le système (globalement). L'impact local est mesuré en faisant une moyenne $\mathcal{H}^{r}_{DivAP}(m, p)$ (cf. section 5.4, page 77) pour chaque pair p ayant changé son ontologie. Une fois de plus, le rayon r est fixé à 3. L'impact global est mesuré avec $\mathcal{H}_{DivAPMoy}$. Dans cette section nous considérons uniquement la version basique du protocole. La version raffinée donne des résultats similaires. Nous pensons que la version raffinée devrait être plus réactive lorsque les pairs découvrent de nouvelles correspondances.

Résultats et observations

La figure 7.6 montre que lorsque des pairs changent leur ontologie, ils trouvent rapidement de nouveaux voisins pertinents. La proximité avec leurs nouveaux voisins n'est pas nécessairement la même qu'avec leurs anciens voisins. Cela dépend du nombre de pairs qui utilisent la même ontologie dans le système. La figure 7.7 montre que lorsque plusieurs pairs changent leur ontologie en même temps, l'hétérogénéité globale du système augmente subitement. Le protocole GoOD-TA permet de réduire l'hétérogénéité de nouveau. Pour accélérer la reconfiguration du système, il est possible de faire en sorte que les pairs réinitialisent leur vue comme cela est suggéré dans [JMB09].

7.4.7 Dynamicité du système

Paramètres de simulation

Dans cette section nous étudions les performances de GoOD-TA dans des systèmes dynamiques. Comme dans le chapitre précédent, nous simulons la dynamicité en faisant en sorte que des pairs rejoignent et quittent le système à chaque cycle. Le nombre de pairs à rejoindre le système est

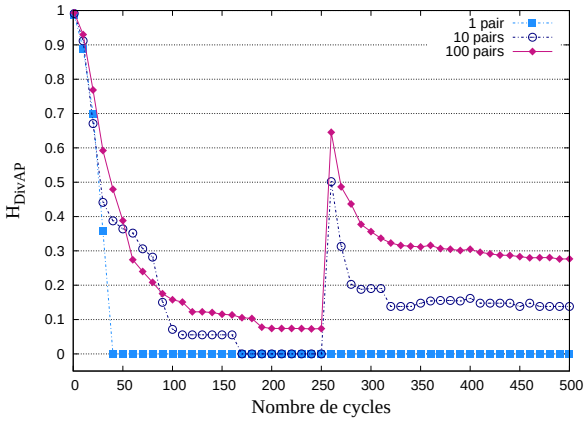


FIGURE 7.6 – *Évolution de l'hétérogénéité $\mathcal{H}_{DivAP}$ autour des pairs ayant changé leur ontologie.*

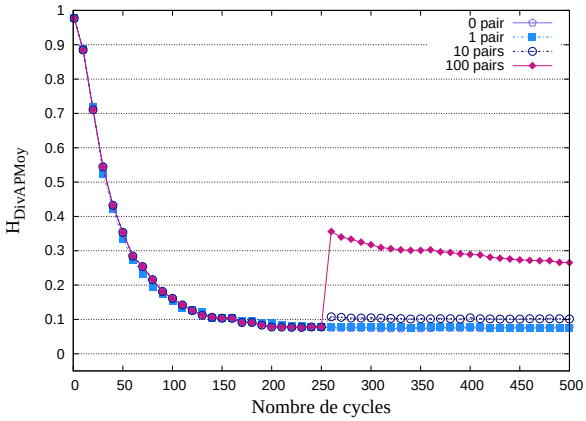


FIGURE 7.7 – *Évolution de $\mathcal{H}_{DivAPMoy}$ pendant l'exécution des deux versions de GoOD-TA. Un certain nombre de pairs changent leur ontologie au 250^e cycle.*

toujours égal au nombre de pairs qui le quittent. De cette manière la taille du système reste constante. Nous considérons que lorsqu'un pair quitte le système, il n'en informe pas ses voisins. Cela correspond à un cas critique car les autres pairs du système vont continuer à propager et/ou à stocker son descripteur jusqu'à ce que ce que l'un d'entre eux découvre qu'il a quitté le système. Dans ces expérimentations nous étudions GoOD-TA lorsque la durée de session moyenne varie entre 1 minute et 120 minutes.

Comme dans les précédentes expérimentations, nous considérons des systèmes de 1000 pairs utilisant 149 ontologies différentes. Le degré de connectivité n est fixé à 4. Les pairs qui rejoignent le système utilisent des ontologies qui sont probablement déjà utilisées par d'autres pairs du système. Nous considérons que la durée d'un cycle est de 5 secondes, c.-à-d. qu'il y a 12 cycles par minute. Nos expérimentations consistent à observer un système durant 300 cycles (c.-à-d. 25 minutes). Nous considérons les configurations présentées dans le tableau 7.6. Si la durée de session est fixée à m minutes (c.-à-d. $(m \times 12)$ cycles), cela signifie que $\frac{|\mathcal{P}|}{m \times 12}$ pairs rejoignent/quittent le système à chaque cycle. Dans ce cas, le taux de churn vaut $(\frac{100}{m \times 12})\%$. Dans ce contexte, un taux de churn supérieur à 1% représente un cas critique et peu probable dans lequel les pairs restent dans le système en moyenne seulement 8 minutes. Ce délai ne semble pas suffisant pour partager ou obtenir des données. Encore une fois nous ne présentons les résultats que pour la mesure $\mathcal{H}_{DivAPMoy}$ car $\mathcal{H}_{DispAPMoy}$ donne des résultats similaires.

Durée de session		Taux de churn
(minute)	(cycle)	(%)
1	12	8,33
5	60	1,67
15	180	0,54
30	360	0,27
60	720	0,14
∞	∞	0 (pas de dynamicité)

TABLE 7.6 – *Configurations considérées pour étudier GoOD-TA dans des systèmes dynamiques.*

Résultats et observations

La figure 7.8 présente l'hétérogénéité sémantique obtenue avec les deux versions de GoOD-TA pour différentes durées de session. Les mesures sont prises lorsque les valeurs d'hétérogénéité n'évoluent plus de manière significative. La référence correspond au cas où aucun protocole n'est exécuté. La figure 7.8 montre que les courbes correspondant aux deux versions coïncident. Cela signifie que les deux versions du protocole supportent la dynamicité de la même manière. La figure 7.8 montre également qu'avec les deux versions de GoOD-TA l'hétérogénéité est réduite même si les pairs restent seulement une minute dans le système. Lorsque la durée de session moyenne devient plus importante, l'hétérogénéité sémantique est davantage réduite. Nous pouvons voir que lorsque la durée de session est de 60 minutes, l'hétérogénéité est réduite de 0,96 à 0,2. Dans [SGG03] les auteurs remarquent que dans les systèmes de partage

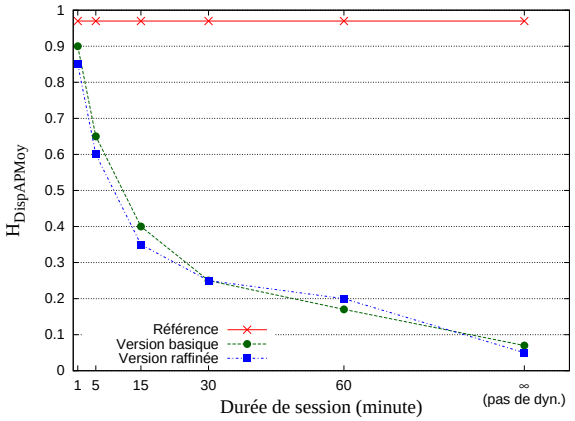


FIGURE 7.8 – *Hétérogénéité sémantique $\mathcal{H}_{DivAPMoy}$ obtenue après 300 cycles en fonction de la durée de session moyenne (c.-à-d. du taux de churn).*

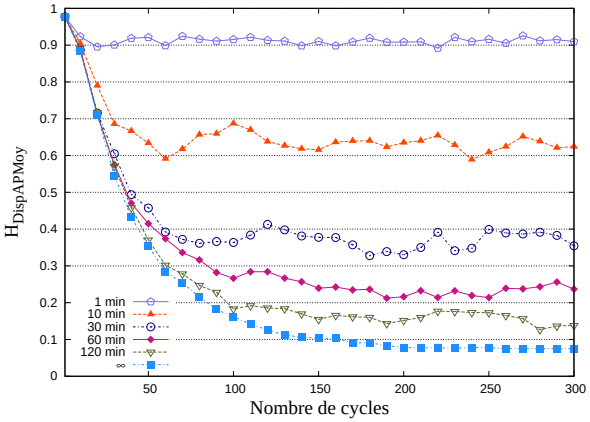


FIGURE 7.9 – *Évolution de $\mathcal{H}_{DispAPMoy}$ pendant l'exécution de la version basique de GoOD-TA en fonction de la durée de session moyenne (c.-à-d. du taux de churn).*

de données tels que Gnutella et Napster, la moitié des participants restent plus d'une heure dans le système. En nous basant sur cette analyse, nous pouvons dire que GoOD-TA est adapté à la dynamicité de ce type de systèmes.

La figure 7.9 présente l'évolution de $\mathcal{H}_{DivAPMoy}$ pour des systèmes dont le taux de churn varie. Nous nous concentrons sur la version basique du protocole. La figure 7.9 montre que plus le taux de churn est élevé, moins l'hétérogénéité est réduite. Lorsque le taux de churn est important, le système est instable : la valeur de l'hétérogénéité oscille.

Pour conclure nous pouvons dire que les deux versions du protocole GoOD-TA sont adaptées aux systèmes P2P dynamiques. L'hétérogénéité est réduite (faiblement) même lorsque les pairs restent seulement une minute dans le système en moyenne. Lorsqu'ils restent plus d'une heure, l'hétérogénéité est réduite de manière importante.

7.5 Discussions

Dans ce chapitre nous avons présenté le protocole GoOD-TA qui permet de réduire l'hétérogénéité sémantique liée à la topologie des systèmes P2P. Ce travail a été présenté à la conférence P2P Computing en

2012 [CCL12a]. Nous avons montré expérimentalement que les deux versions du protocole GoOD-TA réduisent l'hétérogénéité et que les différents mécanismes permettent de limiter le trafic réseau, de gérer l'évolution des connaissances des pairs, et de supporter la dynamicité des pairs. Comme pour le protocole CorDis, nous ne comparons pas notre approche à des approches existantes car à notre connaissance, il n'existe pas d'algorithme qui réorganise des systèmes P2P non-structurés pour réduire l'hétérogénéité sémantique. Nous pourrions éventuellement étudier des méthodes qui créent des overlays pour améliorer la recherche d'information et adapter nos mesures d'hétérogénéité pour pouvoir comparer les approches.

GoOD-TA est assez proche du protocole T-Man, un protocole épidémique permettant d'organiser un système P2P en une topologie particulière (par exemple un torus) [JB05, JMB09]. T-Man est défini de manière très générale et ne permet pas aux pairs d'évoluer. La principale différence entre les deux protocoles est que GoOD-TA inclut un mécanisme permettant aux pairs d'évoluer, c.-à-d. de changer d'ontologie, de découvrir de nouvelles correspondances, etc. De plus, GoOD-TA est appliqué dans un cadre précis (les systèmes P2P sémantiques), ce qui n'est pas le cas de T-Man.

Dans les expérimentations, nous avons considéré des systèmes contenant relativement peu de pairs car nous disposons de seulement 149 ontologies. Le fait de considérer des systèmes de 10000 ou 100000 pairs ne fait que simplifier le problème lié à l'hétérogénéité sémantique. Pour un nombre d'ontologies donné, plus les pairs sont nombreux dans le système, et plus il est simple pour eux de se rapprocher des pairs dont ils sont proches sémantiquement. En clair, plus le système est grand, plus il est facile de le ré-organiser.

Lorsque de nombreuses ontologies sont utilisées, le protocole GoOD-TA a tendance à partitionner le système. Cela n'est pas nécessairement problématique car il est toujours possible d'invoquer un service d'échantillonnage de pairs pour accéder aux pairs d'autres partitions. Néanmoins nous pensons que réduire l'hétérogénéité sémantique (celle liée à la topologie) à 0 n'est pas toujours préférable. Nous pensons que la diversité peut amener les pairs à découvrir des connaissances qu'ils n'avaient pas auparavant. En particulier cela pourrait permettre aux pairs de faire évoluer leurs ontologies. Ce point nécessite une analyse et des expérimentations poussées.

Dans cette partie de notre travail, nous avons fait l'hypothèse que les pairs sont honnêtes et qu'ils ne trichent pas sur la définition de leurs descripteurs. Il serait intéressant d'étudier le cas où certains pairs propagent de faux descripteurs afin de devenir voisins de nombreux pairs et de nuire au système.

8 Recherche d'information sémantique en environnement hétérogène

Dans ce chapitre nous présentons l'algorithme DiQuESH. Il assure le routage et le traitement de requêtes top-k pour les systèmes P2P non-structurés sémantiquement hétérogènes. Il est basé sur l'algorithme Fully Distributed, et laisse la responsabilité à chaque pair recevant une requête de la traduire localement. De cette manière tous les pairs traitent la même requête et les résultats qu'ils fournissent sont ainsi comparables. Nous considérons le modèle vectoriel sémantique : les documents et les requêtes sont représentés par des vecteurs de concepts.

Dans la seconde partie de chapitre nous menons des expérimentations avec un ensemble de documents issus de la base PubMed. Nous montrons que les protocoles définis dans les chapitres 6 et 7 permettent d'améliorer l'interopérabilité. Nous démontrons ainsi que la réduction de l'hétérogénéité sémantique permet d'améliorer l'interopérabilité. Les travaux décrits dans ce chapitre ont été présentés aux conférences CORIA et P2P Computing en 2012 [CCL12b, CCL12a].

8.1 Problématique et objectifs

Dans ce chapitre, nous considérons le même contexte que dans les chapitres précédents. Cette fois nous nous focalisons sur la notion d'interopérabilité, et non plus seulement sur la notion d'hétérogénéité sémantique. Le contexte de notre travail (les systèmes P2P sémantiquement hétérogènes) ne permet pas d'utiliser directement les algorithmes de routage et de traitement de requêtes (comme Fully Distributed, BFS, etc.) car il est parfois nécessaire de traduire les requêtes pour que les pairs puissent les traiter localement. Il nous semble important de faire en sorte que tous les pairs reçoivent et traitent la requête intiale car dans le cas contraire il est difficile de comparer les résultats retournés au pair initiateur de la requête. Cela est d'autant plus important lorsque nous considérons des requêtes top-k, c'est-à-dire des requêtes pour lesquelles l'utilisateur demande à récupérer les k documents les plus pertinents pour sa requête. La problématique que nous abordons dans ce chapitre est la suivante : *Comment effectuer le routage et le traitement de requêtes top-k dans les systèmes P2P non-structurés sémantiquement hétérogènes ?*

En réalité notre objectif est en fait de proposer un système dans lequel les préoccupations sont considérées indépendamment les unes des autres. Pour cela nous proposons une architecture en couche. Une première couche est dédiée aux fonctionnalités relatives au système P2P (maintient de la connectivité, service d'échantillonnage de pairs, etc.). Une deuxième couche est dédiée à ce qui concerne l'hétérogénéité sémantique. C'est à ce niveau que nous implantons les protocoles CorDis et GoOD-TA qui visent à réduire l'hétérogénéité sémantique. Enfin une troisième couche est dédiée à l'application, c'est-à-dire à la recherche d'information. C'est à ce niveau que nous implantons l'algorithme de routage et de traitement de requêtes top-k. Nous présentons cette architecture dans la section 8.2.

Pour répondre à la problématique énoncée ci-dessus, nous proposons l'algorithme DiQuESH[1] de routage et de traitement de requêtes top-k adapté au contexte hétérogène. Cet algorithme permet à un pair du système d'émettre une requête et de récupérer les k documents les plus pertinents. Le classement des documents prend en compte la perte d'information subie lors de la traduction de la requête dans l'espace des documents. Dans cette partie de notre travail, nous considérons le modèle vectoriel sémantique, c.-à-d. que les documents et les requêtes sont représentés par des vecteurs de concepts, et la similarité entre deux vecteurs est donnée par la valeur du cosinus entre ces vecteurs. Nous pensons que les principes de l'algorithme DiQuESH restent valables pour d'autres modèles de RI. Nous présentons l'algorithme DiQuESH dans la section 8.3

Enfin nous voulons montrer dans quelle mesure les protocoles CorDis (chapitre 6, page 89) et GoOD-TA (chapitre 7, page 119) permettent d'améliorer l'interopérabilité, en réduisant l'hétérogénéité sémantique. Dans le cas de la recherche d'information, l'amélioration de l'interopérabilité peut se traduire par une augmentation des taux de précision et de rappel. Nous présentons les résultats d'expérimentations dans la section 8.4.

1. DiQuESH : **Di**stributed **Qu**ery **E**valuation in **S**emantically **H**eterogeneous context.

8.2 Architecture d'un pair du système

Nous proposons un système dont l'architecture, générique et modulaire, est organisée en différentes couches logicielles.

8.2.1 Couche dédiée à la communication

Cette couche est dédiée à la gestion et à la maintenance du système P2P. C'est par exemple à ce niveau que se situe le module assurant le service d'échantillonnage de pairs. Ce dernier fournit un échantillon de pairs du système, et sert de base aux protocoles épidémiques.

8.2.2 Couche dédiée à la gestion de l'hétérogénéité

Cette couche est dédiée à la réduction de l'hétérogénéité sémantique. Elle comprend deux modules correspondant respectivement aux protocoles CorDis et GoOD-TA. Elle supporte les mécanismes mis en œuvre pour faire décroitre l'hétérogénéité sémantique. Les protocoles de cette couche nécessitent des connaissances sur les ontologies et les correspondances données par les alignements.

8.2.3 Couche dédiée à la recherche d'information

Cette dernière couche est dédiée à la recherche d'information en environnement hétérogène. La méthode de RI proposée doit prendre en compte le fait que requêtes et documents sont susceptibles d'être représentés par rapport à des ontologies différentes. En cela, elle est chargée d'assurer l'interoperabilité sémantique, ce qui peut être fait de différentes manières. Dans notre travail, c'est l'algorithme DiQuESH qui s'en charge mais une autre méthode pourrait être mise en place.

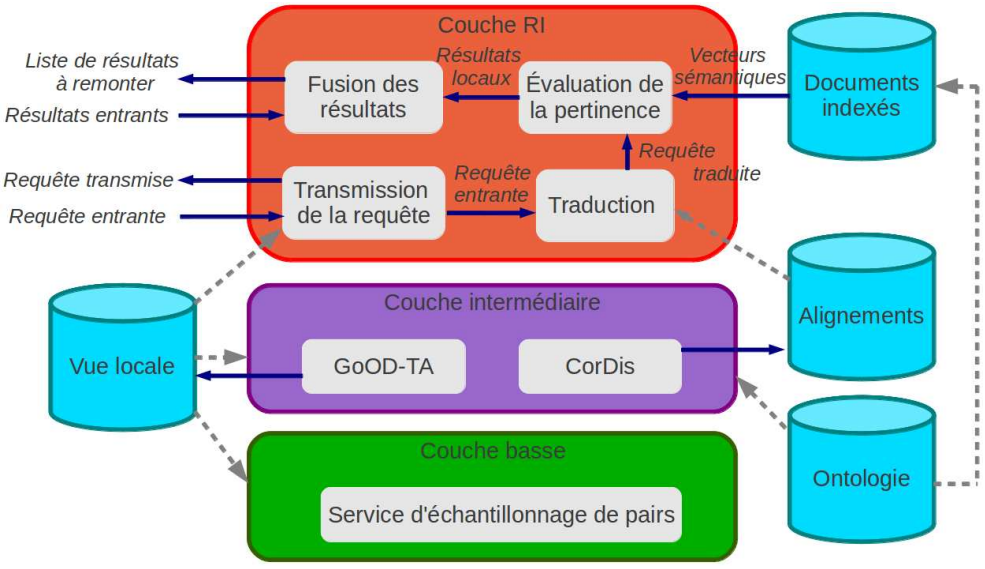


Figure 8.1 – *Architecture d'un pair présentant les trois couches logicielles.*

La figure 8.1 présente l'architecture, elle-même modulaire, d'un pair quelconque du système. Les trois couches logicielles y sont représentées. Ces couches forment une ossature générique qui peut être complétée par d'autres modules ou couches. Par exemple, on peut ajouter une couche gérant le rapprochement des pairs selon leurs intérêts, ou les documents qu'ils stockent. Dans cette figure, les flèches en pointillé représentent des liens d'utilisation, tandis que celles avec des traits pleins représentent des entrées ou des sorties de modules. Par exemple la flèche pleine vers la base des alignements connus symbolise le fait que le protocole CorDis permet au pair d'apprendre des correspondances. Dans la couche supérieure (couche RI), les quatre modules forment le protocole DiQuESH décrit dans la section suivante.

8.3 Évaluation distribuée de requêtes : DiQuESH

Dans cette section nous présentons l'algorithme DiQuESH de traitement distribué de requêtes top-k en environnement sémantiquement hétérogène.

Étant donnée une requête, chaque pair est capable d'affecter un score de pertinence à ses documents. Il contribue à l'interoperabilité en faisant au mieux pour répondre à la requête, même si l'ontologie utilisée pour représenter celle-ci diffère de la sienne. Les scores calculés par les différents pairs sont comparables et ont des valeurs comprises entre 0 et 1. De plus, nous supposons que les pairs ne sont pas malveillants : ils ne trichent pas sur les scores locaux de leurs documents. L'initiateur de la requête désire obtenir un ensemble de k meilleurs documents, pour un voisinage donné, c'est-à-dire qu'il n'existe pas hors de l'ensemble de résultats, un document qui aurait reçu un score plus élevé, en considérant le même voisinage.

Nous nous ramenons donc à une problématique d'évaluation distribuée de requêtes top-k. Nous utilisons le routage des requêtes et la remontée des résultats proposés dans l'algorithme générique Fully Distributed [APV06] où quatre phases se distinguent : transmission de la requête, traitement local, fusion et remontée des résultats, obtention des documents. Pour l'adapter au cas d'une recherche d'information en présence d'hétérogénéité sémantique, nous spécifions le traitement local d'une requête à la fois pour la traduction et le calcul de la pertinence. Notons qu'avec cet algorithme, l'initiateur de la requête reçoit dans un premier temps une liste des k meilleurs résultats, un résultat étant un triplet (id_p, id_d, sc_d) où id_p est l'identifiant du pair possédant le document, id_d est l'identifiant du document d chez le pair p et sc_d est le score de d obtenu chez p. Cette solution ne permet pas de déterminer en cours d'exécution si le même document est retourné plusieurs fois par différents pairs.

8.3.1 Transmission des requêtes

Un pair p initiant une recherche envoie une requête à ses voisins directs ($\mathcal{N}_p$). Le requête Q envoyée dans le système est composée de :

- un identifiant unique de la requête (il peut être construit à partir de l'identifiant du pair initiateur et d'un compteur local) qui sert à déterminer si elle a déjà été traitée par un pair ;

- le vecteur sémantique caractérisant la recherche $\overrightarrow{q_o}$ (exprimé par rapport à l'ontologie o) ;
- l'identifiant de l'ontologie o (c.-à-d. son URI) ;
- le nombre k de documents souhaités ;
- la valeur de TTL (Time-To-Live).

Quand un pair p' reçoit une requête Q, il commence par vérifier s'il l'a déjà reçue. Si ce n'est pas le cas, il la transmet telle quelle à ses propres voisins (si le TTL est supérieur à 0) en décrémentant la valeur du TTL ; puis il la traite localement.

8.3.2 Traitement local des requêtes

Cette phase, qui permet d'extraire la liste des k meilleurs résultats locaux, se décompose en deux étapes : traduction de la requête et évaluation de la pertinence. La première est omise si l'ontologie de la requête est la même que celle du pair. Avant de pouvoir chercher localement les documents pertinents pour une requête, le pair doit vérifier si celle-ci est exprimée à l'aide des concepts de sa propre ontologie. Si c'est le cas, il peut procéder directement à la recherche. Sinon, il doit traduire la requête afin de pouvoir calculer la pertinence entre ses documents et la requête (car requête et documents doivent être définis dans le même espace).

Traduction de la requête

Pour traduire le vecteur d'une requête $\overrightarrow{q_o}$, un pair p' utilisant l'ontologie o' exploite les correspondances qu'il connaît entre o et o'. Le résultat de la traduction est un vecteur sémantique $\overrightarrow{q_{o'}}$ exprimé dans l'espace vectoriel défini par o'. Ce processus est décrit dans l'algorithme 11.

Algorithme 11 : Traduction d'un vecteur sémantique $\overrightarrow{v_o}$ par le pair p'.

Input : un vecteur $\overrightarrow{v_o}$ exprimé dans l'espace de l'ontologie o.
Output : un vecteur $\overrightarrow{v_{o'}}$ exprimé dans l'espace de l'ontologie o'.

```
// Tous les poids de v_o' sont initialisés à 0 :
1 v_o' ← ∅ ;
2 for c ∈ v_o do
3     if ∃c' ∈ C_o' : ⟨id, c, c', ≡, 1⟩ ∈ κ_p(o, o') then
4         v_o'[c'] ← v_o[c] ;
          // v_o[c] désigne le poids de c dans v_o
```

Cet algorithme considère seulement les équivalences dont la valeur de confiance m est égale à 1 (ligne 3). Nous pourrions aussi prendre en compte celles pour lesquelles m dépasse un certain seuil s, en modulant le poids de c' en fonction de m. Nous pourrions également considérer d'autres types de correspondances (subsomption, etc.) pour pondérer davantage de concepts dans l'espace de o', avec les limites connues d'un processus d'expansion.

Évaluation locale de pertinence

Traduire le vecteur de la requête dans l'ontologie du pair récepteur p' a pour effet de le projeter sur les seuls concepts que ce dernier parvient à traduire, c'est-à-dire ceux qui figurent aussi dans l'ontologie o'. La pertinence des documents pourrait être calculée par rapport à cette projection, en oubliant totalement la requête initiale. Le risque est alors d'affecter à un document une valeur de pertinence inadaptée. Par exemple, supposons qu'un document soit représenté par un seul concept c et que la requête initiale le soit sur trois concepts (dont c). Supposons que la requête traduite ne contienne que c. Après cette traduction, si l'on considère directement le cosinus, la pertinence du document sera évaluée à 1. Or, si les trois concepts avaient été traduits, la valeur de pertinence aurait été plus faible. Ainsi, il semble que la pertinence calculée par rapport à une traduction approximative doive être pénalisée. Cela est d'autant plus important que l'algorithme compare ensuite les scores provenant de différents pairs. Ceux qui traduisent bien seraient alors pénalisés à tort.

Pour résoudre ce problème, nous proposons de pondérer le score obtenu par le document vis-à-vis de la requête traduite, en prenant en compte la déviation de cette dernière par rapport à la requête initiale. C'est un calcul qui peut être effectué par le pair récepteur, même s'il n'a pas tous les concepts de la requête dans son ontologie. La déviation correspond à l'erreur introduite lors de la traduction (incomplète) de la requête. Le score d'un document $\overrightarrow{d_{o'}}$ par rapport à la requête initiale $\overrightarrow{q_o}$ est donné par :

$$score(\overrightarrow{d_{o'}}, \overrightarrow{q_o}) = \cos(\overrightarrow{d_{o'}}, \overrightarrow{q_{o'}}) \times \cos(\overrightarrow{q_o}, \overrightarrow{\tilde{q}_o})$$

où $\overrightarrow{\tilde{q}_o}$ est le vecteur correspondant à la requête initiale $\overrightarrow{q_o}$ limité aux concepts qui ont été traduits et pris en compte dans $\overrightarrow{q_{o'}}$.

Exemple 8.1 Dans cet exemple, nous considérons une requête q émise par un pair p_1 utilisant l'ontologie o_1, et un document d stocké par un pair p_2 utilisant l'ontologie o_2 (les ontologies o_1 et o_2 sont celles présentées sur la figure 5.5, page 73). La requête q et le document d sont décrits par :

$$\overrightarrow{q_{o_1}} = [(Fleur_1\ ;\ 1), (Petale_1\ ;\ 0,8), (Sepale_1\ ;\ 0,9)]$$

$$\overrightarrow{d_{o_2}} = [(Fleur_2\ ;\ 1), (Petale_2\ ;\ 0,2), (Lys_2\ ;\ 0,7)]$$

Les concepts pondérés à 0 n'apparaissent pas dans les vecteurs. Si le pair p_2 possède les correspondances $\langle Fleur_1, Fleur_2, \equiv, 1\rangle$ et $\langle Petale_1, Petale_2, \equiv, 1\rangle$, alors il peut traduire la requête $\overrightarrow{q_{o_1}}$ en $\overrightarrow{q_{o_2}}$ avec :

$$\overrightarrow{q_{o_2}} = [(Fleur_2\ ;\ 1), (Petale_2\ ;\ 0,8)]$$

Le vecteur $\overrightarrow{\tilde{q}_{o_1}}$ est alors défini par :

$$\overrightarrow{\tilde{q}_{o_1}} = [(Fleur_1\ ;\ 1), (Petale_1\ ;\ 0,8)]$$

Dans ce cas, le score du document $\overrightarrow{d_{o_2}}$ par rapport à la requête $\overrightarrow{q_{o_1}}$ est donné par :

$$score(\overrightarrow{d_{o_2}}, \overrightarrow{q_{o_1}}) = \cos(\overrightarrow{d_{o_2}}, \overrightarrow{q_{o_2}}) \times \cos(\overrightarrow{q_{o_1}}, \overrightarrow{\tilde{q}_{o_1}}) \simeq 0,67$$

Si d et q sont représentés dans le même espace o, alors $\overrightarrow{q_o} = \overrightarrow{\tilde{q}_o}$. Dans ce cas $\cos(\overrightarrow{q_o}, \overrightarrow{\tilde{q}_o}) = 1$, et donc le score de d par rapport à q est simplement $score(\overrightarrow{d_o}, \overrightarrow{q_o}) = \cos(\overrightarrow{d_o}, \overrightarrow{q_o})$. Le premier avantage de cette méthode est que les scores des documents sont comparables car ils prennent en compte la déviation entre la requête initiale et la requête réellement traitée. Le second avantage est que la déviation est calculée de la même manière que la pertinence entre deux vecteurs sémantiques. Cette méthode est donc générique et peut être appliquée à d'autres mesures que celle du cosinus. L'obtention des documents pertinents s'effectue en sélectionnant les k documents les plus pertinents par rapport à la requête initiale $\overrightarrow{q_o}$.

8.3.3 Fusion et remontée des résultats

Lorsqu'un pair p a fini de traiter la requête localement, il attend les listes de résultats venant de ses voisins. Il effectue un tri-fusion de sa propre liste de résultats avec les listes de résultats qu'il a reçues. Il sélectionne alors les k résultats les plus pertinents et retourne cette liste au pair qui lui avait transmis la requête.

8.3.4 Obtention des documents

La phase d'obtention effective des documents par le pair initiateur de la requête est simple. Elle consiste à contacter les pairs possédant les documents les plus pertinents. Pour récupérer un document id_d, le pair initiateur n'a qu'à contacter le pair qui fournit ce document (son id_p est donné dans la liste des résultats).

8.4 Évaluations

8.4.1 Objectifs

L'objectif des expérimentations est de montrer quel est l'impact de l'hétérogénéité sémantique sur les résultats obtenus par une méthode de recherche d'information. C'est également l'occasion de montrer les bénéfices d'algorithmes de diminution de l'hétérogénéité tels que CorDis et GoOD-TA. Le matériel à mettre en œuvre pour faire de telles expérimentations est important. Sans parler de la distribution, il faut non seulement disposer, comme dans les expérimentations de RI classique, d'un corpus de test (documents, requêtes et jugements de pertinence), mais aussi d'un nombre suffisant d'ontologies, par rapport auxquelles les documents doivent être indexés. Ne connaissant aucun corpus de ce type, nous avons utilisé les ontologies et les services de BioPortal.

8.4.2 Matériel

Simulation de systèmes P2P

Pour ne pas avoir à développer et à déployer un véritable système P2P, nous avons utilisé le simulateur PeerSim [MJ09]. Ce simulateur est très utilisé dans la communauté P2P pour la validation. PeerSim permet

d'instancier des systèmes P2P non-structurés servant de base à notre système.

Ontologies et alignements de BioPortal

Pour ces expérimentations, nous avons considéré les mêmes ontologies et les mêmes correspondances que dans les sections 6.5 (page 104) et 7.4 (page 130). Nous avons réparti les ontologies de manière aléatoire dans le système suivant une loi de distribution uniforme, et chaque pair connaît en moyenne 20% des correspondances qui existent (et qui concernent son ontologie).

Documents

Nous avons extrait les annotations sémantiques de 4163 documents (appelés *ressources* sur BioPortal) à l'aide des services web BioPortal. Les documents correspondent à des articles scientifiques publiés dans des journaux du domaine biomédical. Ils sont issus de la base de données PubMed. Certains documents sont annotés avec des concepts de différentes ontologies. Les annotations sont utilisées pour créer les vecteurs sémantiques correspondant aux documents. Seules 39 ontologies sont utilisées pour représenter les documents.

Distribution des ontologies et des documents

Nous avons distribué aléatoirement dans le système P2P les 39 ontologies utilisées pour représenter les documents : chaque pair utilise une ontologie, et chaque ontologie est utilisée par le même nombre de pairs. Les documents sont également distribués aléatoirement, mais en tenant compte des ontologies utilisées par les pairs : chaque pair stocke entre 50 et 100 documents indexés suivant l'ontologie utilisée. Certains documents peuvent être répliqués dans le système. Des stratégies plus avancées ont été proposées pour placer les documents dans le système. Par exemple, Holz *et al.* [HWH+07] suggèrent de regrouper les documents en fonction de leurs auteurs : chaque pair est responsable des documents d'un auteur, et les pairs responsables de deux auteurs ayant travaillé ensemble sont voisins dans le système. Nous n'avons pas choisi cette stratégie car l'ensemble de documents que nous avons utilisés ne s'y prête pas : il y a presque autant d'auteurs que de documents, et très peu ont collaboré pour produire ces documents.

Requêtes et jugement de pertinence

Comme nous ne disposons pas de requêtes portant sur les documents que nous considérons, nous en avons généré un certain nombre. Chaque requête est un vecteur sémantique composé de concepts d'une ontologie o : certains sont partagés entre o et d'autres ontologies, certains sont présents dans un ou plusieurs documents, et les autres sont choisis aléatoirement dans l'ontologie o.

	$\mathcal{H}_{Div}$	$\mathcal{H}_{Disp}$	$\mathcal{H}_{DispAPMoy}$
DiQuESH seul	0,04	0,97	0,96
CorDis	0,04	0,96	0,96
GoOD-TA	0,04	0,97	0
CorDis et GoOD-TA	0,04	0,96	0

Table 8.1 – *Valeurs d'hétérogénéité sémantique avant exécution des requêtes.*

Le jugement de pertinence nécessaire à l'évaluation de la méthode de recherche d'information est construit à partir de ces requêtes et de ces documents. Pour chaque requête nous identifions les k ($k = 10$) meilleurs documents en exécutant la méthode dans un système centralisé ayant accès à tous les documents et à toutes les correspondances entre ontologies : ce sont eux qui forment l'ensemble des documents pertinents pour la requête. Nous prenons le soin de retirer les requêtes pour lesquelles aucun document pertinent n'est retourné. Finalement, nous disposons de 1353 requêtes.

Métriques

Pour mesurer la qualité des résultats fournis par la méthode de RI pour une requête top-k donnée, nous utilisons les mesures de précision (Pr) et de rappel (Ra). Pour mesurer l'hétérogénéité des systèmes P2P, nous utilisons les mesures présentées dans la section 5.4 (page 77) : $\mathcal{H}_{Div}$ (cf. équation 5.1, page 77), $\mathcal{H}_{Disp}$ (cf. équation 5.4, page 78) et $\mathcal{H}_{DispAPMoy}$ (cf. équation 5.8, page 79).

8.4.3 Paramètres de simulation

Dans les expérimentations nous considérons des systèmes P2P de 1000 pairs, et nous considérons qu'ils sont statiques : les pairs présents dans un système ne le quittent pas, et aucun nouveau pair ne les rejoignent. Initialement tous les pairs sont directement connectés à quatre autres pairs ($c = 4$). Nous considérons des requêtes top-k où $k = 10$: nous cherchons à obtenir les 10 documents les plus pertinents pour chaque requête. Par ailleurs nous fixons la valeur du TTL à 4. Par conséquent, chaque requête, qui est émise par un pair choisi au hasard dans le système, peut atteindre au plus 340 pairs.

Nous considérons quatre situations. La première correspond à un système P2P hétérogène dans lequel seul l'algorithme DiQuESH est mis en œuvre. La deuxième situation correspond au même système que dans la situation précédente, mais dans lequel l'algorithme CorDis est exécuté pendant 100 cycles. Les requêtes sont lancées après la fin de l'exécution de CorDis. La troisième situation est semblable à la deuxième mais l'algorithme exécuté est GoOD-TA. La dernière situation est une combinaison des deux situations précédentes : CorDis et GoOD-TA sont exécutés en parallèle.

8.4.4 Résultats

Le tableau 8.1 montre que l'algorithme CorDis permet de réduire l'hétérogénéité sémantique liée à la disparité entre pairs. La réduction est faible car trop peu de correspondances existent entre les ontologies qu'utilisent les pairs. L'algorithme GoOD-TA permet de réduire l'hétérogénéité sémantique liée à l'organisation du système à 0. Cela est possible car la diversité sémantique est très faible ($\mathcal{H}_{Div}(m) = \frac{39-1}{1000-1} = 0,04$). En moyenne chaque ontologie est utilisée par 25 pairs donc ceux-ci se regroupent et l'hétérogénéité centrée sur chaque participant est nulle.

La figure 8.2 présente les valeurs moyennes de précision et de rappel obtenues dans chacune des situations étudiées pour l'ensemble des requêtes. Cette figure montre que CorDis permet d'augmenter de manière significative la valeur de rappel (de 0,25 à 0,35) tout en améliorant légèrement la précision. Cela est dû au fait que les correspondances apprises par les pairs leurs permettent de mieux traduire les requêtes qu'ils reçoivent, et donc retrouver davantage de documents pertinents. Évidemment plus il y a de correspondances entre les ontologies, plus les pairs utilisant des ontologies différentes sont capables d'interopérer. Dans notre cas, les ontologies sont très peu liées. Par conséquent, l'algorithme CorDis réduit peu l'hétérogénéité, et la précision est peu améliorée.

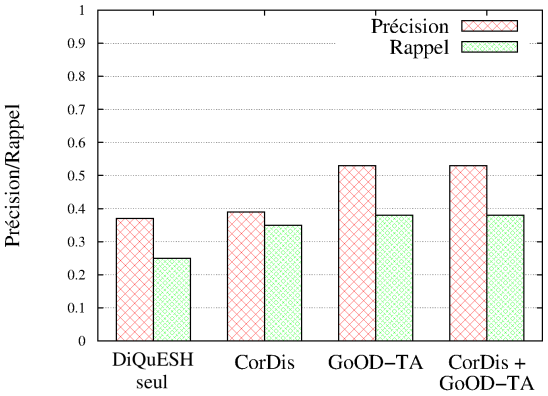


Figure 8.2 – *Valeurs de précision/rappel obtenues dans les différentes situations étudiées.*

Par ailleurs l'algorithme GoOD-TA permet d'augmenter fortement la précision (de 0,25 à 0,39) et le rappel (de 0,38 à 0,53). Cela s'explique par le fait que chaque pair est placé à proximité des pairs susceptibles de comprendre et de traiter les requêtes qu'il émet. En effet il est préférable que les pairs capables de comprendre la requête sans la traduire soient accessibles, c'est-à-dire qu'ils soient dans le voisinage du pair qui l'émet. De plus il y a une forte corrélation entre l'ontologie utilisée par un pair, et ses centres d'intérêts (exprimés au travers des requêtes). Les documents susceptibles d'être pertinents par rapport à la requête sont à priori représentés dans un "vocabulaire" proche de celui de la requête. Par exemple, les requêtes émises par un spécialiste en biochimie métabolique seront vraisemblablement satisfaites par des spécialistes du même domaine (ou au moins d'un domaine proche). Il est donc probable qu'ils

utilisent la même ontologie, ou des ontologies entre lesquelles il existe de nombreuses correspondances.

Enfin la figure 8.2 montre que la combinaison des deux algorithmes n'améliore pas la précision et le rappel par rapport à GoOD-TA seul. Cela s'explique par le fait que les pairs se regroupent en communautés dans lesquelles les pairs utilisent tous la même ontologie. Ce phénomène ne se produirait pas si la diversité sémantique (mesurée par $\mathcal{H}_{Div}$) était plus élevée, ou si les ontologies étaient plus proches sémantiquement.

8.5 Discussions

Dans ce chapitre nous avons présenté l'algorithme DiQuESH de routage et de traitement de requêtes top-k dans les systèmes P2P non-structurés sémantiquement hétérogènes et nous avons montré en quoi les protocoles CorDis (cf. chapitre 6) et GoOD-TA (cf. chapitre 7) permettent d'améliorer l'interopérabilité. Ce travail a été présenté aux conférences CORIA et P2P Computing en 2012 [CCL12b, CCL12a].

Dans ce chapitre, nous avons considéré le modèle vectoriel sémantique : les documents et les requêtes sont représentés par des vecteurs dont les dimensions sont des concepts d'une ontologie. Nous pensons qu'il serait intéressant d'étudier l'algorithme DiQuESH en considérant d'autres modèles de RI, par exemple le modèle booléen ou le modèle booléen étendu dans lesquels les éléments du vocabulaire sont des concepts d'ontologies et non des termes.

Comme autres travaux futurs, nous prévoyons de comparer l'algorithme DiQuESH à d'autres approches. Le problème est que les méthodes existantes mélangent plusieurs préoccupations, par exemple le routage de requêtes et la détection d'incohérences. De plus, elles fonctionnent sur des modes différents puisque les méthodes existantes (par ex. celle utilisé dans le système Piazza [HIM+04]) considèrent que les pairs traduisent les requêtes avant de les transmettre, alors que DiQuESH fait en sorte que tous les pairs traitent la requête originale.

Enfin, pour valider notre approche, nous envisageons de mener de nouvelles expérimentations avec des ensembles de documents et de requêtes plus importants. La difficulté est qu'il est difficile d'obtenir des documents et des requêtes annotés ou indexés par rapport à différentes ontologies. Nous pensons qu'il serait intéressant de considérer une collection communément utilisée dans la communauté travaillant sur la recherche d'information, par exemple un corpus de TREC. Dans ce cas, il faudrait utiliser un outil d'indexation sémantique performant pour éviter d'obtenir des mauvais taux de précision/rappel non pas à cause de la méthode de RI mais à cause de la qualité de l'indexation.

Quatrième partie

Conclusion

Rappel des contributions

Dans ce travail nous avons proposé un ensemble de solutions permettant d'adresser le problème d'hétérogénéité sémantique dans les systèmes P2P non-structurés.

Dans un premier temps, nous avons défini des mesures complémentaires permettant de caractériser finement l'hétérogénéité sémantique d'un système suivant différentes facettes. Elles permettent d'évaluer des méthodes particulières, par exemple de recherche d'information. Elles permettent également de mesurer à quel point un algorithme donné permet de réduire l'hétérogénéité d'un système. Nous pensons que les mesures définies peuvent être adaptées et utilisées dans d'autres contextes : bases de données distribuées hétérogènes, dépôts RDF, etc.. L'intérêt de ces mesures est qu'elles permettent d'évaluer des méthodes de réduction de l'hétérogénéité sans être contraint d'utiliser un ensemble de données (par exemple, un corpus de RI).

Dans un deuxième temps nous avons défini deux protocoles permettant de réduire l'hétérogénéité sémantique d'un système P2P. Les deux protocoles sont des protocoles épidémiques [KvS07]. Le premier, appelé CorDis, réduit l'hétérogénéité liée aux disparités entre pairs. Pour cela il dissémine des correspondances dans le système afin que les pairs puissent apprendre de nouvelles correspondances précédemment connues par d'autres. Le deuxième protocole, appelé GoOD-TA, réduit l'hétérogénéité sémantique liée à la topologie des systèmes. La ré-organisation du système se fait par l'échange de descripteurs de pairs. Ainsi ils peuvent découvrir de nouveaux pairs et les choisir comme voisins. Deux pairs se rapprochent dans le système s'ils utilisent la même ontologie (dans les deux versions de GoOD-TA) ou s'il existe de nombreuses correspondances entre leurs ontologies (dans la seconde version de GoOD-TA). Nous avons montré expérimentalement que chacun des protocoles permet de réduire certaines facettes de l'hétérogénéité sémantique. Pour cela nous avons considéré des ontologies activement utilisées dans le domaine biomédical. Nous les avons obtenu sur BioPortal [FNSW+09]. Par ailleurs les expérimentations ont permis de montrer que les protocoles restent efficaces lorsque le nombre de pairs augmente, lorsque les pairs sont dynamiques, ou lorsqu'ils changent d'ontologies. CorDis et GoOD-TA sont complémentaires dans le sens où ils réduisent différentes facettes de l'hétérogénéité.

Enfin nous avons proposé l'algorithme DiQuESH de routage et de traitement de requêtes top-k dans les systèmes P2P non-structurés sémantiquement hétérogènes. Dans cet algorithme une requête est transmise dans un certain rayon autour du pair initiateur, sans que celle-ci ne soit modifiée. Ainsi tous les pairs reçoivent la même requête et sont alors responsables de la traduire localement si besoin. Ce mode de transmission permet d'éviter une perte d'information due à des traductions incomplètes successives. L'algorithme DiQuESH prend en compte l'erreur commise lors de la traduction de la requête dans le calcul des scores de pertinence attribués aux documents. De cette manière tous les scores attribués par les pairs sont comparables et le pair initiateur est en mesure de classer les documents reçus suivant leurs scores sans se soucier des ontologies avec lesquelles les documents ont été représentés. Nous avons également montré

que les protocole CorDis et GoOD-TA permettent indirectement d'améliorer l'interopérabilité en réduisant l'hétérogénéité. Pour cela nous avons considéré un ensemble de documents obtenus sur BioPortal. Chaque document est un article de la collection PubMed. Par ces expérimentations nous montrons que réduire l'hétérogénéité sémantique permet effectivement d'améliorer l'interopérabilité. Dans le cas d'une méthode de RI, cela se traduit par une amélioration des taux de précision et de rappel.

Pour conclure, nous pouvons dire que nous avons défini un système en couches qui considère séparément les préoccupations : la gestion du système P2P, la gestion de l'hétérogénéité (en particulier sa réduction), et la recherche d'information. Cette proposition répond à la problématique que nous avons proposé d'aborder en introduction. Les perspectives de ce travail sont présentées dans la section suivante.

Perspectives et travaux futurs

Comme travaux futurs à court terme, nous pensons qu'il serait intéressant de mener des évaluations supplémentaires, en particulier pour essayer de situer notre approche par rapport aux travaux existants. Même si nous pensons que cela est difficile, parce qu'aucune méthode à notre connaissance n'a pour objectif de réduire explicitement l'hétérogénéité, cela permettrait de mettre plus en valeur nos contributions et notre approche du problème. Du point de vue de la recherche d'information, il semble pertinent de mener d'autres tests avec des corpus utilisés dans la communauté de recherche d'information : un corpus issu de TREC serait idéal.

Par ailleurs nous envisageons de définir un nouvel algorithme qui permettrait de réduire plusieurs facettes de l'hétérogénéité en même temps. En quelque sorte, il s'agit de fusionner les algorithmes CorDis et GoOD-TA. Bien qu'indépendants et complémentaires, nous pensons qu'il est possible d'améliorer les performances des protocoles (en termes de réduction de l'hétérogénéité et de vitesse de réduction) s'ils sont exécutés conjointement. L'architecture de système que nous avons conçue permet d'intégrer facilement un nouveau protocole dans n'importe quelle couche et de comparer ses performances avec d'autres protocoles. Par exemple, nous pourrions intégrer les méthodes permettant de faire émerger une ontologie globale [NSCC07, DDC10], même si celles-ci nous semblent peu appropriées car elles limitent l'autonomie des pairs.

Dans ce travail, nous avons supposé que chaque participant utilise une seule ontologie pour représenter ces données. Dans certains contextes, il est sans doute plus réaliste de considérer que chaque participant utilise plusieurs ontologies. Chaque ontologie peut alors être utilisée pour décrire certains documents stockés par les pairs en fonction de leurs thématiques ou pour décrire l'ensemble des documents (les documents sont alors décrits suivant plusieurs ontologies). Cette situation nécessite d'adapter notre approche. Nous pensons que la démarche pour gérer l'hétérogénéité sémantique est la même et que nous devons simplement redéfinir certaines mesures d'hétérogénéité et certains aspects des algorithmes de réduction de l'hétérogénéité que nous avons définis.

Certaines hypothèses que nous faisons dans notre travail peuvent sembler fortes, en particulier le fait que les pairs sont coopératifs et qu'ils ne sont pas malveillants. Comme perspective, nous voulons étudier le cas où certains pairs sont malveillants, car cela a probablement un impact sur la manière dont l'hétérogénéité peut être réduite. Les questions auxquelles nous allons nous intéresser sont les suivantes : Dans quelle mesure peut-on réduire l'hétérogénéité liée aux disparités entre pairs si un certain nombre de pairs du système propagent volontairement des correspondances erronées ? Dans quelle mesure peut-on réduire l'hétérogénéité liée à la topologie si des pairs fournissent de mauvais descripteurs (leurs propres descripteurs ou les descripteurs des autres pairs) ? Quels mécanismes, indépendants de ceux liés à la réduction de l'hétérogénéité, doivent être mis en place pour identifier et exclure les pairs malveillants ? Ce travail conduira probablement à l'introduction d'une couche supplémentaire dans l'architecture de notre système (située entre la couche dédiée à l'hétérogénéité et celle dédiée à l'interopérabilité).

Enfin nous envisageons, à plus long terme, de réfléchir en quoi notre travail peut être appliqué dans le cadre du Web sémantique. En particulier nous voudrions nous intéresser au cas où les pairs stockent des triplets RDF, comme c'est le cas dans les systèmes GridVine [ACMHvP04, CMAA07] et Edutella [NWQ+02, NSS03]. Nous pensons que notre approche du problème peut être adaptée à ce contexte. La comparaison de notre solution avec les approches existantes devrait être plus simple car le cadre d'application est le même : même type de requêtes, de données, etc. Nous pensons que l'infrastructure offerte par les systèmes P2P devraient permettre de résoudre les problèmes du Web sémantique liés à la taille que certains dépôts RDF peuvent atteindre et au grand nombre d'utilisateurs qui peuvent vouloir se connecter simultanément à un dépôt RDF.

Annexes

A

A.1 Publications

Revues internationales

- Thomas Cerqueus, Sylvie Cazalens et Philippe Lamarre. Reducing the semantic heterogeneity of unstructured P2P systems : a contribution based on a dissemination protocol. Dans *Transactions on Large-Scale Data- and Knowledge-Centered Systems VII*. pages 62-95, 2012.

Conférences internationales

- Thomas Cerqueus, Sylvie Cazalens et Philippe Lamarre. An Approach to Manage Semantic Heterogeneity in Unstructured P2P Information Retrieval Systems. Dans *12th IEEE International Conference on Peer-to-Peer Computing (P2P'12)*. pages 179-188, 2012.
- Thomas Cerqueus, Sylvie Cazalens et Philippe Lamarre. Gossiping correspondences to reduce semantic heterogeneity of unstructured P2P systems. Dans *4th International Conference on Data Management in Grid and P2P Systems (Globe 2011)*. pages 37-48, 2011.
- Thomas Cerqueus, Sylvie Cazalens et Philippe Lamarre. Semantic heterogeneity measures of unstructured P2P systems. Dans *2011 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT 2011)*. pages 223-226, 2011.
- Anthony Ventresque, Sylvie Cazalens, Thomas Cerqueus, Philippe Lamarre et Gabriela Pasi. Personalization through Query Explanation and Document Adaptation. Dans *4th International Workshop on Personalized Access, Profile Management, and Context Awareness in Databases (PersDB2010, workshop VLDB)*. pages 17-22, 2010.

Conférences nationales

- Thomas Cerqueus, Sylvie Cazalens et Philippe Lamarre. Influence de l'hétérogénéité sémantique sur les performances d'un système de RI distribuée. Dans *COnférence en Recherche d'Information et Applications (CORIA 2012)*. pages 151-166, 2012.
- Thomas Cerqueus. Recherche d'information sémantique dans les systèmes P2P hétérogènes. Dans *COnférence en Recherche d'Information et Applications, Journée Jeunes Chercheurs (CORIA 2012, JJC)*. pages 419-424, 2012.

– Thomas Cerqueus, Sylvie Cazalens et Philippe Lamarre. Mesures d'hétérogénéité sémantique des systèmes P2P non-structurés. Dans *Extraction et Gestion de Connaissances (EGC'2011)*. pages 341-346, 2011.
– Anthony Ventresque, Thomas Cerqueus, Louis-Alexandre Celton, Gaëtan Hervouet, Damien Levin, Philippe Lamarre et Sylvie Cazalens. Mysins : Make Your Semantic INformation System. Dans *Extraction et Gestion de Connaissances (EGC'2010)*. pages 629-630, 2010.

Bibliographie

[Abo01] Olivier Abou. *P2P*. Micro Application, 2001. (Cité page 34.)

[ACMH02] Karl Aberer, Philippe Cudré-Mauroux, and Manfred Hauswirth. A framework for semantic gossiping. *SIGMOD Record*, 31(4) :48–53, 2002. (Cité page 63.)

[ACMH03a] Karl Aberer, Philippe Cudré-Mauroux, and Manfred Hauswirth. The chatty web : emergent semantics through gossiping. In *12th International World Wide Web Conference*, pages 197–206, 2003. (Cité page 62.)

[ACMH03b] Karl Aberer, Philippe Cudré-Mauroux, and Manfred Hauswirth. Start making sense : The chatty web approach for global semantic agreements. *Journal of Web Semantics*, 1(1) :89–114, 2003. (Cité page 62.)

[ACMH06] Karl Aberer, Philippe Cudré-Mauroux, and Manfred Hauswirth. Semantic gossiping : Fostering semantic interoperability in peer data management systems. In *Semantic Web and Peer-to-Peer*, pages 259–278. Springer Verlag, 2006. (Cité page 63.)

[ACMHvP04] Karl Aberer, Philippe Cudré-Mauroux, Manfred Hauswirth, and Tim van Pelt. Gridvine : Building internet-scale semantic overlay networks. In *3rd International Semantic Web Conference*, pages 107–121, 2004. (Cité pages 63 et 159.)

[AD+07] Manuela Angioni, , Roberto Demontis, Massimo Deriu, Emanuela De Vita, Cristian Lai, Ivan Marcialis, Antonio Pintus, Andrea Piras, Alessandro Soro, and Franco Tuveri. DART : The distributed agent based retrieval toolkit. In *1st WSEAS International Conference On Computer Engineering And Applications*, volume 1, 2007. (Cité page 60.)

[ADMR05] David Aumueller, Hong Hai Do, Sabine Massmann, and Erhard Rahm. Schema and ontology matching with COMA++. In *SIGMOD*, pages 906–908, 2005. (Cité pages 4 et 25.)

[AED+08] Erick Antezana, Mikel Egaña, Bernard De Baets, Martin Kuiper, and Vladimir Mironov. ONTO-PERL : An API for supporting the development and analysis of bio-ontologies. *Bioinformatics*, 24(6) :885–887, 2008. (Cité page 104.)

[AEPR11] Manuel Atencia, Jérôme Euzenat, Giuseppe Pirrò, and Marie-Christine Rousset. Alignment-based trust for resource finding in semantic p2p networks. In *10th International Semantic Web Conference*, pages 51–66, 2011. (Cité page 14.)

[AK10] Raddad Al King. *Localisation de sources de données et optimisation de requêtes réparties en environnement pair-à-pair*. PhD thesis, Université Paul Sabatier, 2010. (Cité page 61.)

[AKHM08] Raddad Al King, Abdelkader Hameurlain, and Franck Morvan. Ontology-based data source localization in a structured peer-to-peer environment. In *International Symposium on Database Engineering and Applications*, pages 9–18, 2008. (Cité page 61.)

[AKRW04] Karl Aberer, Fabius Klemm, Martin Rajman, and Jie Wu. An architecture for peer-to-peer information retrieval. In *Workshop on Peer-to-Peer Information Retrieval*, 2004. (Cité page 60.)

[APV06] Reza Akbarinia, Esther Pacitti, and Patrick Valduriez. Reducing network traffic in unstructured P2P systems using top-k queries. *Distributed and Parallel Databases*, 19(2-3) :67–86, 2006. (Cité pages 5, 36 et 147.)

[APV07] Reza Akbarinia, Esther Pacitti, and Patrick Valduriez. Processing top-k queries in distributed hash tables. In *13th International Euro-Par Conference*, pages 489–502, 2007. (Cité page 38.)

[AvH09] Grigoris Antoniou and Frank van Harmelen. Web ontology language : OWL. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 91–110. Springer Berlin Heidelberg, 2009. Second Edition. (Cité page 16.)

[Bay63] Thomas Bayes. An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society of London*, 53 :370–418, 1763. (Cité page 50.)

[BBG+10] Xiao Bai, Marin Bertier, Rachid Guerraoui, Anne-Marie Kermarrec, and Vincent Leroy. Gossiping personalized queries. In *13th International Conference on Extending Database Technology*, pages 87–98, 2010. (Cité page 42.)

[BBR11] Zohra Bellahsene, Angela Bonifati, and Erhard Rahm, editors. *Schema Matching and Mapping*. Springer, 2011. (Cité pages 20 et 24.)

[BCM+03] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider. *The Description Logic Handbook : Theory, Implementation,and Applications*. Cambridge University Press, 2003. (Cité page 17.)

[BEE+04] Paolo Bouquet, Marc Ehrig, Jérôme Euzenat, Enrico Franconi, Pascal Hitzler, Markus Krötzsch, Luciano Serafini, Giorgos Stamou, York Sure, and Sergio Tessaris. Specification of a common framework for characterizing alignment. Knowledge Web Deliverable 2.2.1, 2004. (Cité page 19.)

[BEF+06] Jos de Bruijn, Marc Ehrig, Cristina Feier, Francisco Martíns-Recuerda, François Scharffe, and Moritz Weiten. *Ontology Mediation, Merging, and Aligning*, pages 95–113. John Wiley & Sons, Ltd, 2006. (Cité page 24.)

[BFCS12] David S. Batista, João D. Ferreira, Francisco M. Couto, and Mário J. Silva. Toponym disambiguation using ontology-based semantic similarity. In *10th International Conference on Computational Processing of the Portuguese Language*, pages 179–185, 2012. (Cité page 25.)

[BFG+10] Marin Bertier, Davide Frey, Rachid Guerraoui, Anne-Marie Kermarrec, and Vincent Leroy. The gossple anonymous social network. In *11th ACM/IFIP/USENIX International Middleware Conference*, pages 191–211, 2010. (Cité page 42.)

[BHS09] Franz Baader, Ian Horrocks, and Ulrike Sattler. Description logics. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 21–43. Springer Berlin Heidelberg, 2009. Second Edition. (Cité page 17.)

[BK03] J. Becker and D. Kuropka. Topic-based vector space model. In *6th International Conference on Business Information Systems*, pages 7–12, 2003. (Cité page 48.)

[BK07] Yann Busnel and Anne-Marie Kermarrec. ProxSem : Interest-based proximity measure to improve search efficiency in P2P systems. In *4th European Conference on Universal Multiservice Networks*, pages 62–71, 2007. (Cité page 42.)

[Bor97] Willem Nico Borst. *Construction of Engineering Ontologies for Knowledge Sharing and Reuse*. PhD thesis, Universiteit Twente, 1997. (Cité page 13.)

[BP01] Martin Braschler and Carol Peters. European research letter : Cross-language system evaluation : The clef campaigns. *Journal of the American Society for Information Science and Technology*, 52(12) :1067–1072, 2001. (Cité page 52.)

[BS08] Mohand Boughanem and Jacques Savoy, editors. *Recherche d'information états des lieux et perspectives*. Hermès Science Publications, 2008. (Cité page 45.)

[BVL03] Sean Bechhofer, Raphael Volz, and Phillip W. Lord. Cooking the semantic web with the owl api. In *2nd International Semantic Web Conference*, pages 659–675, 2003. (Cité page 104.)

[BYRN99] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, May 1999. (Cité page 45.)

[CCL11a] Thomas Cerqueus, Sylvie Cazalens, and Philippe Lamarre. Gossiping correspondences to reduce semantic heterogeneity of unstructured P2P systems. In *4th International Conference on Data Management in Grid and P2P Systems*, pages 37–48, 2011. (Cité pages 89, 107 et 115.)

[CCL11b] Thomas Cerqueus, Sylvie Cazalens, and Philippe Lamarre. Mesures d'hétérogénéité sémantique dans les systèmes P2P non-structurés. In *Extraction et Gestion de Connaissances*, pages 341–346, 2011. (Cité pages 69 et 86.)

[CCL11c] Thomas Cerqueus, Sylvie Cazalens, and Philippe Lamarre. Semantic heterogeneity measures of unstructured P2P systems. In *10th IEEE/WIC/ACM International Conference on Web intelligence*, pages 223–226, 2011. (Cité pages 69 et 86.)

[CCL12a] Thomas Cerqueus, Sylvie Cazalens, and Philippe Lamarre. An approach to manage semantic heterogeneity in unstructured p2p information retrieval systems. In *12th IEEE International Conference on Peer-to-Peer Computing*, pages 179–188, 2012. (Cité pages 119, 141, 143 et 154.)

[CCL12b] Thomas Cerqueus, Sylvie Cazalens, and Philippe Lamarre. Influence de l'hétérogénéité sémantique sur les performances d'un système de ri distribuée. In *8ième COnference de Recherche d'Information et Applications*, pages 151–166, 2012. (Cité pages 143 et 154.)

[CCL12c] Thomas Cerqueus, Sylvie Cazalens, and Philippe Lamarre. Reducing the semantic heterogeneity of unstructured p2p systems : a contribution based on a dissemination protocol. *Transactions on Large-Scale Data- and Knowledge-Centered Systems*, 7720(7), 2012. (Cité pages 89 et 116.)

[CF04] Min Cai and Martin R. Frank. Rdfpeers : a scalable distributed rdf repository based on a structured peer-to-peer network. In *13th International Conference on World Wide Web*, pages 650–657, 2004. (Cité page 63.)

[CGM02] Arturo Crespo and Hector Garcia-Molina. Routing indices for peer-to-peer systems. In *22nd International Conference on Distributed Computing Systems*, pages 23–33, 2002. (Cité page 36.)

[CM08a] Michel Chein and Marie-Laure Mugnier. *Graph-based Knowledge Representation : Computational Foundations of Conceptual Graphs*. Springer, 2008. (Cité page 13.)

[CM08b] Philippe Cudré-Mauroux. Peer data management system, 2008. (Cité page 60.)

[CMA04] Philippe Cudré-Mauroux and Karl Aberer. A necessary condition for semantic interoperability in the large. In *3rd International Conference on Ontologies, Databases and Applications of Semantics*, pages 859–872, 2004. (Cité page 71.)

[CMAA07] Philippe Cudré-Mauroux, Suchit Agarwal, and Karl Aberer. GridVine : An infrastructure for peer information management. *IEEE Internet Computing*, 11(5) :36–44, 2007. (Cité pages 63 et 159.)

[Coh03] Bram Cohen. Incentives build robustness in BitTorrent. In *1st Workshop on Economics of Peer-to-Peer Systems*, 2003. (Cité page 3.)

[CPT05] Carmela Comito, Simon Patarin, and Domenico Talia. PARIS : A peer-to-peer architecture for large-scale semantic data integration. In *Databases, Information Systems, and Peer-to-Peer Computing, International Workshops, (DBISP2P)*, pages 163–170, 2005. (Cité page 62.)

[CSC+11] Isabel F. Cruz, Cosmin Stroe, Federico Caimi, Alessio Fabiani, Catia Pesquita, Francisco M. Couto, and Matteo Palmonari. Using agreementmaker to align ontologies for OAEI 2011. In *6th International Workshop on Ontology Matching*, 2011. (Cité page 25.)

[CW04] Pei Cao and Zhe Wan. Efficient top-k query calculation in distributed networks. In *23rd Annual ACM Symposium on Principles of Distributed Computing*, pages 206–215, 2004. (Cité page 5.)

[d'A09] Mathieu d'Aquin. Formally measuring agreement and disagreement in ontologies. In *5th International Conference on Knowledge Capture*, pages 145–152, 2009. (Cité page 29.)

[DDC10] Herminio Camargo De Souza, Ana Maria De C. Moura, and Maria Cláudia Cavalcanti. Integrating ontologies based on P2P mappings. *IEEE Transactions on Systems, Man, and Cybernetics*, 40 :1071–1082, 2010. (Cité pages 64 et 158.)

[DE08] Jérôme David and Jérôme Euzenat. Comparison between ontology distances (preliminary results). In *7th International Conference on The Semantic Web (ISWC)*, pages 245–260, Berlin, Heidelberg, 2008. Springer-Verlag. (Cité pages 21 et 27.)

[DEvZ10] Jérôme David, Jérôme Euzenat, and Ondřej Šváb Zamazal. Ontology similarity in the alignment space. In *9th International Semantic Web Conference*, 2010. (Cité page 28.)

[DGH+88] Alan J. Demers, Daniel H. Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard E. Sturgis, Daniel C. Swinehart, and Douglas B. Terry. Epidemic algorithms for replicated database maintenance. *Operating Systems Review*, 22(1) :8–32, 1988. (Cité pages 40 et 42.)

[Din03] Liviu P. Dinu. On the classification and aggregation of hierarchies with different constitutive elements. *Fundamenta Informaticae*, pages 39–50, 2003. (Cité page 84.)

[DLAV10] William Kokou Dedzoé, Philippe Lamarre, Reza Akbarinia, and Patrick Valduriez. ASAP top-k query processing in unstructured P2P systems. In *10th IEEE International Conference on Peer-to-Peer Computing*, pages 1–10, 2010. (Cité page 37.)

[DMD+03] AnHai Doan, Jayant Madhavan, Robin Dhamankar, Pedro Domingos, and Alon Y. Halevy. Learning to match ontologies on the semantic web. *Very Large DataBase Journal*, 12(4) :303–319, 2003. (Cité page 23.)

[EAD+09] Jérôme Euzenat, Carlo Allocca, Jérôme David, Mathieu d'Aquin, Chan Le Duc, and Ondrej Svab-Zamazal. Ontology distances for contextualisation. Deliverable, NeOn, 2009. (Cité pages 27 et 28.)

[ELBB+04] Jérôme Euzenat, Thanh Le Bach, Jesús Barrasa, Paolo Bouquet, Jan De Bo, Rose Dieng-Kuntz, Marc Ehrig, Manfred

Hauswirth, Mustafa Jarrar, Ruben Lara, Diana Maynard, Amedeo Napoli, Giorgos Stamou, Heiner Stuckenschmidt, Pavel Shvaiko, Sergio Tessaris, Sven Van Acker, and Ilya Zaihrayeu. State of the art on ontology alignment. Knowledge Web Deliverable 2.2.3, 2004. (Cité pages 3, 20, 22 et 24.)

[ES04] Marc Ehrig and York Sure. Ontology mapping - an integrated approach. In *1st European Semantic Web Symposium*, pages 76–91, 2004. (Cité page 24.)

[ES07] Jérôme Euzenat and Pavel Shvaiko. *Ontology matching*. Springer-Verlag, 2007. (Cité pages 4, 19, 20, 21, 24 et 64.)

[EV04] Jérôme Euzenat and Petko Valtchev. Similarity-based ontology alignment in OWL-Lite. In *16th European Conference on Artificial Intelligence (ECAI)*, pages 333–337, 2004. (Cité page 23.)

[FN09] Natalya Fridman Noy. Ontology mapping. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 573–590. Springer Berlin Heidelberg, 2009. Second Edition. (Cité page 20.)

[FNSW+09] Natalya Fridman Noy, Nigam H. Shah, Patricia L. Whetzel, Benjamin Dai, Michael Dorf, Nicholas Griffith, Clement Jonquet, Daniel L. Rubin, Margaret-Anne D. Storey, Christopher G. Chute, and Mark A. Musen. Bioportal : ontologies and integrated data resources at the click of a mouse. *Nucleic Acids Research*, 37(Web-Server-Issue) :170–173, 2009. (Cité pages 104 et 157.)

[GG95] Nicola Guarino and Pierdaniele Giaretta. Ontologies and knowledge bases : Towards a terminological clarification. *Towards Very Large Knowledge Bases : Knowledge Building and Knowledge Sharing*, pages 25–32, 1995. (Cité page 13.)

[GOS09] Nicola Guarino, Daniel Oberle, and Steffen Staab. What is an ontology ? In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 1–17. Springer Berlin Heidelberg, 2009. Second Edition. (Cité page 13.)

[Gru93] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5 :199–220, 1993. (Cité page 13.)

[GS03] Fausto Giunchiglia and Paval Shvaiko. Semantic matching. *The Knowledge Engineering Review*, 18(3) :265–280, 2003. (Cité page 22.)

[Ham50] Richard W. Hamming. Error detecting and error correcting codes. *The Bell System Technical Journal*, 29 :147–160, 1950. (Cité page 21.)

[HBS09] Alice Hertel, Jeen Broekstra, and Heiner Stuckenschmidt. RDF storage and retrieval systems. In Steffen Staab and

Rudi Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 489–508. Springer Berlin Heidelberg, 2009. Second Edition. (Cité page 15.)

[Hep08] Martin Hepp. Ontologies : State of the art, business potential, and grand challenges. In *Ontology Management, Semantic Web, Semantic Web Services, and Business Applications*, volume 7 of *Semantic Web And Beyond Computing for Human Experience*, pages 3–22. Springer, 2008. (Cité page 13.)

[HIM+04] Alon Y. Halevy, Zachary G. Ives, Jayant Madhavan, Peter Mork, Dan Suciu, and Igor Tatarinov. The piazza peer data management system. *IEEE Transactions on Knowledge and Data Engineering*, 16(7) :787–798, 2004. (Cité pages 61 et 154.)

[HPS04] Adil Hameed, Alun Preece, and Derek Sleeman. Ontology reconciliation. *Handbook on ontologies*, pages 231–250, 2004. (Cité pages 3, 20 et 24.)

[HRMC10] Farah Harrathi, Catherine Roussey, Loïc Maisonnasse, and Sylvie Calabretto. Vers une approche statistique pour l'indexation sémantique des documents multilingues. In *INFORSID*, pages 127–141, 2010. (Cité page 48.)

[HS08] Anthony Hunter and Konieczny Sébastien. Measuring inconsistency through minimal inconsistent sets. In *11th International Conference on Principles of Knowledge Representation and Reasoning*, pages 358–366, 2008. (Cité pages 98 et 101.)

[HWH+07] F. Holz, H.F. Witschel, Gregor Heinrich, G. Heyer, and S. Teresniak. An evaluation framework for semantic search in P2P networks. In *Proceedings of the I2CS 2007*, 2007. (Cité page 151.)

[IKKC05] Zachary Ives, Nitin Khandelwal, Aneesh Kapur, and Murat Cakir. Orchestra : Rapid, collaborative sharing of dynamic data. In *2nd Conference on Innovative Data Systems Research*, 2005. (Cité page 61.)

[Jac12] Paul Jaccard. The distribution of the flora in the alpine zone. *New Phytologist*, 11(2) :37–50, 1912. (Cité page 22.)

[Jar89] Matthew A. Jaro. Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical Association*, 84(406) :414–420, 1989. (Cité page 21.)

[JB05] Márk Jelasity and Ozalp Babaoglu. T-man : Gossip-based overlay topology management. In *In 3rd International Workshop on Engineering Self-Organising Applications (ESOA'05)*, pages 1–15. Springer-Verlag, 2005. (Cité pages 42 et 141.)

[JC97] J. J. Jiang and D. W. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. In *10th International Conference Research on Computational Linguistics*, pages 19–33, 1997. (Cité page 26.)

[JGKvS04] Márk Jelasity, Rachid Guerraoui, Anne-Marie Kermarrec, and Maarten van Steen. The peer sampling service : Experimental evaluation of unstructured gossip-based implementations. In *5th ACM/IFIP/USENIX International Middleware Conference*, pages 79–98, 2004. (Cité pages 41, 92 et 122.)

[JMB09] Márk Jelasity, Alberto Montresor, and Özalp Babaoglu. T-Man : Gossip-based fast overlay topology construction. *Computer Networks*, 53(13) :2321–2339, 2009. (Cité pages 42, 137 et 141.)

[KF01] Michel C. A. Klein and Dieter Fensel. Ontology versioning on the semantic web. In *SWWS*, pages 75–91, 2001. (Cité page 20.)

[KGZY02] Vana Kalogeraki, Dimitrios Gunopulos, and D. Zeinalipour-Yazti. A local search mechanism for peer-to-peer networks, 2002. (Cité page 36.)

[Kon05] Grzegorz Kondrak. N-gram similarity and distance. In *12th International Conference on String Processing and Information Retrieval*, pages 115–126, 2005. (Cité page 21.)

[Kro02] Amr Z. Kronfol. FASD : A fault-tolerant, adaptive, scalable, distributed search engine. Technical report, Princeton Uniservity, 2002. (Cité page 59.)

[KvS07] Anne-Marie Kermarrec and Maarten van Steen. Gossiping in distributed systems. *Operating Systems Review*, 41(5) :2–7, 2007. (Cité pages 40, 42, 122 et 157.)

[LCC+02] Qin Lv, Pei Cao, Edith Cohen, Kai Li, and Scott Shenker. Search and replication in unstructured peer-to-peer networks. In *16th ACM International Conference on Supercomputing*, pages 84–95, 2002. (Cité page 36.)

[Lev66] Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8) :707–710, 1966. (Cité pages 21 et 27.)

[Lin98] Dekang Lin. An information-theoretic definition of similarity. In *15th International Conference on Machine Learning*, pages 296–304, 1998. (Cité page 26.)

[LNS+03] Alexander Löser, Felix Naumann, Wolf Siberski, Wolfgang Nejdl, and Uwe Thaden. Semantic overlay clusters within super-peer networks. In *1st International Workshop, Databases, Information Systems, and Peer-to-Peer Computing*, pages 33–47, 2003. (Cité page 65.)

[LSK+08] Toan Luu, Gleb Skobeltsyn, Fabius Klemm, Maroje Puh, Ivana Podnar Zarko, Martin Rajman, and Karl Aberer. AlvisP2P : Scalable peer-to-peer text retrieval in a structured P2P network. *34th International Conference on Very Large Data Bases*, 1(2) :1424–1427, 2008. (Cité page 60.)

[MCH09] Kelly Moran, Kajal T. Claypool, and Benjamin J. Hescott. Compositematch : Detecting n-ary matches in ontology alignment. In *OM*, 2009. (Cité page 4.)

[ME96] Alvaro Monge and Charles Elkan. The field matching problem : Algorithms and applications. In *2nd International Conference on Knowledge Discovery and Data Mining*, pages 267–270, 1996. (Cité page 21.)

[MGMR02] Sergey Melnik, Hector Garcia-Molina, and Erhard Rahm. Similarity flooding : A versatile graph matching algorithm and its application to schema matching. In *18th International Conference on Data Engineering*, pages 117–128, 2002. (Cité page 23.)

[MH09] Ralf Möller and Volker Haarslev. Tableau-based reasoning. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 509–528. Springer Berlin Heidelberg, 2009. Second Edition. (Cité page 19.)

[Mil95] George A. Miller. Wordnet : a lexical database for english. *Communications of the ACM*, 38(11) :39–41, 1995. (Cité page 17.)

[MJ09] Alberto Montresor and Márk Jelasity. Peersim : A scalable P2P simulator. In *9th IEEE International Conference on Peer-to-Peer Computing*, pages 99–100, 2009. `http://peersim.sf.net`. (Cité pages 105 et 150.)

[MM02] Petar Maymounkov and David Mazières. Kademlia : A peer-to-peer information system based on the xor metric. In *IPTPS*, pages 53–65, 2002. (Cité page 38.)

[MNR02] Dahlia Malkhi, Moni Naor, and David Ratajczak. Viceroy : a scalable and dynamic emulation of the butterfly. In *21st ACM Symposium on Principles of Distributed Computing*, pages 183–192, 2002. (Cité page 38.)

[MPR00] Udi Manber, Ash Patel, and John Robison. The business of personalization : experience with personalization of yahoo ! *Communications of the ACM*, 43(8) :35–39, 2000. (Cité page 3.)

[MQH11] Yue Ma, Guilin Qi, and Pascal Hitzler. Computing inconsistency measure based on paraconsistent semantics. *Journal of Logic and Computation*, 21(6) :1257–1281, 2011. (Cité page 101.)

[MQHL07] Yue Ma, Guilin Qi, Pascal Hitzler, and Zuoquan Lin. Measuring inconsistency for description logics based on paraconsistent semantics. In *9th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pages 30–41, 2007. (Cité page 101.)

[MRS08a] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. Boolean retrieval. In *Introduction to information retrieval*, chapter 1, pages 1–18. Cambridge University Press, 2008. (Cité page 45.)

[MRS08b] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. Evaluation in information retrieval. In *Introduction to information retrieval*, chapter 8, pages 151–175. Cambridge University Press, 2008. (Cité page 51.)

[MRS08c] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*. Cambridge University Press, 2008. (Cité page 51.)

[MS02] Alexander Maedche and Steffen Staab. Measuring similarity between ontologies. In *13th International Conference on Knowledge Engineering and Knowledge Management*, pages 251–263, 2002. (Cité pages 22 et 27.)

[MTW05] Sebastian Michel, Peter Triantafillou, and Gerhard Weikum. KLEE : A framework for distributed top-k query algorithms. In *Proceedings of 31st Internal Conference on Very Large Data Bases*, pages 637–648, 2005. (Cité page 5.)

[NSCC07] Andronikos Nedos, Kulpreet Singh, Raymond Cunningham, and Siobhán Clarke. A gossip protocol to support service discovery with heterogeneous ontologies in MANETs. In *3rd IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*, pages 53–61, 2007. (Cité pages 64 et 158.)

[NSS03] Wolfgang Nejdl, Wolf Siberski, and Michael Sintek. Design issues and challenges for RDF- and schema-based peer-to-peer systems. *SIGMOD Record*, 32, 2003. (Cité pages 63 et 159.)

[NW70] Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3) :443–453, 1970. (Cité page 21.)

[NWQ+02] Wolfgang Nejdl, Boris Wolf, Changtao Qu, Stefan Decker, Michael Sintek, Ambjörn Naeve, Mikael Nilsson, Matthias Palmér, and Tore Risch. EDUTELLA : a P2P networking infrastructure based on RDF. In *11th International World Wide Web Conference*, pages 604–615, 2002. (Cité pages 39, 63 et 159.)

[NWS+04] Wolfgang Nejdl, Martin Wolpers, Wolf Siberski, Christoph Schmitz, Mario T. Schlosser, Ingo Brunkhorst, and Alexander Löser. Super-peer-based routing strategies for RDF-based peer-to-peer networks. *Journal of Web Semantic*, 1(2) :177–186, 2004. (Cité page 64.)

[Pan09] Jeff Z. Pan. Resource description framework. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 71–90. Springer Berlin Heidelberg, 2009. Second Edition. (Cité page 14.)

[PBMW99] Lawrence Edward Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking : Bringing order to the web, 1999. (Cité page 3.)

[PHG+00] Alun D. Preece, Kit-ying Hui, W. A. Gray, Philippe Marti, Trevor J. M. Bench-Capon, Dean M. Jones, and Zhan Cui.

The KRAFT architecture for knowledge fusion and transformation. *Knowledge-Based Systems*, 13(2-3) :113–120, 2000. (Cité page 61.)

[PLM+08] Wilma Penzo, Stefano Lodi, Federica Mandreoli, Riccardo Martoglia, and Simona Sassatelli. Semantic peer, here are the neighbors you want ! In *11th International Conference on Extending Database Technology*, pages 26–37, 2008. (Cité page 85.)

[PMd08] Silvio Peroni, Enrico Motta, and Mathieu d'Aquin. Identifying key concepts in an ontology, through the integration of cognitive principles with statistical and topological measures. In *3rd Asian Semantic Web Conference*, pages 242–256, 2008. (Cité pages 28 et 99.)

[Pow03] Shelley Powers. *Practical RDF*. O'Reilly & Associates, Inc., 2003. (Cité pages 3 et 14.)

[PS06] Éric Prud'hommeaux and Andy Seaborne. SPARQL query language for RDF. Technical report, 2006. (Cité page 15.)

[PSPS09] Carlos Eduardo S. Pires, Damires Souza, Thiago Pachêco, and Ana Carolina Salgado. A semantic-based ontology matching process for PDMS. In *2nd International Conference on Data Management in Grid and P2P Systems*, pages 124–135, 2009. (Cité page 64.)

[QF93] Yonggang Qiu and Hans-Peter Frei. Concept based query expansion. In *16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 160–169, 1993. (Cité page 25.)

[Rah11] Erhard Rahm. Towards large-scale schema and ontology matching. In *Schema Matching and Mapping*, pages 3–27. 2011. (Cité page 22.)

[RD01] Antony I. T. Rowstron and Peter Druschel. Pastry : Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *7th IFIP/ACM International Conference on Distributed Systems Platforms*, pages 329–350, 2001. (Cité page 38.)

[Rei87] Raymond Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1) :57–95, 1987. (Cité page 99.)

[Res95] Philip Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *In Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 448–453, 1995. (Cité page 26.)

[RFH+01] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard M. Karp, and Scott Shenker. A scalable content-addressable network. In *SIGCOMM*, pages 161–172, 2001. (Cité page 38.)

[RFI02] Matei Ripeanu, Ian Foster, and Adriana Iamnitchi. Mapping the gnutella network : Properties of large-scale peer-

to-peer systems and implications for system design. 2002. (Cité page 35.)

[RK02] Sean C. Rhea and John Kubiatowicz. Probabilistic location and routing. In *21st Annual Joint Conference of the IEEE Computer and Communications Societies*, pages 1248–1257, 2002. (Cité page 36.)

[RMBB89] R. Rada, H. Mili, E. Bicknell, and M. Blettner. Development and application of a metric on semantic nets. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(1) :17–30, 1989. (Cité page 26.)

[RP08] P. Raftopoulou and E.G.M. Petrakis. iCluster : a self-organizing overlay network for P2P information retrieval. In *Proceedings of 30th European ECIR Conference*, Glasgow, Scotland, 2008. (Cité page 59.)

[RSJ76] Stephen E. Robertson and Karen Spärck Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3) :129–146, 1976. (Cité page 50.)

[RSM94] R. Richardson, A. F. Smeaton, and J. Murphy. Using wordnet as a knowledge base for measuring semantic similarity between words. Technical Report CA-1294, Dublin City University, 1994. (Cité page 26.)

[RWHB00] Stephen E. Robertson, Steve Walker, and Micheline Hancock-Beaulieu. Experimentation as a way of life : Okapi at trec. *Information Processing and Management*, 36(1) :95–108, 2000. (Cité page 50.)

[Saï07] Fatiha Saïs. *Intégration Sémantique de Données guidée par une Ontologie*. PhD thesis, Université Paris-Sud, 2007. (Cité page 24.)

[Sal90] Gerard Salton. Full text information processing using the smart system. *IEEE Data Engineering Bulletin*, 13(1) :2–9, 1990. (Cité page 52.)

[SE08] Paul R. Smart and Paula C. Engelbrecht. An analysis of the origin of ontology mismatches on the semantic web. In *EKAW*, pages 120–135, 2008. (Cité pages 3 et 19.)

[SFW83] Gerard Salton, Edward A. Fox, and Harry Wu. Extended boolean information retrieval. *Communications of the ACM*, 26(11) :1022–1036, 1983. (Cité page 49.)

[SGG03] Stefan Saroiu, Krishna P. Gummadi, and Steven D. Gribble. Measuring and analyzing the characteristics of napster and gnutella hosts. *Multimedia Systems*, 9(2) :170–184, 2003. (Cité pages 3, 33, 34, 35, 114 et 139.)

[SGY09] Pavel Shvaiko, Fausto Giunchiglia, and Mikalai Yatskevich. Semantic matching with s-match. In *Semantic Web Information Management*, pages 183–202. 2009. (Cité page 25.)

[Sha48] Claude Elwood Shannon. A mathematical theory of communication. *Bell Systems Technical Journal*, 27 :379–423,623–656, 1948. (Cité page 77.)

[SHCVH07] Stephan Schlobach, Zhisheng Huang, Ronald Cornet, and Frank Van Harmelen. Debugging incoherent terminologies. *Journal of Automated Reasoning*, 39(3) :317–349, 2007. (Cité page 100.)

[Sim49] E.H. Simpson. Measurement of diversity. *Nature*, 163 :688, 1949. (Cité page 77.)

[SMK+01] Ion Stoica, Robert Morris, David R. Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord : A scalable peer-to-peer lookup service for internet applications. In *7th Conference of the Special Interest Group on Data Communication*, pages 149–160, 2001. (Cité page 37.)

[SMLN+03] Ion Stoica, Robert Morris, David Liben-Nowell, David R. Karger, M. Frans Kaashoek, Frank Dabek, and Hari Balakrishnan. Chord : a scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Transactions on Networking*, 11(1) :17–32, 2003. (Cité page 3.)

[SPG+07] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet : A practical OWL-DL reasoner. *Web Semantics : science, services and agents on the World Wide Web*, 5(2) :51–53, 2007. (Cité pages 19 et 98.)

[SS04] Steffen Staab and Studi Studer. *Handbook on Ontologies*. Springer, 2004. (Cité page 3.)

[SVH04] Nuno Seco, Tony Veale, and Jer Hayes. An intrinsic information content metric for semantic similarity in wordnet. In *16th Eureopean Conference on Artificial Intelligence*, pages 1089–1090, 2004. (Cité page 26.)

[Tch12] Andon Tchechmedjiev. État de l'art : mesures de similarité sémantique locales et algorithmes globaux pour la désambiguïsation lexicale à base de connaissances. In *Conférence JEP-TALN-RECITAL 2012*, pages 295–308, 2012. (Cité page 83.)

[TH06] Dmitry Tsarkov and Ian Horrocks. FaCT++ description logic reasoner : System description. In Ulrich Furbach and Natarajan Shankar, editors, *Automated Reasoning*, volume 4130 of *Lecture Notes in Computer Science*, pages 292–297. Springer Berlin / Heidelberg, 2006. (Cité pages 19 et 98.)

[TIM+03] Igor Tatarinov, Zachary G. Ives, Jayant Madhavan, Alon Y. Halevy, Dan Suciu, Nilesh N. Dalvi, Xin Dong, Yana Kadiyska, Gerome Miklau, and Peter Mork. The piazza peer data management project. *SIGMOD Record*, 32(3) :47–52, 2003. (Cité page 61.)

[TR03] Dimitrios Tsoumakos and Nick Roussopoulos. Adaptive probabilistic search for peer-to-peer networks. In *3rd International Conference on Peer-to-Peer Computing*, pages 102–109, 2003. (Cité page 36.)

[VCC+10] Anthony Ventresque, Thomas Cerqueus, Louis-Alexandre Celton, Gaëtan Hervouet, Damien Levin, Philippe Lamarre,

and Sylvie Cazalens. Mysins : make your semantic information system. In *Extraction et Gestion de Connaissances*, pages 629–630, 2010. (Cité page 48.)

[VCLV08] Anthony Ventresque, Sylvie Cazalens, Philippe Lamarre, and Patrick Valduriez. Improving interoperability using query interpretation in semantic vector spaces. In *5th European Semantic Web Conference*, pages 539–553, 2008. (Cité pages 3, 25, 48 et 83.)

[VE97] Petko Valtchev and Jérôme Euzenat. Dissimilarity measure for collections of objects and values. In *2nd International Symposium on Advances in Intelligent Data Analysis, Reasoning about Data*, pages 259–272, 1997. (Cité page 21.)

[Voo94] Ellen M. Voorhees. Query expansion using lexical-semantic relations. In *17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 61–69, 1994. (Cité pages 3 et 48.)

[vSB+05] O. Šváb, V. Svátek, P. Berka, D. Rak, and P. Tomášek. Ontofarm : Towards an experimental collection of parallel ontologies. In *5th International Semantic Web Conference*, 2005. Poster Track. (Cité page 116.)

[WDKF02] Steve R. Waterhouse, David M. Doolin, Gene Kan, and Yaroslav Faybishenko. Distributed search in P2P networks. *IEEE Internet Computing*, 6(1) :68–72, 2002. (Cité page 59.)

[Win99] William E. Winkler. The state of record linkage and current research problems. Technical report, Statistical Research Division, U.S. Census Bureau, 1999. (Cité page 21.)

[WP94] Zhibiao Wu and Martha Palmer. Verb semantics and lexical selection. In *32nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 133–138, 1994. (Cité page 26.)

[WZH+10] Zhichun Wang, Xiao Zhang, Lei Hou, Yue Zhao, Juanzi Li, Yu Qi, and Jie Tang. RiMOM results for OAEI 2010. In *5th International Workshop on Ontology Matching*, 2010. (Cité page 25.)

[WZW85] S. K. M. Wong, Wojciech Ziarko, and Patrick C. N. Wong. Generalized vector spaces model in information retrieval. In *8th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 18–25, 1985. (Cité page 48.)

[YGM02] Beverly Yang and Hector Garcia-Molina. Improving search in peer-to-peer networks. In *22nd International Conference on Distributed Computing Systems*, pages 5–14, 2002. (Cité page 36.)

[ZHS+04] Ben Y. Zhao, Ling Huang, Jeremy Stribling, Sean C. Rhea, Anthony D. Joseph, and John Kubiatowicz. Tapestry : a resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications*, 22(1) :41–53, 2004. (Cité pages 3 et 38.)

Contributions au problème d'hétérogénéité sémantique dans les systèmes pair-à-pair : application à la recherche d'information

Nous considérons des systèmes pair-à-pair (P2P) pour le partage de données dans lesquels chaque pair est libre de choisir l'ontologie qui correspond le mieux à ses besoins pour représenter ses données. Nous parlons alors d'hétérogénéité sémantique. Cette situation est un frein important à l'interopérabilité car les requêtes émises par les pairs peuvent être incomprises par d'autres. Dans un premier temps nous nous focalisons sur la notion d'hétérogénéité sémantique. Nous définissons un ensemble de mesures permettant de caractériser finement l'hétérogénéité d'un système suivant différentes facettes. Dans un deuxième temps nous définissons deux protocoles. Le premier, appelé CorDis, permet de réduire l'hétérogénéité sémantique liée aux disparités entre pairs. Il dissémine des correspondances dans le système afin que les pairs apprennent de nouvelles correspondances. Le second protocole, appelé GoOD-TA, permet de réduire l'hétérogénéité sémantique d'un système liée à son organisation. L'objectif est d'organiser le système de sorte que les pairs proches sémantiquement soient proches dans le système. Ainsi deux pairs deviennent voisins s'ils utilisent la même ontologie ou s'il existe de nombreuses correspondances entre leurs ontologies respectives. Enfin, dans un trois temps, nous proposons l'algorithme DiQuESH pour le routage et le traitement de requêtes top-k dans les systèmes P2P sémantiquement hétérogènes. Cet algorithme permet à un pair d'obtenir les k documents les plus pertinents de son voisinage. Nous montrons expérimentalement que les protocoles CorDis et GoOD-TA améliorent les résultats obtenus par DiQuESH.

Mots-clés : Système P2P, hétérogénéité, sémantique, ontologie, interopérabilité, recherche d'information.

Contributions to the problem of semantic heterogeneity in peer-to-peer systems : application to information retrieval

We consider peer-to-peer (P2P) data sharing systems in which each peer is free to choose the ontology that best fit its needs to represent its data. This is what we call semantic heterogeneity. This situation prevents from perfect interoperability because queries issued by peers may be misunderstood by other peers. First we focus on the notion of semantic heterogeneity because it seems to us that it is a complex notion. We define several measures allowing to precisely characterize semantic heterogeneity of a P2P system according to different facets. Second we define two protocols. The first one, called CorDis, allows to reduce semantic heterogeneity related to the disparities between peers. It disseminates correspondences in the system so that peers learn new correspondences. The second protocol, called GoOD-TA, allows to reduce semantic heterogeneity related to the topology of a system. The goal is to organize it in way that semantically close peers are close in the system. Thus two peers are neighbours if they use the same ontology, or if numerous correspondences exist between their respective ontologies. Third we propose an algorithm called DiQuESH for the routing and the treatment of top-k queries in semantically heterogeneous P2P systems. This algorithm allows a peer to retrieve the k most relevant documents from its neighbourhood. We experimentally show that CorDis and GoOD-TA improve results obtained by DiQuESH.

Keywords: P2P system, heterogeneity, semantics, ontology, interoperability, information retrieval.