

ALGORITHMS FOR COMPLETE, EFFICIENT, AND SCALABLE ALIGNMENT OF LARGE ONTOLOGIES

by

UTHAYASANKER THAYASIVAM

(Under the direction of Prashant Doshi)

ABSTRACT

As ontology repositories proliferate on the web, many contain ontologies that overlap in scope. Ontology alignment (OA) is the process of identifying this overlap, which is important for the discovery and exchange of knowledge. Consequently, aligning ontologies gains importance. OA algorithms are faced with crucial challenges: improving the correctness and completeness of the alignment, scaling to large ontologies and quickly producing the alignment without compromising its quality. In this dissertation, we present algorithms for complete, efficient and scalable ontology alignment.

Many existing algorithms unconditionally utilize lexicons such as, WordNet for the potential improvement in the alignment accuracy. We empirically analyzed the impact on alignment quality and execution time when using WordNet for OA. We provide useful insights on the types of ontology pairs for which WordNet-based alignment is potentially worthwhile. We also noticed that many algorithms either do not consider the complex concepts in their alignment procedures or model them naively. We introduce axiomatic and graphical canonical forms for modeling value and cardinality restrictions and Boolean combinations, and present a similarity-measure for them. OA algorithms may utilize this approach to model complex concepts for participation in the alignment process. Our results indicate a significant improvement in the quality of the alignment produced.

Several algorithms use iterative approaches for better alignment quality though they consume more time than others. We present a novel and general approach to speed up the convergence of the iterative OA algorithms to produce similar or improved alignment using block-coordinate descent (BCD) technique. We also provide useful insights on how to identify an appropriate partitioning and ordering scheme for a given algorithm. As ontologies are submitted or updated in repositories, their alignment with others must be quickly computed. We project the problem of aligning several pairs of ontologies as that of batch alignment and demonstrate dramatic speedup in the alignment using the distributed computing paradigm of MapReduce. Using a representative set of algorithms; we empirically analyzed and evaluated the performance of all the approaches presented. This dissertation introduces algorithms and insights for OA algorithms to scale up for large ontologies and efficiently align them.

INDEX WORDS: Scalability, Ontology alignment, MapReduce, WordNet, Optima+, Complex Concepts, Parallelization

ALGORITHMS FOR COMPLETE, EFFICIENT, AND SCALABLE ALIGNMENT OF
LARGE ONTOLOGIES

by

UTHAYASANKER THAYASIVAM

B.Sc Eng. University of Moratuwa, Sri Lanka, 2006

A Dissertation Submitted to the Graduate Faculty
of The University of Georgia in Partial Fulfillment
of the
Requirements for the Degree

DOCTOR OF PHILOSOPHY

ATHENS, GEORGIA

2013

© 2013

UTHAYASANKER THAYASIVAM

All Rights Reserved

ALGORITHMS FOR COMPLETE, EFFICIENT, AND SCALABLE ALIGNMENT OF
LARGE ONTOLOGIES

by

UTHAYASANKER THAYASIVAM

Approved:

Major Professor: Prashant Doshi

Committee: John A. Miller
Krzysztof J. Kochut
T.N. Sriram

Electronic Version Approved:

Maureen Grasso
Dean of the Graduate School
The University of Georgia
August 2013

DEDICATION

To my dad (appa) Dr.R.Thayasivam, mom (amma) Mrs.T.Naguleswary and brothers.

ACKNOWLEDGMENTS

There are several people who have aided me directly or indirectly in my journey through the rigors of accomplishing this dissertation. First and foremost, I would like to express my sincere gratitude to my advisor, Prof. Prashant Doshi for his expert guidance, support and motivation. I am very much grateful to him for giving me an opportunity to carry out an interesting research work in Semantic Computing and for his constant and distinct encouragement throughout my research work. Second, I would like to thank my committee members, Prof. John A. Miller, Prof. Krzysztof J. Kochut and Prof. T.N. Sriram for their numerous suggestions and help. Third, a special thanks to all my lab-mates (THINCers), especially Ekhlās, Muthu, Roi and Tejas, for their friendship and constant support. Among them I should explicitly mention Tejas who was part of some of my research efforts and interesting philosophical debates. It is also my duty to thank the supporting staff in the Boyd Graduate Studies Research Center and Department of Computer Science, UGA for their assistance in many ways. Many thanks to National Heart, Lung, And Blood Institute for providing me a research assistantship from the grant number R01HL087795. Finally, I acknowledge my indebtedness to my family members – Amma, Appa, Anna, Gobi-anna and Uma-anna – for their love, encouragement, and support throughout my life. Special thanks to my love – Janani – for helping me to handle the pressure especially during the final few months.

PREFACE

My dissertation research focuses on principled ways of scaling the automated alignment of ontologies without compromising on the quality of the alignment. The wealth of ontologies and many of those overlap in their scope, have made aligning ontologies an important problem for the semantic Web. Crucial challenges for the alignment algorithms involve scaling to large ontologies and performing the alignment in a reasonable amount of time without compromising on the quality of the alignment. Though ontology alignment is traditionally perceived as an offline and one-time task, the second challenge is gaining importance, especially, continuously evolving ontologies and applications involving real-time ontology alignment such as semantic search and Web service composition stress the importance of computational complexity considerations. My research focuses on identifying techniques to improve the efficiency and scalability of ontology alignment task.

Jointly with my advisor Prof. Prashant Doshi, I have endeavored to disseminate the research outcome by means of workshops, conferences, journal and posters submissions. The list of papers given below along with this dissertation forms an accurate description of the work that I have completed towards my dissertation.

Publication List

1. Uthayasanker Thayasivam, Prashant Doshi, “Improved Efficiency of Iterative Ontology Alignment using Block-Coordinate Descent”, in *Journal of Artificial Intelligence Research (JAIR)*, under review.
2. Uthayasanker Thayasivam, Prashant Doshi, “Speeding up Batch Alignment of Large Ontologies Using MapReduce”, in *International Conference on Semantic Computing (ICSC) 2013*.

3. Tejas Chaudhari, Uthayasanker Thayasivam, Prashant Doshi, “Canonical Forms and Similarity of Complex Concepts for Improved Ontology Alignment”, in *International Conference on Web Intelligence (WI) 2013*.
4. Uthayasanker Thayasivam, Prashant Doshi, “Optima+’s Results in OAEI 2012”, in *Ontology Matching (OM) workshop in International Semantic Web Conference (ISWC)*, Boston, MA USA, November 2012, pp 204 – 211.
5. Uthayasanker Thayasivam, Prashant Doshi, “Improved Convergence of Iterative Ontology Alignment using Block-Coordinate Descent”, in *26th International Conference of Association for the Advancement of Artificial Intelligence (AAAI)*, Toronto, Canada, September 2012, pp. 150 – 156.
6. Uthayasanker Thayasivam, Prashant Doshi, “On the Utility of WordNet for Ontology Alignment: Is it Really Worth It?”, in *IEEE International Conference on Semantic Computing (ICSC)*, Palo Alto, California, USA, September 2011, pp. 267 – 274.
7. Uthayasanker Thayasivam, Prashant Doshi, “Optima’s Results in OAEI 2011”, in *Ontology Matching (OM) workshop in International Semantic Web Conference (ISWC)*, Bonn, Germany, October 2011, pp. 204 – 211.
8. Uthayasanker Thayasivam, Kunal Verma, Alex Kass, Reymonrod Vasquez, in “Automatically Mapping Natural Language Requirements to Domain-Specific Process Models” *Innovative Applications Of Artificial Intelligence Conference (IAAI)*, San Francisco, August 2011, pp. 1695 – 1700.

TABLE OF CONTENTS

	Page
ACKNOWLEDGMENTS	v
PREFACE	vi
LIST OF FIGURES	xi
LIST OF TABLES	xiv
CHAPTER	
1 INTRODUCTION	1
1.1 Ontology Alignment	1
1.2 Sources Of Complexity	3
1.3 Biomedical Ontology Alignment	7
1.4 Optima+ And Its Performance In OAEI	9
1.5 Contributions	16
1.6 Dissertation Organization	21
2 BACKGROUND AND RELATED WORK	23
2.1 Alignment Problem	23
2.2 Architecture	24
2.3 Survey of Automated Alignment Algorithms	30
2.4 Scalable Alignment Algorithms	40
3 ON THE UTILITY OF WORDNET FOR ONTOLOGY ALIGNMENT	44
3.1 WordNet And Ontology Alignment	45
3.2 Integrating WordNet	47

3.3	Experiments	50
3.4	Recommendations	55
4	MODELING COMPLEX CONCEPTS FOR COMPLETE ONTOLOGY ALIGN- MENT	57
4.1	OWL 2 to RDF Graph Transformation	60
4.2	Representative Alignment Algorithms	61
4.3	Modeling Complex Concepts Using Canonical Representation	61
4.4	Computing Similarity between Canonical Representation	68
4.5	Integrating Complex Concepts	70
4.6	Experiments	71
4.7	Discussion	73
5	SPEEDING UP CONVERGENCE OF ITERATIVE ONTOLOGY ALIGNMENT . .	74
5.1	Representative Alignment Algorithms	77
5.2	Block-Coordinate Descent	78
5.3	Integrating BCD into Iterative Alignment	80
5.4	Empirical Analysis	87
5.5	Optimizing BCD using Partitioning and Ordering Schemes	92
5.6	Discussion	100
6	BATCH ALIGNMENT OF LARGE ONTOLOGIES USING MAPREDUCE	102
6.1	Representative Algorithms	104
6.2	Overview of MapReduce Paradigm	105
6.3	Distributed Ontology Alignment Using MapReduce	107
6.4	MapReduce Algorithm	110
6.5	Performance Evaluation	111
6.6	Discussion	115
7	LARGE BIOMEDICAL ONTOLOGY ALIGNMENT	118

7.1	Improvement Using Complex Concepts Modeling	120
7.2	Evaluating Using BCD Enhanced Algorithms	122
7.3	Scaling Using MapReduce Paradigm	123
8	CONCLUSIONS AND FUTURE WORK	128
8.1	Conclusions	128
8.2	Future Work	131
	BIBLIOGRAPHY	134
	Appendix	
A	ONTOLOGIES USED IN OUR EVALUATIONS	145
B	ADDITIONAL RESULTS ON WORDNET UTILITY	155

LIST OF FIGURES

1.1	An alignment between parasite experiment ontology and ontology of biomedical investigations.	2
2.1	The general architecture of ontology alignment process.	26
2.2	An example redundant correspondence and an example inconsistent correspondence.	27
2.3	Iterative approach.	28
2.4	General algorithms for iterative update, and search approaches toward aligning ontologies	29
2.5	Iterative update in the structural matcher, GM , in Falcon-AO	33
2.6	Iterative search in MapPSO. Objective function, Q , is as given in Eq. 2.4.	35
2.7	OLA's alignment algorithm iteratively updates the alignment matrix using a combination of neighboring similarity values.	36
2.8	Optima 's expectation-maximization based iterative search; it uses binary matrix, M^i , to represent an alignment. The objective function, Q , is as defined in Eq. 2.6.	38
3.1	All four synsets of term <i>sample</i> in WordNet are illustrated.	46
3.2	Integrated similarity measure	49
3.3	(a) Final recall and (b) final F-measure generated by Optima on 6 representative ontology pairs, with the integrated similarity measure and with just the syntactic similarity between entity labels.	51
3.4	Recall and F-measure for 6 of the 23 ontology pairs that I used in my evaluations.	52
4.1	People and Animal ontologies that classify people and animals respectively	58
4.2	The nodes and edges in bold constitute the canonical form RDF subgraph for value restrictions	63
4.3	Canonical RDF graph representation of cardinality restrictions.	65

4.4	The nodes and edges in bold constitute the canonical form subgraph for a Boolean combination	67
5.1	General iterative algorithms are modified to obtain, iterative update enhanced with BCD, and iterative search enhanced with BCD	82
5.2	Iterative update in GMO modified to perform BCD.	84
5.3	OLA's BCD-integrated iterative ontology alignment algorithm.	86
5.4	Average execution times of the four iterative algorithms	89
5.5	Average execution time consumed by, (a) Falcon-AO , (b) MapPSO , (c) OLA , and (d) Optima in their original form and with BCD	91
5.6	Average execution times of, (a) Falcon-AO , (b) OLA , and (c) Optima , with BCD that uses the initial ordering scheme and with BCD ordering the blocks from root(s) to leaves	93
5.7	Average execution time consumed by, (a) Falcon-AO , (b) OLA and (c) Optima with BCD utilizing the previous ordering scheme and with BCD ordering the blocks by similarity distribution	94
5.8	Partitioning schemes.	96
5.9	Execution times consumed by, (a) Falcon-AO , (b) OLA , and (c) Optima with BCD that uses blocks obtained by partitioning a single ontology and with BCD that utilizes partitions of both the ontologies	97
5.10	Execution times consumed by, (a) Falcon-AO , (b) OLA , and (c) Optima , with BCD that uses the default partitioning approach and with BCD that uses subtree-based partitioning	99
6.1	The MapReduce framework for ontology alignment.	106
6.2	Two types of inconsistent correspondences, which must be resolved while merging subproblem alignments.	109
6.3	Average execution times of Falcon-AO , Logmap , Optima+ , and YAM++ in their original form on a single node and using MapReduce	112

6.4	The plot demonstrates the exponential decaying of average total execution time with increasing number of nodes by Falcon-AO , Logmap , Optima+ , and YAM++ for large ontologies from OAEI.	116
7.1	Performance on the biomedical testbed.	121
7.2	Total recall (left y-axis) attained and total time (right y-axis) consumed, by Falcon-AO and Optima with optimized BCD for 50 and 26 pairs of our large biomedical ontology testbed	124
7.3	Average execution times of Falcon-AO , Logmap , Optima+ , and YAM++ in their original form on a single node and using MapReduce	125
7.4	The plot demonstrates the exponential decaying of average total execution time with increasing number of nodes by Falcon-AO , Logmap , Optima+ , and YAM++ for biomedical ontologies.	126
B.1	Recall and F-measure for 2 ontology pairs of the same trend where the final recall and F-measure with WN integrated is higher than the recall and F-measure with just syntactic similarity.	156
B.2	Recall and F-measure for 2 ontology pairs of the same trend where the final recall and F-measure with WN integrated did not improve on the recall and F-measure without WN.	157
B.3	Both the ontology pairs shown here exhibit a final recall with WordNet that is same as the recall without it. However, the F-measure with WordNet is less than the F-measure without WordNet.	158

LIST OF TABLES

1.1	Average recall, precision, and F-measure of Optima+ in OAEI 2012 for benchmark track. Note Optima+ performs well in test cases in the range of 201-247. . . .	13
1.2	Comparison between the performances of Optima+ in OAEI 2012 and Optima in OAEI 2011 for conference track. Optima+ significantly improved its alignment quality and efficiency.	14
1.3	Comparison between the performances of top 4 alignment algorithms (YAM++ , Logmap , CODI, and Optima+) in OAEI 2012 for conference track.	15
3.1	The different ontology pairs could be grouped into 4 trends of alignment performance based on the recall and F-measure evaluations.	54
6.1	The precision (P), recall (R) and F-measure (F) of the output alignments by Falcon-AO , Logmap , Optima+ , and YAM++ in MapReduce setup for the large ontology pairs from OAEI.	113
A.1	Ontologies from OAEI’s benchmark and conference tracks participating in our evaluation and the number of named classes, complex concepts and properties in each.	145
A.2	Large ontologies from OAEI 2012 used in our evaluations and the number of named classes and properties in each of them.	146
A.3	Selected ontologies from NCBO in the biomedical ontology alignment testbed 1 and the number of named classes and properties in each.	147
A.4	The biomedical ontology pairs in our testbed 1 sorted in terms of $ V_1 \times V_2 $. This metric is illustrative of the complexity of aligning the pair.	148

- A.5 Selected ontologies from NCBO in the biomedical ontology alignment testbed 2 and the number of named classes, anonymous classes and different type of properties in each. 151
- A.6 The 35 biomedical ontology pairs from our second testbed are listed above using their NCBO acronym. These ontologies contains significant amount of complex concepts within them. 153

CHAPTER 1

INTRODUCTION

The growing usefulness of the semantic Web is fueled in part by the development and publication of an increasing number of ontologies. Ontologies are formalizations of commonly agreed upon knowledge, often specific to a domain. An ontology consists of a set of concepts (classes) and relationships (properties) between the concepts. As opposed to having a centralized repository of ontologies, we witness a growth of disparate communities of ontologies that cater the specific applications [50, 63, 92]. Naturally, many of these communities contain ontologies that describe the same or overlapping domains but use different names for concepts and may exhibit varying structure. For example, the National Center for Biomedical Ontologies (NCBO) [63] currently hosts more than 320 ontologies pertaining to the life sciences. Among these ontologies, about 30% have more than 2,000 entities and relationships, making them very large in size. Because many of these ontologies overlap in their scope, aligning ontologies is important for the utility of the repositories [2] and several semantic web applications [42].

1.1 Ontology Alignment

The ontology alignment problem is to find a set of correspondences between two ontologies, $\mathcal{O}_1$ and $\mathcal{O}_2$. A correspondence, $m_{a\alpha}$, between two entities, $x_a \in \mathcal{O}_1$ and $y_\alpha \in \mathcal{O}_2$ consists of a relation, $r \in \{=, \subseteq, \supseteq\}$, and confidence, $c \in \mathbb{R}$. A partial alignment between two ontologies – parasite experiment ontology (PEO) and ontology for biomedical investigations (OBI), hosted by NCBO, is illustrated in Fig. 1.1. It shows mappings between classes created by the Agreement-Maker [17] tool. Identifying an equivalence correspondence between the nodes **peo:region** from PEO and **obi:region** from OBI is trivial since they share the same label. However, identifying

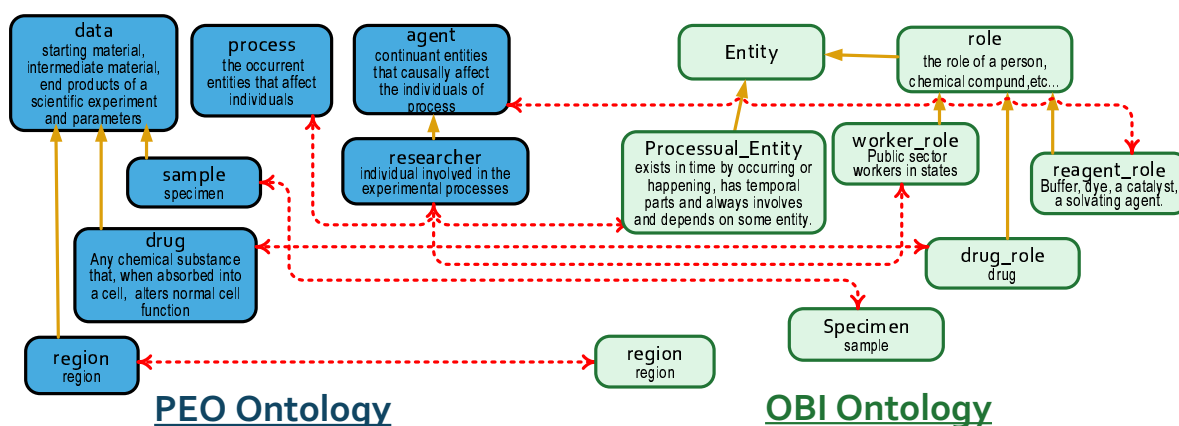


Figure 1.1: Alignment (shown in dashed red) between portions of the parasite experiment ontology (PEO) and the ontology of biomedical investigations (OBI) as discovered by an automated algorithm called AgreementMaker [17]. Both of these ontologies are available at NCBO. Each identified map in the alignment signifies an equivalence relation between the concepts.

that **peo:sample** and **obi:specimen** are equivalent is not straightforward, yet it could be achieved with the help of a lexical database (e.g. WordNet or UMLS). Finding the correspondence between **peo:sample** and **obi:drug_role** is even more challenging since their association is not present even in lexical databases such as WordNet or UMLS.

Several ontology alignment algorithms [12, 20, 24, 35, 45–47, 51, 66] are now available that utilize varying techniques to semi- or fully automatically generate mappings between entities in the ontology pair. They can be broadly classified based on 1) the level of human intervention needed 2) the amount of prior training needed 3) the way ontologies are modeled and 4) the selection of similarity measures used. Over 50 ontology alignment algorithms have been submitted to the ontology alignment evaluation initiative (OAEI) with mix successes in their performances. Despite the increasing number of alignment approaches, modern large scale ontologies still pose serious challenges to existing ontology matching tools.

1.2 Sources Of Complexity

Crucial challenges for the ontology alignment algorithms involve improving the alignment quality, performing the alignment in a reasonable amount of time without compromising on the quality of the alignment and scaling to large ontologies. Quality of an alignment is twofold – correctness and coverage. Correctness of an alignment is measured by percentage of the correct correspondences in an alignment. This measure is called *precision* which is defined in Eq. 1.1. The *recall* of an alignment depicted in Eq. 1.2 is a ratio between the number of correct correspondences in an alignment and the total number of correct correspondences between the ontologies, which measures the coverage of an alignment. A collective measure of both correctness and coverage of an alignment is known as F_β -measure, which is a harmonic mean of precision and recall, defined in Eq. 1.3. F_β -measure indicates the quality of an alignment where the weight of precision and recall can be controlled using the positive real-value parameter β . When β is set to one both precision and recall get the same importance, then it is also known as F-measure. If it gets higher than 1 then recall gains more importance than precision and when it gets lower than 1 precision gets more importance than recall.

$$\text{Precision} = \frac{\text{Number of correct correspondences in an alignment}}{\text{Total number of correspondences in an alignment}} \quad (1.1)$$

$$\text{Recall} = \frac{\text{Number of correct correspondences in an alignment}}{\text{Total number of correct correspondences between the ontologies}} \quad (1.2)$$

$$F_\beta\text{-measure} = (1 + \beta^2) \cdot \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + (\beta^2 \cdot \text{Precision})} \quad (1.3)$$

1.2.1 Producing Quality Alignment

Existing ontology alignment approaches rely heavily on lexical attributes of entities such as internationalized resource identifiers (IRI), labels, and descriptive comments to identify correspondences between them. Additionally structures of the ontologies are also exploited in the alignment

process. Often, alignment algorithms are given ontologies from similar domains to compute an alignment. Yet, producing a quality alignment is challenging due to the lexical and structural disparity between the ontologies. Because these ontologies are developed independently they exhibit significant difference in structuring and naming. For example, as shown in Fig. 1.1, the entities *sample* and *specimen* from the ontologies PEO and OBI respectively render the same concept using different naming and structure. Note, in PEO *sample* is a subclass of *data* but OBI defines *specimen* without any super classes.

Many ontology alignment algorithms augment syntactic matching with the use of WordNet in order to improve their performance. For example, identifying the equivalence correspondence between **peo:sample** and **obi:specimen** from the PEO and OBI ontologies shown in Fig. 1.1 becomes possible with WordNet. Specifically, alignment algorithms [20,47,51,66] utilize WordNet due to the potential improvement in recall of the alignment. However, we strike a more cautionary note. We analyze the utility of WordNet in the context of the reduction in precision and increase in execution time that its use entails. We report distinct trends in the performance of WordNet-based alignment in comparison with alignment that uses syntactic matching only. We analyze the trends and their implications, and provide useful insights on the types of ontology pair for which WordNet-based alignment may potentially be worthwhile and those types where it may not be. This study and the useful insights of its results are presented in Chapter 4.

Alignment algorithms primarily focus on lexical attributes and neighboring named entities when evaluating a correspondence between a pair of entities. Many of them either do not consider the complex concepts in their alignment procedures or model them naively, thereby producing a possibly incomplete alignment. We introduce axiomatic and graphical canonical forms for modeling value and cardinality restrictions and Boolean combinations, and present a way of measuring the similarity between these complex concepts in their canonical forms. We show how our approach may be integrated in multiple ontology alignment algorithms. Our results indicate a significant improvement in the F-measure of the alignment produced by these algorithms. However,

this improvement is achieved at the expense of increased run time due to the additional concepts modeled. Our approach and its performance evaluation are presented in Chapter 4.

1.2.2 Efficient and Scalable Alignment

Traditionally, ontology alignment is perceived as an offline and a one-time task. However, efficiency and scalability of ontology alignment are gaining more importance. In particular, as Hughes and Ashpole [42] note, continuously evolving ontologies and applications involving real-time ontology alignment, such as semantic search and Web service composition, stress the importance of computational complexity considerations. Additionally, established benchmarks, such as the OAEI [83], recently began reporting the execution times of the participating alignment systems as well. In last year's OAEI campaign [84], out of 21 total participants, only 13 tools participated in large ontology matching tracks namely, the library and large biomedical ontology tracks. Especially in the large biomedical ontology track, only 8 tools were able to complete the tasks. Moreover, OAEI points out that the sizes of the input ontologies significantly affect the efficiency of many tools. Clearly, despite the prior investigations on matching larger ontologies, there is still significant room for improvement in ontology alignment algorithms in terms of their scalability.

Key challenges for making ontology alignment computationally feasible involve managing its alignment space growing exponential to the sizes of the ontologies and improving the alignment efficiency. In general there may be $2^{|V_1| \cdot |V_2|} + 2^{|E_1| \cdot |E_2|}$ different alignments in aligning the ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$. Here, $|V_i|$ denote the number of concepts in an ontology $\mathcal{O}_i$ using $|V_i|$ and the number of properties from the same ontology using $|E_i|$. An important challenge for alignment algorithms is to search this space of alignments which grows exponentially with the sizes of the ontologies. Regularly, alignment algorithms restrict their focus to either many-to-one or one-to-one mapping to reduce the search space. In the case of many-to-one the space shrinks to $(|V_1| + 1)^{|V_2|} + (|E_1| + 1)^{|E_2|}$. The space get even smaller $-(|V_1| + 1)! / (|V_1| - |V_2|)! + (|E_1| + 1)! / (|E_1| - |E_2|)!$ with one-to-one restriction. Here, without lose of generality we assumed that $|V_1| \geq |V_2|$ and $|E_1| \geq |E_2|$.

Previous approaches explore ways to reduce the space of alignments for scalability [20, 34, 41], improve the efficiency of the algorithms [22, 47, 66] and automatically adjust the alignment workflow [51, 66] for speedup. Often the reduction in execution time obtained by these approaches is at the expense of the quality of the alignment. However, with the help of indexing [47] and caching [66], alignment algorithms could gain efficiency without compromising the alignment quality. Yet, these techniques are not enough to scale up for very large ontologies. Some alignment algorithms [46, 47, 51, 66] adopt a self-configuring mechanism to disable computationally expensive components or to choose a light-weight alignment workflow when aligning large ontologies. The associated tradeoff of this strategy is the reduction in the quality of the output alignment. Approaches to managing the memory and processing requirements in aligning large ontologies frequently utilize partitioning techniques [20, 34, 41]. By partitioning the ontologies and only aligning the parts which share significant alignment between them, algorithms could gain significant speedup again at the expense of alignment quality.

A class of algorithms that performs automated alignment is *iterative* in nature [12, 20, 35, 46, 51, 93]. These algorithms repeatedly improve on the previous preliminary solution by optimizing a measure of the solution quality. Often, this is carried out as a guided search through the alignment space using techniques such as gradient descent or expectation-maximization. These algorithms run until convergence after which the solution stays fixed but, in practice, they are often terminated after an ad hoc number of iterations. Through repeated improvements, the computed alignment is usually of high quality but these approaches also consume more time in general than their non-iterative counterparts. While the focus on computational complexity has yielded ways of scaling the alignment algorithms to larger ontologies, such as through ontology partitioning [41, 77, 88], there is a general absence of effort to speed up the ontology alignment process. We think that these considerations of space and time go hand in hand in the context of scalability. We present a novel and general approach for speeding up the multivariable optimization process utilized by these algorithms in Chapter 5. Specifically, we use the technique of block-coordinate descent (BCD) in order to possibly improve the speed of convergence of the iterative alignment techniques.

Hosting ontologies from a specific domain in a repository has become prevalent [50, 63, 92]. These repositories also provide the alignment between the hosted ontologies to facilitate the discovery and exchange of knowledge. As new ontologies are submitted or ontologies are updated, their alignment with others must be quickly computed. Though improving the ontology alignment algorithms' efficiency helps to speed up the alignment process, it is not enough for many alignment algorithms to scale up for very large ontologies. Consequently, quickly aligning several pairs of ontologies becomes a challenge for these repositories.

Regularly, ontology alignment algorithms approach the complexity in aligning large ontologies by simply slicing the ontologies into smaller pieces and aligning some of them [20, 41]. Also, scalability is achieved by parallelizing the alignment process using either intra-matcher or inter-matcher parallelization [31]. Introducing parallelization within the alignment algorithm is called intra-matcher parallelization; on the other hand, aligning several ontology parts in parallel using ontology alignment algorithms is called inter-matcher parallelization. While Rahm [72] points out a general absence of inter-matcher parallelization, Chapter 6 presents a novel and general method for batch alignment of large ontology pairs using the distributed computing paradigm of MapReduce [18]. Our approach allows any alignment algorithm to be utilized on a MapReduce framework. Experiments using four representative alignment algorithms demonstrate flexible and significant speedup of batch alignment of large ontology pairs using MapReduce.

1.3 Biomedical Ontology Alignment

At present, we find momentous interest for ontologies in the biomedical domain, where a significant number of ontologies have been built, covering different aspects of medical research. Due to the complexity and the specialized vocabulary of the domain, matching biomedical ontologies is one of the hardest alignment problems. Life science researchers use these ontologies to annotate biomedical data and literature in order to facilitate improved information exchange. An agreement between these ontologies enables interoperability between the users and applications of them.

Evaluation of general ontology alignment algorithms has benefited immensely from standard setting benchmarks like OAEI. The annual competition evaluates the algorithms along a number of tracks, each of which contains a set of ontology pairs. While the emphasis of the competition is on comparison tracks, which contain test pairs that are modifications of a single small ontology pair in order to systematically identify the strengths and weaknesses of the algorithms, real-world test cases are also included. One of these involves aligning the ontology of adult mouse anatomy with the human anatomy portion of the NCI thesaurus [30]. OAEI included a new track called large biomedical ontology track in year 2012. This track aims at finding alignments between the large and semantically rich biomedical ontologies FMA, SNOMED CT, and NCI, which contain 78,989, 306,591 and 66,724 classes, respectively. However, aligning biomedical ontologies poses its own unique challenges. In particular,

1. Entity names are often identification numbers instead of descriptive names. Hence, the alignment algorithm must rely more on the labels and descriptions associated with the entities, which are expressed differently in different formats.
2. Although annotations using entities of some ontologies such as the gene ontology [5] are growing rapidly, for other ontologies they continue to remain sparse. Consequently, we may not overly rely on the entity instances while aligning biomedical ontologies.
3. Finally, biomedical ontologies tend to be large with many including over a thousand entities. This motivates the alignment approaches to depend less on **brute-force** steps, and compels assigning high importance to issues related to scalability.

Given these unique challenges in aligning biomedical ontologies, we created two novel biomedical ontology testbeds, using the ontologies from the NCBO, which provide an important application context to the alignment research community. Due to the large sizes of biomedical ontologies, these testbeds could serve as a comprehensive large ontology benchmark. However, the second testbed specifically focuses on ontology pairs with a significant amount of complex concepts. More details on these testbeds and performance evaluations using them are detailed in Chapter 7.

1.4 Optima+ And Its Performance In OAEI

Optima [20] is an automatic inexact ontology alignment tool developed here at THINC lab, Department of Computer Science, University of Georgia. It models ontologies as graphs, and formulates alignment as a maximum likelihood problem and uses expectation-maximization to solve this optimization problem. It iteratively searches through the space of candidate alignments evaluating the expected log likelihood of each candidate alignment, which is generated using heuristics that consider neighboring correspondences. More details on **Optima**'s iterative algorithm is provided later in Section 2.3.8. This chapter details the enhancements I made to **Optima** and its performances in the OAEI benchmarking for the past 2 years.

1.4.1 Enhancements to Optima

Throughout my research, I constantly contributed to the **Optima** alignment algorithm to improve its performance. This includes better software engineering practices and applying the learnings from my research. Additionally the general and novel algorithms I devised for complete efficient and scalable alignment of ontologies, which are the corner stones of this dissertation are also implemented in **Optima**. These algorithms are detailed later in chapters 4 to 6. This improved version of **Optima** is named **Optima+**. **Optima** debuted in OAEI benchmarking in 2011 [83, 89] with acceptable middle tier performance. Then the new and improved **Optima+** participated in the next year and ranked second in the important track called *conference* track with very good results in few other tracks. Note conference track consists of medium to large sized real world ontologies with varying lexical and structural features, thus the improvements due to the enhancements are significant. Some of the noticeable enhancements of **Optima+** are,

- Improved and efficient ontology preprocessing and ontology modeling
- Improved and efficient use of similarity measures
- Improved convergence using BCD

- Improved and efficient alignment postprocessing

Optima+ models ontologies as RDF graphs [57] and includes complex concepts within its modeling. During preprocessing it tokenizes and indexes the lexical attributes and prefetches the tokens from WordNet for improved efficiency. The complex concepts are modeled using the RDF graph-based canonical representations presented in Chapter 4. It uses the three-gram index of WordNet terms [69] to perform three-gram tokenization for improved evaluation of similarity. It integrates two syntactic similarity measures (I-sub similarity measure [87] and Needleman-Wunsch [64]) and two semantic similarity measures (Lin [52] and gloss based cosine [96]) to evaluate correspondences. **Optima+** uses WordNet version 3.0 to evaluate the semantic similarity measures.

The BCD based approach for improving the convergence of iterative alignment algorithms presented in Chapter 5 is also implemented in **Optima+** to speed up its convergence. During alignment postprocessing, **Optima+** prunes the alignment to achieve a minimal and coherent final alignment. A minimal alignment is achieved by removing the correspondences, which can be inferred by an existing correspondence. A coherent alignment is achieved by resolving conflicting correspondences. Specifically, in addition to duplicate correspondences, for each correspondence between N_1 and N_2 , **Optima+** removes the following correspondences:

- any correspondence among the descendants of N_1 with N_2
- any correspondence among the descendants of N_1 with N_2 's ancestors
- any correspondence among the descendants of N_2 with N_1
- any correspondence among the descendants of N_2 with N_1 's ancestors

With the above mentioned enhancements **Optima+** improved its F-measure by 81% compared to its previous year in the conference track which place it second in this track with 65% F-measure. Moreover, it completed the whole conference track in 23 minutes, which is dramatically small compare to its previous year's run time of more than 15 hours. Yet, **Optima+** finds it difficult to scale up

to very large ontologies. Subsequently, I integrated it with the algorithms presented in Chapter 6 for scaling ontology alignment algorithms for very large ontologies using the MapReduce paradigm. Using this approach it gains tremendous speed up. For example, it completed aligning all the ontology pairs of conference track in 30 seconds without compromising the alignment quality.

1.4.2 Ontology Alignment Evaluation Initiative (OAEI)

The Ontology Alignment Evaluation Initiative (OAEI) [23] is an international initiative that annually organizes the evaluation of ontology matching systems. Every year OAEI organizes a workshop [79–85] for ontology alignment tools and the participated tools are benchmarked. This evaluation is operated on SEALS [6] platform to automate and streamline the evaluation process. The OAEI benchmark is a collection of tracks such that each track focuses on a specific capability of the ontology matching system or a specific domain of ontologies. For example, the test cases from *multifarm* track are tailored with a special focus on multilingualism. On the other hand, expressive ontologies in the *conference* track structure knowledge related to conference organization, and the *anatomy* track, consists of a pair of large ontologies from the life sciences, describing the anatomy of an adult mouse and human.

Last year – 2012 – OAEI evaluated algorithms using seven different tracks: *benchmark*, *anatomy*, *conference*, *multifarm*, *library*, *large biomedical ontologies* and *instance matching* [84]. Tracks contain tasks/datasets which consist of ontologies of similar domain to be aligned. The OAEI campaign consists of both the tailored ontologies and the real world ontologies. Note, the *benchmark* track consists of systematically generated test cases. Ontologies in *anatomy*, *conference*, *library* and *large biomedical ontologies* were either acquired from the Web or created independently of each other and based on real-world resources. A subset of the ontologies from *conference* track and their translation in eight different languages (Chinese, Czech, Dutch, French, German, Portuguese, Russian, and Spanish) form the *multifarm* track. The *instance matching* track aims at evaluating the ability of tools to identify similar instances among different RDF and OWL datasets.

I extensively used the ontology pairs from OAEI in my several experiments. Specifically, I focus on the test cases that involve real-world ontologies for which the reference (true) alignment was provided by OAEI. This includes all ontology pairs in the 300 range of the benchmark, which relate to *bibliography*, expressive ontologies in the *conference* track all of which structure knowledge related to conference organization, and the *anatomy* track, which consists of large ontologies from life sciences, describing anatomy of adult mouse and human. I list the ontologies participating in my evaluations in Table A.1 and provide an indication of their sizes.

1.4.3 Performance of Optima+ in OAEI

Optima participated in the last 2 years' OAEI campaigns. In 2011, it debuted in 3 tracks and performed with favorable middle tier results. Next year, **Optima** participated with its new version **Optima+** and out of 23 tools participated, it was placed second along with two other algorithms in a key track called *Conference* track. Last year, I mainly focused on three tracks – Benchmark, Conference, and Anatomy. However, we were evaluated in all the tracks of the campaign offered by the SEALS platform of OAEI: Benchmark, Conference, Anatomy, Multifarm, Library, and LargeBioMed. This year, I am preparing to participate in all five tracks including the large ontology track. The following sections analyze the performances of **Optima+** in benchmark, conference and anatomy tracks.

Benchmark Track

The Benchmark test library consists of 5 different test suites [54]. Each of the test suits is based on individual ontologies and consist of a number of test cases. Each test case discards a certain amount of information from the ontology to evaluate the change in the behavior of the algorithm. There are six categories of such alterations – changing the names of the entities, suppression or translation of comments, changing hierarchy, suppressing instances, discarding properties with restrictions, or suppressing all properties and expanding classes into several classes or vice versa. Suppressing entities and replacing their names with random strings result in scrambled labels of entities. Test

Table 1.1: Average recall, precision, and F-measure of **Optima+** in OAEI 2012 for benchmark track. Note **Optima+** performs well in test cases in the range of 201-247. However, it struggles to maintain the same level for the testcases above 247, which contain tailored ontologies with scrambled labels.

		Precision	Recall	F-measure
Bibliography	100 Series	1	1	1
	201-247	0.88	0.85	0.85
	248-266	0.65	0.35	0.43
2	100 Series	1	1	1
	201-247	1	0.84	0.87
	248-266	1	0.36	0.46
3	100 Series	1	1	1
	201-247	0.97	0.88	0.89
	248-266	0.98	0.38	0.49
4	100 Series	1	1	1
	201-247	0.93	0.77	0.79
	248-266	0.96	0.34	0.43
Finance	100 Series	1	1	1
	201-247	0.96	0.80	0.83
	248-266	0.96	0.38	0.49

cases from 248 to 266 consist of such entities with scrambled labels. Table. 1.1 shows **Optima+**'s performance in benchmark track on 100 series test cases, 200 series test cases without scrambled labels test cases, and all the scrambled labels test cases. The average precision for **Optima+** is 0.95, while average recall is 0.83 for all the test cases in the 200 series except those with scrambled labels. For test cases with scrambled labels, the average recall is dropped by 0.53, while precision is dropped only by 0.04. When labels are scrambled, lexical similarity becomes ineffective. For **Optima+** algorithm, structural similarity stems from lexical similarity, hence scrambling the labels makes the alignment more challenging. As a result a 46% decrease in average F-measure from 0.85 to 0.46 is observed. This trend of reduction in precision, recall, and F-measure can be observed throughout all different test suites of the benchmark track.

Anatomy Track

In 2011, **Optima** could not successfully complete aligning the *anatomy* track. Last year, with the help of naive partitioning technique and the improved efficiency due to BCD, **Optima+** was able to successfully align the ontologies of this track. In this track, **Optima+** yields 85.4% precision and 58.4% recall in 108 minutes. We hope with biomedical lexical databases like Unified Medical Language System (UMLS) [10], **Optima+** can improve its recall. Note it was able to increase its speedup by more than 15 when aligning these large ontology pairs in the MapReduce setup. I present more details about this approach in Chapter 6.

Conference Track

Table 1.2: Comparison between the performances of **Optima+** in OAEI 2012 and **Optima** in OAEI 2011 for conference track. **Optima+** significantly improved its alignment quality and efficiency. Specifically, it improved its F-measure by 81% and gained a speed up of 40.

Year	Precision	Recall	F-measure	Total Runtime
2011	0.26	0.60	0.36	15hrs
2012	0.62	0.68	0.65	1349sec

Conference track consist of 16 ontologies all of which organize knowledge related conference organization, which forms 120 unique ontology pairs. However, it only has 21 reference alignments which corresponds to the complete alignment space between 7 ontologies from the data set. More details about these 7 ontologies are provided in Appendix A. For this track, **Optima+** achieves a recall of 0.68 and a precision of 0.62, which are significantly improved compared to its previous year’s performance. Overall, there is 81% increase in F-measure as compared to OAEI 2011. With this performance leap, **Optima+** was placed second, along with two other algorithms – their performances were too close to each other to distinguish any of them – with the top spot held by **YAM++**. It is unique in its uniform emphasis on recall (discovering more maps) and precision (making sure the discovered maps are correct). Table 1.2 lists the precision, recall, and F-measure along with total runtime for *Conference* track of **Optima** in OAEI 2011 and **Optima+** in OAEI 2012. The

alignment quality improvement in the *Conference* track arises from the improved similarity measure and the alignment extraction mentioned above. **Optima+** also utilizes improved design and optimization techniques to drastically reduce runtime. The runtimes reported in Table 1.2 cannot be compared directly, as the underlying systems used for evaluations differ. However, the improvement in runtime from 15+ hours to around 23 minutes is perspicuous. Note, in the MapReduce setup presented in Chapter 6, **Optima+** is able to align this whole track in 30 seconds without compromising the output quality.

Table 1.3: Comparison between the performances of top 4 alignment algorithms (**YAM++**, **Logmap**, CODI, and **Optima+**) in OAEI 2012 for conference track. F2-measure weights recall higher than precision. Note, **Optima+** produces the second highest recall and F2-score while the leading algorithm **YAM++** has difficulties in completely aligning this track.

Algorithms	Precision	Recall	F1-measure	F2-measure	Total Runtime (seconds)
YAM++	0.78	0.65	0.71	0.67	N/A
Logmap	0.77	0.53	0.63	0.57	211
CODI	0.74	0.55	0.63	0.58	2353
Optima+	0.60	0.63	0.61	0.62	1349

The Table 1.3 compare precision, recall, F1-measure, F2-measure and run times of top 4 algorithms in conference track. Note, F2-measure is obtained using the equation in Eq. [refeq:fmeasure](#) with $\beta = 2$ hence it weights recall higher than precision. Though, **YAM++** ranked first in terms of both F1-score and F-2 score, it used the ontologies and reference alignments of this track to train its algorithm. While **Optima+** produces the second highest F2-measure, **Logmap** produces the second best F1-score. Importantly, **Optima+** also demonstrates the second best recall for conference track. **YAM++** could not align the 120 pairs within the 5 hour time limit set by the OAEI. However, it was able to finish those 21 pairs for which reference alignments are available within the time limit. Therefore we provide measures of its alignment quality but not the runtime. Note that all the precision, recall and F-measure information presented in this table are based on the 21 reference alignments only. **Logmap** which is known for its scalability able to quickly produce the alignments. However, its approach for scalability is known for low recall. CODI ties up with **Logmap** interns of F1-score but consumes significantly more time than the rest of the algorithms.

Noticeably **Optima+** performed significantly well in this track with the enhancements I listed earlier. This demonstrates the significance and applicability of the algorithms and insights presented in this dissertation.

1.5 Contributions

This dissertation addresses some of the key challenges for ontology alignment, such as (1) efficiently aligning ontologies without compromising the quality of the output (2) improving the alignment quality (3) scaling up for very large ontologies. Subsequently its major contributions are:

1. An algorithm for complete ontology alignment
2. Algorithms and insights for alignment algorithms to improve their efficiency without compromising the quality of the output
3. Approaches for ontology alignment algorithms to scale up to very large ontologies
4. An approach for ontology alignment algorithms to scale up to batch alignment of ontologies
5. Biomedical ontology alignment testbeds for evaluations

Although the main contributions of my work falls under these categories, I believe that this dissertation as a whole can serve as a good reference for ontology alignment community. Also, I discuss some exciting and useful future avenues of the works presented here and thus which provides a source of useful research directions for ontology alignment researchers. Altogether this dissertation contributes to the communities of ontology alignment users, ontology alignment researchers and ontology repositories. In the following sections, I summarize the specific contributions of the above mentioned efforts which are part of my dissertation research that I have accomplished thus far and give further details in subsequent chapters.

1.5.1 Utilizing WordNet for Efficient and Improved Ontology Alignment

Many ontology alignment algorithms augment syntactic matching with the use of WordNet in order to improve their performance. The advantage of using WordNet in alignment seems apparent. I analyzed the utility of WordNet in the context of the reduction in precision and increase in execution time that its use entails. I observed distinct trends in the performance of WordNet-based alignment in comparison with alignment that uses syntactic matching only. I analyzed the trends and their implications and provided useful insights on the types of ontology pair for which WordNet-based alignment may potentially be worthwhile and those types where it may not be. I think that many of the outcomes of this analysis are novel and useful in evaluating the use of computationally intensive add-ons such as WordNet.

The major contributions of this work are listed below:

Contributions

- Recommendation to the ontology alignment research community to not to discourage the use of WordNet, but allow WordNet usage within the alignment process to be *optional*
- Provided a few rules of thumb related to characteristics of ontologies for which WordNet should be utilized cautiously

These contributions are outlined in Chapter 3.

1.5.2 Modeling Complex Concepts for Complete Alignment of Ontologies

Modern ontology languages, such as the Web Ontology Language (OWL), allow defining complex concepts that involve restrictions, Boolean combinations, and exhaustive enumeration of individuals. About 40% of the ontologies in BioPortal repository at the NCBO have more than a thousand complex concepts, and in about 60% of the ontologies, such concepts constitute 25% or more of all concepts. Complex concepts are either ignored or naively utilized in alignment algorithms. Hence, the resulted alignments of these algorithms are possibly incomplete. For value and cardinality

restrictions and Boolean combinations, we introduce axiomatic and graphical canonical representations. Using a representative set of well known alignment algorithms, we show that the existing alignment algorithms can integrate this approach within their alignment process. Our results indicate a significant improvement in the F-measure of the alignment produced by these algorithms. However, this improvement is achieved at the expense of increased run time due to the additional concepts modeled.

Contributions

- Provided a general approach for modeling complex concepts in ontology alignment
- Demonstrated that modeling complex concept in ontology alignment may improve the quality of the alignment, specifically the precision, at the expense of run time.

These contributions are outlined in Chapter 4.

1.5.3 Improved Convergence of Iterative Ontology Alignment

A class of alignment algorithms that is iterative and often consumes more time than others while delivering solutions of high quality. I presented a novel and general approach for speeding up the multivariable optimization process utilized by these algorithms. Specifically, I used the technique of block-coordinate descent in order to possibly improve the speed of convergence of the iterative alignment techniques. I integrated this approach into four well-known alignment systems and showed that the enhanced systems generate similar or improved alignments in significantly less time on a comprehensive testbed of ontology pairs. This represents an important step toward making alignment techniques computationally more feasible.

Because BCD does not overly constrain how we partition or order the parts, I varied the partitioning and ordering schemes in order to empirically determine the best schemes for each of the selected algorithms. My study shows that the choice of partitioning and ordering schemes varies between algorithms. I provided insights that are useful toward identifying a scheme that is appropriate for a given iterative alignment algorithm.

Contributions

- Presented a novel approach based on BCD to increase the speed of convergence of iterative alignment algorithms with no observed adverse effect on the final alignments
- Provided insights for selecting partitioning and ordering schemes for ontology alignment algorithms when using BCD

These contributions are presented in Chapter 5.

1.5.4 Speeding up Batch Alignment of Large Ontologies Using MapReduce

While my previous approaches allowed alignment algorithms to efficiently align medium to large ontologies, they do not enable these algorithms to scale up for very large ontologies such as FMA, SNOMED, and NCI which contain 78,989, 306,591 and 66,724 classes respectively. A prevalent way of managing the alignment complexity posed by large ontologies is to simply dissect the ontologies into smaller pieces and align some of the ontology parts [20,41]. Parallelizing the alignment process is another way of approaching scalability. Intra-matcher parallelization introduces parallelization within the alignment algorithm. On the other hand, inter-matcher parallelization aligns several ontology parts in parallel using ontology alignment algorithms [31]. In the context of a general absence of inter-matcher parallelization, I proposed a novel and general framework for aligning very large ontologies in parallel using MapReduce. My approach allows any alignment algorithm to be utilized on the MapReduce framework. This approach allows previously unscalable alignment algorithms to scale up for very large ontologies with small reduction in alignment quality for some algorithms.

Ontologies are increasingly hosted in repositories, which often compute the alignment between the ontologies. As new ontologies are submitted or ontologies are updated, their alignment with others must be quickly computed. Therefore, aligning several pairs of ontologies quickly becomes a challenge for these repositories. I project this problem as one of batch alignment and show how it may be approached using the distributed computing paradigm of MapReduce. Experiments using

four representative alignment algorithms demonstrate flexible and significant speedup of batch alignment of large ontology pairs using MapReduce.

Contributions

- Provided a general and novel approach for speeding up batch alignment of several ontology pairs using MapReduce.
- Provided a general and novel approach for aligning very large ontologies using MapReduce.

These contributions are detailed in Chapter 6.

1.5.5 Optima+: Efficient, Improved and Open-sourced Ontology alignment Tool

As mentioned earlier in Section 1.4 I constantly contributed to the development of the ontology alignment tool, **Optima**. I redesigned it to produce improved alignment in relatively less time. This new and enhanced version is called **Optima+**, which participated in OAEI 2012 and ranked second along with two other algorithms in a key track called Conference track. The **Optima+** tool brings the following contributions to both ontology alignment users and researchers.

Contributions

- Provided an improved and efficient ontology alignment tool call **Optima+** with various user interfaces for ontology alignment users
- Provided the source code and documentation of **Optima+** for ontology alignment researchers to experiment, extend and reuse

1.5.6 Biomedical Ontology Alignment Testbeds

Ontologies are becoming increasingly critical in the life sciences [13,49] with multiple repositories such as Bio2RDF [9], OBO Foundry [4] and NCBO's BioPortal [63] publishing a growing number of biomedical ontologies from different domains such as anatomy and molecular biology. Given the emerging importance of ontology alignment in the biomedical domain, I combed through more

than 300 ontologies hosted at NCBO [63] and OBO Foundry [4], and created two distinct testbeds for ontology alignment. One testbed is of 50 biomedical ontology pairs. Thirty-two ontologies with sizes ranging from a few hundred to tens of thousands of entities constitute the pairs. It serves as an extensive testbed for analyzing the scalability of alignment algorithms. The second testbed contains 35 ontology pairs which have significant amount of complex concepts within them. I have evaluated the performances of the algorithms presented in this dissertation using these testbeds in addition to the testbeds provided by OAEL.

Contributions

- Provided a novel biomedical ontology alignment testbed for analyzing the scalability of alignment algorithms
- Provided a comprehensive testbed of 35 ontology pairs to evaluate the ability of ontology alignment algorithms to utilize the complex concepts in ontology alignment

These contributions are outlined in Chapter 7.

1.6 Dissertation Organization

The rest of this dissertation is outlined as follows. Chapter 2 formally defines the ontology alignment problem and illustrates the general architecture of ontology alignment algorithms. Here, I extensively review ontology alignment algorithms and the approaches they take to scale up for very large ontologies. In the next four chapters I present three distinct and novel approaches and a comprehensive study to make alignment algorithms complete, efficient and scalable. Each of these four chapters empirically analyzes the presented approach using representative algorithms and various data sets. Subsequently, these chapters discuss insights supported by the results from their experiments.

In the first chapter, I introduce axiomatic and graphical canonical forms for modeling value and cardinality restrictions and Boolean combinations, and present a way of measuring the similarity between these complex concepts in their canonical forms in Chapter 4. Note, that many of the

current ontology alignment algorithms either do not consider the complex concepts in their alignment procedures or model them naively, thereby producing a possibly incomplete alignment. Here I also show how our approach may be integrated in multiple ontology alignment algorithms and evaluate its impact on performance. This approach helps alignment algorithms towards complete alignment of ontologies. In the next chapter, I analyze the utility of WordNet in the context of the reduction in precision and increase in execution time that its use entails for ontology alignment. The details of this empirical study and useful insights are discussed in Chapter 3. Specifically, this chapter provides useful insights and recommendations in utilizing WordNet for efficient and complete alignment of ontologies. In Chapter 5, I describe a novel algorithm for iterative ontology alignment algorithms to improve their speedup using block coordinate descent. I also present the integration of this approach into multiple well-known alignment algorithms and provide the performance analysis of these enhanced algorithms. Various directions for optimizing this approach using several partitioning and ordering schemes and a comprehensive analysis of these schemes on the selected algorithms are also presented in this chapter. Next chapter projects a crucial challenge of ontology repositories as batch alignment of ontologies and shows how it may be approached using the distributed computing paradigm of MapReduce. I also present performance analysis of this approach using four representative alignment algorithms and demonstrate flexible and significant speedup of batch alignment of large ontology pairs using MapReduce in the same chapter.

To facilitate ontology alignment evaluations in the domain of life science I have created two novel testbeds made from biomedical ontologies from NCBO. Details on how these testbeds are created and the evaluations of presented algorithms using these novel testbeds are outlined in Chapter 7. Finally, Chapter 8 gives a brief discussion of the accomplished work and outlines some avenues of future work.

CHAPTER 2

BACKGROUND AND RELATED WORK

An ontology is a specification of knowledge pertaining to a domain of interest formalized into concepts and relationships between the concepts. Contemporary ontologies utilize description logics [8] that are represented in XML, such as the Web Ontology Language (OWL) [58], in order to facilitate publication on the Web. OWL allows the use of classes to represent entities, different types of properties to represent relationships, and individuals to include instances. Ontology alignment has become popular in solving interoperability issues across heterogeneous systems in the semantic web.

2.1 Alignment Problem

As we stated earlier, the ontology alignment task is to find a set of correspondences between two ontologies, $\mathcal{O}_1$ and $\mathcal{O}_2$. Though OWL is based on description logic, several alignment algorithms model ontologies as labeled graphs (with some possible loss of information) due to the presence of a class hierarchy and properties that relate classes, in order to facilitate alignment. For example, **Falcon-AO** and **Optima** transform OWL ontologies into a bipartite graph [36] and **OLA** utilizes an OL-graph [24]. Consequently, the alignment problem is often cast as a matching problem between such graphs. An ontology graph, $\mathcal{O}$, is defined as, $\mathcal{O} = \langle V, E, L \rangle$ where V is the set of labeled vertices representing the entities, E is the set of edges representing the relations, which is a set of ordered 2-subsets of V , and L is a mapping from each edge to its label. A correspondence, $m_{\alpha\alpha}$, between two entities, $x_\alpha \in \mathcal{O}_1$ and $y_\alpha \in \mathcal{O}_2$, consists of the relation, $r \in \{=, \subseteq, \supseteq\}$, and confidence, $c \in \mathbb{R}$. However, many alignment algorithms [12, 20, 24, 46, 47] focus on the possible presence of $=$ relation (also called *equivalentClass* in OWL) between entities only. In

this case, an alignment may be represented as a $|V_1| \times |V_2|$ -dimensional matrix that represents the correspondence between the two ontologies, $\mathcal{O}_1 = \langle V_1, E_1, L_1 \rangle$ and $\mathcal{O}_2 = \langle V_2, E_2, L_2 \rangle$:

$$M = \begin{bmatrix} m_{11} & m_{12} & \cdots & m_{1|V_2|} \\ m_{21} & m_{22} & \cdots & m_{2|V_2|} \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ m_{|V_1|1} & m_{|V_1|2} & \cdots & m_{|V_1||V_2|} \end{bmatrix}$$

Note that if the ontologies are not modeled as graphs, the rows and columns of M are the concepts in $\mathcal{O}_1$ and $\mathcal{O}_2$ defined in the description logic. Each assignment variable, $m_{a\alpha} \in M$, is the confidence of the correspondence between entities, $x_a \in V_1$ and $y_\alpha \in V_2$. Consequently, M could be a real-valued matrix, commonly known as the similarity matrix between the two ontologies. However, the confidence may also be binary, with 1 indicating a correspondence, otherwise 0, due to which the match matrix M becomes a binary matrix representing the alignment. Two of the algorithms that we use maintain a binary M while the others use a real M .

An alignment is not limited to correspondences between entities alone and may include correspondences between the relationships as well. In order to facilitate matching relationships, alignment techniques [20, 24, 46], transform the edge-labeled graphs into unlabeled ones by elevating the edge labels to first-class citizens of the graph. This process involves treating the relationships as resources thereby adding them as nodes to the graph. Subsequently, the transformed graph is a bipartite graph [36].

2.2 Architecture

Several algorithms [12, 16, 20, 24, 35, 40, 45–47, 51, 93] now exist for automatically aligning ontologies, with mixed success in their performances. Fig. 2.1 depicts an abstract architecture of the ontology alignment algorithms. The alignment process produces a set of correspondences between

the given pair of ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$. For a faster computation, the ontologies may be preprocessed (e.g., prefetching the list of neighboring entities to fasten the structural similarity calculation, removing redundant information from schema and tokenizing the lexical attributes). Then for each pair of entities of the Cartesian product of entities in $\mathcal{O}_1$ and $\mathcal{O}_2$ the similarity is evaluated using the element level matchers. An element level matcher measures the similarity between a pair of entities. Predominantly, element level matchers exploit lexical attributes of entities. Element level lexical matchers only use the lexical attributes of entities such as, *name*, *label*, and *comment*, while element level structural matchers exploit the neighboring entities when evaluating the similarity. Yatskevich and Giunchiglia surveyed [96] several WordNet based element level matchers for semantic matching. Thus far I never witnessed alignment algorithms exploiting element level matchers which exploit structural attributes only. However, they use several element level matchers [12, 71] which use both structural and lexical features together. **Falcon-AO** uses a hybrid element level matcher known as VDOC [71] which concatenates lexical features of neighboring classes to evaluate similarity. The element level matchers which utilize the instances to evaluate an entity pair known as element level instance matcher [19, 20, 43, 51]. These algorithms use instance level matcher to match the instances. Note element level instance matchers are different from instance level matchers which match instances.

An element level lexical matcher is uniquely identified by the lexical similarity measure it uses. Commonly, the measure of similarity is a value between 0 and 1 where a value 1 indicates equivalence and a value 0 means disjoint relation. The lexical similarity measures attempt to evaluate the similarity between two *concepts* expressed in natural language. A concept may be expressed using a word, a phrase or even using a sentence. Lexical similarity measures may be broadly categorized into syntactic and semantic. Syntactic similarity between two concepts is entirely based on the sequence similarity between the texts. For example, Smith-Waterman similarity measure [86] determines similar regions between two strings to evaluate similarity. Semantic similarity measures attempt to utilize the meaning behind the concept names to ascertain the similarity of the concepts. A popular way of doing this is to exploit lexical databases suchlike WordNet, which

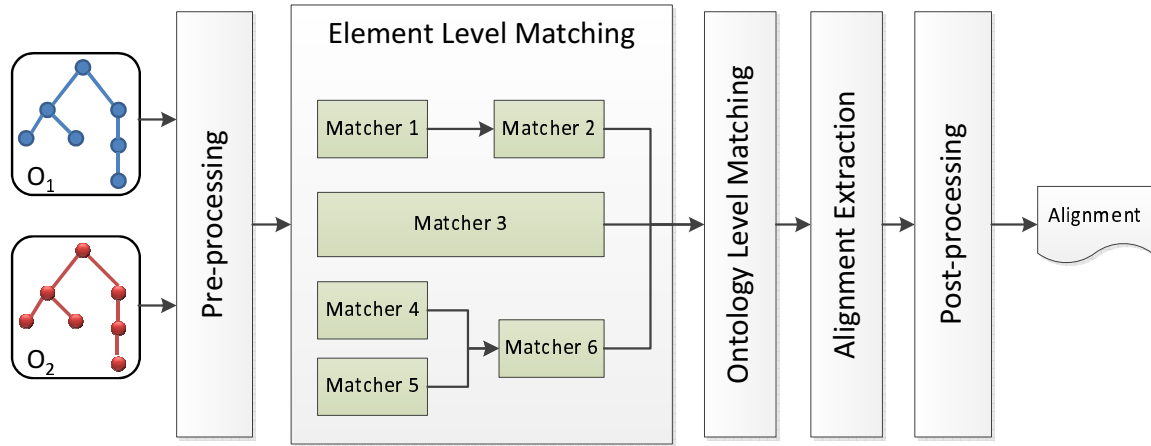


Figure 2.1: The general architecture of ontology alignment process. The alignment process consume ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$ and produce an alignment output. Ontologies are preprocessed and both element level and ontology level matchers perform matching and produce correspondences. An alignment is extracted from this set of correspondences and preprocessed for inconsistent and redundant correspondences.

provide words related in meaning. As an example, Lin similarity measure [52] implementation in **Optima+** exploits taxonomical relationships between the concepts in WordNet and their frequency of occurrence in WordNet to evaluate similarity. Note an element level matcher may be comprised of several lexical matchers.

An alignment algorithm often utilizes several element level matchers to evaluate correspondences. These matchers can be operated in either sequential, parallel, or some mixed fashion. For example, the *matcher 1* and *matcher 2* in the Fig. 2.1 are arranged to operate sequentially while *matcher 3* is set to operate in parallel to them. *Matcher 4*, *matcher 5* and *matcher 6* are set up in a mixed fashion. Alignment algorithms utilize various techniques to derive correspondences from the similarity measures obtained from these matchers. A straightforward technique is to threshold the similarity values to identify correspondences. Some alignment algorithms [12, 20, 71] derive the weighted summation of similarity measures before thresholding. **YAM++** uses a decision tree to classify correspondences using various similarity measures.

Some of the leading ontology alignment algorithms, such as **Optima+** [20], **YAM++** [66], **Logmap** [47] and **Falcon-AO** [40] employ ontology level matchers to enhance their performance. Ontology level matchers utilize the complete models of ontologies while aligning them. For example, **YAM++** uses similarity flooding [61], and **Falcon-AO** employs graph matching for ontologies (GMO) [39]. Both of these algorithms iteratively update the similarity matrix M where the update is controlled by the complete models of ontologies. **Optima+** adopts the inexact matching of ontology graphs algorithm [20] which exploits the bipartite graph model of ontologies. **Logmap** uses Dowling-Gallier’s algorithm [21] to identify unsatisfiability in a Horns [38] representation of a pair of ontologies and repair them. Predominantly these ontology level matchers are iterative in nature. They extract an initial seed alignment using the element level matchers and iteratively improve it till convergence. Chapter 5 provides specific details on iterative alignment approaches.

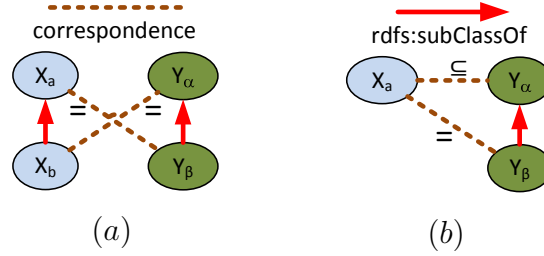


Figure 2.2: An example redundant correspondence (a) and an example inconsistent correspondence (b) which are often resolved during the post processing of alignment work-flow. The inconsistent correspondence shown here is known as crisscross correspondence.

Finally, alignment algorithms need to post-process the collection of correspondences they obtain from various matchers they utilize. During post-processing, in order to keep the alignment minimal, some algorithms **Optima+** [20] and **YAM++** [66] remove those correspondences which may be inferred from another. Duplicate correspondences and redundant correspondences form such scenarios. An example redundant correspondence is shown in Fig. 2.2(a). Let there exist correspondences, $\langle x_a, y_\alpha, \subseteq, c_{a\alpha} \rangle$ and, $\langle x_a, y_\beta, =, c_{a\beta} \rangle$, in the set of correspondences obtained. Here, x_a is an entity of ontology, $\mathcal{O}_1$, and y_α and y_β are entities of ontology, $\mathcal{O}_2$. If y_β is a subclass of

y_α , then we may remove the correspondence, $\langle x_a, y_\alpha, \subseteq, c_{a\alpha} \rangle$, which can be inferred. Some algorithms [45, 47] additionally remove logically inconsistent mappings during post-processing. For example, the crisscross inconsistency which is illustrated in Fig. 2.2(b) occurs when merging the correspondences, $\langle x_a, y_\beta, =, c_{a\beta} \rangle$ and $\langle x_b, y_\alpha, =, c_{b\alpha} \rangle$, where x_a and x_b are entities in ontology, $\mathcal{O}_1$, y_α and y_β are entities in ontology, $\mathcal{O}_2$, and $c_{a\beta}$ and $c_{b\alpha}$ are confidence scores in the equivalence correspondences. If x_b is a subclass of x_a and y_β is a subclass of y_α then these crisscross correspondences are inconsistent. In practice, ontology alignment algorithms remove the one with the lower confidence score while merging.

2.2.1 Iterative Ontology Alignment

As I mentioned earlier a large class of ontology level matchers is iterative in nature [12, 20, 24, 35, 46, 51, 61, 66, 93]. Iterative algorithms utilize a seed matrix, M^0 , either input by the user or generated automatically. Beginning with the seed, the match matrix is iteratively improved until it converges as I abstractly illustrate in Fig. 2.3.

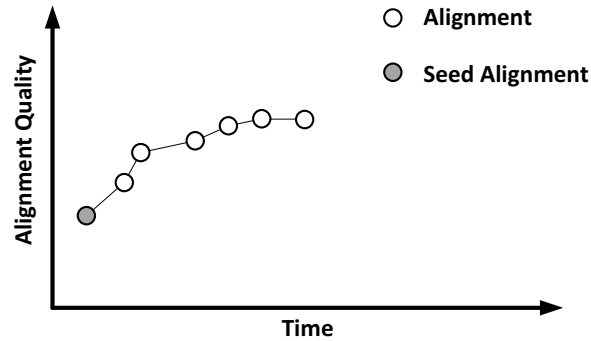


Figure 2.3: An iterative approach jumps from one alignment to another simultaneously improving on the previous one. Implementations differ in how they select the next alignment in each iteration and in the qualitative metric used for assessing it. Iterative process starts with a seed alignment. An alignment that cannot be improved further signifies convergence.

Two types of iterative techniques are predominant: The first type of iterative algorithms improve the real-valued similarity matrix from the previous iteration, M^{i-1} , by directly updating it as shown below:

$$M^i = U(M^{i-1}) \quad (2.1)$$

where, U is a function that updates the similarities. These type of algorithms converge to a fixed point, M_* , such that, $M_* = U(M_*)$.

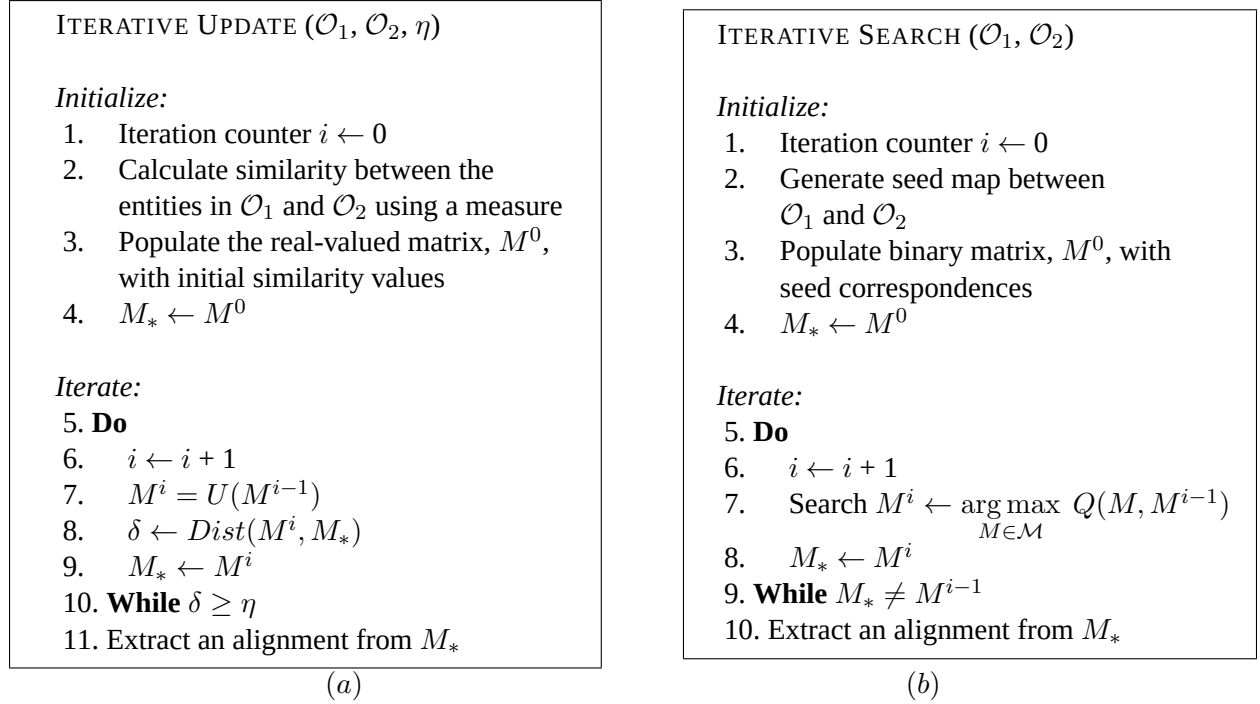


Figure 2.4: General algorithms for iterative (a) update, and (b) search approaches toward aligning ontologies. The distance function, $Dist$, in line 8 of (a) is a measure of the difference between two real-valued matrices.

The second type of iterative algorithms repeatedly search over the space of match matrices, denoted as $\mathcal{M}$. The goal is to find the alignment that optimizes an objective function, which gives a measure of the quality of the alignment in the context of the alignment from the previous iteration. This approach is appropriate when the search space is bounded such as when the match matrix is binary. Nevertheless, with a cardinality of $2^{|V_1||V_2|}$ this space could get very large. Some of the algorithms sample this space to reduce the effective search space though scaling to large ontologies continues to remain challenging. Formally,

$$M_*^i = \arg \max_{M \in \mathcal{M}} Q(M, M_*^{i-1}) \quad (2.2)$$

where, M_*^i is the alignment that optimizes the Q function in iteration i given the best alignment from previous iteration, M_*^{i-1} . Convergence of these algorithms occurs when the iteration reaches

a point, M_* , which cannot be improved further on searching for an alignment matrix, $M \in \mathcal{M}$, such that $Q(M, M_*) > Q(M_*, M_*)$.

Equations 2.1 and 2.2 help to solve a multidimensional optimization problem iteratively with $m_{a\alpha}$ in M as the variables. In Fig. 2.4, I show the abstract algorithms for the two types of iterative approaches. In the iterative update of Fig. 2.4(a), I may settle for a near fixed point by calculating the distance between a pair of alignment matrices (line 8) and terminating the iterations when the distance is within a parameter, η . As $\eta \rightarrow 0$, I get closer to the fixed point and obtain the fixed point in the limit. Iterative search in Fig. 2.4(b) often requires a seed map (line 3) to obtain M^0 , which is generated in various ways.

2.3 Survey of Automated Alignment Algorithms

The Ontology Alignment Evaluation Initiative (OAEI) [23] is a coordinated international initiative that organizes the evaluation of the ontology matching systems. Every year, OAEI organizes a workshop for ontology alignment systems and participated tools are benchmarked. SEALS [6] platform, which facilitates the formal evaluation of semantic technologies has been extended for ontology alignment evaluation by OAEI. Thus far we witness more than 50 different automatic ontology alignment tools submitted to SEALS platform. While there exists more extensive surveys [1, 15, 78] on ontology matching, here I limit my focus to a selected representative set of leading ontology matching algorithms. I have picked 10 alignment algorithms which have performed well in the previous OAEI benchmarkings [79–85] and briefly review them below.

2.3.1 Anchor Flood (Aflood)

Anchor flood [35] is an ontology matching system known for its efficiency in aligning large ontologies. It implements an efficient neighborhood search based on the *quick ontology matching* technique [22]. It first determines a set of seed correspondences where entity pairs share exact labels or names. Based on this seed, Aflood collects blocks of neighboring concepts. The concepts and properties within these blocks are compared and possibly aligned. This process is repeated where

each newly found correspondence is used as seed. This strategy to reduce the search space cut down the time required to generate an alignment significantly. The Aflood system participated in the OAEI in 2008 and 2009 and aligned ontologies from 5 different tracks. In OAEI 2009, it performed significantly well in the conference track and ranked second in the conference track with a F-measure of 52%. Despite its efficient neighborhood exploration, it is not able to scale-up for very large ontologies with more than several thousand entities.

2.3.2 AgreementMaker

AgreementMaker [16] offers a user interface built on an extensible architecture. This architecture allows flexible and deep configuration of the matching process. It defines several similarity measures and similarity aggregation methods which can be combined by users as required. In addition to various lexical similarity measures it offers couple of hybrid similarity measures which exploit either siblings or descendants lexical properties in evaluating similarities of a pair of concepts. It uses various classifiers (KStar, Naive Bayes, and Multilayer Perceptron) [33] to extract correspondences from the similarity matrix produced by matchers. AgreementMaker participated in 2009 and 2010 with good results in the conference and anatomy track. Especially it is ranked first in the conference track in 2009 with F-measure of 57%. Though it has instance matching module it has not shown good performances as in the anatomy or conference tracks.

2.3.3 Automated Semantic Mapping of Ontologies with Validation(ASMOV)

Automated semantic matching of ontologies with semantic verification (ASMOV) [45] alignment algorithm uses both lexical and structural properties and iteratively evaluates a set of correspondences. Then it employs a logic based post processing, in which it resolves any semantic inconsistencies. It also produces subsumption relationships in addition to equivalence relationships in its alignment. It exploits the extensive owl schema (restrictions, types, domains, range, and data values) and external knowledge stores such as WordNet [62] and Unified Medical Language

System (UMLS) [10] in obtaining these relationships. ASMOV participated in the OAEI consecutively from 2007 to 2010. It was one of the top performers in the benchmark track and participated also in many other tracks with good results. It is ranked first in the directory track of OAEI 2009 and 2010 with 63% F-measure.

2.3.4 Falcon-AO

Falcon-AO [40] is an automated ontology alignment algorithm that uses two matchers: a linguistic matcher and a graph-based matcher (GMO) [39] for structural matching. It models an OWL ontology as a bipartite RDF graph [36]. **Falcon-AO** adopts the partition-based block matching of large ontologies [40,41] to scale up for very large ontologies.

Falcon-AO [46] is a well-known automated ontology alignment system combining output from multiple components including a linguistic matcher (LMO) [71], an iterative structural graph matching algorithm called **GMO** [39], and a method for partitioning large ontologies and focusing on some of the parts [40,41]. LMO uses virtual documents, which are string concatenations of neighboring classes, and string similarity measures to produce the linguistic alignment. The virtual document for an anonymous class is a string formed from concatenating neighboring concepts. Consequently, LMO measures the lexical similarities between RDF statements involving named classes, which could be related to anonymous classes.

GMO measures the structural similarity between the ontologies that are modeled as bipartite graphs [36]. Calculation of the structural similarity by **GMO** is independent of the lexical similarity. Matrix M in **GMO** is real-valued and this similarity matrix is iteratively updated (Eq. 2.1) by updating each variable, $m_{\alpha\alpha}$, with the average of its neighborhood similarities, until M stops changing significantly. Equation 2.1 manifests in **GMO** as a series of matrix operations:

$$M^i = G_1 M^{i-1} G_2^T + G_1^T M^{i-1} G_2 \quad (2.3)$$

Here, G_1 and G_2 are the adjacency matrices of the bipartite graph models of the two ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$, respectively. In the first term of the summation, the outbound neighborhood of entities in $\mathcal{O}_1$ and $\mathcal{O}_2$ is considered, while the second term considers the inbound neighborhood. **GMO**

terminates its iterations when the cosine similarity between successive matrices, M^i and M^{i-1} , is less than a parameter, η .

FALCON-AO/GMO ($\mathcal{O}_1, \mathcal{O}_2, \eta$)

Initialize:

1. Iteration counter $i \leftarrow 0$
2. $G_1 \leftarrow \text{AdjacencyMatrix}(\mathcal{O}_1)$
3. $G_2 \leftarrow \text{AdjacencyMatrix}(\mathcal{O}_2)$
4. **For each** $m_{a\alpha} \in M^0$ **do**
5. $m_{a\alpha} \leftarrow 1$
6. $M_* \leftarrow M^0$

Iterate:

7. **Do**
8. $i \leftarrow i + 1$
9. $M^i \leftarrow G_1 M^{i-1} G_2^T + G_1^T M^{i-1} G_2$
10. $\delta \leftarrow \text{CosineSim}(M^i, M_*)$
11. $M_* \leftarrow M^i$
12. **While** $\delta \geq \eta$
13. Extract an alignment from M_*

Figure 2.5: Iterative update in the structural matcher, **GMO**, in **Falcon-AO**.

The iterative update of Fig. 2.4(a) manifests in **Falcon-AO** as shown in Fig. 2.5. Adjacency-Matrix ($\mathcal{O}_1$) (line 2) produces a binary matrix, G_1 , of size $|V_1| \times |V_1|$, where a value of 1 in the i^{th} row and j^{th} column represents an edge from the vertex indexed by i to the vertex indexed by j in the bipartite graph model of $\mathcal{O}_1$; analogously for AdjacencyMatrix ($\mathcal{O}_2$). The update and distance functions are implemented as shown in lines 9 and 10, respectively, of the algorithm. In particular, the cosine similarity computes the cosine of the two matrices from consecutive iterations serialized as vectors. **Falcon-AO** has participated in OAEI from 2005 to 2010 and has shown good performances in several tracks. It obtained top results in the benchmark track in the early years of OAEI.

2.3.5 Logic based and Scalable Ontology Matching (Logmap)

Logic based and scalable ontology matching (**Logmap**) [47], is a scalable system that performs reasoning and inconsistency checking of discovered correspondences using a fast OWL reasoner. It models ontologies as a set of axioms using the OWL API. It builds an inverted index of lexical attributes of entities for quick lookup. Seed correspondences are generated by matching named classes exactly followed by iteratively checking the alignment for semantic consistency on a Horn knowledge base and repairing the alignment. New correspondences are generated simply by pairing the entities with high lexical similarity in the neighborhood of previously found correspondences. However, this limits **Logmap**'s focus to named entities only while both building the knowledge base and generating new correspondences. **Logmap** has participated in OAEI from 2010 till 2012 and performed well. Noticeably, it ranked second in the conference track and placed among the top systems in many tracks (conference, anatomy, and the large biomedical ontology) in the 2012 edition of OAEI. Importantly, it scales significantly well for large ontologies.

2.3.6 MapPSO

MapPSO [12] utilizes discrete particle swarms to perform the optimization. A particle swarm is used to search for the optimal alignment. Each of K particles in a swarm represents a valid candidate alignment, which is updated iteratively. In each iteration, given the particle(s) representing the best alignment(s) in the swarm, alignments in other particles are adjusted as influenced by the best particle.

Equation 2.2 manifests in **MapPSO** as a two-step process consisting of retaining the best particle(s) (alignment(s)) and replacing all others with improved ones influenced by the best alignment in the previous iteration. The measure of the quality of an alignment in the k^{th} particle is determined by the mean of the measures of its correspondences as shown below:

$$Q(M_k^i) = \frac{\sum_{a=1}^{|V_1|} \sum_{\alpha=1}^{|V_2|} m_{a\alpha} \times f(x_a, y_\alpha)}{|V_1||V_2|} \quad (2.4)$$

where, $m_{\alpha\alpha}$ is a correspondence in M_k^i and f represents a weighted combination of multiple syntactic, semantic and structural similarity measures between the entities in the two ontologies. Improved particles are generated by keeping aside a random number of best correspondences according to f in the alignment in the particle, and replacing others based on the correspondences in the previous best particle. Iterations in **MapPSO** terminate when the increment in Q due to a new alignment matrix is lower than a parameter, η .

MAPPSO ($\mathcal{O}_1, \mathcal{O}_2, K, \eta$)

Initialize:

1. Iteration counter $i \leftarrow 0$
2. Generate seed map between $\mathcal{O}_1$ and $\mathcal{O}_2$
3. Populate binary matrix, M^0 , with seed correspondences
4. Generate K particles using the seed M^0 : $P = \{M_1^0, M_2^0, \dots, M_K^0\}$
5. Search $M_*^0 \leftarrow \arg \max_{M_k^0 \in P} Q(M_k^0)$

Iterate:

6. **Do**
7. $i \leftarrow i + 1$
8. **For** $k \leftarrow 1, 2, \dots, K$ **do**
9. $M_k^i \leftarrow \text{UpdateParticle}(M_k^i, M_*^{i-1})$
10. Search $M_*^i \leftarrow \arg \max_{M_k^i \in P} Q(M_k^i)$
11. **While** $|Q(M_*^i) - Q(M_*^{i-1})| \geq \eta$
12. Extract an alignment from M_*^i

Figure 2.6: Iterative search in MapPSO. Objective function, Q , is as given in Eq. 2.4.

The general iterative search approach of Fig. 2.4(b) manifests in **MapPSO** as shown in Fig. 2.6. The algorithm takes as input the number of particles, K , and the threshold, η , in addition to the two ontologies to be aligned. It iteratively searches for an alignment until it is unable to find one that improves on the previous best alignment by more than or equal to η . As per our knowledge, it is the only alignment algorithm which is naturally parallelizable. MapPSO participated in the OAEI from 2008 to 2010 and has shown acceptable performance.

2.3.7 OWL-Lite Alignment (OLA)

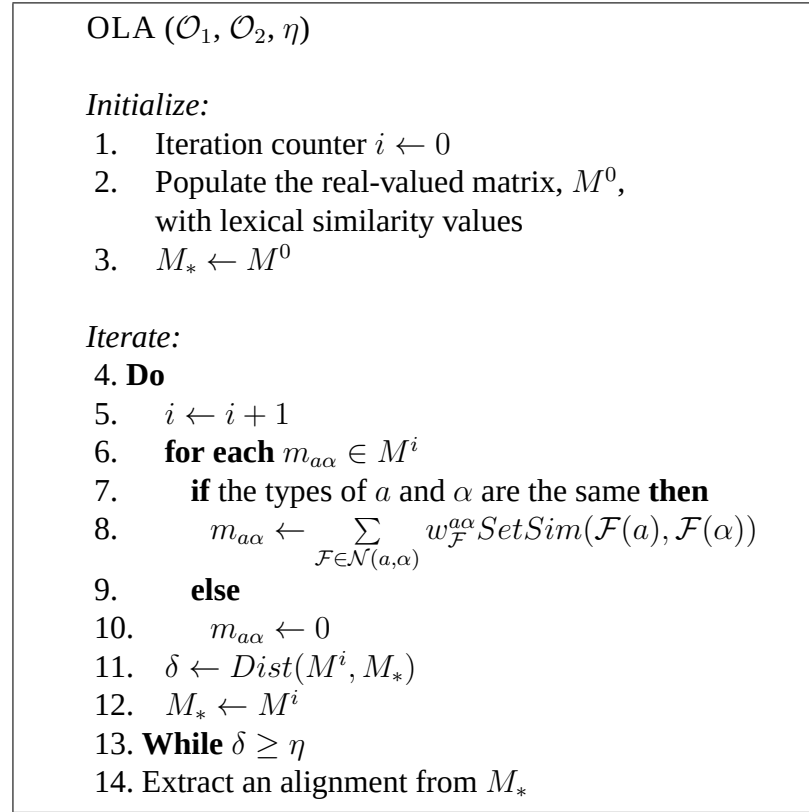


Figure 2.7: OLA's alignment algorithm iteratively updates the alignment matrix using a combination of neighboring similarity values.

OWL-Lite alignment (**OLA**) [24] is limited to aligning ontologies expressed in OWL with an emphasis on its most restricted dialect called OWL-Lite. It has participated in the 2007 version of OAEI and performed reasonable. **OLA** adopts a bipartite graph model of an ontology, and distinguishes between 8 types of nodes such as classes, objects, properties, restrictions and others; and between 5 types of edges: *rdfs:subClassOf*, *rdf:type*, between classes and properties, objects and property instances, *owl:Restriction*, and valuation of a property in an individual.

OLA computes the similarity between a pair of entities from two ontologies as a weighted aggregation of the similarities between the respective neighborhood entities. However, due to its consideration of multiple types of edges, cycles are common. Consequently, it computes the similarities between entities as the solution of a large system of linear equations, which is solved iteratively for the fixed point.

Let $\mathcal{F}(a)$ be the set of all nodes in $\mathcal{O}_1$, which are connected to the node a via an edge type, $\mathcal{F}$. Formally, similarity $Sim(a, \alpha)$, between vertex, $a \in \mathcal{O}_1$, and vertex, $\alpha \in \mathcal{O}_2$, is defined as,

$$Sim(a, \alpha) = \sum_{\mathcal{F} \in \mathcal{N}(a, \alpha)} w_{\mathcal{F}}^{a\alpha} SetSim(\mathcal{F}(a), \mathcal{F}(\alpha)) \quad (2.5)$$

where, $\mathcal{N}(a, \alpha)$ is the set of all edge types in which a, α participate. Weight, $w_{\mathcal{F}}^{a\alpha}$, for an entity pair, a, α , and edge type, $\mathcal{F}$, is normalized, i.e., $\sum_{\mathcal{F} \in \mathcal{N}(a, \alpha)} w_{\mathcal{F}}^{a\alpha} = 1$. Function, $SetSim$, evaluates the similarity between sets, $\mathcal{F}(a)$ and $\mathcal{F}(\alpha)$, as the average of maximal pairing.

OLA initializes a real-valued similarity matrix, M^0 , with values based on lexical attributes only, while the iterations update each variable, $m_{a\alpha}$, in the matrix using the structure of the two ontologies. In particular, if two entities, a and α are of the same type, then $m_{a\alpha}$ is updated using Eq. 2.5, otherwise the value is set to 0. Iterative update of Fig. 2.4(a) is realized by **OLA** as shown in Fig. 2.7. The distance function of line 11 measures the similarity between the updated alignment matrix with that from the previous iteration. The iterations terminate when the distance falls below the parameter, η .

2.3.8 Optima

I have briefly introduced **Optima** [20] algorithm previously in Section 1.4.3. Here I provide details about its iterative alignment approach. **Optima** formulates ontology alignment as a maximum likelihood problem, and searches for the match matrix, M_* , which gives the maximum conditional probability of observing the ontology $\mathcal{O}_1$, given the other ontology, $\mathcal{O}_2$, under the match matrix M_* .

It employs generalized expectation-maximization to solve this optimization problem in which, it iteratively evaluates the expected log likelihood of each candidate alignment and picks the one which maximizes it. It implements Eq. 2.2 as a two-step process of computing expectation followed by maximization, which is iterated until convergence. The expectation step consists of evaluating the expected log likelihood of the candidate alignment given the previous iteration's alignment:

$$Q(M^i | M^{i-1}) = \sum_{a=1}^{|V_1|} \sum_{\alpha=1}^{|V_2|} Pr(y_\alpha | x_a, M^{i-1}) \times \log Pr(x_a | y_\alpha, M^i) \pi_\alpha^i \quad (2.6)$$

where, x_a and y_α are entities in ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$ respectively, and π_α^i is the prior probability of y_α . $Pr(x_a|y_\alpha, M^i)$ is the probability that node x_a is in correspondence with node y_α given the match matrix M^i . The prior probability is computed as,

$$\pi_\alpha^i = \frac{1}{|V_1|} \sum_{a=1}^{|V_1|} Pr(y_\alpha|x_a, M^{i-1})$$

The generalized maximization step involves finding an alignment matrix, M_*^i , that improves on the previous one:

$$M_*^i = M^i \in \mathcal{M} : Q(M^i|M_*^{i-1}) \geq Q(M_*^{i-1}|M_*^{i-1}) \quad (2.7)$$

OPTIMA+ ($\mathcal{O}_1, \mathcal{O}_2$)

Initialize:

1. Iteration counter $i \leftarrow 0$
2. **For all** $\alpha \in \{1, 2, \dots, |V_2|\}$ **do**
3. $\pi_\alpha^0 \leftarrow \frac{1}{|V_2|}$
4. Generate seed map between $\mathcal{O}_1$ and $\mathcal{O}_2$
5. Populate binary matrix, M_*^0 , with seed correspondences

Iterate:

6. **Do**
7. $i \leftarrow i + 1$
8. Search $M_*^i \leftarrow \arg \max_{M \in \mathcal{M}} Q(M|M_*^{i-1})$
9. $\pi_\alpha^i \leftarrow \frac{1}{|V_1|} \sum_{a=1}^{|V_1|} Pr(y_\alpha|x_a, M_*^{i-1})$
10. **While** $M_*^i \neq M_*^{i-1}$
11. Extract an alignment from M_*^i

Figure 2.8: **Optima**'s expectation-maximization based iterative search; it uses binary matrix, M^i , to represent an alignment. The objective function, Q , is as defined in Eq. 2.6.

I show the iterative alignment algorithm of **Optima** in Fig. 2.8, which implements the general iterative search of Fig. 2.4(b). The search for an improved alignment in line 8 is implemented using the two steps of expectation and maximization. Iterations within **Optima** terminate when it does not find any sample $M^i \in \mathcal{M}$, which improves the objective function, Q , further. The similarity matrix maintained by **Optima**, M , consists of the named concepts in the two ontologies

and does not include anonymous classes. It participated in OAEI 2011 and 2012 and ranked second in the conference track of OAEI 2012 with a F-measure of 61%. A detail analysis about Optima's performance in OAEI 2012 is previously presented in Section 1.4.3.

2.3.9 Risk Minimized Based Ontology Matching (RiMOM)

Risk minimized based ontology matching (RiMOM) [51] participated in the OAEI from 2006 to 2010 in many tracks. It uses a combination of a name based strategy (edit distance between labels), a vector based strategy (cosine similarity between vectors), and a strategy taking instances of concepts into account. RiMOM models the alignment problem as a decision problem instead of the traditional similarity problem in other tools. RiMOM brings instance matching along with concept matching into its Bayesian decision theoretic model. It has participated in OAEI 2007 to 2010. RiMOM has shown very good results in instance matching track and ranked first on both 2009 and 2010 with an average F-measure of 80%. Noticeably, it is the only alignment tool which successfully completed all the data sets from instance track in 2010 OAEI.

2.3.10 Yet Another Matcher (YAM++)

Yet Another Matcher (**YAM++**) [66] is an automatic, flexible and self-configuring ontology alignment algorithm for identifying semantic correspondences. **YAM++** utilizes techniques based on machine learning, information retrieval, and graph matching within its alignment process. In particular, it uses a decision tree to combine different similarity measures. A similarity propagation method is used to discover correspondences by exploiting the structure of the ontology. It has participated in the last couple of years (2011–2012) of OAEI campaigns. **YAM++** placed first in the conference and large biomedical ontology tracks in the 2012 OAEI edition, while placing second in the anatomy track. **YAM++** is widely regarded as generating the most accurate and complete alignments among all algorithms. Yet, these algorithms may not align very large ontologies due to memory issues or are unable to produce an alignment in a reasonable amount of time. **YAM++**,

was the slowest in the large biomedical track and was unable to complete the conference track within 5 hours.

2.4 Scalable Alignment Algorithms

Ontology matching is seen by many as an offline process, and systems are often not designed with scalability in mind. As a case in point, less than half the alignment algorithms that participated in the 2012 instance of the annual OAEI competition [84] generated acceptable results for aligning moderately large ontologies. Crucial challenges for many alignment algorithms involve scaling to large ontologies and performing the alignment in a reasonable amount of time without compromising on the quality of the alignment. On the other hand, real-world ontologies tend to be very large with several containing thousands of entities. For example, popular biomedical ontologies FMA, SNOMED and NCI contain 78,989, 306,591 and 66,724 classes respectively. Increasingly, ontologies are hosted in repositories, which often compute the alignment between the ontologies. As the changes emerge with submission and editing of ontologies, new alignments must be computed in order to accompany consistency to the alignments. Therefore, aligning several pairs of ontologies quickly becomes a challenge for these repositories.

An element level matcher searches the space of all correspondences formed by the Cartesian product of entities of the given two ontologies, $\mathcal{O}_1$ and $\mathcal{O}_2$. Therefore, the time complexity of such a matcher is $O(|\mathcal{O}_1| \times |\mathcal{O}_2|)$. Here $|\mathcal{O}_i|$ is the total numbers of entities in ontology $\mathcal{O}_i$. Often, alignment algorithms employ several element level matchers for improved recall. Subsequently, the complexity gets multiplied by the number of matchers used. Some alignment algorithms [7, 47] provide a lighter version for large ontologies in which they employ methods to quickly identify correspondences, called anchors, in which the pair of entities share a similar name or label. Here they often use just exact string matching. Recently, **Logmap** presented an algorithm to identify entity pairs with exact labels or names with a time complexity of $O(|\mathcal{O}_1| + |\mathcal{O}_2|)$ using efficient indexing [47]. However, these approaches suffer from low recall. In order to improve the recall without drastically increasing the execution time, some algorithms limit their scope to

search around the neighborhood of anchors or previously found correspondences. For example, the ASMOV [45] alignment algorithm limits its scope to all the entities that are at most at a edge-distance of 2 from an entity participating in a previously found correspondence. Wang et. al. presented an approach [94], called *quick ontology matching*, to reduce the search space by pruning incompatible entity pairs around previously found correspondences. For each anchor this approach avoids evaluating the entity pairs formed by pairing ancestors of one entity from that anchor with descendants of the other in the anchor. This reduces the time complexity to $O(n \cdot \lg(n))$ where n is the maximum of $|\mathcal{O}_1|$ and $|\mathcal{O}_2|$. These efforts to improve efficiency of element level matchers did not enable alignment algorithms to align large ontologies with more than thousands of concepts. Specifically, alignment algorithms with ontology level matchers found it difficult to scale-up for very large ontologies by reducing search space using early pruning of incompatible entity pairs.

With a view to scale-up alignment algorithms, especially the ones with ontology level matching, researchers often adopt divide and conquer strategy. This strategy significantly reduces the search space by partitioning the ontologies [20, 34, 41] and only aligns the parts which share significant alignment between them. Hu et. al. proposed a partitioning technique [41], for **Falcon-AO**, by clustering the entities based on structural cohesiveness. Then, among the Cartesian product between parts of one ontology and the other, the pair of ontology parts which share significant anchors are aligned. Unlike some other approaches which simplify the algorithms, this technique applies the original algorithm to the selected partition pairs. Hence, these techniques may provide better recall than those crude approaches while maintaining the precision but, have associated overhead of partitioning. Hamdi et. al. provided an improved partitioning algorithm [34] specifically tailored for ontology matching based on the technique [41] by Hu et. al. Instead of independently dissecting each ontology, this technique utilizes anchors within partitioning such that, partitions are centered at anchors.

Often ontology level matchers are iterative in nature, suchlike similarity flooding [61], graph matching for ontologies (GMO) [39], and inexact matching of ontology graph algorithm [20]. Through repeated improvements, the computed alignment is usually of high quality but these

approaches also consume more time in general than their non-iterative counterparts. While the focus on computational complexity has yielded ways of scaling the alignment algorithms to larger ontologies, such as through ontology partitioning [41, 77, 88], there is a general absence of an effort to speed up the ontology alignment process. In this dissertation, I introduce an approach for speeding up the convergence of iterative ontology alignment techniques using *block-coordinate descent*. While BCD forms a standard candidate tool for multidimensional optimization and has been applied in contexts such as image reconstruction [26, 70] and channel capacity computation [3, 11], my dissertation research presents its first application toward ontology alignment.

The other technique to tackle the complexity in aligning very large ontologies is to exploit parallelization. As we mentioned before there exists two major classes of parallelization in ontology alignment algorithms, *intra-matcher parallelization* and *inter-matcher parallelization*. Intra-matcher parallelization introduces parallelization within the alignment algorithm. For example, parallelly evaluating element level matchers falls under intra-matcher parallelization. On the other hand, inter-matcher parallelization aligns several ontology parts in parallel using ontology alignment algorithms [31].

We know of only one contemporary alignment algorithm, MapPSO [12], whose *inherent* computations may be distributed in a straightforward way. Some existing algorithms [31, 43, 97] parallelize the lexical similarity calculations. Gross et. al. [31] discussed some element-level and instance-level matchers, whose internal computations may be parallelized and illustrated the effectiveness of parallelization on aligning biomedical ontologies. The popular distributed MapReduce [18] paradigm has been utilized to implement intra-matcher parallelization of element level lexical matchers. For example, the SILK framework [43] allows several element level similarity measures to be calculated on linked-open data using MapReduce and the similarity is merged. Recently, Zhang et. al. [97] computed VDOC similarity measure [71] using MapReduce leading to the enhanced VDOC+.

Though intra matcher parallelization brings in some amount of speedup it is not suitable for established structural matchers and logic based matchers such as similarity flooding [66], graph

matching [20, 46] and **Logmap** [47] to gain speedup. Interestingly, Paulheim demonstrated [68] that parallel alignment of ontology partitions may help to scale the alignment with 5% reduction in the quality. However, as Rahm notes [72], there is a general absence of inter-matcher parallelization frameworks for parallel execution of complex ontology alignment process, which are independent of algorithm. Importantly, the lack of ontology level alignment algorithms amenable to parallelization is noted.

In the context of a general absence of inter-matcher parallelization, one of our primary contributions in this dissertation is a novel and general method for batch alignment of large ontology pairs using the distributed computing paradigm of MapReduce [18]. As distributed computing clusters, including cloud computing, proliferate, the significance of this approach is that it allows us to exploit these parallel computing resources toward automatically aligning several ontologies whose scale takes them out of the reach of many of the current algorithms, and simultaneously align in a reasonable amount of time.

CHAPTER 3

ON THE UTILITY OF WORDNET FOR ONTOLOGY ALIGNMENT

Many ontology alignment algorithms augment syntactic matching with the use of WordNet(WN) in order to improve their performance. The advantage of using WordNet in alignment seems apparent. However, we strike a more cautionary note. We analyze the utility of WordNet in the context of the reduction in precision and increase in execution time that its use entails. For this analysis, we particularly focus on real-world ontologies. We report distinct trends in the performance of WordNet-based alignment in comparison with alignment that uses syntactic matching only. We analyze the trends and their implications, and provide useful insights on the types of ontology pair for which WordNet-based alignment may potentially be worthwhile and those types where it may not be.

For this study¹ we select a recognized ontology alignment algorithm based on iterative expectation-maximization, which produces the most likely match between two given ontologies [20]. This algorithm uses both the structure of the ontologies and their lexical similarity in arriving at the match. We perform this experiment comprehensively using ontology pairs that appear in the real-world ontologies track of the OAEI 2009 edition [82]. For this analysis, I think that the real-world ontology pairs are most appropriate due to the nature of this study.

I uncover some surprising trends while comparing the performance of ontology alignment enhanced with WordNet and that of alignment that uses syntactic matching only. While, in many cases, the WordNet-enhanced alignment expectedly achieved a better recall and F-measure, it did so while taking significantly more time and aligning without it achieved nearly identical performance in less time. I also report on several pairs where the WordNet-enhanced alignment did not

¹This study was conducted at 2009, hence the **Optima** used in this study was the version available at that time and the data used is from OAEI 2009.

improve on the performance of the original alignment algorithm. Consequently, I investigate characteristics of the ontology pair that would likely facilitate improved performance when a lexical database such as WordNet is used during the alignment, and particularly those which would hinder its performance. I think that many of the outcomes of this analysis are novel and useful in evaluating the use of computationally intensive add-ons such as WordNet.

This study has insights for both ontology alignment researchers and users, and provides useful guidance on utilizing lexical knowledge sources for ontology alignment. Its results provide clear evidence against commonly held beliefs that, (a) the use of WordNet in ontology alignment *always* improves the recall of the alignment; and (b) any improvement in the recall supersedes the loss in precision that WordNet may bring, and this is notwithstanding the excessive execution time due to using WordNet. The contributions of this novel study in the context of alignment are two-fold: First, it shows that the utility of WordNet in aligning ontologies is not always clear, and the use of WordNet not always advisable. This is demonstrated by comparing the performance of ontology alignment with WordNet and that of alignment without WordNet. For example, we show that multiple benchmark ontology pairs do not exhibit improvements in recall when WordNet is used despite the larger execution time. More importantly, several benchmark ontology pairs do not show a marked improvement in F-measure when WordNet is utilized to help the alignment process. Second, it recommends a set of “rules of thumb” for ontology alignment users in order to decide whether WordNet would be worthwhile for a given ontology pair. For example, I discover that ontologies with deep hierarchies take far more time when aligned with WordNet than ontologies with shallow hierarchies.

3.1 WordNet And Ontology Alignment

As mentioned earlier the basic building block of ontology alignment is the element level matchers. Regularly, alignment algorithms use lexical matchers to evaluate similarity between entities. Similarity measures may be broadly categorized into syntactic and semantic. Syntactic similarity between concepts is entirely based on the sequence similarity between the concepts’ names, labels

and other associated text. Semantic similarity measures attempt to utilize the meaning behind the concept names to ascertain the similarity of the concepts. A popular way of doing this is to exploit lexical databases such as WordNet, which provide words related in meaning. WordNet is a lexical database from the psycholinguistic theorist which defines words and their association with other words along with a descriptive gloss. WordNet consists of a set of synonyms *synsets*. A synset defines a sense of a group of terms. All the terms in a synset are synonymous to each other in the sense of the concept they represent. All the different synsets of the term *sample* in the WordNet is illustrated in Fig. 3.1. It appears in three noun senses and one verb sense. Notice, that the second sense of the term *sample* have synonym terms *sample distribution* and *sampling*. Synsets are also related via different semantic relationships such as antonymy (opposite), hypernymy (superconcept)/hyponymy (subconcept)(also called Is-A hierarchy / taxonomy), meronymy (part-of) and holonymy (has-a) [53]. Semantic similarity measures exploit these relationships and gloss in evaluating the similarity between terms [96].

noun: sample <i>sample</i> <i>a small part of something intended as representative of the whole</i>	noun: sample distribution, sample, sampling <i>items selected at random from a population and used to test hypotheses about the population</i>	noun: sample <i>all or part of a natural object that is collected and preserved as an example of its class</i>	verb: sample, try, try out, taste <i>take a sample of "Try these new crackers"; "Sample the regional dishes"</i>
--	--	--	--

Figure 3.1: All four synsets of term *sample* in WordNet are illustrated. The term *sample* has 3 senses as a noun and has only one meaning as a verb. The meaning of a synset can be represented using all the terms in it. For example, the terms *sample* and *taste* are synonymous verbs point to the action of **take a sample of**. In WordNet each synset is annotated with a descriptive gloss which.

The use of WordNet enhances the traditional syntactic or string-based matching between the labels of entities with the ability to match words that could be synonyms, hypernyms, and in other lexical senses. Alignment algorithms utilize WordNet due to the potential improvement in recall of the alignment. This predicted improvement is reinforced by previous studies of using WordNet [96], which cite the improved recall to unconditionally recommend using WordNet in

alignment. However, I strike a more cautionary note on the utility of WordNet in ontology alignment. Although its use may improve recall, one trade off is that precision typically suffers. This has been studied by Mandala et al. [55] in the context of information retrieval with the revelation that WordNet’s significant negative impact on precision cannot be ignored while deciding on its use. Additionally, in contrast to the previous studies [53, 96], I consider the increased computational expenditure in the form of execution time as well while evaluating the performance gains. I think that execution time is a critical component of the evaluation because automatically aligning ontologies is computationally intensive, which is exacerbated as the ontologies become larger. *While alignment is often viewed as an offline and one-time task, continuously evolving ontologies and applications involving real-time ontology alignment such as semantic search and Web service composition stress the importance of computational complexity considerations [42].* Consequently, in this study I position the possibly improved performance gains from using WordNet in the context of the increased computational time that the enhanced alignment entails.

3.2 Integrating WordNet

As I mentioned before both syntactic and semantic similarity measures are widely used for ontology alignment. **Optima** utilizes the well-known Smith-Waterman [86] technique for ascertaining the syntactic similarity between concept and relationship names. I enhance the syntactic similarity to include knowledge from WordNet [62] as a representative lexical database, popularly used by many ontology alignment tools. In a comparison of different ways of using WordNet to match concept names, Yatskevich and Giunchiglia [96] demonstrate that gloss-based similarity measuring algorithms (matchers) showed the best matching performance. These matchers compute the cosine similarity between the glosses (definitions) provided by WordNet for the given words. Consequently, I integrate these matchers with the syntactic matching in **Optima**. However, these matchers do not utilize the structure of WordNet – *synsets* and how they relate to each other – and associated statistical knowledge. Hence, I also include another popular and competitive method [52], which uses WordNet’s structure. As I seek to evaluate the incremental utility

of WordNet, I augment the existing syntactic similarity in **Optima** with these WordNet-based similarity measures.

3.2.1 Adding WordNet-based Similarity

A known limitation of Lin’s method [52] is, its poor performance when the concept labels are word phrases instead of single words. In this case, I evaluate the WordNet-based similarity using the gloss-based matcher that accumulates the glosses of each word in the phrase. Consequently, I use Lin’s approach if both labels are single words, otherwise the gloss-based matcher is utilized. I denote this way of utilizing WordNet using *Sem*.

Lin proposes the use of information content in computing the semantic similarity between labels using WordNet:

$$Lin(x_a, y_\alpha) = \frac{2 \times IC(lcs(x_a, y_\alpha))}{IC(x_a) + IC(y_\alpha)} \quad (3.1)$$

Here, the information content (IC) is computed by looking up the frequency count of its argument word in standard corpora [56]. The term, $lcs(x_a, y_\alpha)$, is the least common subsumer of the two words, x_a and y_α , within the WordNet hierarchy. *Lin* is guaranteed to be between 0 and 1.

Let x_a, y_α be the two concepts for which the similarity to be measured and the number of words in each concepts be w_a and w_α respectively, then the time complexity of the Lin similarity is $\mathcal{O}(w_a \cdot w_\alpha \cdot s_a \cdot s_\alpha \cdot h)$ [44]. Here, the number of senses in WordNet for x_a and y_α are s_a and s_α respectively with h being the maximum depth of both the concepts in WordNet hierarchy. The time complexity of the gloss based similarity would be then, $\mathcal{O}(w_a \cdot w_\alpha \cdot s_a \cdot s_\alpha \cdot g_a \cdot g_\alpha)$, where g_a and g_α are the maximum number of words in any single gloss in WordNet for concepts x_a and y_α respectively. Note that the number of words in a concept and the depth of the words in the WordNet hierarchy determine the complexity of computing its similarity using WordNet.

There is no standard way of integrating WN-based similarity with syntactic measures. We define a normalized 3D function that maps a given pair of semantic and syntactic similarity to the integrated value. In order to generate this function, we observe that labels that are syntactically

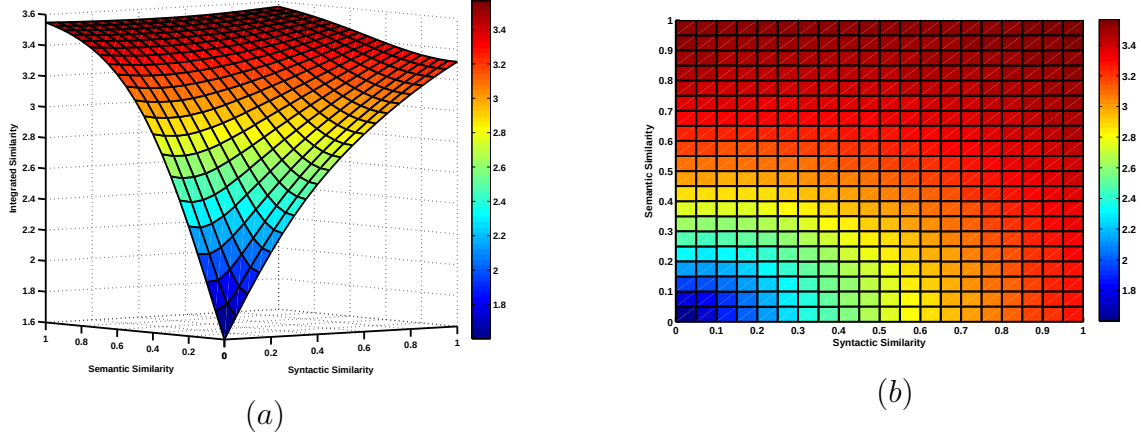


Figure 3.2: Integrated similarity measure as a function of the WordNet-based semantic similarity (*Sem*) and Smith-Waterman based syntactic similarity (*Syn*). Notice that the value is lower if semantic similarity is low but syntactic is high compared to vice versa.

similar (such as *cat* and *bat*) may have different meanings. Because we wish to meaningfully map entities, semantic similarity takes precedence over syntactic. Consequently, high syntactic but low semantic similarity results in a lower integrated similarity value in comparison to low syntactic but high semantic similarity. We model such an integrated similarity measure as shown in Fig. 3.2 and give the function in Eq. 3.2. Our integrated similarity function is similar to a 3D sigmoid restricted to the quadrant where the semantic and syntactic similarities range from 0 to 1. One difference from the exact sigmoid is due to the specific property it must have because semantic similarity takes precedence over syntactic.

$$Int(x_a, y_\alpha) = \gamma \frac{1}{1 + e^{t \cdot r - c(Sem)}} \quad (3.2)$$

Here, γ is a normalization constant; $r = \sqrt{Syn^2 + Sem^2}$, which produces the 3D sigmoid about the origin; t is a scaling factor and $c(Sem)$ is a function of the semantic similarity as shown below:

$c(Sem) = \frac{2}{1 + e^{t' \cdot Sem(x_a, y_\alpha) - c'}}$ where t' is the scaling factor and c' is the translation factor, if needed. The specific function in Fig. 3.2 is obtained when $t = 4$, $t' = 3.5$, and $c' = 2$.

3.3 Experiments

As I mentioned previously, alignment algorithms have used lexical databases such as WordNet based on the potential improvement in the alignment that it could generate. Furthermore, past studies of using WordNet do not take into account the increased computational load that utilizing WordNet entails. I analyze the implications of using WordNet on the alignment performance in the context of **Optima**.

3.3.1 Methodology

We utilized execution time as an indicator of the computational load. In order to incorporate execution time within my experimentation, we measure the maximum *recall* and *F-measure* that **Optima** attains on a pair of ontologies given varying execution time. We evaluated recall and F-measure because integrating WordNet typically results in improved recall but reduced precision, which would be collectively reflected in the F-measure.

The alignment performance was measured with the integrated similarity measure and independently using just the syntactic similarity between node labels, in order to evaluate the utility of WordNet. I used OAEI in its recent version, 2009, as the testbed for benchmarking. Within the benchmark, I mostly focus on the track that involves real-world ontologies for which the reference (true) alignment was provided by OAEI. These ontologies are not created or altered for purposes related to the benchmark and were obtained by OAEI from the Web. This includes all ontology pairs in the 300 range which relate to *bibliography*, and expressive ontologies in the *conference* track all of which structure knowledge related to conference organization. Because I wish to evaluate the utility of WordNet in practical use, I focused on real-world ontologies. However, I selected one pair of ontologies specifically tailored by the benchmark that contained synonyms of node labels. I list the ontologies participating in my evaluation in Table A.1 and provide an indication of their sizes.

I ran each execution – with WordNet and without – until there was no improvement in the performance. During the execution, I recorded the recall and F-measure every time it changed

along with the time consumed till then. Because of the iterative nature of **Optima**, the alignment performance usually improves as more time is allocated until the EM converges to a maxima. I note that I seed both executions with the same initial alignment to facilitate comparison.

3.3.2 Results and Analysis

While I ran my evaluations on 23 pairs of ontologies, in this section I focus on a set of 6 pairs, which are representative of the different trends that I obtained. I show my evaluations on some of the remaining pairs in the Appendix B. Because of the large number of pairs that I evaluated on (23 in all), I ran the tests on three different computing platforms. Two of these were Red Hat machines with Intel Xeon Core 2, processor speed of about 3 GHz with 4GB of memory, while the third was a Windows Vista machine with Intel Xeon Core 2, 2.4 GHz processor and 4GB of memory.

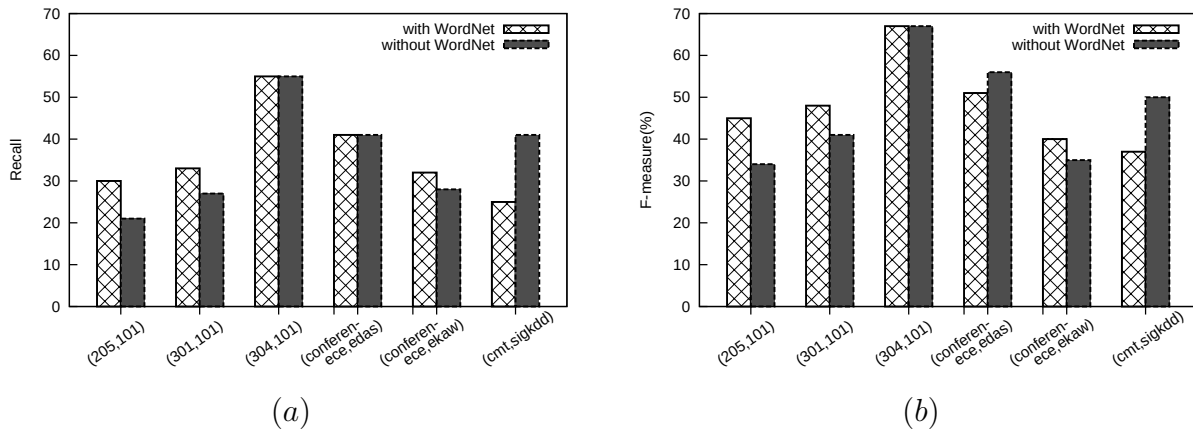


Figure 3.3: (a) Final recall and (b) final F-measure generated by **Optima** on 6 representative ontology pairs, with the integrated similarity measure and with just the syntactic similarity between entity labels.

I show a summary of the final recall and F-measure that was obtained on the 6 pairs with WordNet integrated and with just the syntactic similarity measure, in Figs. 3.3 (a, b). My focus is on the change in these measures, and not their overall values which could be poor for some ontology pairs. As we may expect, for many of the ontology pairs, the final recall with WordNet integrated is higher than the recall with just the syntactic similarity. For example, while aligning the ontology pair (101, 301), the alignment process with WordNet matches the concept *Monograph* against

the concept *Book*, which is not possible with using just the syntactic similarity. The difference in performance is statistically significant with p -value of 0.057 as measured using a paired Student's t -test. On the other hand, integrating WordNet decreased the recall for a single pair, (*cmt,sigkdd*). However, the improvement in F-measure due to WordNet reduces to the extent where it loses significance (p -value=0.184).

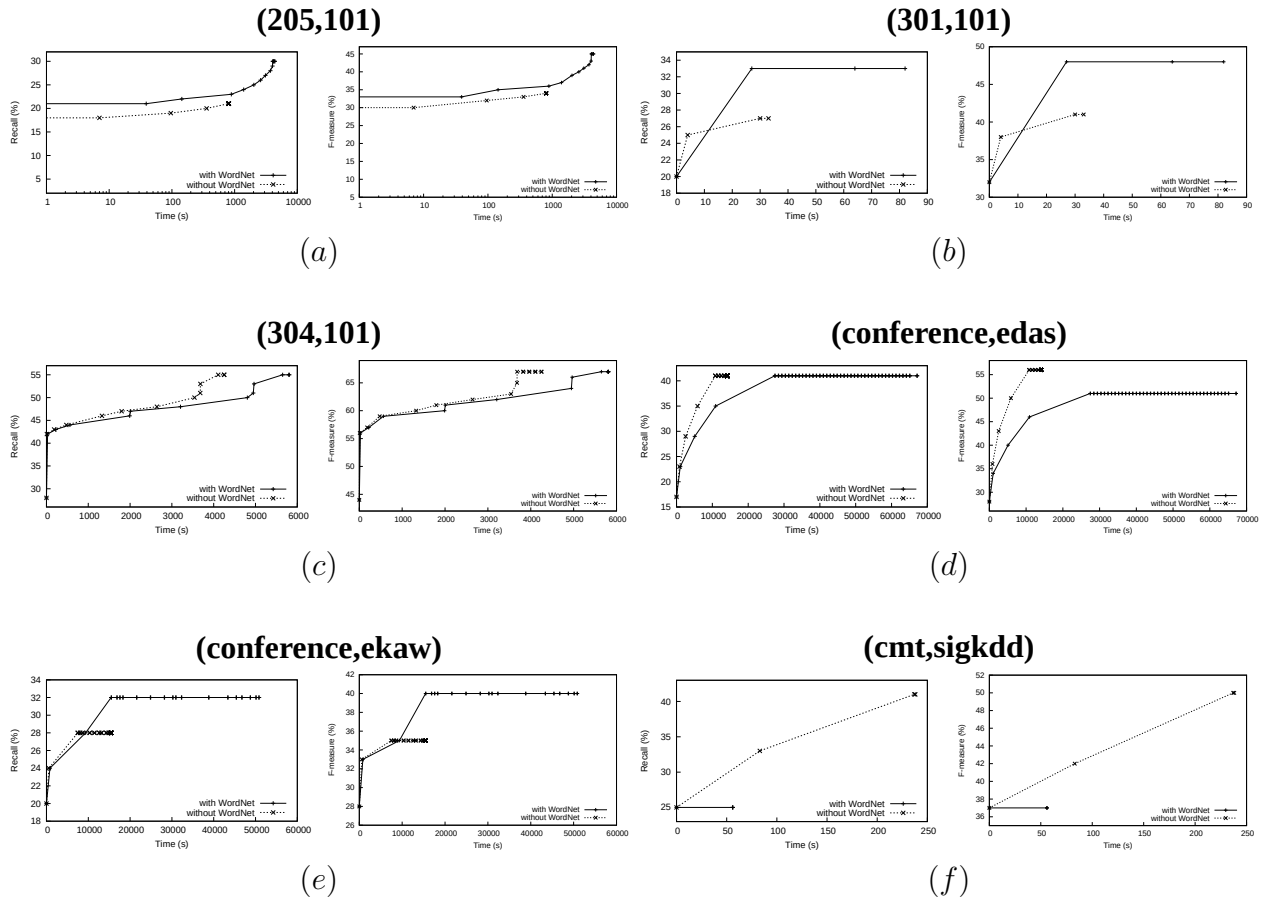


Figure 3.4: (a) – (f): Recall (left) and F-measure (right) for 6 of the 23 ontology pairs that I used in my evaluations. I show the evaluations when the alignment algorithm utilized an integrated similarity involving WordNet and just the string-based similarity without WordNet. Notice the different trends in my evaluations. Ontologies related to *conference* consume more time because they are larger.

In Fig. 3.4, I detail the performances w.r.t. execution time. Each data point is the maximum recall or F-measure, as appropriate, that could be obtained given the execution time. Notice that Figs. 3.4(a, b, e) all show an improved recall with WordNet integrated. In particular, ontology 205

in the pair (205, 101) is altered by OAEI to include synonyms of labels in 101, as its entity labels. For example, *title* is altered to *heading*. In some cases, the WordNet-based integrated similarity leads to better recall eventually. However, the improvement is obtained after spending significantly more time on the alignment process; in some cases approximately an order of magnitude more time was consumed to achieve a significant increase, as in Fig. 3.4(a). The additional time is spent on initializing WordNet and querying the database. Further, in two of these, aligning without WordNet results in better recall for an initial short time span (Fig. 3.4(b, e)), before the performance with WordNet exceeds it.

On the other hand, some ontology pairs did not exhibit an improved recall with WordNet (see Figs. 3.4(c, d, f)). Surprisingly, conference ontology pair (*cmt*, *sigkdd*) results in worse recall with WordNet integrated (Fig. 3.4(f)). This is because (*cmt*, *sigkdd*) pair has several concepts with compound words or phrases as labels. As one example, *Meta-Review* appears in *cmt* ontology and *Registration_Non-Member* appears in *sigkdd* ontology. Tokenizing these correctly and locating individual glosses in WordNet is often challenging², resulting in low semantic and therefore low integrated similarity. However, the string-based similarity resulted in better label matching.

The F-measure evaluations of the alignments tell another story. I predominantly found that the improvement in F-measure due to WordNet was smaller in comparison to the improvement in recall. Thus, the use of WordNet often leads to reduced precision than if I did not use it. Due to its consideration of synonyms and other lexical senses, semantic similarity is often high for multiple concepts across the two ontologies. However, not all of these possible matches appear in the true alignment. For example, while the final recall in Fig. 3.4(d) does not change when WordNet is utilized, the final F-measure drops to below what I could get when just the syntactic similarity is used in **Optima** for the alignment. The mapping between concepts *Conference_part* and *ConferenceEvent* in ontologies *Conference* and *edas*, respectively, is one such example that is found

²The concept *Meta-Review* should be tokenized into two words (*Meta*, *Review*) while *Registration_Non-Member* needs to be tokenized into two words (*Registration*, *NonMember*) but should not be tokenized into three words (*Registration*, *Non*, *Member*). The hyphen (–) is a delimiter in the former concept but should be just ignored in the later concept. This tokenization is demanded by WordNet matchers since *MetaReview* does not exist in WordNet but the word *NonMember* exists in WordNet.

Table 3.1: The different ontology pairs could be grouped into 4 trends of alignment performance based on the recall and F-measure evaluations.

	Max. recall with WordNet > Max. recall without WordNet	Max. recall with WordNet = Max. recall without WordNet	Max. recall with WordNet < Max. recall without WordNet	Count
Max. F-measure with WordNet > Max. F-measure without WordNet	(205, 101); (301, 101); (confOf, ekaw); (edas, iasted); (cmt, conference); (conference, iasted); (conference, ekaw)	none	none	7
Max. F-measure with WordNet = Max. F-measure without WordNet	none	(304,101); (cmt, confOf); (ekaw, iasted); (ekaw, sigkdd); (iasted, sigkdd); (conference, sigkdd)	none	6
Max. F-measure with WordNet < Max. F-measure without WordNet	none	(302, 101); (303, 101); (confOf, edas); (edas, sigkdd); (edas, ekaw); (cmt, edas); (cmt, ekaw); (conference, confOf); (conference, edas)	(cmt, sigkdd)	10
Count	7	15	1	23

by **Optima** with WordNet but is incorrect and therefore leads to lower precision. Furthermore, the increased execution time due to WordNet for achieving an F-measure is significant (p -value = 0.013).

Overall, we saw general trends where, (i) the final recall and F-measure due to WordNet improved considerably although the lower values of recall and F-measure were achieved without the use of WordNet in much less time; (ii) alignment with WordNet exhibited similar or better recall but poorer F-measure due to reduced precision; and (iii) integrating WordNet degraded the alignment performance, although this was rare. I tabulate the alignment performance on all the 23 different pairs based on the trends, in Table 3.1. Interestingly, 15 of the 23 pairs that I used did not exhibit an increase in recall due to the additional use of WordNet, and 9 of these showed a decrease in overall F-measure.

3.4 Recommendations

The results in the previous section demonstrate that integrating a lexical database such as WordNet may not always be worthwhile especially if the execution time is a concern as well. In particular, the performance in terms of recall or F-measure did not improve for 15 of the 23 ontology pairs when an integrated similarity measure involving WordNet was utilized. However, the execution time increased considerably. Clearly, the utility of WordNet for these ontology pairs is negligible. I investigated these pairs in greater detail to ascertain the differential properties that could lead to minimal performance improvement on using WordNet. These would allow us to make an informed decision on whether WordNet would be worthwhile for a given ontology pair.

- Interestingly, ontologies that have a deep hierarchy (“tall” ontology) may consume an excessive amount of time when aligned using WordNet. This is because such ontologies tend to have several specialized classes, and identifying the least common subsumer in WordNet required by algorithms such as Lin [52] requires traversing a large portion of the WordNet hierarchy (see section 3.2.1). An example of this is the ontology pair, (*conference*, *edas*), in which the ontology *edas* is a tall ontology.
- Furthermore, if such ontologies need to be aligned with those that have a shallow hierarchy (“short” ontology), WordNet will likely suggest several matches between the specific con-

cepts³ of the tall ontology and more general concepts of the short ontology, thereby leading to reduced precision.

- We may search WordNet using single words only. Consequently, compound words or phrases appearing as entity labels in an ontology need to be appropriately tokenized and a single representative word or WordNet-based similarity measure must be obtained. This is further complicated if the phrases are not formatted in a uniform manner making tokenization challenging. An example of this is the ontology pair, (*cmt*, *sigkdd*), which leads to poor performance with WordNet due to the difficulty in improving over the seed map (see Fig. 3.4(f)).

Of course, my study could be enhanced by evaluating the utility of WordNet in the context of multiple alignment algorithms and more ways of using WordNet. However, my focus on the relative change in performance due to WordNet reduces the effect of the choice of the underlying algorithm on the results, and I sought to select multiple competitive WordNet-based matchers with prior support. As such, I think that my results reflect the general pattern. Additionally, I used 23 independently developed real-world ontology pairs from two distinct domains (bibliography and conferences), which I think is a relatively versatile dataset from which to generalize my conclusions. Furthermore, emerging applications of ontology alignment such as in semantic Web services and search bring new emphasis on alignment execution time.

³Specific concepts (e.g., *Presenter* in "tall" *edas* ontology) appear at the lower part of the WordNet hierarchy tree compared to general concepts (e.g., *Person* in "short" *confOf* ontology) which stay closer to the root of the WordNet tree.

CHAPTER 4

MODELING COMPLEX CONCEPTS FOR COMPLETE ONTOLOGY ALIGNMENT

Contemporary languages for describing ontologies such as the Web ontology language (OWL 2) [59] and the resource description framework schema (RDFS) [57] identify concepts using the internationalized resource identifier (IRI). RDF allows using *blank nodes* to represent resources that do not have an IRI. Analogously, OWL utilizes anonymous classes to represent certain class descriptions. These include concepts involving restrictions, Boolean combinations of classes and an exhaustive enumeration of individuals. We call these *complex concepts*; these are part of class expressions in OWL 2 [59].

Due to the absence of distinguishing labels, complex concepts are insufficiently utilized by many alignment algorithms. Being an important part of the conceptualization in ontologies, ignoring complex concepts often leads to an incomplete and inaccurate alignment. As an illustration, about 40% of the ontologies in BioPortal have more than a thousand complex concepts, and in about 60% of the ontologies, these concepts constitute 25% or more of all.

Modeling complex concepts for participation in the alignment is challenging. Though these concepts do not possess the attributes of a named class such as a label, comment and IRI, they contain significant semantics in the context of their respective ontology. This meaning is especially helpful when other attributes are insufficient. For example, consider the two ontologies containing complex concepts in Fig. 4.1. Ontology in Fig. 4.1(a) classifies people based on their eating norms while the ontology in Fig. 4.1(b) classifies animals based on their diet. Finding the correspondence between classes *Vegetarian*, *NonVegetarian* in Fig. 4.1(a) and classes *Herbivore*, *Carnivore* and *Omnivore* in Fig. 4.1(b) is challenging because the names do not contain enough lexical similarity. None of the state-of-the-art alignment algorithms such as RiMOM [51], LogMap [47],

Declaration(Class(Food))	Declaration(Class(People))
Declaration(Class(Plant))	Declaration(Class(Vegetarian))
Declaration(Class(Meat))	Declaration(Class(NonVegetarian))
SubClassOf(Meat, Food)	SubClassOf(Vegetarian, Animal)
SubClassOf(Plant, Food)	SubClassOf(NonVegetarian, Animal)
Declaration(ObjectProperty(eats))	
SubClassOf(Vegetarian, ObjectAllValuesFrom(eats, Plant))	
SubClassOf(NonVegetarian, ObjectAllValuesFrom(eats, ObjectUnionOf(Meat, Plant)))	

(a) PeopleOntology

Declaration(Class(Food))	SubClassOf(Meat, Food)
Declaration(Class(Plant))	SubClassOf(Plant, Food)
Declaration(Class(Meat))	Declaration(Class(Animal))
Declaration(Class(Herbivore))	SubClassOf(Herbivore, Animal)
Declaration(Class(Omnivore))	SubClassOf(Omnivore, Animal)
Declaration(Class(Carnivore))	SubClassOf(Carnivore, Animal)
Declaration(ObjectProperty(eats))	
SubClassOf(Herbivore, ObjectAllValuesFrom(eats, Plant))	
SubClassOf(Carnivore, ObjectAllValuesFrom(eats, Meat))	
SubClassOf(Omnivore, ObjectAllValuesFrom(eats, ObjectUnionOf(Meat, Plant)))	

(b) AnimalOntology

Figure 4.1: (a) People and (b) Animal ontologies that classify people and animals respectively, depending on the type of food they eat. Both the ontologies have multiple restricted concepts.

YAM++ [66], Optima+ [20], or Falcon-AO [40] yielded a complete alignment. While the algorithms yielded trivial correspondences, $(Animal, People)$, $(Food, Food)$, $(eats, eats)$, $(Meat, Meat)$, $(Plant, Plant)$, target correspondences, $(Herbivore, Vegetarian)$, $(Omnivore, NonVegetarian)$ were not discovered.

In cases as the one above, when the typical lexical and structural similarity are insufficient to suggest a correspondence, finding a complete alignment between two ontologies is not straightforward. To infer such difficult correspondences, the complex concepts should be utilized by the alignment process. Understanding the semantics and structure of the complex concepts is a first step toward this goal. Involving complex concepts adds inferencing capabilities to the alignment algorithms, which may help in improving their performance both in finding more correspondences and pruning the incorrect ones. In theory, semantic approaches such as S-Match [27] have the poten-

tial to discover correspondences between the complex concepts. However, in practice, semantic approaches do not scale – **S-Match** is limited to small taxonomies – necessitating a combination of fast lexical techniques for discovery of correspondences with partial consideration of semantics, such as for validation as in **Logmap**. As We pointed out, **Logmap** too did not identify the correspondences between the complex concepts in Fig. 4.1.

In this chapter, I present a novel and general way of modeling complex concepts. To the best of our knowledge, this study is the first in its *explicit* focus on modeling complex concepts for improving ontology alignment. We seek to find the similarity between the anonymous classes that appear in the definition of these concepts, so that it may be utilized by existing algorithms analogously to the other named entities. Alignment algorithms model OWL ontologies either as a set of axioms [27, 45, 47] or as a graph [20, 40, 51, 66]. Consequently, I introduce axiomatic representations of the different types of complex concepts in canonical forms, and additionally derive RDF graph-based canonical representations that model the associated OWL axioms without any loss in meaning. Subsequently, We compare the corresponding entities represented either axiomatically or as subgraphs in their canonical forms (canonicalized), in order to obtain a similarity between the anonymous classes or their graphical representation as blank nodes. This similarity is seamlessly integrated into ontology alignment algorithms.

We study the impact of our approach in the context of three ontology alignment algorithms: **Falcon-AO** [40], **Logmap** [47] and **Optima** [20]. Using 2 different testbeds – 300-level ontology pairs from the benchmark track and the entire conference track of OAEI 2012 [84], and a novel biomedical testbed of 35 large BioPortal ontology pairs, We demonstrate significant positive impact on the precision of the alignment, with improvement in the recall for some of the algorithms as well at the expense of computation time.

4.1 OWL 2 to RDF Graph Transformation

In OWL, the restricted class is the only subclass of an anonymous class. The latter is a class that is devoid of an informative IRI and is the class of all individuals that satisfy the restriction. On declaring a restriction, an anonymous class associated with that restriction is created implicitly.

Analogous to a restricted class, a Boolean class is a Boolean combination of two or more classes in the ontology. Boolean combination operators include the union, intersection and complement. The combination is implicitly an anonymous class that is associated with a RDF list of all the classes involved using a property whose name is the Boolean operator. The Boolean class is named and is associated with the anonymous class using *owl:equivalentClass*. Next, I briefly review a W3C specification for transforming OWL 2 axioms into RDF graphs without loss in meaning, followed by an overview of the three alignment algorithms that We use and their extant modeling of complex concepts.

The recent OWL 2 to RDF graph mapping [67] provides a transformation, T , that can be used to translate any OWL 2 ontology axiom, O , into an RDF graph, $G = T(O)$, without loss of generality. A reverse mapping, T^{-1} , is also presented, which can be used to transform an RDF graph, G , satisfying certain restrictions into an OWL 2 DL ontology, OG . These transformations do not incur any change in the formal meaning of the ontology [67]. Formally, for any OWL 2 DL ontology O , let $G = T(O)$ be the RDF graph obtained by transforming O as specified, and let $OG = T^{-1}(G)$ be the OWL 2 DL ontology obtained by applying the reverse transformation to G ; then O and OG are logically equivalent because they have exactly the same set of models. Blank nodes in RDF are used to represent a resource that is not provided with an IRI. Therefore, the anonymous classes of OWL are mapped to blank nodes in RDF. Because existing ontologies seldom contain annotations, We focus on the transformation of axioms that do not contain annotations in this study. In summary, the translated RDF graph, G , provides a graphical representation for an OWL 2 ontology without loss of meaning.

4.2 Representative Alignment Algorithms

In order to select the representative alignment algorithms, We seek open-sourced implementations. Though several different ontology alignment algorithms exist, few source codes are freely accessible. Among these, We select three different alignment algorithms, Falcon-AO [40], LogMap [47] and Optima+ [20], as representatives. These established algorithms successfully participated in multiple editions of the OAEI competition [79, 83, 84]. **Logmap** and **Optima+** placed in the top five in OAEI 2012's conference track.

4.3 Modeling Complex Concepts Using Canonical Representation

Many alignment algorithms heavily rely on the lexical attributes of named concepts, with some exploiting the ontology structure as well. Complex concepts are typically composite and often implicitly involve anonymous classes in their descriptions. The absence of lexical attributes of the anonymous classes complicates both lexical and structural matching of these concepts. In order to include these concepts within the alignment process, algorithms need a general way of measuring similarity between them.

Alignment algorithms either adopt an axiomatic model of an ontology or an intermediate RDF graph-theoretic model. As I mentioned in Section 4.1, each axiom in OWL may be transformed to an equivalent RDF graph [67]. Consequently, the specific OWL constructs that constitute complex concepts may also be represented using subgraphs within the full graphical representation of the ontology.

Our insight is that for the purpose of alignment, the axiomatic structural specifications of the differing types of restrictions and types of Boolean combinations may be partially standardized into canonical forms, and transformed into subgraph representations in canonical form, which are useful for equivalence comparisons. This is significant because there exist 12 different types of property restrictions and 3 different Boolean combination operators in OWL making their comparisons challenging.

Property restrictions may be classified into value restrictions and cardinality restrictions, which are addressed next.

4.3.1 Canonical Form for Value Restrictions

A value restriction on a class restricts the values of its property's range. The *ObjectAllValuesFrom* or *ObjectSomeValuesFrom* restrictions defined for an object property limits its values to individuals of a class, while data properties are restricted to a data range by *DataAllValuesFrom* or *DataSomeValuesFrom*. The *ObjectHasValue* limits the value of the object property to an individual, a , and *DataHasValue* limits the data property to a literal, lt . For example, the *ObjectAllValuesFrom*(*has_output_value*, *Spectral_Count*) restriction defined in the parasite experiment ontology (PEO) for the class *Proteome_analysis* restricts the range of *has_output_value* to take individuals from *Spectral_Count* only. Let me denote an object property expression as OPE and a data property expression as DPE . We refer to a class expression using CE and a datarange using DR .

We observe that the different types of value restrictions admit structural specifications, which may be represented equivalently (though their semantics differ). Subsequently, We introduce the generalized value restriction complex concept and define it in a canonical form, $CE_V = RE_V(PE_V, R_V)$, where $PE_V \in \{OPE, DPE, (DPE_1, \dots, DPE_k)\}$, $k \geq 2$, is a property expression(s) and $R_V \in \{DR, CE, a, lt\}$ is the value that restricts the range. RE_V is one of the value restriction expressions in $\{ObjectSomeValuesFrom, ObjectAllValuesFrom, ObjectHasValue, DataSomeValuesFrom, DataAllValuesFrom, DataHasValue\}$.

Next, We introduce the general transformation, $\mathcal{T}(CE_V) = SG_V$, which is defined as transforming the canonical form value restriction, CE_V , into a subgraph for value restrictions, $SG_V = \langle V_V, E_V, \lambda_V \rangle$, in a canonical form. Here, the set of vertices, $V_V = \{T(PE_V), T(R_V), owl:restriction, _ : x\}$, and the set of directed edges, $E_V = \{\{ _ : x, T(PE_V) \}, \{ _ : x, T(R_V) \}, \{ _ : x, owl:restriction \}\}$, where $_ : x$ is the blank node for the restriction and $T(\cdot)$ is the transformation mentioned in Section 4.1. $\lambda_V : E_V \rightarrow L_V$ is the edge-labeling function defined below, where $L_V = \{owl:onProperty, owl:onProperties, owl:allValuesFrom, owl:someValuesFrom, owl:hasValue,$

$\text{rdf:type}\}$:

$$\lambda_V = \begin{cases} \{- : x, T(PE_V)\} & \rightarrow \text{owl:OnProperty} \\ \{- : x, R_V\} & \rightarrow \bar{T}(RE_V) \\ \{- : x, \text{owl:restriction}\} & \rightarrow \text{rdf:type} \end{cases}$$

Here, $\bar{T}(RE_V)$ maps the edge to either *owl:allValuesFrom*, *owl:someValuesFrom*, or *owl:hasValue*.

We show the derived subgraph for value restrictions, SG_V , in Fig. 4.2(a). Note that the subgraph produced by $T(\cdot)$ for a specific value restriction canonicalizes to SG_V generated by $\mathcal{T}_V(\cdot)$, as we illustrate in Fig. 4.2(b).

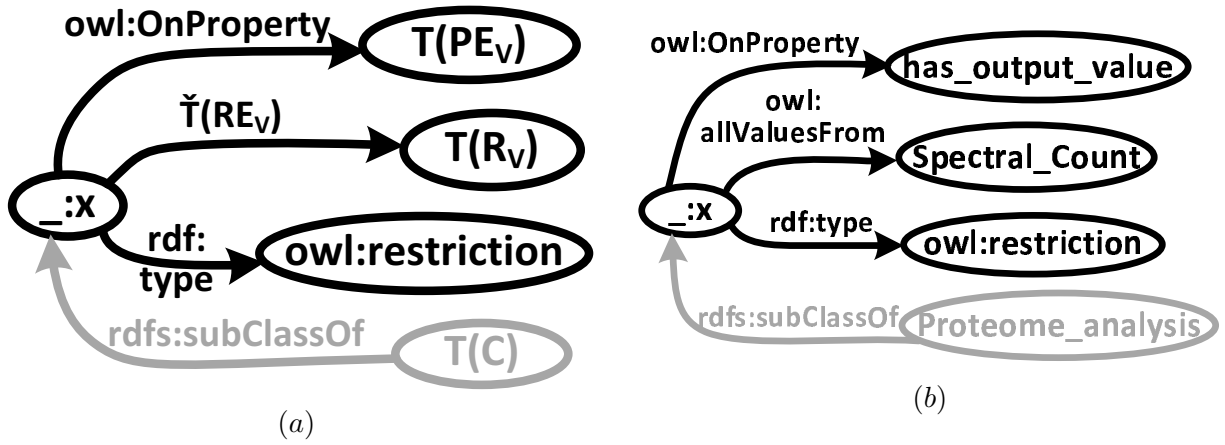


Figure 4.2: (a) The nodes and edges in bold constitute the canonical form RDF subgraph for value restrictions, while the grayed node is the restricted concept. (b) Canonicalized RDF subgraph for an extract from PEO. The specific value restriction, *owl:allValuesFrom*, on the *proteome_analysis* class restricts *has_output_value* property to take values from *Spectral_Count* only.

A reverse transformation function, $\mathcal{T}^{-1}(SG_V) = CSG_V$ is also defined which transforms a canonical form subgraph, SG_V , to a value restriction in the axiomatic structural specification, CSG_V . It applies $\mathcal{T}^{-1}(\cdot)$ as mentioned in Section 4.1 to each RDF triple in SG_V . Note that $\mathcal{T}^{-1}(\mathcal{T}(\cdot))$ produces an ontology that is logically equivalent to its input. The following theorem shows that the subgraph as derived above may be used to represent any value restriction complex concept without loss in meaning.

Theorem 1 (Canonical value restriction subgraph) *For any OWL 2 DL value restriction, CE_V , if SG_V is the canonical form subgraph obtained by transforming CE_V using $\mathcal{T}$, and CSG_V is the*

OWL 2 DL restriction in canonical form obtained by applying the reverse transformation, T^{-1} , to SG_V , then CE_V and CSG_V are logically equivalent for any value restriction.

Proof For each type of value restriction in CE_V , the RDF subgraph obtained by applying $T(\cdot)$ to the specific value restriction canonicalizes to SG_V . Furthermore, CSG_V is the canonical form of the OWL 2 ontology obtained by applying the reverse mapping, $T^{-1}(\cdot)$, to the RDF subgraph. The theorem holds because $T^{-1}(T(O))$ is equivalent in meaning to OWL 2 ontology, O , for any O including any value restriction, as We mention in Section 4.1.

4.3.2 Canonical Form for Cardinality Restrictions

Cardinality restrictions declare the minimum, maximum and exact cardinality of a property range. A cardinality restriction expression is defined using a property and a cardinality value, n .

For example, the cardinality restriction, *ObjectMaxCardinality*(2, *has_output_value*, *StandardDeviation*), defined in PEO for *Proteome_analysis* restricts the cardinality of property, *has_output_value*, to a maximum of 2 individuals of the *StandardDeviation* class.

Analogously to value restrictions, different types of cardinality restrictions admit structural specifications, which may be represented equivalently. Using the cardinality value n and property expression, $PE_C \in \{OPE, DPE\}$, We introduce the generalized cardinality restriction complex concept in a canonical form, $CE_C = RE_C(n, PE_C, R_C)$, where RE_C is one of the cardinality restriction expressions in $\{ObjectMinCardinality, ObjectMaxCardinality, ObjectExactCardinality, DataMinCardinality, DataMaxCardinality, DataExactCardinality\}$. $R_C \in \{CE, DR\}$ is the specific class or datarange whose cardinality is restricted. R_C could be empty unless the cardinality is qualified.

We expand the general transformation function, $T(CE_C) = SG_C$, to translate a canonical form cardinality restriction, CE_C , into a RDF subgraph in a canonical form, $SG_C = \langle V_C, E_C, \lambda_C \rangle$. Here, vertices $V_C = \{n \sim \text{xsd:nonNegativeInteger}, T(PE_C), T(R_C), \text{owl:restriction}, _ : x\}$ and directed edges $E_C = \{\{ _ : x, n \}, \{ _ : x, T(PE_C) \}, \{ _ : x, \text{owl:restriction} \}, \{ _ : x, T(CE) \} \}$. Function, $\lambda_C: E_C \rightarrow L_C$ gives the edge labels, where $L_C = \{\text{owl:onProperty}, \text{owl:onClass},$

`owl:onDataRange`, `owl:minCardinality`, `owl:maxCardinality`, `owl:cardinality`, `rdf:type`}, and is defined as:

$$\lambda_C = \begin{cases} \{ _ : x, T(PE_C) \} & \rightarrow \text{owl:OnProperty} \\ \{ _ : x, T(CE) \} & \rightarrow \text{owl:OnClass} \\ \{ _ : x, T(DR) \} & \rightarrow \text{owl:onDataRange} \\ \{ _ : x, n^{\sim\text{xsd:nonNegativeInteger}} \} & \rightarrow \bar{T}(RE_C) \\ \{ _ : x, \text{owl:restriction} \} & \rightarrow \text{rdf:type} \end{cases}$$

Here, $\bar{T}(RE_C)$ maps RE_C to one of three corresponding restriction types: `owl:minCardinality`, `owl:maxCardinality`, `owl:cardinality`.

Subsequently, the reverse transformation function, $\mathcal{T}^{-1}(SG_C) = CE_C$ may also be defined by applying $\mathcal{T}^{-1}(\cdot)$ to each RDF triple in SG_C . The following theorem establishes that the transformation, $\mathcal{T}(\cdot)$, produces a general RDF subgraph, SG_C , that is a canonical form for cardinality restrictions with no loss in meaning.

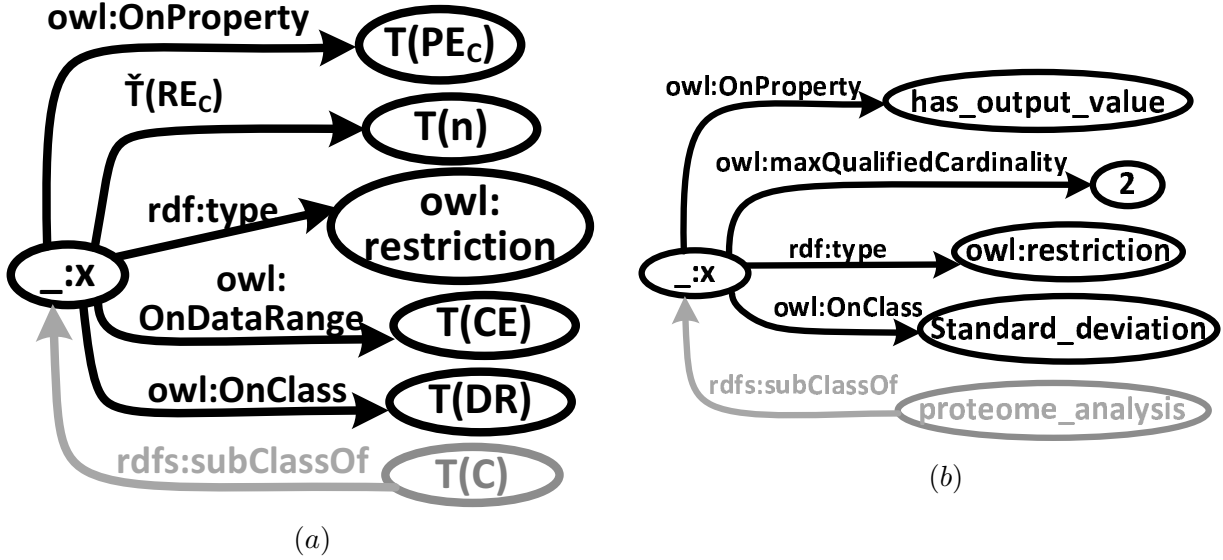


Figure 4.3: (a) Canonical RDF graph representation of cardinality restrictions. The nodes and edges in bold constitute the canonical form subgraph for a cardinality restriction. The restricted concept is grayed. (b) An example cardinality restriction obtained from PEO in the canonical form. A cardinality restriction on the *proteome_analysis* class restricts the *has_output_value* property to have a cardinality of 2 on the *Standard_deviation* class.

Theorem 2 (Canonical cardinality restriction subgraph) For any OWL 2 DL cardinality restriction CE_C , let $SG_C = \mathcal{T}(CE_C)$ be the canonical form subgraph obtained by transforming CE_C as

specified previously, and let CSG_C be the OWL 2 DL restriction obtained by applying the reverse transformation $\mathcal{T}^{-1}$ to SG_C ; then, CE_C and CSG_C are logically equivalent for any cardinality restriction.

Proof For each type of cardinality restriction in CE_C , the RDF subgraph obtained by applying $T(\cdot)$ to the specific cardinality restriction is identical to SG_C . Furthermore, the OWL 2 ontology, CSG_C , is the canonical form of the OWL 2 ontology obtained by applying the reverse mapping, $T^{-1}(\cdot)$, to the RDF subgraph. The theorem holds because $T^{-1}(T(O))$ is equivalent in meaning to OWL 2 ontology, O , for any O including any cardinality restriction.

We illustrate cardinality restrictions represented using the canonical RDF graph, and the previous example cardinality restriction canonicalized, in Fig. 4.3. The canonical form subgraph in Fig. 4.3(a) is not parsimonious. During canonicalization, either the edge $\{- : x, T(CE)\}$ or $\{- : x, T(DR)\}$ is retained while the other is absent based on whether the property is an object or data property, respectively.

4.3.3 Canonical Form for Boolean Combinations

Complex concepts that are Boolean combinations are primarily defined using one of the set operators: union, intersection or complement. Union and intersection are applied to a sequence of classes or datatypes while complement is applied on a single class or datatype. Structural specifications of these complex concepts may be represented identically in a canonical form as, $CE_B = BE(B)$. Here, the operand, $B \in \{(CE_1, \dots, CE_k), CE, (DR_1, \dots, DR_k), DR\}$, and the Boolean operator expression is denoted by, $BE \in \{\text{ObjectUnionOf}, \text{ObjectIntersectionOf}, \text{ObjectComplementOf}, \text{DataUnionOf}, \text{DataIntersectionOf}, \text{DataComplementOf}\}$. An example Boolean combination from PEO defines the range of the object property *has_output_value* as a union of *data_collection* and *parameter*. Its structural specification in the canonical form is, *ObjectUnionOf(data_collection, parameter)*.

Analogously to our previous approach, We expand the generalized transformation function, $\mathcal{T}$, to Boolean combinations, which when applied to CE_B yields a RDF subgraph SG_B

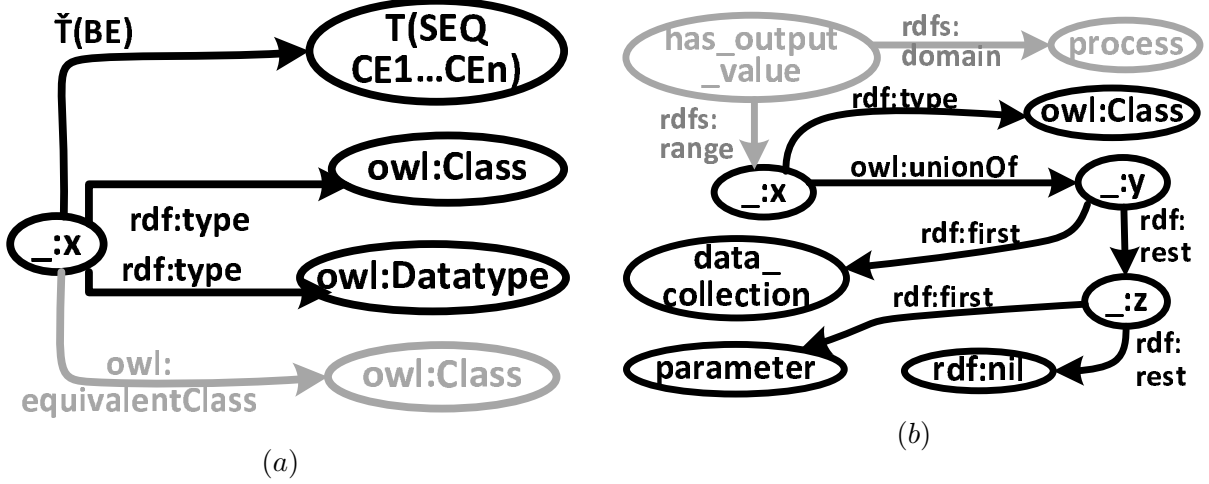


Figure 4.4: (a) The nodes and edges in bold constitute the canonical form subgraph for a Boolean combination. (b) RDF graph representation of an example Boolean combination from PEO in its canonical form. Property *has_output_value* has a Boolean combination as its range, which is a *owl:unionOf* classes, *data_collection* and *parameter*.

that gives the canonical form, $\mathcal{T}(CE_B) = SG_B$. Graph, $SG_B = (V_B, E_B, \lambda_B)$ where the set of vertices is, $V_B = \{T(B), \text{owl:class}, \text{rdfs:Datatype}, _ : x\}$. $E_B = \{ \{ _ : x, \text{owl:class} \}, \{ _ : x, \text{rdfs:Datatype} \}, \{ _ : x, T(B) \} \}$ is a set of edges, and $\lambda_B : E_B \rightarrow L_B$, where $L_B = \{ \text{owl:unionOf}, \text{owl:intersectionOf}, \text{owl:complementOf}, \text{rdf:type} \}$. λ_B labels the edge $\{ _ : x, T(B) \}$ with $\bar{T}(BE)$, which maps to *owl:unionOf*, *owl:intersectionOf* or *owl:complementOf*. Edges $\{ _ : x, \text{owl:class} \}$ and $\{ _ : x, \text{rdfs:Datatype} \}$ are both labeled with *rdf:type*. The corresponding canonical form subgraph for Boolean combinations is shown in Fig. 4.4(a). We also define a reverse transformation, $\mathcal{T}^{-1}(SG_B) = CE_B$, which transforms any canonical form Boolean combination subgraph back to a structural specification by applying the transformation, $T^{-1}(\cdot)$, to each RDF triple in the graph. The following theorem holds for complex concepts involving Boolean combinations as well.

Theorem 3 (Canonical Boolean combination subgraph) *For any OWL 2 DL Boolean combination, CE_B , let $SG_B = \bar{T}(CE_B)$ be the canonical form subgraph and let CSG_B be the OWL 2 DL*

Boolean combination obtained by applying the reverse transformation, T^{-1} , to SG_B ; then, CE_B and CSG_B are logically equivalent for any Boolean combination.

Proof For each type of Boolean combination in CE_B , the RDF subgraph obtained by applying $T(\cdot)$ to the specific Boolean combination canonicalizes to SG_B . Additionally, CSG_B gives the canonical form of the OWL 2 ontology obtained by applying the reverse mapping, $T^{-1}(\cdot)$, to the RDF subgraph. Because $T^{-1}(T(O))$ is equivalent in meaning to OWL 2 ontology, O , for any O including any Boolean combination, the theorem holds.

Note that the canonical form subgraph in Fig. 4.4(a) is not parsimonious. During canonicalization, either the edge $\{ _ : x, owl:class \}$ or $\{ _ : x, rdfs:Datatype \}$ is retained based on whether the specific concept is a Boolean combination of classes or data types.

Well developed ontologies such as PEO have several Boolean combinations, as illustrated in Fig. 4.4(b).

4.4 Computing Similarity between Canonical Representation

The first step toward matching complex concepts present in an ontology pair is to identify them in each ontology, followed by canonicalizing them to the appropriate axiomatic or graph forms based on the concept type. We adopt a cautious approach in comparing the complex concepts for alignment. Specifically, due to the differing semantics of their interpretations, We do not match a value or cardinality restriction with a Boolean combination. This leads to a limitation of our approach: some concepts may admit descriptions using both restrictions and Boolean combinations, which may not be matched. Furthermore, We draw a strict distinction between cardinality and value restrictions by noting that their semantics are often complementary. Therefore, We do not seek a match between these different types of restrictions.

Let $\mathcal{CE}_{CC}$ denote the set of all types of complex concepts, and Sim denote the similarity function between two complex concepts, $Sim : \mathcal{CE}_{CC} \times \mathcal{CE}_{CC} \rightarrow \mathbb{R}$. Then,

$$Sim(CE_{CC}^1, CE_{CC}^2) = \begin{cases} Sim_R(CE_{CC}^1, CE_{CC}^2) & CE_{CC}^1, CE_{CC}^2 \in \{CE_V, CE_C\} \\ Sim_B(CE_{CC}^1, CE_{CC}^2) & CE_{CC}^1, CE_{CC}^2 \in \{CE_B\} \\ -1 & \text{otherwise} \end{cases} \quad (4.1)$$

where Sim_R and Sim_B are the similarity functions that operate on restriction and Boolean complex concepts. Notice that I return a value of -1 instead of 0, which signifies that a match between the two concepts has not been attempted.

Similarity between property restrictions in their canonical form subgraphs is an aggregation of the similarities between their corresponding transformed property expressions, $T(PE^1)$ and $T(PE^2)$, corresponding transformed class expressions or dataranges, $T(R^1)$ and $T(R^2)$, and literals, n^1 and n^2 . If one of the canonical representations has a nonempty set of literals while the other is empty indicating that the latter is a value restriction while the former is a restriction on cardinality, no similarity is computed.

$$Sim_R(CE_{V|C}^1, CE_{V|C}^2) = \begin{cases} w \cdot H_{\text{mean}} \left(Sim'(T(PE_{V|C}^1), T(PE_{V|C}^2)), \right. & n^1, n^2 \neq \{\} \\ \left. Sim'(T(R_{V|C}^1), T(R_{V|C}^2)), Sim'(n^1, n^2) \right) & n^1, n^2 = \{\} \\ -1 & \text{otherwise} \end{cases} \quad (4.2)$$

Here, Sim' measures the similarity between the expressions. If the expressions are complex concepts themselves, this becomes a *recursive* call to the Sim function defined in Eq. 4.1, otherwise the lexical similarity is evaluated.

We utilize the weight, w , to emphasize the similarity in the types of value and cardinality restrictions. For example, the weight w between the same cardinality types could be 1, between a *owl:minCardinality* and a *owl:maxCardinality*, 0, and between the remaining cardinality type combinations, 0.75. Instead of taking a simple average, a modified harmonic mean, H_{mean} , is utilized. This mitigates the influence of extreme outliers in the similarity values and tends toward the lower values Sim' in the list.

In the context of Boolean combinations, We match complex concepts representing the same Boolean operators. Because property expressions and literals are not present in Boolean canonical subgraphs, Eq. 4.2 reduces to:

$$Sim_B(CE_B^1, CE_B^2) = w \cdot Sim'(\bar{T}(BE^1), \bar{T}(BE^2)) \quad (4.3)$$

Here, w becomes 0 if $\bar{T}(BE^1)$ and $\bar{T}(BE^2)$ are not the same indicating that the two canonicalized subgraphs contain different operators.

If the ontologies are modeled axiomatically with the complex concepts canonicalizing to structural specifications, Eqs. 4.2 and 4.3 measure the similarity between the participating expressions directly instead of their graph transformations.

4.5 Integrating Complex Concepts

We integrate our approach for computing the similarity between the complex concepts described in the previous section within the alignment algorithms outlined in Section 4.1. Because many of the alignment algorithms precluded complex concepts while loading the ontologies, our first step is to update the ontology models of the algorithms to include complex concepts. We aim to integrate the similarity of complex concepts within the algorithms as seamlessly as possible. This allows the different alignment algorithms to treat the complex concepts analogously to the named concepts, thereby requiring minimal changes in the algorithms themselves.

Canonicalized subgraphs of complex concepts are integrated into the RDF bipartite graphs of the two ontologies, and utilized by **Falcon-AO**'s structural matcher, GMO. While initializing the class similarity matrix, M , in GMO, We extend it to include anonymous classes also and provide our Sim function to evaluate similarity between anonymous classes, while leaving the similarity between the named entities undisturbed. Now, GMO iteratively evolves the similarity matrix M as before, which also includes complex concepts. This allows the complex concepts to influence named entity similarity where appropriate.

As mentioned earlier, **Logmap** limits its focus to named entities while building the Horn propositional representation and discovering candidate correspondences. We update its Horn knowledge

base to include anonymous classes. Similarity between the anonymous classes in their canonicalized axiomatic forms is computed using the function in Eq. (4.1), and included while generating candidate correspondences. This enhancement aids in the discovery of candidate correspondences in the neighborhood of the anonymous classes. Furthermore, it helps to prune additional inconsistent correspondences by considering the complex concepts within its knowledge base.

By default **Optima+** precludes complex concepts. I identify the complex concepts and include their canonicalized subgraphs in the ontology models. Analogous to **Falcon-AO**, We extend the similarity matrix to include complex concepts with the similarity scores of the blank nodes provided by *Sim*. **Optima+** utilizes various lexical similarity measures for the named entities. Consequently, it now additionally utilizes the similarities between the blank nodes for evaluating the quality of an alignment sample. One of the heuristics used by it for generating alignment samples is to create correspondences between the neighboring entities of two particular entities, if they are matched. Consequently, We expect the explicit modeling of complex concepts to generate different samples than in the default.

4.6 Experiments

We analyze the improvements in precision and recall along with the associated trade off in the runtime by modeling complex concepts in various alignment algorithms. For evaluation, We use a comprehensive testbed of several ontology pairs spanning multiple domains. One of the testbed comprises of 25 pairs of ontologies from the 2012 edition of OAEI. We use 4 ontology pairs from its Benchmark track and 21 ontology pairs from its Conference track. The selection of these tracks is based on the fact that they include real-world ontologies for which the reference alignments are also provided by OAEI. These ontologies were either acquired from the Web or created independently using real-world resources. This includes all ontology pairs in the 300 range of the Benchmark track, which relate to *bibliography*, and expressive ontologies in the *conference* domain all of which structure knowledge about conference organization. We list the participating ontologies in Table A.1. We created another novel testbed for evaluation using biomedical ontologies

from NCBO. This testbed contains 35 ontology pairs organizing knowledge in various biomedical domains. The ontologies were selected based on having 10% percent or more of complex concepts and a good amount of reference correspondences available in NCBO (10% or more of each ontology's concepts are present in the reference). The biomedical testbed is available for use at <http://tinyurl.com/aulcezm>.

On modeling complex concepts, there is no change in the overall precision and recall for **Falcon-AO** across all the pairs in the *bibliography* and *conference* domains (precision=62%, recall=60%). For **Optima+**, We obtained an overall 1% improvement in the precision increasing it to 54% and 1% improvement in recall thereby making it 70%. **Logmap**'s overall precision improved by 1% to 59% but modeling complex concepts did not affect its recall of 80%.

Increase in **runtime** caused by modeling complex concepts is minimal for each algorithm for these tracks. This is due to the scarcity of compatible complex concepts in the involved ontologies. **Logmap** and **Falcon-AO** consumed 1 and 8 seconds more, respectively, than the default across all 25 pairs in the bibliography and conference domains. **Optima+**, which is slowest among the three, consumed 52 seconds in addition to the default.

As We will show later in Section 7.1, modeling complex concept has benefited all three algorithms when aligning ontologies from the novel biomedical testbed. The performance of (a) Falcon-AO, (b) LogMap, and (c) Optima+ with complex concept modeling and in their default mode on the biomedical testbed is shown in Fig. 7.1. The overall improvement in F-measure is significant on this testbed for all three algorithms. The overall improvement in F-measure for **Falcon-AO** to 31% (precision=48%, recall=23%) is significant (Student's paired t-test, $p < 0.05$). Complex concept modeling increased **Logmap**'s precision to 62% and recall to 35%, both of which increases are significant ($p < 0.05$). For **Optima+** the overall F-measure improved by 4%. This improvement in F-measure to 56% (precision=55%,recall=56%) on the biomedical testbed is significant ($p < 0.01$).

4.7 Discussion

We observed that different types of value restrictions could be modeled uniformly thereby allowing an axiomatic canonical form in OWL 2's structural specification and derived an equivalent RDF graph-based canonical form. Analogously, canonical forms were provided for different cardinality restrictions and the various Boolean combinations. This allowed me to improve ontology alignment by canonicalizing the complex concepts often present in an ontology and providing a simple way to measure the similarity between the anonymous classes.

Ideally, We seek to match composite complex concepts of different types such as one involving a value restriction and another containing a Boolean combination if they are semantically equivalent. Therefore, a single canonical representation that would identify the same concept despite being differently defined is preferred. This is challenging and requiring robust DL inferencing. In this chapter, We provide separate canonical representations for three types of complex concepts, which is a first step toward this goal. To the best of our knowledge, this study is the first in its explicit focus on modeling complex concepts for inclusion in the ontology alignment process.

CHAPTER 5

SPEEDING UP CONVERGENCE OF ITERATIVE ONTOLOGY ALIGNMENT

As mentioned earlier, several algorithms exist for automatically aligning ontologies using various techniques [12, 20, 24, 35, 45, 45–47, 51, 51, 66], with mixed levels of performance. Crucial challenges for these algorithms involve scaling to large ontologies and performing the alignment in a reasonable amount of time without compromising on the quality of the alignment. As a case in point, less than half the alignment algorithms that participated in the 2012 instance of the annual ontology alignment evaluation initiative (OAEI) competition [84] generated acceptable results for aligning moderately large ontologies.

Although ontology alignment is traditionally perceived as an offline and one-time task, the second challenge is gaining importance. In particular, as Hughes and Ashpole [42] note, continuously evolving ontologies and applications involving real-time ontology alignment such as semantic search and Web service composition stress the importance of computational complexity considerations. Recently, established competitions such as OAEI [83] began reporting the execution times of the participating alignment algorithms as well. As ontologies continue to become larger, efficiency and scalability become key properties of alignment algorithms.

As I mentioned earlier in Section 2.2.1, a large class of algorithms that performs automated alignment is *iterative* in nature [12, 20, 24, 35, 46, 51, 61, 93]. These algorithms repeatedly improve on the previous preliminary solution by optimizing a measure of the solution quality. Often, this is carried out as a guided search through the alignment space using techniques such as gradient descent or expectation-maximization. These algorithms run until convergence, which means the solution cannot be improved further because it is a, possibly local, optima. However, in practice, the runs are often terminated after a number of iterations determined in an ad hoc manner. Through

repeated improvements, the computed alignment is usually of high quality but these approaches also consume more time in general than their non-iterative counterparts. For example, algorithms performing among the top three in OAEI 2012 in terms of alignment quality such as YAM++ [66], which ranked first in the *conference* track, **Optima+**, ranked third in the *conference* track, and GOMMA [48], which ranked first in *anatomy* and *library* tracks, are iterative. On the other hand, YAM++ consumed an excessive amount of time in completing the *conference* track (greater than 5 hours) and GOMMA consumed comparatively more time as well.

Furthermore, iterative techniques tend to be anytime algorithms, which deliver an alignment even if the algorithm is interrupted before its convergence. While considerations of computational complexity has delivered ways of scaling the alignment algorithms to larger ontologies, such as through ontology partitioning [41, 77, 88] and the use of inverted indices [47], there is a general absence of effort to speed up the ontology alignment process. I think these considerations of space and time go hand in hand in the context of scalability.

In this chapter, I introduce a novel approach for significantly speeding up the convergence of iterative ontology alignment techniques. Objective functions that measure the quality of the solution are typically multidimensional. Instead of the traditional approach of modifying the values of a large number of variables in each iteration, I decompose the problem into optimization sub-problems, in which the objective function is optimized with respect to a single or a small subset, also called a *block*, of variables while holding the other variables fixed. This approach of *block-coordinate descent* (BCD) is theoretically shown to converge faster under considerably relaxed conditions on the objective function such as pseudoconvexity – and even the lack of it in certain cases – or the existence of optima in each variable (coordinate) block [91]. While it forms a standard candidate tool for multidimensional optimization in statistics, and has been applied in contexts such as image reconstruction [26, 70] and channel capacity computation [3, 11], this chapter presents its first application towards ontology alignment.

I evaluate this approach by integrating it into multiple ontology alignment algorithms. Although several iterative alignment techniques have been proposed, I selected **Falcon-AO** [46],

MapPSO [12], **OLA** [24] and **Optima** [20, 90] as representative algorithms. These algorithms have all participated in OAEI competitions in the past, and some of them have placed in the top tier. Consequently, these algorithms in their default forms exhibit favorable alignment performance. Additionally, their implementations and source codes are freely accessible.

Using a comprehensive testbed of several ontology pairs – some of which are very large – spanning multiple domains, I show a significant reduction in the execution times of the alignment processes thereby indicating faster convergences. Corresponding alignment quality continues to remain the same as before or is improved by a small amount in some cases. This enables the application of these algorithms toward aligning more ontology pairs in a given amount of time, or to more subsets in large ontology partitions. Also, it allows these techniques to run until convergence in contrast to a predefined ad hoc number of iterations, which possibly leads to the similar or improved alignments.

Intuitively, the coordinate blocks in my application of BCD involve alignment variables between entities at specific heights in the ontology graph. Blocks with entities at the lowest heights are considered first followed by those of increasing height. However, BCD does not constrain how the alignment variables are divided into blocks except for the rule that each block be chosen at least once in a cycle through all blocks. Furthermore, I may order the blocks for consideration in any manner within a cycle.

Consequently, I empirically study the impact of different ordering and partitioning schemes on the improvement that BCD brings to the alignment. In addition to the default ordering scheme based on increasing height of blocked entities, I consider reversing this ordering, and a third approach in which I sample the blocks based on a probability distribution that represents the estimated likelihood of finding a large alignment in a block. In the context of partitioning, I additionally consider grouping alignment variables such that the entities are divided in a breadth-first search based partition. While my default approach partitions one of the ontologies in a pair, I also consider the impact of partitioning both. My experiments show that MapPSO’s run time and align-

ment performance continues to remain significantly lower compared to that of others. Therefore, I exclude **MapPSO** and focus on the other algorithms in subsequent empirical analyzes.

Surprisingly, the algorithms differ in which ordering and partitioning scheme optimizes their alignment performance. In order to comprehensively evaluate the efficiency of the BCD-enhanced and optimized algorithms, I construct a novel biomedical ontology alignment testbed. In addition to being an important application domain, aligning biomedical ontologies has its own unique challenges. I selected biomedical ontologies published in NCBO for my testbed, which also provides a crowd-sourced but incomplete reference alignment. Thirty-two different biomedical ontologies form the 50 pairs in my testbed with about half of these having 3,000+ named classes. Details on this biomedical testbed evaluation is presented in Section 7.2.

The rest of this chapter is organized as follows. In the next section, I briefly explain the representative iterative approaches selected for this study and their selection criteria. In the following section, I briefly review the technical approach of BCD. I show how BCD may be integrated into iterative ontology alignment algorithms in Section 5.3. In Section 5.4, I empirically evaluate the performances of the BCD enhanced algorithms using a comprehensive data set. Then, in Section 5.5, I explore other ways of ordering the blocks and partitioning the alignment variables in the context of the representative algorithms. Note, in Section 7.2 of Chapter 7, I detail a new biomedical ontology benchmark and report the performances of the BCD enhanced and optimized iterative techniques on this benchmark. New alignments discovered in these experiments are reported to NCBO's BioPortal for public use and curation. Finally, in Section 5.6, I discuss the impact of BCD toward iterative ontology alignment algorithms and its limitations.

5.1 Representative Alignment Algorithms

Though several iterative approaches exist, I chose *four* ontology alignment algorithms – **Falcon-AO**, **MapPSO**, **OLA** and **Optima** as representatives. Previously, in Section 2.3 of Chapter 2 I briefly reviewed these algorithms and their iterative approach. The selection of these algorithms is

based on their accessibility and competitive performance on previous OAEI competitions, and is meant to be representative of iteration-based alignment algorithms.¹

Altogether, the four alignment algorithms that I chose represent a broad variety of iterative update and search techniques, realized in different ways. This facilitates a broad evaluation of the usefulness of BCD. Over the years, algorithms such as **Falcon-AO**, **OLA** and **Optima** have performed satisfactorily in the annual OAEI competition, with **Falcon-AO** and **Optima** demonstrating strong performances with respect to the comparative quality of the generated alignment. For example, **Falcon-AO** often placed in the top 3 systems when it participated in OAEI competitions between 2005 and 2010, and its performance continues to remain a benchmark for other algorithms. **Optima** enhanced with BCD (called **Optima+**) placed second in the *conference* track (F2-measure and recall) in the 2012 edition of the OAEI competition [90]. Consequently, these representative algorithms exhibit strong alignment performances. On other hand, **MapPSO**'s performance is comparatively poor but it's particle-swarm based iterative approach motivates its selection in my representative set.

5.2 Block-Coordinate Descent

Large-scale multidimensional optimization problems maximize or minimize a real-valued continuously differentiable objective function, Q , of N real variables. Block-coordinate descent (BCD) [91] is an established iterative technique to gain faster convergence in the context of such large-scale N -dimensional optimization problems. In this technique, within each iteration, a set of the variables referred to as coordinates are chosen and the objective function, Q , is optimized with respect to one of the coordinate blocks while the other coordinates are held *fixed*.

Let S denote a block of coordinates, which is a non-empty subset of $\{1, 2, \dots, N\}$. I may define a set of such blocks as, $B = \{S_0, S_1, \dots, S_C\}$, which is a set of subsets each representing a coordinate block with the constraint that, $S_0 \cup S_1 \cup \dots \cup S_C = \{1, 2, \dots, N\}$. Note that B could be a

¹I sought to include YAM++ as well in my evaluation, which was the top performer in the *conference* track of OAEI 2012. However, its source code is not freely available and I could not access it.

partition of the coordinates, although this is not required and the blocks may intersect. I also define the complement of a coordinate block, S_c , where $c \in \{0, 1, \dots, C\}$, as, $\tilde{S}_c = B - S_c$. To illustrate, let the domain of a real-valued, continuously differentiable, multidimensional function, Q , with $N = 10$ be, $M = \{m_1, m_2, m_3, \dots, m_{10}\}$, where each element is a variable. I may partition this set of coordinates into two blocks, $C = 2$, so that, $B = \{S_0, S_1\}$. Let $S_0 = \{m_2, m_5, m_8\}$, and therefore, $S_1 = \{m_1, m_3, m_4, m_6, m_7, m_9, m_{10}\}$. Finally, $\tilde{S}_0$ denotes the block, S_1 .

BCD converges to a fixed point such as a local or a global optima of the objective function under relaxed conditions such as pseudoconvexity of the function and requires the function to have bounded level sets [91]. While pseudoconvex functions continue to have fixed points, they may have non-unique optima along different coordinate directions. In the absence of pseudoconvexity, BCD may oscillate without approaching any fixed point of the function. Nevertheless, BCD still converges if the function has unique optima in each of the coordinate blocks.

In order to converge using BCD, I must satisfy the following rule, which ensures that each coordinate is chosen sufficiently often [91].

Definition 1 (Cyclic rule) *There exists a constant, $T \leq N$, such that every block, S_c , is chosen at least once between the i^{th} iteration and the $(i + T - 1)^{th}$ iteration, for all i .*

In the context of the cyclic rule, BCD does not mandate a specific partitioning or an ordering scheme for the blocks. A simple way to meet this rule is by sequentially iterating through each block although I must continue iterating until each block converges to the fixed point.

Very recently, Saha and Tewari [76] show that the nonasymptotic convergence rate ² of BCD under the cyclic rule is faster than that of gradient descent (GC) if they both start from the same point, under some relaxed conditions. Starting from the same initial map, $M_{BCD}^0 = M_{GC}^0$, let M_{BCD}^i and M_{GC}^i denote the alignment at iteration i by BCD with cyclic rule and GC, respectively. Under the condition that the objective function, Q , which must be say, minimized, is isotonic and convex, $\forall i \geq 1, Q(M_{BCD}^i) \leq Q(M_{GC}^i)$. Furthermore, the nonasymptotic convergence rate of BCD

²This is the rate of convergence effective from the first iteration itself.

under the cyclic rule for objective functions with the previously mentioned properties is, $\mathcal{O}(1/i)$, where i is the iteration count.

5.3 Integrating BCD into Iterative Alignment

As I mentioned previously, ontology alignment may be approached as a principled multivariable optimization of an objective function, where the variables are the correspondences between the entities of the two ontologies. Different algorithms formulate the objective function differently. As the objective functions are often complex and difficult to differentiate, numerical iterative techniques are appropriate but these tend to progress slowly. In this context, I may speed up the convergence using BCD as I describe below.

5.3.1 General Approach

In Section 2.2.1, I identify two types of iterative ontology alignment algorithms. BCD may be integrated into both these types. In order to integrate BCD into the iterations, the match matrix, M , must be first suitably partitioned into blocks.

Though a matrix may be partitioned using one of several ways, I adopt an approach that is intuitive in the context of ontology alignment. An important heuristic, which has proved highly successful in both ontology and schema alignment, matches parent entities in two ontologies if their respective child entities were previously matched. This motivates grouping together those variables, $m_{a\alpha}$ in M , into a coordinate block such that the x_a participating in the correspondence belong to the same height leading to a partition of M . The height of an ontology node is the length of the shortest path from a leaf node.

Let the partition of M into coordinate blocks be $\{M_{S_0}, M_{S_1}, \dots, M_{S_C}\}$, where C is the height of the largest class hierarchy in ontology $\mathcal{O}_1$. Thus, each block is a submatrix with as many rows as the number of entities of $\mathcal{O}_1$ at a height and number of columns equal to the number of all entities in $\mathcal{O}_2$. For example, correspondences between the leaf entities of $\mathcal{O}_1$ and all entities of $\mathcal{O}_2$ will form the block, M_{S_0} . In the context of a bipartite graph model as utilized by **Falcon-AO** and OLA,

which represents properties in an ontology as vertices as well and are therefore part of M , these would be included in the coordinate blocks.

Iterative ontology alignment integrated with BCD optimizes with respect to a single block, M_{S_c} , at an iteration while keeping the remaining blocks fixed. In order to meet the cyclic rule, I choose a block, M_{S_c} , at iterations, $i = c + qC$ where $q \in \{0, 1, 2, \dots\}$. I point out that BCD is applicable to both types of iterative alignment techniques as outlined in Section 2.2.1. Alignment algorithms which update the similarity matrix iteratively as in Eq. 2.1 will now update only the current block of interest, M_{S_c} , and the remaining blocks are carried forward as is, as shown below:

$$\begin{aligned} M_{S_c}^i &= U_{S_c}(M^{i-1}) \\ M_{\tilde{S}}^i &= M_{\tilde{S}}^{i-1} \quad \forall \tilde{S} \in \tilde{S}_c \end{aligned} \tag{5.1}$$

where $\tilde{S}_c$ is the complement of S_c in B as defined previously. Note that $M_{S_c}^i$ combined with $M_{\tilde{S}}^i$ for all $\tilde{S} \in \tilde{S}_c$ forms M^i . Update function, U_{S_c} , modifies U in Eq. 2.1 to update just a block of the coordinates.

Analogously, iterative alignment which search for the candidate alignment that maximizes the objective function as in Eq. 2.2, will now choose a block, M_{S_c} , at each iteration. They will search over the *reduced search space* pertaining to the subset of all variables included in M_{S_c} , for the best candidate coordinate block. Formally,

$$\begin{aligned} M_{S_c,*}^i &= \arg \max_{M_{S_c} \in \mathcal{M}_{S_c}} Q_S(M_{S_c}, M_*^{i-1}) \\ M_{\tilde{S},*}^i &= M_{\tilde{S},*}^{i-1} \quad \forall \tilde{S} \in \tilde{S}_c \end{aligned} \tag{5.2}$$

where, $\mathcal{M}_{S_c}$ is the space of alignments limited to block, S_c . The original objective function, Q , is modified to Q_S such that it provides a measure of the quality of the block, M_{S_c} , given the previous best match matrix. Note that the previous iteration's matrix, M_*^{i-1} , contains the best block that was of interest in that iteration.

Algorithms in Fig. 5.1 revise the iterative update and search algorithms of Fig. 2.4 in order to integrate BCD. The primary differences in both involve creating a partition of the alignment matrix, M (line 4), and iterations that sequentially process each coordinate block only while keeping the

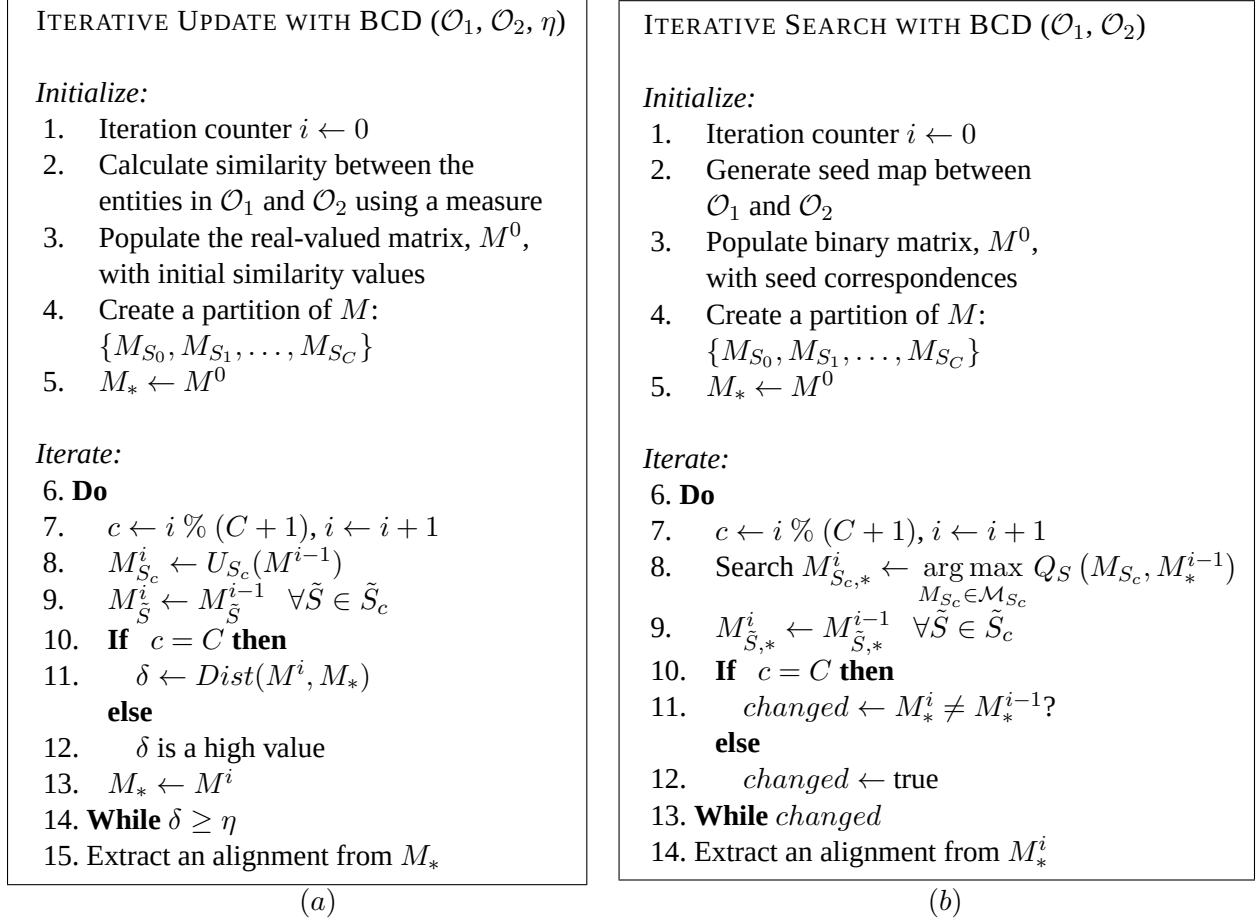


Figure 5.1: General iterative algorithms of Fig. 2.4 are modified to obtain, (a) iterative update enhanced with BCD, and (b) iterative search enhanced with BCD. The update or search steps in line numbers 8 and 9 are modified to update only the current block of interest.

others fixed (lines 7-9). On completing a cycle through all coordinate blocks, I evaluate whether the new alignment matrix differs from the one in the previous iteration, and continue the iterations if it does (lines 10-12).

Performing the update, U_{S_c} , or evaluating the objective function, Q_S , while focusing on a coordinate block may be performed in significantly reduced time as compared to performing these operations on the entire alignment matrix. While I may perform more iterations as I cycle through

the blocks, the use of partially updated matrices from the previous iteration in evaluating the next block facilitates faster convergence.

Given the general modifications brought about by BCD, I describe how these manifest in the four iterative alignment systems that form my representative set.

5.3.2 BCD Enhanced Falcon-AO

I enhance **Falcon-AO** by modifying **GMO** to utilize BCD as it iterates. As depicted in Fig. 5.2(a), I begin by partitioning the similarity matrix used by **GMO** into $C + 1$ blocks based on the height of the entities in $\mathcal{O}_1$ that are part of the correspondences, as mentioned previously. **GMO** is then modified so that at each iteration, a block of the similarity matrix is updated while the other blocks remain unchanged. If block, S_c , is updated at iteration i , then Eq. 2.3 becomes:

$$\begin{aligned} M_{S_c}^i &= G_{1,S_c} M^{i-1} G_2^T + G_{1,S_c}^T M^{i-1} G_2 \\ M_{\tilde{S}}^i &= M_{\tilde{S}}^{i-1} \quad \forall \tilde{S} \in \tilde{S}_c \end{aligned} \quad (5.3)$$

Here, G_{1,S_c} focuses on that portion of the adjacency matrix of $\mathcal{O}_1$ that corresponds to the outbound neighborhood of entities participating in correspondences of block S_c , while G_{1,S_c}^T focuses on the inbound neighborhood of entities in S_c . Adjacency matrix, G_2 , is utilized as before. The outcome of the matrix operations is a similarity matrix, with as many rows as the variables in S_c and columns corresponding to all the entities in $\mathcal{O}_2$. The complete similarity matrix is obtained at iteration, i , by carrying forward the remaining blocks unchanged, which is then utilized in the next iteration.

The general iterative update modified to perform BCD of Fig. 5.1(a) may be realized in **Falcon-AO** as in the algorithm of Fig. 5.2(a). A block of coordinates are updated using Eq. 5.3 while holding the remaining blocks fixed (lines 10 and 11). This yields a partially updated but complete alignment matrix in reduced time, which is utilized in the next iteration.

5.3.3 BCD Enhanced MapPSO

We may integrate BCD into **MapPSO** by ordering the particles in a swarm based on a measure of the quality of a coordinate block, S_c , in each particle in an iteration. Equation 2.4 is modified to

FALCON-AO/GMO-BCD ($\mathcal{O}_1, \mathcal{O}_2, \eta$)*Initialize:*

1. Iteration counter $i \leftarrow 0$
2. $G_1 \leftarrow \text{AdjacencyMatrix}(\mathcal{O}_1)$
3. $G_2 \leftarrow \text{AdjacencyMatrix}(\mathcal{O}_2)$
4. **For each** $m_{a\alpha} \in M^0$ **do**
5. $m_{a\alpha} \leftarrow 1$
6. Create a partition of M :
 $\{M_{S_0}, M_{S_1}, \dots, M_{S_C}\}$
7. $M_* \leftarrow M^0$

Iterate:

8. **Do**
9. $c \leftarrow i \% (C + 1), i \leftarrow i + 1$
10. $M_{S_c}^i \leftarrow G_{1,S_c} M^{i-1} G_2^T + G_{1,S_c}^T M^{i-1} G_2$
11. $M_{\tilde{S}}^i \leftarrow M_{\tilde{S}}^{i-1} \quad \forall \tilde{S} \in \tilde{S}_c$
12. **If** $c = C$ **then**
13. $\delta \leftarrow \text{CosineSim}(M^i, M_*)$
14. **else**
15. δ is a very high value
16. $M_* \leftarrow M^i$
17. **While** $\delta \geq \eta$
18. Extract an alignment from M_*

(a)

MAPPSO-BCD ($\mathcal{O}_1, \mathcal{O}_2, K, \eta$)*Initialize:*

1. Iteration counter $i \leftarrow 0$
2. Generate seed map between $\mathcal{O}_1$ and $\mathcal{O}_2$
3. Populate binary matrix, M^0 , with seed correspondences
4. Generate K particles using the seed M^0 : $P = \{M_1^0, M_2^0, \dots, M_K^0\}$
5. Create a partition of M :
 $\{M_{S_0}, M_{S_1}, \dots, M_{S_C}\}$
6. Search $M_*^0 \leftarrow \arg \max_{M_k^0 \in P} Q(M_k^0)$

Iterate:

7. **Do**
8. $c \leftarrow i \% (C + 1), i \leftarrow i + 1$
9. **For** $k \leftarrow 1, 2, \dots, K$ **do**
10. $M_{k,S_c}^i \leftarrow \text{UpdateBlock}(M_{k,S_c}^i, M_*^{i-1})$
11. $M_{k,\tilde{S}}^i \leftarrow M_{k,\tilde{S}}^{i-1} \quad \forall \tilde{S} \in \tilde{S}_c$
12. Search $M_*^i \leftarrow \arg \max_{M_k^i \in P} Q_S(M_k^i)$
13. **If** $c = C$ **then**
14. $changed \leftarrow |Q(M_*^i) - Q(M_*^{i-1})| \geq \eta?$
15. **else**
16. $changed \leftarrow \text{true}$
17. **While** $changed$
18. Extract an alignment from M_*^i

(b)

Figure 5.2: (a) Iterative update in **GMO** modified to perform BCD. In each iteration, just a block of variables are updated while holding the remaining fixed, and an updated alignment matrix is obtained which is utilized in the next iteration. (b) **MapPSO**'s particle swarm based iterative algorithm enhanced with BCD. Notice that the objective function, Q , is modified to Q_S , such that it is calculated for the coordinate block of interest. Furthermore, only the block in each particle is updated.

measure the quality of the correspondences in just the coordinate block S_c , in the k^{th} particle by

taking the average:

$$Q_S(M_k^i) = \frac{\sum_{a=1}^{|V_{1,c}|} \sum_{\alpha=1}^{|V_2|} m_{a\alpha} \times f(x_a, y_\alpha)}{|V_{1,c}| |V_2|} \quad (5.4)$$

where, $V_{1,c}$ denotes the set of entities of ontology, $\mathcal{O}_1$, of identical height participating in the correspondences included in block S_c . As before, I retain the best particle(s) based on this measure and improve on the alignment in a coordinate block, M_{k,S_c}^i , in the remaining particles using the best particle in the previous iteration. The remaining coordinates are held unchanged.

Iterative search of **MapPSO** modified using BCD is shown in the algorithm of Fig. 5.2(b). A coordinate block in each particle is updated while keeping the remaining blocks unchanged (lines 10 and 11), followed by searching for the best particle based on a measure of the alignment in the block (line 12). Both these steps may be performed in reduced time.

5.3.4 BCD Enhanced OLA

As explained earlier, **OLA** evolves its similarity matrix M by similarity exchange between pairs of neighboring entities. In each iteration, it performs an element-wise matrix update operation. **OLA** is enhanced with BCD by adopting Eq. 5.1. Specifically, the similarity values of the coordinates of the chosen block, S_c , will be updated using the similarity computations (Eq. 2.5). The remaining blocks, $M_{\tilde{S}_c}^i$, are kept unchanged.

$$m_{a\alpha}^i = \begin{cases} Sim(a, \alpha) & \text{if types of } a \text{ and } \alpha \text{ are the same} \\ 0 & \text{otherwise} \end{cases}, \forall m_{a\alpha}^i \in M_{S_c}^i \quad (5.5)$$

$$M_{\tilde{S}}^i = M_{\tilde{S}}^{i-1} \quad \forall \tilde{S} \in \tilde{S}_c$$

5.3.5 BCD Enhanced Optima

As I mentioned previously, **Optima** utilizes generalized expectation-maximization to iteratively improve the likelihood of candidate alignments. Jeffery and Alfred [25] discuss a BCD inspired

OLA-BCD ($\mathcal{O}_1, \mathcal{O}_2, \eta$)*Initialize:*

1. Iteration counter $i \leftarrow 0$
2. Populate the real-valued matrix M^0 with lexical similarity values
3. Create a partition of M : $\{M_{S_0}, M_{S_1}, \dots, M_{S_C}\}$
4. $M_* \leftarrow M^0$

Iterate:

5. **Do**
6. $c \leftarrow i \% (C + 1), i \leftarrow i + 1$
7. **for each** $m_{a\alpha} \in M_{S_c}^i$
8. **if** the types of a and α are the same **then**
9. $m_{a\alpha} \leftarrow \sum_{\mathcal{F} \in \mathcal{N}(a, \alpha)} w_{\mathcal{F}}^{a\alpha} \text{SetSim}(\mathcal{F}(a), \mathcal{F}(\alpha))$
10. **else**
11. $m_{a\alpha} \leftarrow 0$
12. $M_{\tilde{S}}^i = M_{\tilde{S}}^{i-1} \quad \forall \tilde{S} \in \tilde{S}_c$
13. **If** $c = C$ **then**
14. $\delta \leftarrow \text{Dist}(M^i, M_*)$
15. **else**
16. δ is a high value
17. $M_* \leftarrow M^i$
18. **While** $\delta \geq \eta$
19. Extract an alignment from M_*

(a)

OPTIMA+-BCD ($\mathcal{O}_1, \mathcal{O}_2$)*Initialize:*

1. Iteration counter $i \leftarrow 0$
2. **For all** $\alpha \in \{1, 2, \dots, |V_2|\}$ **do**
3. $\pi_{\alpha}^0 \leftarrow \frac{1}{|V_2|}$
4. Generate seed map between $\mathcal{O}_1$ and $\mathcal{O}_2$
5. Populate binary matrix, M_*^0 , with seed correspondences
6. Create a partition of M : $\{M_{S_0}, M_{S_1}, \dots, M_{S_C}\}$

Iterate:

7. **Do**
8. $c \leftarrow i \% (C + 1), i \leftarrow i + 1$
9. Search $M_{S_c, *}^i \leftarrow \arg \max_{M_{S_c} \in \mathcal{M}_{S_c}} Q_S(M_{S_c}^i | M_*^{i-1})$
10. $M_{\tilde{S}, *}^i \leftarrow M_{\tilde{S}, *}^{i-1} \quad \forall \tilde{S} \in \tilde{S}_c$
11. $\pi_{\alpha, c}^i \leftarrow \frac{1}{|V_{1, c}|} \sum_{a=1}^{|V_{1, c}|} Pr(y_{\alpha} | x_a, M_*^{i-1})$
12. **If** $c = C$ **then**
13. $changed \leftarrow M_*^i \neq M_*^{i-1}?$
14. **else**
15. $changed \leftarrow \text{true}$
16. **While** $changed$
17. Extract an alignment from M_*^i

(b)

Figure 5.3: (a) OLA's BCD-integrated iterative ontology alignment algorithm. Notice that I cycle through the blocks and only the coordinates belonging to the current block, $M_{S_c}^i$, are updated. (b) Expectation-maximization based iterative ontology alignment of **Optima** with BCD. The search is modified to explore a reduced search space, $\mathcal{M}_{S_c}$, as I cycle through the blocks.

expectation-maximization scheme and call it the space alternating generalized expectation-maximization (SAGE). Intuitively, SAGE maximizes the expected log likelihood of a block of coordinates thereby limiting the hidden space, instead of maximizing the likelihood of the complete alignment. The sequence of block updates in SAGE monotonically improves the objective likelihood. For a regular objective function, the monotonicity property ensures that the sequence

will not diverge, but it does not guarantee convergence. However, proper initialization lets SAGE converge locally.³ In each iteration, **Optima** enhanced using SAGE chooses a block of the match matrix, $M_{S_c}^i$, and its expected log likelihood is estimated. As in previous techniques, I choose the blocks in a sequential manner such that all the blocks are iterated in order.

Equation 2.6 is modified to estimate the expected log likelihood of the block of a candidate alignment as:

$$Q_S(M_{S_c}^i | M^{i-1}) = \sum_{a=1}^{|V_{1,c}|} \sum_{\alpha=1}^{|V_2|} Pr(y_\alpha | x_a, M^{i-1}) \times \log Pr(x_a | y_\alpha, M_{S_c}^i) \pi_{\alpha,c}^i \quad (5.6)$$

Recall that $V_{1,c}$ denotes the set of entities of ontology, $\mathcal{O}_1$, participating in the correspondences included in S_c . Notice that the prior probability, $\pi_{\alpha,c}^i$, is modified as well to utilize just $V_{1,c}$ in its calculations.

The generalized maximization step now involves finding a match matrix block, $M_{S_c,*}^i$, that improves on the previous one:

$$M_{S_c,*}^i = M_{S_c}^i \in \mathcal{M}_{S_c} : Q_S(M_{S_c,*}^i | M_*^{i-1}) \geq Q_S(M_{S_c,*}^{i-1} | M_*^{i-1}) \quad (5.7)$$

Here, $M_{S_c,*}^{i-1}$ is a part of M_*^{i-1} .

At iteration i , the best alignment matrix M_*^i , is formed by combining the block matrix $M_{S_c,*}^i$, which improves Q_S as defined in Eq. 5.7 with the remaining blocks from the previous iteration, $M_{\tilde{S},*}^{i-1}$, in the complement of S_c , unchanged.

The algorithm in Fig. 5.3(b) shows how **Optima** may be enhanced with BCD. I expect significant savings in time because of the search over a reduced space of alignments, $\mathcal{M}_{S_c}$, in each iteration. Additionally, both the objective function, Q_S , and the prior operate on a single coordinate block, in reduced time.

5.4 Empirical Analysis

While the use of BCD is expected to make the iterative approaches more efficient, I seek to empirically determine:

³Furthermore, the convergence rate may be improved by choosing the hidden space with less Fisher information [37].

1. The amount of speed up obtained for the various alignment algorithms by integrating BCD; and
2. Changes in the quality of the final alignment, if any, due to BCD. This may happen because the iterations converge to a different local optima.

I use a comprehensive testbed of several ontology pairs – some of which are very large – spanning multiple domains. I used ontology pairs from the OAEI competition in its most recent version, 2012, as the testbed for my evaluation [84]. Among the OAEI tracks, I focus on the test cases that involve real-world ontologies for which the reference (true) alignment was provided by OAEI. These ontologies were either acquired from the Web or created independently of each other and based on real-world resources. This includes all ontology pairs in the 300 range of the competition, which relate to *bibliography*, expressive ontologies in the *conference* track all of which structure knowledge related to conference organization, and the *anatomy* track, which consists of a pair of large ontologies from the life sciences, describing the anatomy of an adult mouse and human. I list the ontologies from OAEI participating in my evaluation in Table A.1 and provide an indication of their sizes.

I align ontology pairs using the four representative alignment algorithms, in their original forms and with BCD using the same seed alignment, M^0 , if applicable. The iterations were run until the algorithm converged and I measured the total execution time, final recall, precision and F-measure, and the number of iterations performed until convergence. Recall measures the fraction of correspondences in the reference alignment that were found by an algorithm while precision measures the fraction of all the found correspondences that were in the reference alignment thereby indicating the fraction of false positives. F-measure represents a harmonic mean of recall and precision.

I averaged results of 5 runs on every ontology pair using both the original and the BCD enhanced version of each algorithm. Because of the large number of total runs, I ran the tests on two different computing platforms while ensuring comparability. One of these is a Red Hat machine with Intel Xeon Core 2, processor speed of about 3 GHz with 8GB of memory (anatomy

ontology pair) and the other one is a Windows 7 machine with Intel Core i7, 1.6 GHz processor and 4GB of memory (benchmark and conference ontology pairs). While comparing the performance metrics for statistical significance, I tested the data for normality and used Student's paired t-test if it exhibits normality. Otherwise, I employed the Wilcoxon signed-rank test. I utilized the 1% level ($p \leq 0.01$) to deem significance.

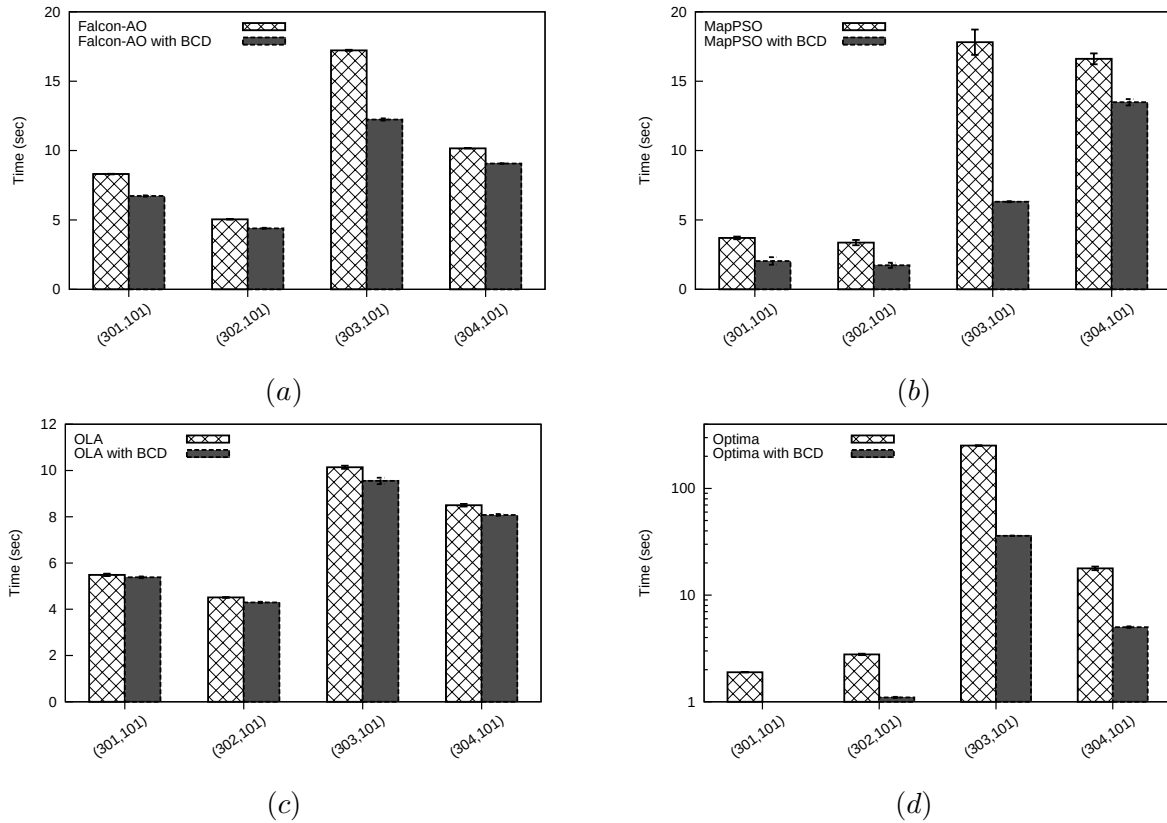


Figure 5.4: Average execution times of the four iterative algorithms, (a) **Falcon-AO**, (b) **MapPSO**, (c) **OLA**, and (d) **Optima**, in their original form and with BCD when aligning the 4 ontology pairs of the *bibliography* domain. Note that the time axis of (d) is in log scale. While the overall differences in run time are statistically significant, I point out an order of magnitude reduction for the (303,101) pair in (d). For a majority of the pairs, the algorithms converged in a lesser number of iterations as well. The total run time reductions across the pairs due to BCD are 8 seconds for **Falcon-AO**, 18 seconds for **MapPSO**, 2 seconds for **OLA**, and nearly 4 minutes for **Optima**.

For the 4 ontology pairs formed using the ontologies in the *bibliography* domain, I show the average execution time consumed by each algorithm until convergence in its default form and with BCD in Fig. 5.4. While the introduction of BCD significantly reduces the total execution time of

all four iterative techniques (Wilcoxon signed-rank test, $p < 0.01$), the reduction is nearly an order of magnitude for the ontology pair (303,101) in the context of **Optima**.

The final recall and precision of the resulting alignment remained unchanged for **Falcon-AO**. Enhancing **MapPSO**, which has a random component, with BCD improved the precision over all the pairs and averaged over the runs from 37% to 88%, while the recall remained steady at about 37%. OLA's precision and recall reduced slightly causing its F-measure to reduce by 1% for the ontology pair (302,101), while the the alignments for the other pairs remained the same. The integration of BCD in **Optima** caused the precision for ontology pair (302,101) to improve by 4% from 77% to 81% with no change in the recall of 35%. However, for ontology pair (303,101) it slightly lost precision from 77% to 74% along with a 1% reduction in recall from 68% to 67%. The precision and recall for all other pairs remain unchanged. Despite the small changes in recall and precision for the individual pairs, the Wilcoxon signed-rank test did not deem these changes to be significant for any of the four tools.

The ontologies in the *conference* domain vary widely in their size and structure. As shown in Fig. 5.5, the introduction of BCD to the four iterative techniques clearly improves their speed of convergence and the differences for each algorithm are significant (Student's paired t-test, $p \ll 0.01$). In particular, I observed an order of magnitude reduction in time for aligning relatively larger ontologies such as *iasted* and *edas*. For example, pairs (*conference*, *iasted*) on **MapPSO** and (*edas*, *iasted*) on **Optima** showed such reductions. Overall, I observed a total reduction of 50 seconds for **Falcon-AO**, 1 minute and 37 seconds for **MapPSO**, 11 seconds for **OLA**, and 29 minutes and 20 seconds for **Optima**.

Falcon-AO shows no change due to BCD in its alignment, holding its precision at 25% and recall at 66%. **Falcon-AO** with BCD saved a total of about 50 seconds in its run time across all pairs. **Optima** shows a 2% improvement in average precision from 60% to 62% but average recall reduced from 74% to 71%. Nevertheless, this causes a 1% improvement in average F-measure to 65%. **MapPSO** with BCD resulted in a significant improvement in final precision from 9% to 43%

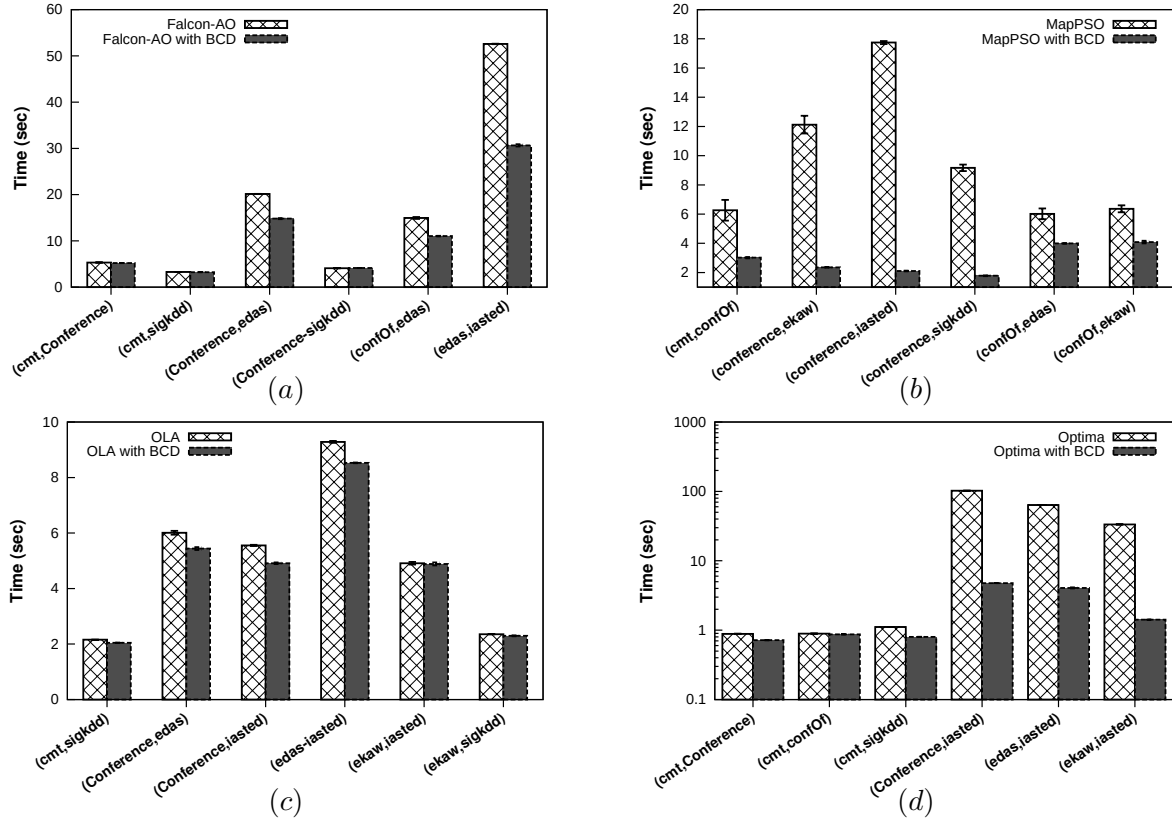


Figure 5.5: Average execution time consumed by, (a) **Falcon-AO**, (b) **MapPSO**, (c) **OLA**, and (d) **Optima** in their original form and with BCD, for 6 of the 21 ontology pairs from *conference* domain. I ran the algorithms for *all the pairs*, and selected ontology pairs which exhibited the three highest and the three lowest differences in average execution times for clarity. Note that the time axis of (d) is in log scale. Notice the improvements in execution time for the larger pairs. Specifically, about a 50% reduction in average execution time for the ontology pair *(edas, iasted)* by **Falcon-AO** and an order of magnitude reductions in average run time for ontology pairs *(conference, iasted)* in **MapPSO** and *(edas, iasted)* in **Optima**, were observed.

on average, although the difference in recall was not significant. The precision and recall for OLA remained unchanged.

The very large *anatomy* ontologies for mouse and human were not successfully aligned by **MapPSO**, **OLA** and **Optima** despite the use of BCD. However, BCD drastically reduced **Falcon-AO**'s average execution time for aligning this ontology pair from 162 minutes to 85 minutes.

Furthermore, the alignment generated by **Falcon-AO** with BCD gained in precision from 74% to 76% while keeping the recall unchanged.

In summary, the introduction of BCD led to significant reductions in convergence time for all four iterative algorithms on multiple ontology pairs, some extending to an order of magnitude. Simultaneously, the quality of the final alignments as indicated by F-measure improved for a few pairs, with one pair showing a reduction in the context of **Optima**. However, I did not observe a change in the F-measure for many of the pairs. Therefore, my empirical observations indicate that BCD does not have a significant adverse impact on the quality of the alignment.

5.5 Optimizing BCD using Partitioning and Ordering Schemes

As I mentioned previously, BCD does not overly constrain the formation of the coordinate blocks and neither does it impose an ordering on the consideration of the blocks, other than satisfying the cyclic rule. Consequently, I explore other ways of ordering the blocks and partitioning the alignment variables in the context of the representative algorithms. While the partitioning and ordering utilized previously are intuitive, my objective is to discover if other ways may further improve the run time performances of the algorithms. In subsequent experimentation, I exclude **MapPSO** from my representative set due to the randomness in its algorithm, which leads to comparatively high variability in its run times.

5.5.1 Ordering The Blocks

The order in which the blocks are processed may affect performance. This is because updated correspondences from the previous blocks are used in generating the alignment for the current block. Initially, blocks with participating entities of increasing height beginning with the leaves were used. Other ordering schemes could improve performance:

- I may reverse the previous ordering by cycling over blocks of decreasing height, beginning with the block that contains entities with the largest height. This leads to processing parent entities first followed by the children.

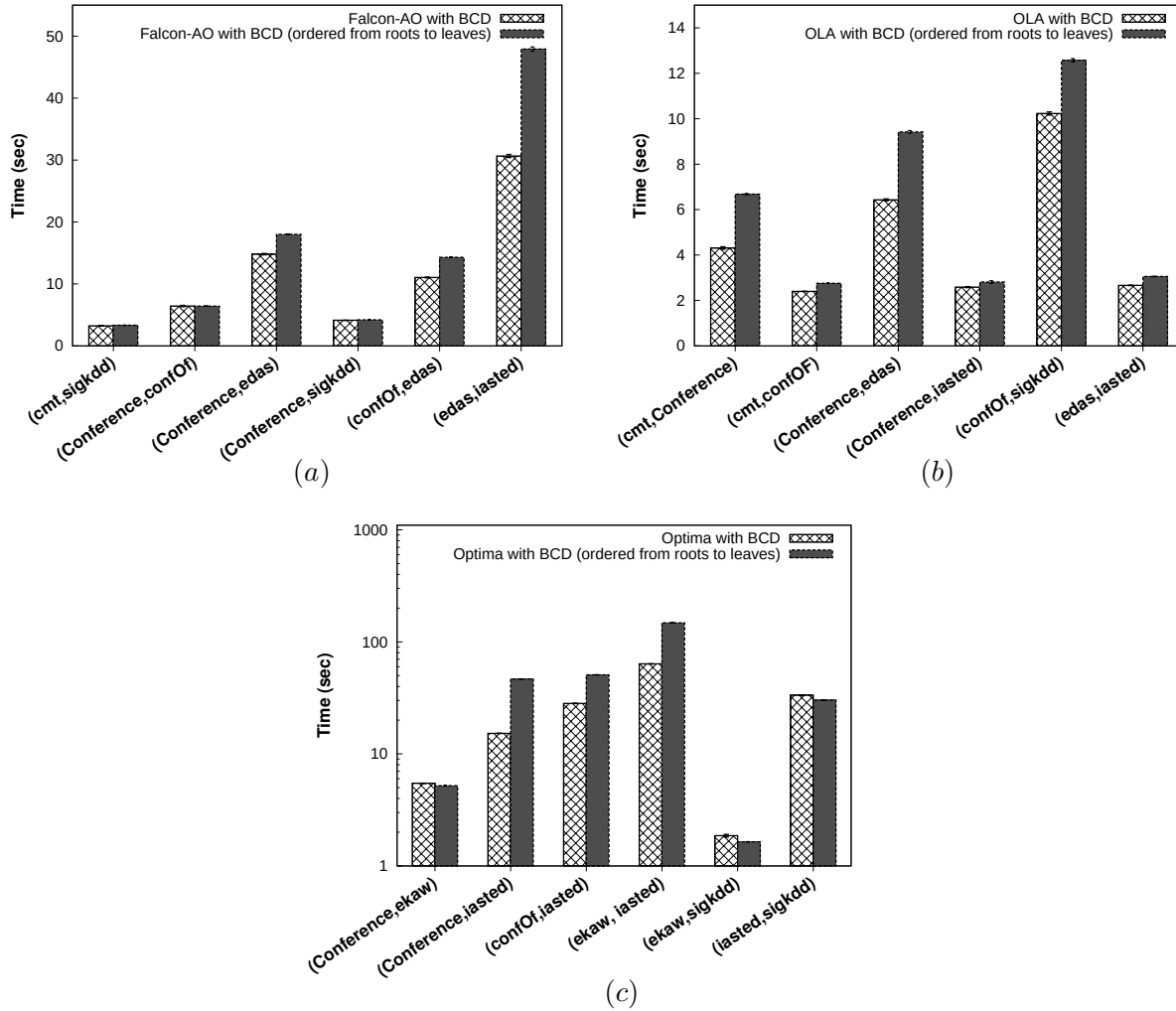


Figure 5.6: Average execution times of, (a) **Falcon-AO**, (b) **OLA**, and (c) **Optima**, with BCD that uses the initial ordering scheme and with BCD ordering the blocks from root(s) to leaves, for 6 of the 21 ontology pairs from the *conference* domain. While I ran the algorithms for all the pairs, I selected ontology pairs which exhibited the highest and lowest differences in average execution times. While this alternate ordering increases the run times to convergence, I did not observe significant improvements in the F-measures.

- I may obtain a quick and approximate estimate of the amount of alignment in a block of variables. One way to do this is to compute an aggregate measure of the lexical similarity between the entities of the two ontologies participating in the block. Assuming the similarity to be an estimate of the amount of alignment in a block, I may convert the estimates

into a probability distribution that gives the likelihood of finding multiple correspondences in a block. The block to process next is then sampled from this distribution. This approach requires a relaxation of the cyclic rule because a particular block is not guaranteed to be selected. In this regard, an expectation of selecting each block is sufficient to obtain asymptotic convergence of BCD [65].

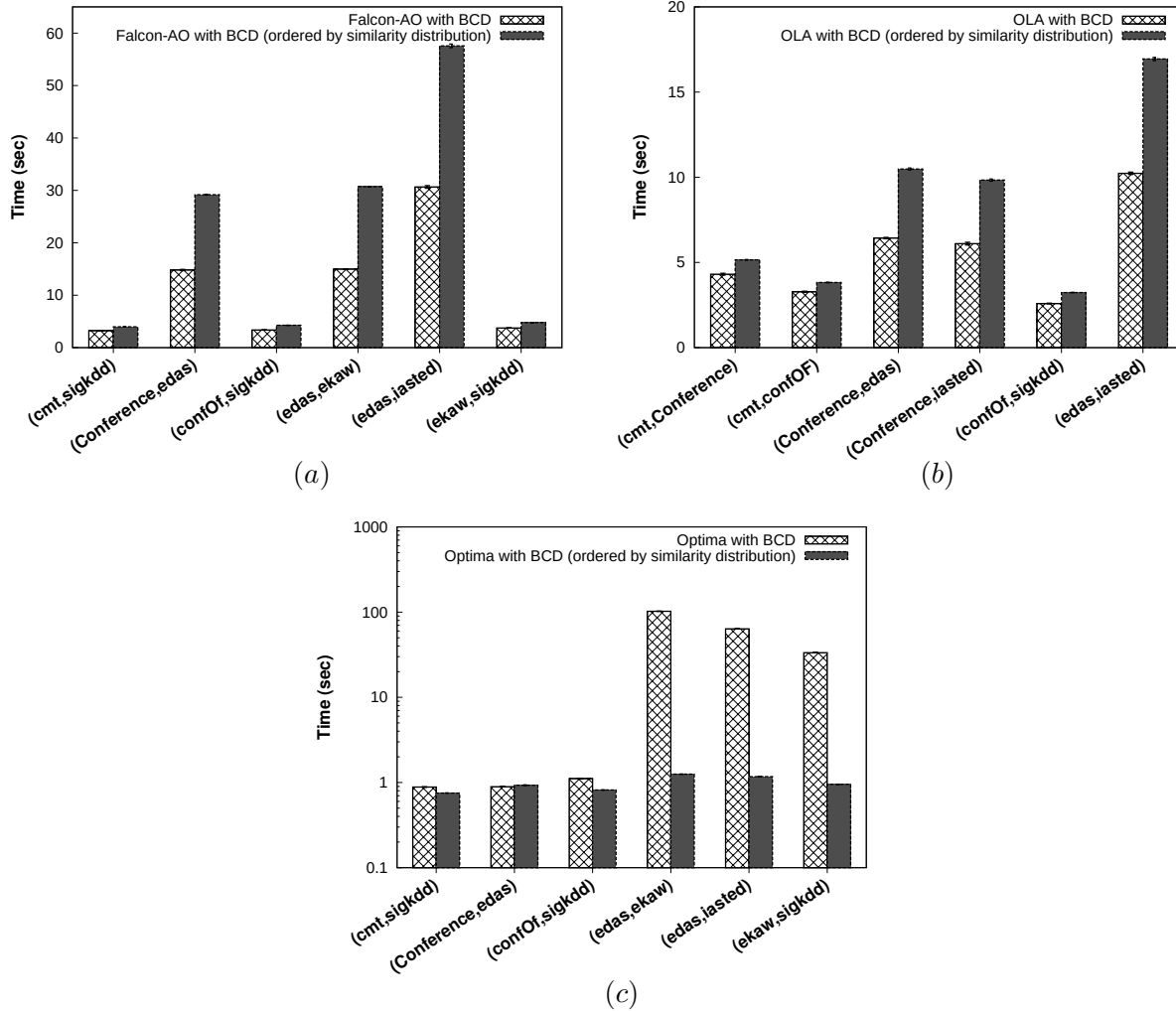


Figure 5.7: Average execution time consumed by, (a) **Falcon-AO**, (b) **OLA** and (c) **Optima** with BCD utilizing the previous ordering scheme and with BCD ordering the blocks by similarity distribution, for 6 of the 21 ontology pairs from *conference* domain. Although I ran the algorithms for all the pairs, I selected ontology pairs which exhibited the highest and lowest differences in average execution times. The new ordering helped **Optima** further cut down the total execution time by 262 seconds while finding 1 more correct correspondence and 6 false positives across all pairs changing the final F-measure slightly.

I compare the performances of the alternate ordering schemes with the initial on the 21 ontology pairs in the *conference* domain. The results of reversing the order of the original scheme are shown in Fig. 5.6. Clearly, the original ordering allows all three BCD-enhanced approaches to converge faster in general. While **Optima**'s average recall across all pairs improved slightly from 68% to 70%, average precision reduced by 4% to a final of 56%. **Falcon-AO**'s average F-measure improved insignificantly at the overall expense of 40 seconds in run time. Reversing the order has no impact on the precision and recall of **OLA**. These results are insightful in that they reinforce the usefulness of the alignment heuristic motivating the original ordering scheme.

my second alternate ordering scheme involves determining the aggregate lexical similarity between the entities participating in a block. The distribution of the similarities is normalized and the next block to consider is sampled from this distribution. Notice from Fig. 5.7 that **Falcon-AO** and **OLA** demonstrate significant increases in convergence time ($p \ll 0.01$) compared to utilizing BCD with the initial ordering scheme; on the other hand, the overall time reduces for **Optima** and by orders of magnitude for some of the pairs containing the larger ontologies such as *edas* and *iasted*. I select 6 ontology pairs, which exhibit the highest and lowest differences in average execution times to show in Fig. 5.7 for clarity. **Falcon-AO**'s precision and recall show no significant change and its F-measure remains unchanged. **OLA** loses both precision and recall with the similarity distribution scheme. The precision across all pairs went down to 13% from 37% along with a 24% drop in recall from 58% leading to a drop in F-measure to 19%. However, **Optima**'s F-measure remains largely unaffected.

Recall that both **Falcon-AO** and **OLA** perform iterative updates while **Optima** conducts an iterative search. While all sampled blocks undergo updates by the iterative update algorithms, search algorithms may not improve the blocks having low similarity. Consequently, blocks with high similarity that are sampled more often are repeatedly improved. This results in quicker convergence to a different and peculiar local optima where the blocks with high similarity have converged while the others predominantly remain unchanged. Thus, the alignment quality remains largely unaffected while the convergence time is reduced, as I see in the context of **Optima**.

5.5.2 Partitioning the Alignment Variables

Because BCD does not impose a particular way of grouping variables, other well-founded partitioning schemes may yield significant improvements:

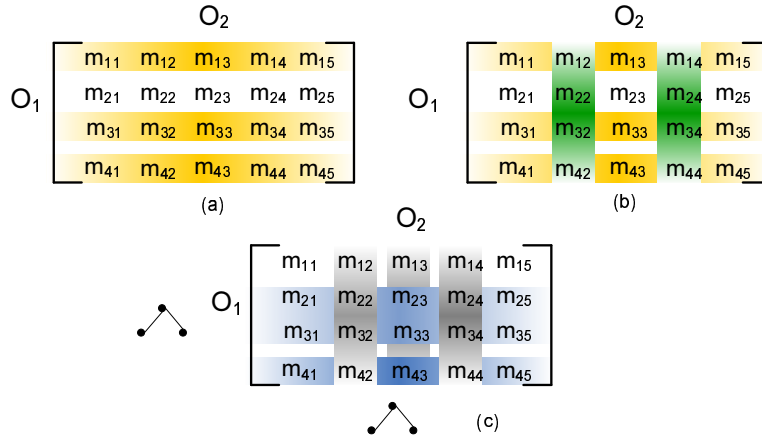


Figure 5.8: Matrices representing an intermediate alignment between entities of $\mathcal{O}_1$ and $\mathcal{O}_2$. (a) Identically shaded rows form a block of variables because the corresponding entities of $\mathcal{O}_1$ are at the same height. (b) Identically shaded rows and columns correspond to entities at the same height in $\mathcal{O}_1$ and $\mathcal{O}_2$, respectively. Variables in overlapping regions form a block. (c) Entities corresponding to identically shaded rows or columns form subtrees.

- An extension of the initial scheme (Fig. 5.8(a)) would be to group variables representing correspondences such that the participating entities from each of $\mathcal{O}_1$ and $\mathcal{O}_2$ are at the same height in relation to a leaf entity in the ontology, as I illustrate in Fig. 5.8(b). Note that the entity heights may differ between the two ontologies. This is based on the observation that the generalization-specialization hierarchy of concepts pertaining to a subtopic is usually invariant across ontologies.
- A more sophisticated scheme founded on the same observation is to temporarily transform each ontology, which is modeled as a labeled graph, into a tree. I may utilize any graph search technique that handles repeated nodes, such as breadth-first search for graphs [74], to obtain the tree. If the ontology has isolated graphs leading to separate trees, I use the owl:thing node to combine them into a single tree. Subsequently, I group those variables such that participating entities from each ontology are part of a subtree of a predefined size

(Fig. 5.8(c)). I may discard the ontology trees after forming the blocks. While the previous schemes form blocks of differing numbers of variables, this scheme forms all but one block with the same number of variables by limiting the subtree size.

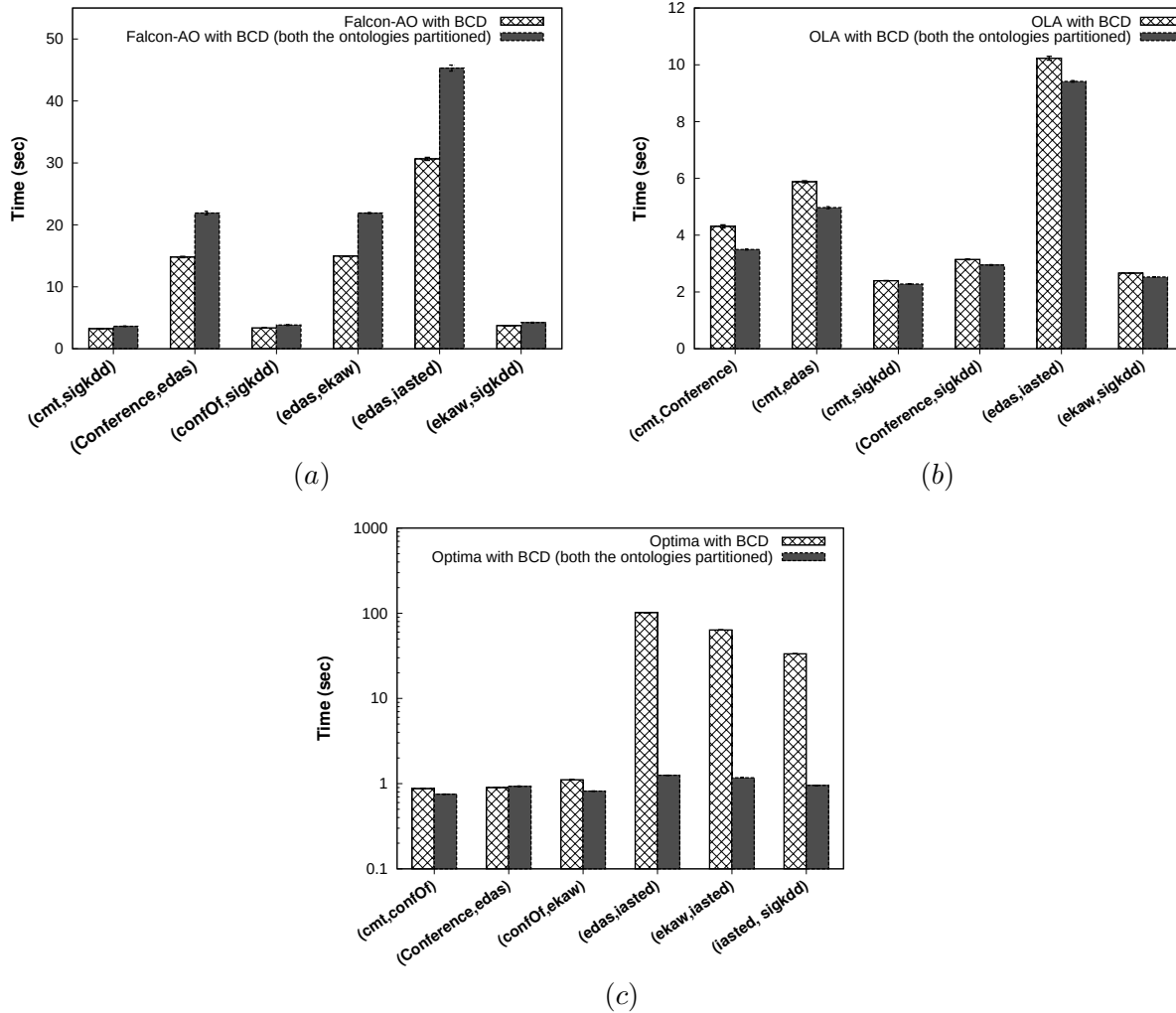


Figure 5.9: Execution times consumed by, (a) **Falcon-AO**, (b) **OLA**, and (c) **Optima** with BCD that uses blocks obtained by partitioning a single ontology and with BCD that utilizes partitions of both the ontologies, for 6 of the 21 ontology pairs from *conference* domain. Although I ran the algorithms for all the pairs, I selected ontology pairs which exhibited the highest and lowest differences in execution times. **Optima**'s total execution time over all pairs reduced by 274 seconds. False positive correspondences reduced by 37 at the expense of 3 correct correspondences. **OLA** cut down 10 seconds of the total execution time and 2 incorrect correspondences.

Based on the findings in the previous subsection, the blocks are ordered based on height of the participating entities or the subtrees' root nodes for **Falcon-AO** and **OLA**. I begin with the blocks of smaller height and proceed to those with increasing height. For **Optima**, I sample the blocks using a distribution based on the lexical similarity between participating entities.

As illustrated in Fig. 5.9, partitioning both the ontologies helped **Optima** the most and significantly saves on its execution times ($p \ll 0.01$). For the pairs involving some of the larger ontologies, it reduced by more than an order of magnitude. Furthermore, **Optima** gains in precision over all pairs by 6% with a 1% reduction in recall resulting in a 3% gain in F-measure to 67%. **OLA** saves on execution time as well – relatively less than **Optima** – with a slight improvement in its alignment quality. On the other hand, **Falcon-AO** experienced an increase in its total execution time over all the pairs. **Optima**'s improved performance is attributed to blocks that are now smaller allowing a more comprehensive coverage of the search space in less time. On the other hand, iterative update techniques such as **Falcon-AO** do not show any improvement because the smaller blocks may be a sign of overpartitioning.

Figure 5.10 illustrates the impact of subtree-based partitioning in all three algorithms. **Falcon-AO** exhibited a significant reduction in execution times ($p < 0.01$) simultaneously with an improvement in precision and F-measure over all the pairs by 3%. Similar to the previous optimization, **OLA**'s execution time reduces significantly as well ($p < 0.01$) while keeping its output unchanged. On the other hand, this partitioning technique reduces the efficiency of **Optima** with a small reduction in alignment quality as well. **Falcon-AO**'s GMO employs an approach that relies on inbound and outbound neighbors, which is benefited by using blocks whose participating entities form subtrees. As structure-based matching in **Optima** is limited to looking at the correspondences between the immediate children, including larger subtrees in blocks may not be of benefit.

Given the BCD-based enhancement and optimizations, how well do these algorithms compare in terms of execution time and alignment quality with the state of the art? In order to answer this

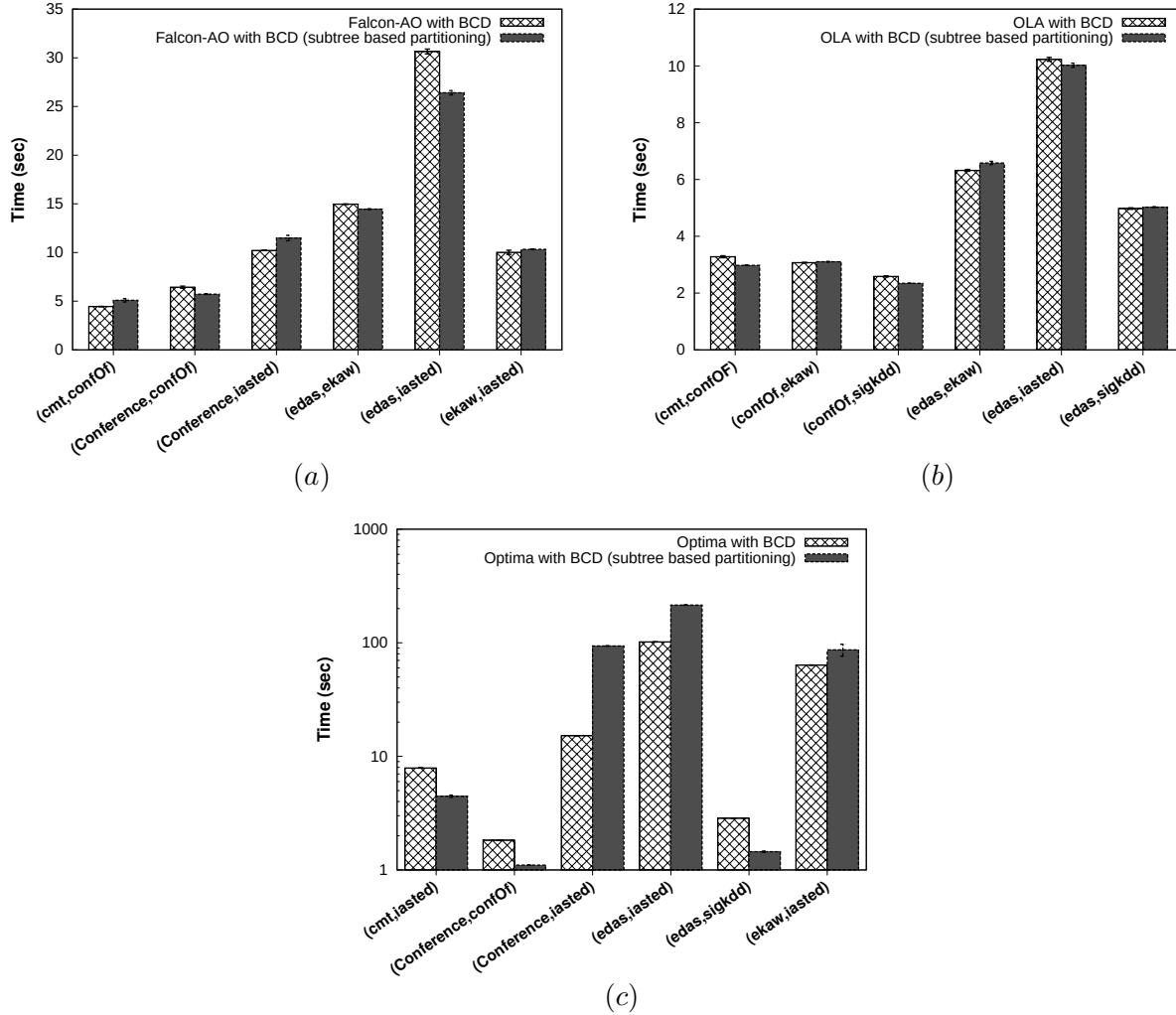


Figure 5.10: Execution times consumed by, (a) **Falcon-AO**, (b) **OLA**, and (c) **Optima**, with BCD that uses the default partitioning approach and with BCD that uses subtree-based partitioning, for 6 of the 21 ontology pairs from *conference* domain. I ran the algorithms for all the pairs of which I selected ontology pairs that exhibited the highest and lowest differences in execution times. The total execution time of **Falcon-AO** for the complete conference track reduces by 8 sec along with a reduction of 71 false positives. **OLA** saves 1.5 sec in total execution time while keeping the output alignments unchanged. However, **Optima** consumes 192 seconds more.

question, I compare with the performances of 18 algorithms that participated in the *conference* track of OAEI 2012 [84]. Among these, an iterative alignment algorithm, **YAM++**, produced the best F-measure for the 21 pairs followed by **LogMap**, which does not utilize optimization, **CODI**,

and **Optima+**, which is **Optima** augmented with BCD. These latter approaches all produced F-measures, which were tied or within 2% of each other.

OAEI reports run time on a larger task of aligning 120 *conference* ontology pairs. On this task, while YAM++ consumed more than 5 hours for all the pairs, LogMap took slightly less than 4 minutes and **Optima+** consumed 22 minutes. Because **Falcon-AO** and **OLA** did not participate in OAEI 2012, I utilized them separately on the 120 pairs on the machines, whose configurations are comparable to that utilized by OAEI. **Falcon-AO** and **OLA** enhanced with BCD consumed 11 and 5 minutes respectively although their alignment quality is lower than that of **Optima+**. This would place all three representative algorithms in the top two-thirds among the 18 that participated in the *conference* track of OAEI, in terms of run time, and **Optima** and **OLA** in group 1 with respect to alignment quality.

5.6 Discussion

While techniques for scaling automated alignment to large ontologies have been previously proposed, there is a general absence of effort to *speed up* the alignment process. I presented a novel approach based on BCD to increase the speed of convergence of an important class of alignment algorithms with no observed adverse effect on the final alignments. I demonstrated this technique in the context of four different iterative algorithms and evaluated its impact on both the total time of execution and the final alignment's precision and recall. I reported significant reductions in the total execution times of the algorithms enhanced using BCD. These reductions were most noticeable for larger ontology pairs. Often the algorithms converged in a lesser number of iterations. Simultaneously, the integration of BCD improved the precision of the alignments generated by some of the algorithms while retaining the recall. However, BCD does not promote scalability to large ontologies.

The capability to converge quickly, allows an iterative alignment algorithm to run until convergence, in contrast to the common practice of terminating the alignment process after an arbitrary

number of iterations. As predefining a common bound for the number of iterations is difficult, speeding up the convergence becomes vital.

I believe that the observed increase in precision of the alignment due to BCD is because of the optimized correspondences found for the previous coordinate block, which influence the selection of the mappings for the current coordinate block. Additionally, the randomly generated mappings in MapPSO are limited to the block instead of the whole ontology, due to which the search becomes more guided.

Given that on integrating BCD the iterative algorithms produced better quality alignments, I infer that the original algorithms were converging to local optima, instead of the global optima, and that using BCD has likely resulted in convergence to (better) local optima as well. This is a significant insight because it uncovers the presence of local optima in the alignment space of these algorithms. This may limit the efficacy of iterative alignment techniques.

Interestingly, performances of the iterative update and search techniques are impacted differently by various ways of formulating the blocks and the order of processing them. Nevertheless, the approach of grouping alignment variables into blocks based on the height of the participating entities in the ontologies is intuitive and leads to competitive performance. However, different ontology pairs may lead to a differing number of blocks of various sizes: in particular, “tall” ontologies that exhibit a deep class hierarchy result in more blocks than “short” ontologies.

CHAPTER 6

BATCH ALIGNMENT OF LARGE ONTOLOGIES USING MAPREDUCE

We are witnessing a growing number of ontology repositories hosting several ontologies on specific domains [50, 63, 92]. Simultaneously, ontologies in these repositories are significantly large (more than 1,000 concepts). For example, the National Center for Biomedical Ontologies (NCBO) [63] currently hosts more than 320 ontologies pertaining to the life sciences. Among these, about 30% have more than 2,000 entities and relationships, which makes them very large in size. Because many of these ontologies overlap in their scope, aligning ontologies is important to the success and usefulness of the repositories [2].

Although ontology alignment is traditionally perceived as an offline and one-time task, issues of scaling to large ontologies and performing the alignment in a reasonable amount of time without much qualitative compromise are gaining importance. In the recent edition of the annual ontology alignment evaluation initiative (OAEI) 2012 [84], only 8 out of the 21 alignment algorithms completed the very large biomedical ontology track. Subsequently, OAEI pointed out that the sizes of the input ontologies significantly affect the efficiency of many algorithms. As services and applications such as search engines [14, 75], ontology management tools for repositories [32], thesaurus management [29] and semantic Web service composition [73] begin to rely on the alignment provided by repositories, the significance of keeping the ontologies aligned increases. As new ontologies are submitted or ontologies are updated, their alignment with others must be quickly computed. As existing algorithms find it difficult to scale up for very large ontologies, aligning several pairs of ontologies quickly becomes a challenge for these repositories.

A prevalent way of managing the alignment complexity posed by large ontologies is to simply dissect the ontologies into smaller pieces and align some of the ontology parts [20, 41]. Parallelizing the alignment process is another way of approaching scalability. Intra-matcher parallelization introduces parallelization within the alignment algorithm. On the other hand, inter-matcher parallelization aligns several ontology parts in parallel using ontology alignment algorithms [31].

In the context of a general absence of inter-matcher parallelization, my primary contribution in this chapter is a novel and general method for batch alignment of large ontology pairs using the distributed computing paradigm of MapReduce [18]. As distributed computing clusters including cloud computing proliferate, the significance of this approach is that it allows me to exploit these parallel computing resources toward automatically aligning several ontologies whose scale takes them out of the reach of many of the current algorithms, and simultaneously align in a reasonable amount of time.

I identify three key challenges in casting the ontology alignment problem for MapReduce. Given an ontology alignment problem of finding correspondences between an ontology pair, $\mathcal{O}_1$, and $\mathcal{O}_2$, I am interested in decomposing the problem such that, a part of an ontology, $\mathcal{O}_1$, corresponds predominantly to one other part of the second ontology, $\mathcal{O}_2$, rather than distributing its correspondences among many parts. This helps in both aligning the parts as independently as possible and in merging the correspondences together to obtain the final alignment between the ontologies easily. I utilize the recent partition-anchor-partition approach [34] to identify the ontology parts.

Given the subproblems, I show how the *mapper* and *reducer* methods of MapReduce may be implemented. My input file is a list of data records, where each record is a pair consisting of ontology parts that constitutes an alignment subproblem and an associated key. *Mapper* reads the input and creates intermediate files in the local file system of the nodes in the cluster. Multiple *reducers*, one on each node, align the paired parts and generate the output alignment. Finally, the alignment from each *reducer* is merged with the others. The challenge here is formulating the alignment of subontologies in keeping with the simple functional paradigm of MapReduce.

Postprocessing of the correspondences is needed in order to produce a final consistent alignment. I identify two important inconsistencies which could occur while merging alignments of subproblem and resolve them during postprocessing. I do not seek inconsistencies within an alignment of a subproblem – these are often resolved by the algorithm itself – but rather address the inconsistencies between alignments of two different subproblems while merging them. I avoid complex postprocessing such as [45,60] because it could get computationally expensive for a large number of correspondences.

In order to demonstrate the efficiency that MapReduce brings in general, I utilize **Falcon-AO** [46], **Logmap** [47] **Optima+** [20], and **YAM++** [66] as representative algorithms and the open-source **Hadoop** implementation [28, 95] of MapReduce. These established algorithms participated in previous OAEI competitions [79, 83, 84] and performed well. Using batches of several ontology pairs spanning multiple domains, I show: (a) my formulation of distributed alignment using MapReduce demonstrates more than an order of magnitude in speedup for aligning multiple ontology pairs; (b) small changes in the quality of the alignment when using some of the algorithms while no change for others; (c) batch alignment of large ontologies using scalable algorithms such as **Logmap** may be further speeded up through distributed computing despite the overhead.

6.1 Representative Algorithms

I propose a framework for parallel execution of ontology alignment in a distributed cluster using the MapReduce model. Existing alignment algorithms may be used within my approach to align a set of very large ontologies in parallel on a computing cluster. Open source implementations of MapReduce such as **Hadoop** address the housekeeping tasks involved in distributed computing such as a simple partitioning of the input data, managing node failures, and administering communications while expecting users to implement the *map* and *reduce* steps.

I select automated alignment algorithms, which have participated in multiple editions of OAEI and performed well¹. Availability of the source code is not critical to my study. However, its access does facilitate a better integration with the distributed computing architecture's file system and time keeping.

Altogether, these four algorithms represent a mix of alignment techniques. Over the years, these algorithms have performed well in the annual OAEI competitions [79, 83, 84]. **Falcon-AO** and **Optima+** demonstrated strong performances with respect to the comparative quality of the generated alignment on moderately-sized ontologies (less than 200 named classes and 100 properties). For example, **Falcon-AO** often placed in the top 3 systems when it participated in OAEI competitions between 2005 and 2010. **Optima+** placed second (F-2 score) in an important track in the 2012 edition of the OAEI competition [90]. These two algorithms are competitive for medium-sized ontologies. **YAM++** placed first in the conference and large biomedical ontology tracks in the 2012 OAEI edition, while placing second in the anatomy track. **YAM++** is widely regarded as generating the most accurate and complete alignments among all algorithms. Yet, these algorithms may not align very large ontologies due to memory issues or are unable to produce an alignment in a reasonable amount of time. **YAM++**, was the slowest in the large biomedical track, and unable to complete the conference track within 5 hours. **Logmap** placed among the top systems in many tracks (conference, anatomy, and the large biomedical ontology) in the 2012 OAEI. Importantly, it scales significantly well for large ontologies. Consequently, my representatives are state of the art alignment algorithms.

6.2 Overview of MapReduce Paradigm

MapReduce [18] is a popular programming framework for processing large data sets in parallel using a distributed computing environment. MapReduce involves two steps: **Map** and **Reduce**. The *map* function maps the input data to an intermediate data-set which is processed by a *reduce*

¹I tried six different alignment algorithms –CODI, **Falcon-AO**, **Logmap**, **Optima+**, GOMMA and **YAM++** – for this study. Among these, CODI would not be used due to proprietary dependencies and GOMMA required special access rights to a database server which were not available to me.

function. The *reduce* function reads the output of map, processes it and generates the final output. MapReduce defines a *master* node and several *worker* nodes. The master node manages the distribution of tasks and data to worker nodes. A worker node is a *mapper* if it performs the map step or is labeled as a *reducer* if it is assigned a reduce task.

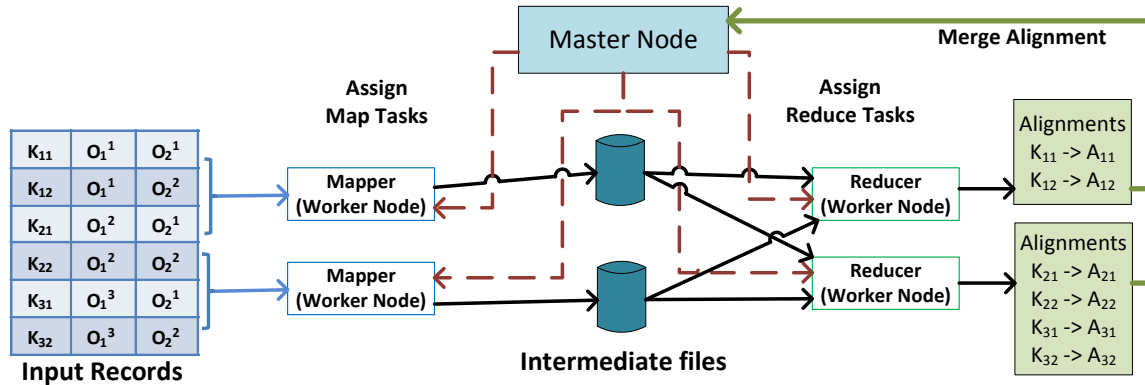


Figure 6.1: The MapReduce framework for ontology alignment. The input is a list of key-value pairs, which is split. A mapper reads a record and writes intermediate key-value pairs to the different nodes' local file systems. A reducer reads the intermediate output allocated to it and aligns the subontologies. Finally, output alignments between the subontology pairs are merged.

Input to the MapReduce framework is a list of data records, where each record has a unique key and a value. The master node splits the input and assigns each part to a mapper. The mapper reads each record in the given part and generates intermediate key-value pairs. The master node then processes the intermediate output from mappers and assigns a set of keys and for each key the list of all associated values to a reducer. For each key, reducer processes the set of values and writes out the output in key-value pair format. Distributed implementations of MapReduce such as **Hadoop** [28] provide functionalities such as a simple partitioning of the input data, managing node failures, and administering communications, while expecting users to program the *map* and *reduce* steps. Some of these aspects may be flexibly configured in **Hadoop** to suite the problem context on hand. Approaches adopting this functional model may be naturally parallelized and executed on a large cluster of commodity machines. In this distributed setup, several mappers and reducers could be independently working in parallel. MapReduce provides a simple programming framework for tasks to scale up to large data while keeping the overhead of distributed computation transparent.

However, not every large-scale data processing task is appropriate for MapReduce. For example, problems that may not be decomposed and solved independently are not suitable for MapReduce.

6.3 Distributed Ontology Alignment Using MapReduce

MapReduce requires that the input data be split. The generic scheme in implementations such as **Hadoop** sequentially reads the data and groups a configurable number of records into a block, with each record representing a subproblem. *However, this naive splitting is not directly suited for the ontology alignment task because it would dissect the input ontologies (RDF triples) without any attention to the semantic cohesiveness of each block and its potential for alignment with parts of the other ontology.* Consequently, the ontology pairs need to be preprocessed into a list of subproblems. Below, I introduce my approach to generate these subproblems and align them in the MapReduce paradigm.

6.3.1 Identifying Alignment Subproblems

I formulate alignment subproblems by partitioning each pair of ontologies, $\mathcal{O}_1$ and $\mathcal{O}_2$ from the batch, and aligning pairs of parts. Let $\mathcal{O}_1$ and $\mathcal{O}_2$ be partitioned into k_1 subontologies, $\{\mathcal{O}_1^1, \mathcal{O}_1^2, \dots, \mathcal{O}_1^{k_1}\}$, and k_2 subontologies, $\{\mathcal{O}_2^1, \mathcal{O}_2^2, \dots, \mathcal{O}_2^{k_2}\}$, respectively.

Among the few existing partitioning approaches, **Falcon-AO** generates structurally cohesive subontologies using clustering [41]. Hamdi et al. [34] noted that in this approach each ontology is decomposed independently of the other without considering the alignment objective. This limitation is mitigated by first identifying anchors, which are entities in the two ontologies that have identical names or labels, followed by forming subontologies around these anchors based on the structural neighborhood. In this study, I utilize this technique to cluster the concepts. To possibly avoid losing relationships between entities in different clusters, I duplicate one of the participating entities in the other cluster and add the relationship. Note that this step may lead to overlapping

subontologies, and therefore the parts do not technically form a partition. Given the subontologies, I formulate alignment subproblems, $(\mathcal{O}_1^i, \mathcal{O}_2^j)$ such that parts i and j have a correspondence between their anchors.

6.3.2 Aligning Ontologies Using MapReduce

As stated earlier, the MapReduce programming framework facilitates distributed processing of large data using computing clusters.

An alignment subproblem, S_{ij} , is defined as, $S_{ij} = \langle K_{ij}, (\mathcal{O}_1^i, \mathcal{O}_2^j) \rangle$ where, K_{ij} is the unique key for the subproblem, and $\mathcal{O}_1^i$ and $\mathcal{O}_2^j$ are the subontologies that have a correspondance between their anchors, as discussed in the previous subsection. As shown in Fig. 6.1, the input to MapReduce is a set of key-value pairs such that the key uniquely identifies a subproblem and the value is a pair of subontologies associated with that subproblem. This list is split by the master node and the parts are sent to the mapper nodes. The map function reads in a data record, say S_{ij} , and writes out two intermediate key-value pairs, one for each subontology – $\langle K_{ij}, \mathcal{O}_1^i \rangle$ and $\langle K_{ij}, \mathcal{O}_2^j \rangle$. An instance of the reducer node will get these new key-value pairs, and possibly more with other keys. The reduce function aligns the subontologies associated with the same key, and writes out the output as another key-value pair where the key remains the same, K_{ij} , and the value is the alignment between the corresponding blocks. Alignments for all subontology pairs from all reducers are transferred to the master node where they are merged. The overhead of distributed execution is usually transparent.

Alternately, the mapper on processing an input record may write out a key-value pair where the key is a subontology, $\mathcal{O}_1^i$, and the value is some subontology, $\mathcal{O}_2^j$, which is to be aligned with $\mathcal{O}_1^i$. Subsequently, a reducer node is tasked with multiple alignment subproblems where the first subontology remains the same, $\mathcal{O}_1^i$. Because alignments from all reducers are obtained by the master node and then merged, I may use either approach for the mapper, and I select the former.

6.3.3 Merging Subproblem Alignments

Alignment algorithms may process the correspondences. The goal of this postprocessing is to remove inconsistent and duplicate correspondences. Despite this post processing, the final alignment between two ontologies may not simply be a concatenation of the alignments for each subproblem. I may need to postprocess them further to remove specific inconsistencies. In addition to removing duplicate correspondences, I identify two inconsistencies, which must be resolved:

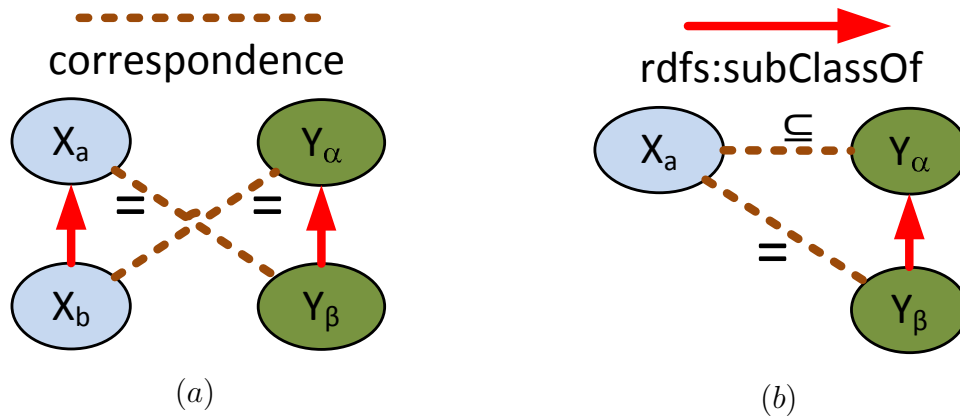


Figure 6.2: Two types of inconsistent correspondences, which must be resolved while merging subproblem alignments. (a) Crisscross correspondences, and (b) redundant correspondences.

1. *Crisscross mappings*, as illustrated in Fig. 6.2(a). While merging alignments from two subproblems, let there exist correspondence, $\langle x_a, y_\beta, =, c_{a\beta} \rangle$, in one alignment and, $\langle x_b, y_\alpha, =, c_{b\alpha} \rangle$, in the other, where x_a and x_b are entities in ontology, $\mathcal{O}_1$, y_α and y_β are entities in ontology, $\mathcal{O}_2$, and $c_{a\beta}$ and $c_{b\alpha}$ are confidence scores in the equivalence correspondences. If x_b is a subclass of x_a and y_β is a subclass of y_α then these crisscross correspondences are inconsistent. I remove the one with the lower confidence score while merging.
2. *Redundant mappings* are illustrated in Fig. 6.2(b). In order to keep the alignment minimal, I remove those correspondences which may be inferred from another. Let there exist correspondence, $\langle x_a, y_\alpha, \subseteq, c_{a\alpha} \rangle$, in one subproblem alignment and, $\langle x_a, y_\beta, =, c_{a\beta} \rangle$, in the other alignment. Here, x_a is an entity of ontology, $\mathcal{O}_1$, and y_α and y_β are entities of ontology, $\mathcal{O}_2$.

If y_β is a subclass of y_α , then I may remove the correspondence, $\langle x_a, y_\alpha, \subseteq, c_{a\alpha} \rangle$, which can be inferred.

Though these inconsistencies are similar to those previously discussed [45, 60], and can be resolved using the same techniques, they are not obtained in a similar manner. Importantly, I do not seek inconsistencies within an alignment of a subproblem, but address the inconsistencies between alignments of two different subproblems while *merging* them. I may enrich this postprocessing further using the techniques detailed in [45, 60]. However, more sophisticated postprocessing of a large number of correspondences could get computationally expensive.

6.4 MapReduce Algorithm

I outline the algorithms for the mapper and reducer steps of MapReduce. The master node allocates a set of key-value pairs to each mapper node, where the key uniquely identifies a subproblem and the value is the associated subontology pair. For a key-value, a mapper applies the MAP function detailed in Algorithm 1. During mapping, each subontology is emitted as a value in a key-value format, where the key remains the same as before. A set of mappers may read the subproblems in parallel and distribute them.

Algorithm 1 Algorithm for mapping an input record.

MAP ($\langle K_{ij}, \text{Value} \rangle$)

- 1: $\{O_1^i, O_2^j\} \leftarrow$ parse the Value in the record
 - 2: emit($\langle K_{ij}, O_1^i \rangle$)
 - 3: emit($\langle K_{ij}, O_2^j \rangle$)
-

Algorithm 2 Reducer aligns subontologies using an alignment algorithm.

REDUCE ($\langle K_{ij}, \{O_1^i, O_2^j\} \rangle$)

- 1: $A_{ij} \leftarrow$ align O_1^i, O_2^j using an alignment algorithm
 - 2: emit($\langle K_{ij}, A_{ij} \rangle$)
-

As soon as some data has been mapped, the master node collects all values with the same key and starts allocating a set of such records to a reducer. The reducer applies the REDUCE function depicted in Algorithm 2. For the key and subontology pair received, it aligns the two subontologies

using an alignment algorithm. This is followed by writing out the alignment result as a value in key-value pair format where the key remains unchanged. Finally, the master node collects all the output and merges it while resolving any inconsistencies as mentioned in the previous subsection. Several reducers may align in parallel as soon as the master node allocates the key-value pairs to them. Both, distributing subproblems and aligning subontologies are carried out in parallel on different nodes, which provides a speedup in the alignment.²

6.5 Performance Evaluation

I study the impact of distributing ontology alignment using MapReduce in terms of average speedup of the alignment time and its impact on the quality of the alignment. Specifically, I compare the total execution time when using MapReduce with the time required by the default setup of each alignment algorithm for aligning batches of ontology pairs together. In order to evaluate whether my formulation in MapReduce is properly distributed, I measure the speedup obtained as I allocate an increasing number of nodes to **Hadoop** for each algorithms. For this study, I utilize the four representative alignment algorithms: **Falcon-AO**, **Logmap**, **Optima+**, and **YAM++** described in Section 6.1.

I use three comprehensive batches of several ontology pairs spanning multiple domains. The first batch, labeled *conference testbed* consists of 120 medium-sized ontology pairs from the *conference* track of OAEI 2012, all of which structure knowledge related to conference organization. OAEI provides reference alignments for only 21 pairs in this track. The second batch includes very large ontology pairs from *anatomy*, *library* and *large biomedical ontologies* tracks of OAEI 2012 along with their reference alignments. I call this batch as, *large OAEI testbed*. Participating ontologies in this batch are detailed in Table A.2; these ontologies are semantically rich and contain tens of thousands of classes. Finally, I used the novel batch of 50 large ontology pairs mentioned in the previous chapter for evaluation purposes made of ontologies from the NCBO and labeled it as the

²I provide an implementation of my algorithm at http://thinc.cs.uga.edu/thinclabwiki/index.php/Distributed_Alignment_of_Ontologies for reuse. Note, that this implementation utilizes the alignment API provided by OAEI.

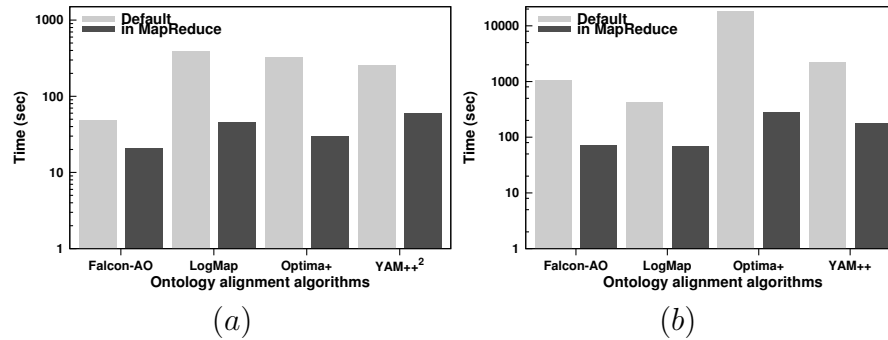


Figure 6.3: Average execution times of **Falcon-AO** , **Logmap**, **Optima+**, and **YAM++** in their original form on a single node and using MapReduce, for aligning the (a) *conference* track ontologies and (b) large OAEI ontologies. Note, **YAM++** has difficulties in aligning some ontology pairs in conference track and biomedical track⁴. I observed an order of magnitude reduction in time, specifically for the large ontologies from OAEI for all four tools. Note that the time axes are in log scale.

biomedical testbed. This biomedical testbed contains 36 ontologies from NCBO repository organizing knowledge in various biomedical domains.³ The results and analysis of *biomedical testbed* is presented in Chapter 7 - Section 7.3.

I use a **Hadoop** cluster of 12 CentOS 6.3 systems, each with 24 2.0GHz Intel Xeon processors and memory limited to a maximum of 2GB per task in each node. All the ontologies I use, except those in the first testbed have cryptic ids as the names of the entities, but labels are descriptive. Because **Falcon-AO** heavily relies on names to identify correspondences, I configure it to use entity labels as well; **Optima+**, **Logmap**, and **YAM++** automatically adjust by analyzing the ontology. All timing results are averages of 3 runs; I observed very small variances in the execution times.

³The biomedical testbed in http://thinc.cs.uga.edu/thinclabwiki/index.php/Biomedical_Domain_Benchmark is available for download.

Table 6.1: The precision (P), recall (R) and F-measure (F) of the output alignments by **Falcon-AO**, **Logmap**, **Optima+**, and **YAM++** in MapReduce setup for the large ontology pairs from OAEI. The default performance of **Logmap** and **YAM++** are also presented in the table for comparison⁴. The output of default **Falcon-AO** and **Optima+** are same as the MapReduce setup, because they employ partitioning by default.

Ontology Pairs	MapReduce/Default Falcon-AO			MapReduce Logmap			MapReduce/Default Optima+			MapReduce YAM++			Default Logmap			Default YAM++		
	P%	R%	F%	P%	R%	F%	P%	R%	F%	P%	R%	F%	P%	R%	F%	P%	R%	F%
(mouse,human)	73	74	73	96	75	84	78	73	76	95	77	85	92	85	88	94	86	90
(STW,TheSoz)	57	50	53	57	51	54	18	40	25	55	52	53	69	64	67	60	75	66
(fma,nci)	95	81	88	95	83	89	96	83	89	97	84	90	95	86	90	98	85	91
(fma,snomed)	85	63	72	85	63	72	84	61	71	86	63	73	97	66	78	97	70	81
(snomed,nci)	69	58	63	67	58	62	70	58	63	71	58	64	90	64	75	95	60	74

In Fig. 6.3, I show the average execution time consumed by **Falcon-AO**, **Logmap**, **Optima+**, and **YAM++** in batch aligning ontology pairs from the first two testbeds mentioned previously, in their default form on a single node and with MapReduce in the **Hadoop** framework ⁴. I observe an order of magnitude reduction in average execution time brought about by MapReduce for all four algorithms in aligning large ontology pairs from OAEI. In the conference testbed, MapReduce enhanced **Falcon-AO**, **Logmap**, **Optima+**, and **YAM++** to demonstrate a speedup of 2, 9, 11, and 4 respectively. For the second testbed, **Falcon-AO** and **Logmap** used in MapReduce achieved a speedup of 15 and 16, respectively. MapReduce with **Optima+** – the slowest among the selected algorithms showed an average speedup of 64, while distributed alignment using **YAM++** speeded up by 22. The

I tabulate the precision (P), recall (R) and F-measure (F) of the output alignments by all four algorithms in MapReduce setup for the large ontology pairs from OAEI in Table 6.1. Because, both **Falcon-AO** and **Optima+** by default employ partitioning, the performance metrics do not change between their default setup and MapReduce. I observed a significant reduction in F-measure when aligning subontology pairs in MapReduce using both **Logmap** and **YAM++**. A maximum of 13% reduction in F-measure is observed on using **Logmap**, for the (*snomed,nci*) ontology pair and on using **YAM++** for the (*STW,TheSoz*) ontology pair. I believe that with improved partitioning techniques I may reduce this impact.

As an aside, partitioning is not mandatory for my approach. For example, I also observe a significant speedup in aligning the medium-sized conference ontology pairs in MapReduce without partitioning. Batch alignment of conference testbed using MapReduce and **Falcon-AO**, **Logmap**, **Optima+**, and **YAM++** obtained 59%, 63%, 61% and 71% F-measure. Since I do not partition these medium-sized ontologies there is no change in output using MapReduce. For my biomedical

⁴As pointed out by OAEI [84], **YAM++** has known difficulties in aligning some ontologies from conference track. However, it is able to align the 21 ontology pairs for which the reference alignments are provided by OAEI. Hence, I evaluate the performance of **YAM++** in conference batch using these 21 ontology pairs. Also, it fails to complete my large biomedical testbed. Because its source-code is not available I could not investigate its failure. However, it was able to complete the biomedical testbed with partitioning. Subsequently, I compare the performance of default **YAM++** aligning ontology parts of the biomedical testbed with its MapReduce setup.

testbed **Falcon-AO** generated a recall of 49% while with **Optima+** a recall of 58% is obtained. **Logmap** and **YAM++** produced 51% and 56% recall respectively.

To analyze the maximum speedup the MapReduce approach could offer for batch aligning a given set of ontology pairs and the minimum number of nodes required to achieve it, I gradually increased the number of nodes allocated to MapReduce and measured the average execution time of alignment. The average execution time to align, (a) large OAEI testbed and (b) biomedical testbed using MapReduce with increasing number of nodes is shown in Fig. 6.4. The execution time decreases exponentially with an increasing number of nodes until it reaches a minimum. For a given testbed, the number of nodes required to achieve this minimum is upper bounded by the total number of alignment subproblems. On the other hand, the lower bound of the minimum time required by MapReduce to align a set of ontology pairs is the longest time required for any subproblem and the time required for initialization and communications of MapReduce. I observed that the minimum number of nodes required to reach the minimum execution time varies between using different algorithms and data-sets. This is because, execution times required for aligning subproblems vary between algorithms.

My performance study shows that the batch alignment of ontologies can be significantly speeded up using my approach with any ontology alignment algorithm. Using my approach, algorithms which already employ partitioning to manage memory and time complexity posed by large ontologies would produce the same alignment output. However, reduction in alignment quality is observed for some other alignment algorithms while aligning subontologies in MapReduce. I also observed that the time consumed by alignment algorithms in MapReduce follows an exponential decay with increasing number of nodes allocated.

6.6 Discussion

This chapter showed how automated ontology alignment may be performed in a distributed manner using the popular distributed computing model, MapReduce, thereby allowing ontology alignment to exploit the proliferating cloud computing paradigm. Importantly, my general approach

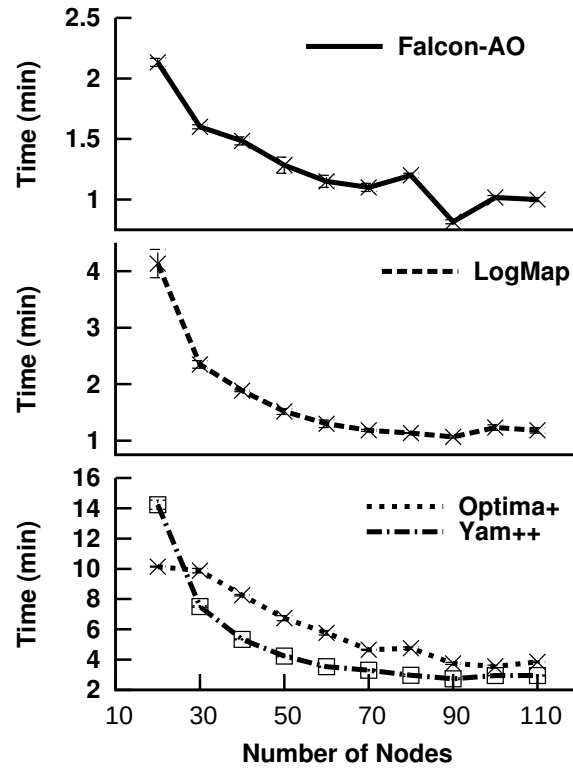


Figure 6.4: The plot demonstrates the exponential decaying of average total execution time with increasing number of nodes by **Falcon-AO**, **Logmap**, **Optima+**, and **YAM++** for large ontologies from OAEI. Note, the average execution time gradually converges to a minimum time. However, the minimum number of nodes required to attain this convergence differ between algorithms and data-sets.

offers significant speedup for batch alignment of ontology pairs using any alignment algorithm without modifying it. I contrast this approach with the alternative of parallelizing an alignment algorithm itself (referred to as intra-matcher parallelization). In the latter, while element-level lexical similarity computation may be parallelized for large ontology pairs, structural and logic-based matching is thought to be not amenable to parallelization. Speeding up batch alignment of ontology pairs will benefit ontology repositories such as NCBO that publish maps between all housed ontologies.

MapReduce demonstrated significant speedup when aligning three different batches of ontology pairs using four representative alignment algorithms. This included recent efficient

algorithms such as **Logmap**, whose alignment time for performing batch alignment reduced when deployed in MapReduce compared to its default execution on a single node without MapReduce. This represents an important step toward making alignment techniques computationally more scalable.

As additional analysis, I note that MapReduce also decreases the execution time of aligning a *single* ontology pair when used with any of the representative algorithms other than **Logmap**. For example, alignment using MapReduce with **Falcon-AO** gained speedup by a factor of 3.8 while with **Optima+**, it offered a speedup factor of 58, for aligning the very large ontology pair, (*mouse, human*), from OAIE's anatomy track. MapReduce with **YAM++** achieved a speedup of 22 for the same ontology pair. However, **Logmap** designed to be scalable from the ground up when used in MapReduce consumed 22 seconds more in aligning these ontologies. Here, I note that **YAM++** provides the best F-measure on this and many other ontology pairs, so its improved scalability is of import.

The reduction in alignment quality while aligning subontology pairs on using some of the algorithms may be reduced by improved partitioning approaches. Consequently, I think that the selection of the partitioning approach matters. As future work, I am interested in investigating ways of minimizing the loss in alignment quality by exploring other approaches of partitioning.

CHAPTER 7

LARGE BIOMEDICAL ONTOLOGY ALIGNMENT

Ontologies are becoming increasingly critical in the life sciences [13, 49] with multiple repositories such as Bio2RDF [9], OBO Foundry [4] and NCBO's BioPortal [63] publishing a growing number of biomedical ontologies from different domains such as anatomy and molecular biology. For example, BioPortal hosts more than 320 ontologies whose domains fall within the life sciences. These ontologies are primarily being used to annotate biomedical data and literature in order to facilitate improved information exchange. With the growth in ontology usage, reconciliation between those that overlap in scope gains importance.

Evaluation of general ontology alignment algorithms has benefited immensely from the standard-setting benchmark – OAEI [84]. The annual competition evaluates algorithms along a number of tracks each of which contains a set of ontology pairs. In addition to the real-world test cases, the competition emphasizes on comparison tracks by using test pairs that are modifications of single ontology pair in order to systematically identify the strengths and weaknesses of the alignment algorithms. One of these tracks involves aligning the ontology of the adult mouse anatomy with the human anatomy portion of NCI thesaurus [30], while another seeks to align the foundational model of anatomy (FMA), SNOMED CT and the national cancer institute thesaurus (NCI). However, aligning biomedical ontologies poses its own unique challenges. In particular,

1. Entity names are often identification numbers instead of descriptive names. Hence, the alignment algorithm must rely more on the labels and descriptions associated with the entities, which are expressed differently using different formats.

2. Although annotations using entities of some ontologies such as the gene ontology [5] are growing rapidly, for other ontologies they continue to remain sparse. Consequently, we may not overly rely on the entity instances while aligning biomedical ontologies.
3. Finally, biomedical ontologies tend to be large with many including over a thousand entities. This motivates the alignment approaches to depend less on “brute-force” steps, and compels assigning high importance to issues related to efficiency and scalability.

Given these specific challenges, I combed through more than 300 ontologies hosted at NCBO [63] and OBO Foundry [4], and created two distinct testbeds for ontology alignment. One testbed is of 50 different large biomedical ontology pairs. Thirty-two ontologies with sizes ranging from a few hundred to tens of thousands of entities constitute the pairs. It serves as an extensive testbed for analyzing the scalability of alignment algorithms. The primary criteria for including a pair in the benchmark was an expectation of a sufficient amount of correspondences between the ontologies in the pair, as determined from NCBO’s BioPortal. In particular, I calculated the ratio of the crowd-sourced correspondences posted in BioPortal for each ontology pair to the largest number of possible correspondences that could exist. I selected the 50 pairs with the largest such ratio. Existing correspondences will serve in the reference alignment, although our analysis reveals that the existing correspondences represent just a small fraction of the total alignment that is possible between two ontologies. This testbed is available for public use at http://thinc.cs.uga.edu/thinclabwiki/index.php/Biomedical_Domain_Benchmark. The second testbed contains 35 ontology pairs which have significant amount of complex concepts within. The ontologies were selected based on having 10% percent or more of complex concepts and a good amount of reference correspondences available in NCBO (10% or more of each ontology’s concepts are present in the reference). This biomedical testbed is available for public use at http://thinc.cs.uga.edu/thinclabwiki/index.php/Modeling_Complex_Concepts.

I provide the list of ontologies participating in both the benchmarks and the ontology pairs in Appendix A. These new benchmarks guide comparative evaluation of alignment algorithms to the context of a key application domain of biomedicine.

7.1 Improvement Using Complex Concepts Modeling

As I mentioned earlier I created a novel testbed for ontology alignment algorithms to evaluate the impact of complex concepts modeling. This testbed contains 35 ontology pairs organizing knowledge in various biomedical domains. The ontologies were selected based on having 10% percent or more of complex concepts and a good amount of reference correspondences available in NCBO (10% or more of each ontology's concepts are present in the reference). I analyze the improvements in precision and recall along with the associated trade off in the runtime by modeling complex concepts in various alignment algorithms using this testbed.

The performance of (a) **Falcon-AO**, (b) **LogMap**, and (c) **Optima+** with complex concept modeling and in their default mode on the biomedical testbed is shown in Fig. 7.1. For the large ontology pairs in this novel biomedical testbed, all the three algorithms ((a) **Falcon-AO**, (b) **LogMap**, and (c) **Optima+**) benefit from modeling the complex concepts. The enhanced **Falcon-AO** identified a total of 88 less false positive correspondences thus improving precision significantly as shown in Fig. 7.1(a). It found 2 additional correct correspondences resulting in a small increase in overall recall of 0.12%. However, it also identified 4 more false positives. The overall improvement in F-measure for **Falcon-AO** to 31% (precision=48%, recall=23%) is significant (Student's paired t-test, $p < 0.05$). The enhanced **Logmap** pruned 81 false positives in addition to finding 3 more correct correspondences (see Fig. 7.1(b)). This increased its precision to 62% and recall to 35%, both of which increases are significant ($p < 0.05$). In particular, modeling complex concepts improved the F-measure of aligning the *OPL*, *ERO* pair by 4%. Note that **Logmap** is designed for aligning large biomedical ontologies. The enhanced **Optima+** identified a total of 74 less false positives in addition to finding 9 more correct correspondences. **Optima+** generates more useful samples by modeling complex concepts, which improves the recall noticeably for some of

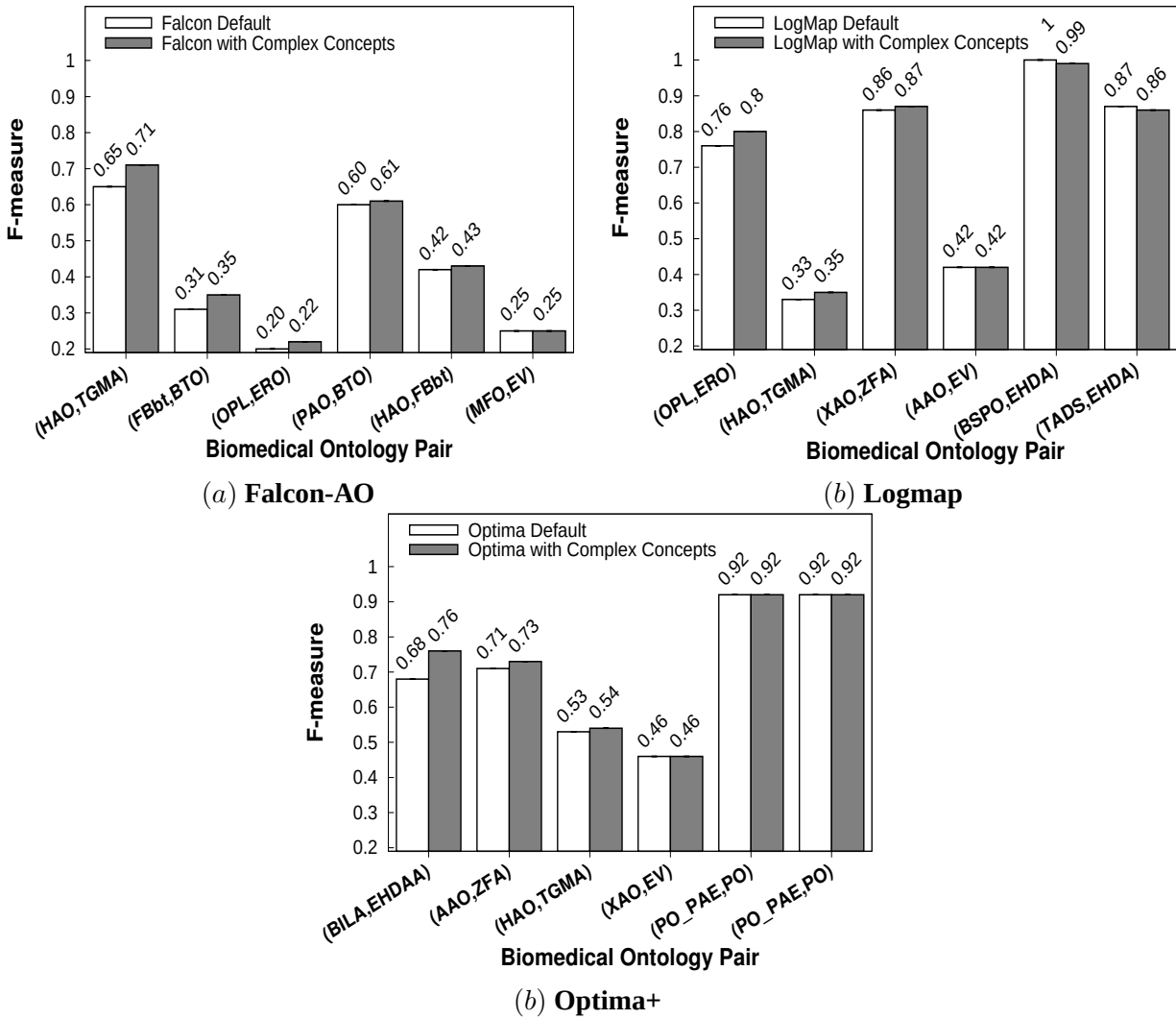


Figure 7.1: Performance on the biomedical testbed. (a) Falcon-AO, (b) LogMap, and (c) Optima+ with complex concept modeling exhibits significantly improved precision (Student's paired t-test, $p < 0.05$) resulting in improved F-measure. I ran the algorithms on all 35 pairs of which I show the 3 ontology pairs that exhibit the highest and lowest differences in F-measure. Ontology names are NCBO abbreviations.

the pairs; for example, the *BILA*, *EHDA* pair gained recall by 6%. The overall F-measure improved by 4%. This improvement in F-measure to 56% (precision=55%, recall=56%) on the biomedical testbed is significant ($p < 0.01$).

Falcon-AO's total run time of 20 minutes for aligning the entire biomedical testbed increases threefold to 57 minutes due to including complex concepts. **Optima+** with complex concepts took 25.6 hours compared to 18 hours consumed by the default resulting in a 1.5 times increase. **Logmap** with complex concepts modeled took 1.8 hours compared to 0.5 hour by the default, resulting in more than a threefold increase. The overhead associated with modeling and computing the similarity of complex concepts is evident in all three algorithms on the large biomedical testbed, which contain a large number of complex concepts. Consideration of the complex concepts affects the structural matching. The time complexity of the structural matchers of these algorithms is exponential in the size of the input ontologies, which exacerbated the total execution time. **Logmap** displays a particularly high increase in execution time because modeling complex concepts significantly increases the number of candidate correspondences that are considered. In the case of **Falcon-AO**, GMO takes more iterations to converge when complex concepts are modeled. As **Optima+** performs a limited amount of structural matching focusing on the class hierarchy only, its increase in execution time is relatively less compared to other algorithms.

7.2 Evaluating Using BCD Enhanced Algorithms

We sought to align the pairs in our first biomedical ontology alignment testbed using the BCD-enhanced representative algorithms. The obtained alignments are evaluated using the existing correspondences previously submitted to BioPortal; the reference alignments between the pairs are likely incomplete. A secondary objective is to discover new correspondences between the ontologies and submit them to NCBO's BioPortal for curation.

Informed by the experimentation described in Section 5.5, blocks for the BCD in **Falcon-AO** were formed using subtree-based partitioning of one ontology and ordered as they were created. Blocks in **OLA** were formed similarly though both ontologies were partitioned while blocks in **Optima** were formed by partitioning both ontologies on the basis of the height of the entities and ordered from leaves to root. The final recall and total execution time for all the pairs successfully aligned within an arbitrary window of 5 hours per pair by the BCD-enhanced algorithms are shown

in Fig. 7.2. We point out that BCD makes the algorithms efficient but does not explicitly promote scalability. In other words, while it reduces the time to convergence it does not provide a way to manage the memory in order to align large ontologies.

OLA with BCD failed to align a single pair within our time window. **Optima** enhanced with BCD aligned 26 pairs each within the window compared to just 8 pairs without BCD. **Falcon-AO** aligned all 50 pairs each within the time window compared to 21 previously. **Optima** produced a recall of 58% for the pairs it aligned while **Falcon-AO** generated a recall of 49% over all the pairs. For the sake of completeness, we utilized a recent scalable algorithm called LogMap [47] specifically designed for aligning very large biomedical ontologies, on our new testbed. LogMap, a non-iterative algorithm, aligned all the pairs within an hour of total execution time and produced a recall of 51% for all the pairs. For the 26 pairs that **Optima** aligned, LogMap exhibited a recall of 55%, which is less than that of **Optima**. Consequently, iterative techniques continue to be among the best in the quality of the obtained alignment including very large ontology pairs. This motivates ways of making them efficient, such as BCD, and more scalable.

Finally, we submitted 15 new correspondences between entities in the pairs of the testbed to NCBO for curation and publication. These are nontrivial correspondences that were discovered by both algorithms, not present in the reference alignments and appropriately validated by us.

I demonstrated the benefits of BCD toward aligning pairs in a new biomedical ontology testbed. Due to the large number of ontologies in biomedicine, there is a critical need for ontology alignment in this vast domain. I believe that this community benchmark could potentially drive future research toward pragmatic ontology alignment. In that line, I demonstrate in the next section, that existing alignment techniques could scale up to these large biomedical ontologies using the MapReduce approach I presented earlier.

7.3 Scaling Using MapReduce Paradigm

I study the impact of distributing ontology alignment using MapReduce in terms of average speedup of the alignment time and its impact on the quality of the alignment. Specifically, I

[illegible]

compare the total execution time when using MapReduce with the time required by the default setup of each alignment algorithm for aligning batches of ontology pairs together. In order to evaluate whether my formulation in MapReduce is properly distributed, I measure the speedup obtained as I allocate an increasing number of nodes to **Hadoop** for each algorithms. For this study, I utilize the four representative alignment algorithms: **Falcon-AO**, **Logmap**, **Optima+**, and **YAM++** described in Section 6.1.

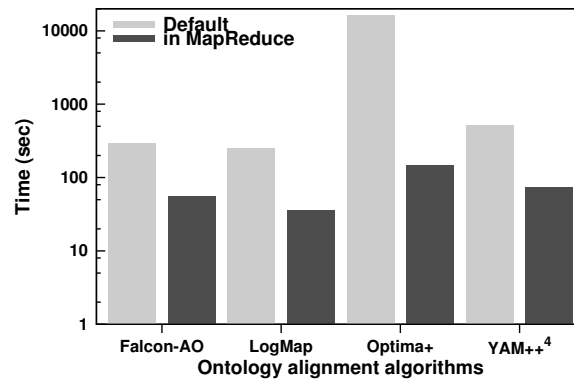


Figure 7.3: Average execution times of **Falcon-AO**, **Logmap**, **Optima+**, and **YAM++** in their original form on a single node and using MapReduce, for aligning the biomedical ontologies from NCBO. Note, that the time axes are in log scale.

I use the same experimental setup as before – **Hadoop** cluster of 12 CentOS 6.3 systems, each with 24 2.0GHz Intel Xeon processors and memory limited to a maximum of 2GB per task in each node. All the ontologies I use have cryptic ids as the names of the entities, but labels are descriptive. Because **Falcon-AO** heavily relies on names to identify correspondences, I configure it to use entity labels as well; **Optima+**, **Logmap**, and **YAM++** automatically adjust by analyzing the ontology. All timing results are averages of 3 runs; I observed very small variances in the execution times.

In Fig. 7.3, I show the average execution time consumed by **Falcon-AO**, **Logmap**, **Optima+**, and **YAM++** in batch aligning biomedical ontology pairs from the large biomedical testbeds, in their default form on a single node and with MapReduce in the **Hadoop** framework. MapReduce demonstrated significant speedup when aligning the batch of biomedical ontology pairs using the four representative alignment algorithms. For the biomedical testbed, **Falcon-AO** and **Logmap**

with MapReduce achieved a speedup of 5, while **YAM++** gained 7. **Optima+** gained a speedup of 110 in the MapReduce setup. While **Logmap** is designed to be the scalable, distributed alignment of a set of pairs using **Logmap** demonstrates significant speedup.

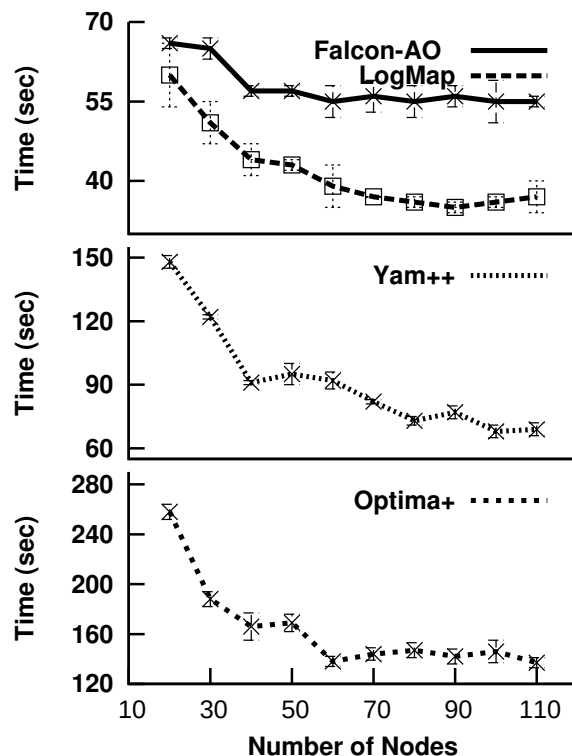


Figure 7.4: The plot demonstrates the exponential decaying of average total execution time with increasing number of nodes by **Falcon-AO**, **Logmap**, **Optima+**, and **YAM++** for biomedical ontologies. Note, the average execution time gradually converges to a minimum time. However, the minimum number of nodes required to attain this convergence differ between algorithms and data-sets.

As before (Chapter 6), to analyze the maximum speedup the MapReduce approach could offer for batch aligning the set of biomedical ontology pairs and the minimum number of nodes required to achieve it, I gradually increased the number of nodes allocated to MapReduce and measured the average execution time of alignment. The average execution time to align the biomedical testbed using MapReduce with increasing number of nodes is shown in Fig. 7.4. I observed the similar trend in execution time as before. The execution time decreases exponentially with an increasing number of nodes until it reaches a minimum. For a given testbed, the number of nodes required to achieve this minimum is upper bounded by the total number of alignment subproblems. On

the other hand, the lower bound of the minimum time required by MapReduce to align a set of ontology pairs is the longest time required for any subproblem and the time required for initialization and communications of MapReduce. I observed that the minimum number of nodes required to reach the minimum execution time varies between using different algorithms and data-sets. This is because, execution times required for aligning subproblems vary between algorithms.

CHAPTER 8

CONCLUSIONS AND FUTURE WORK

Crucial challenges for ontology alignment algorithms involve improving the correctness and completeness of the alignment, scaling to large ontologies and performing the alignment in a reasonable amount of time without compromising on the quality of the alignment. Importantly, emerging applications of ontology alignment such as semantic Web services and search bring new emphasis on alignment execution time and scalability. In this dissertation I presented novel and general algorithms and insights for complete, efficient, and scalable alignment of large ontologies and evaluated their performances using several ontology alignment algorithms and multiple sets of ontology pairs from various domains.

8.1 Conclusions

Many alignment algorithms heavily rely on the lexical attributes of named concepts, with some exploiting the ontology structure as well. However, complex concepts are either not considered or modeled naively in many of the current ontology alignment algorithms, thereby producing possibly an incomplete alignment. I introduced axiomatic and graphical canonical forms for modeling value and cardinality restrictions and Boolean combinations, and presented a way of measuring the similarity between these complex concepts in their canonical forms in Chapter 4. We observed that different types of value restrictions could be modeled uniformly, thereby allowing an axiomatic canonical form in OWL 2's structural specification and derived an equivalent RDF graph-based canonical form. Similarly, canonical forms were provided for different cardinality restrictions and the various Boolean combinations. This allowed us to improve ontology alignment by canonicalizing the complex concepts often present in the ontology and providing a simple way to measure

the similarity between the anonymous classes. I showed how our approach may be integrated in multiple ontology alignment algorithms. My results indicate a significant improvement in the F-measure of the alignment produced by these algorithms. However, this improvement is achieved at the expense of increased run time due to the additional concepts modeled. This enables the alignment algorithms towards a possibly complete alignment. Ideally, we seek to match composite complex concepts of different types such as one involving a value restriction and another containing a Boolean combination if they are semantically equivalent. Therefore, a single canonical representation that would identify the same concept despite being differently defined is preferred. This is challenging and requires robust DL inferencing. Here, we provided separate canonical representations for three types of complex concepts, which is a first step toward this goal. To the best of our knowledge, this is the first effort in its explicit focus on modeling complex concepts for inclusion in the ontology alignment process.

Existing algorithms frequently augment syntactic matching with WordNet based semantic matching to improve their performance. While the advantage of using WordNet seems trivial, I present an analysis on the utility of WordNet for ontology alignment in the context of the reduction in precision and increase in execution time in Chapter 3. My results demonstrate that although using WordNet in addition to syntactic string-based similarity measures does improve the quality of the alignment in many cases, it does so after consuming significantly more time. Furthermore, the precision of the alignment typically reduces leading to much reduced improvement in F-measure. I also reported on many ontology pairs where WordNet did not improve on the final recall or F-measure, but consumed more time. Clearly, the utility of WordNet is questionable in these cases. I analyzed the ontologies for which using WordNet did not improve the performance, and provided a few rules of thumb related to characteristics of ontologies for which WordNet should be utilized cautiously. Based on these results and analyses, my recommendation to the ontology alignment research community is not to discourage the use of WordNet but to allow WordNet usage within the alignment process to be *optional*, and its use may be recommended after analyzing the characteristics of the ontologies.

While techniques for scaling automated alignment to large ontologies have been previously proposed, there is a general absence in the effort to *speed up* the alignment process. Chapter 5 presented a novel approach based on BCD to increase the speed of convergence of an important class of alignment algorithms with no observed adverse effect on the final alignments. Here, I demonstrated this technique in the context of four different iterative algorithms and reported significant reductions in the total execution times of the BCD enhanced algorithms. Also, the integration of BCD improved the alignments' precision by some of the algorithms while retaining the recall. Interestingly, performances of the iterative update and search techniques are impacted differently by various ways of formulating the blocks and the order of processing them. Nevertheless, the approach of grouping alignment variables into blocks based on the height of the participating entities in the ontologies is intuitive and leads to competitive performance. However, BCD does not promote scalability to large ontologies.

Although, improving the efficiency of alignment algorithm helps to speed up them, it is not enough for many of them to scale up for very large ontologies. On the other hand, as new ontologies are submitted or ontologies are updated in the repositories, their alignment with others must be quickly computed. As existing algorithms find it difficult to scale up for very large ontologies, quickly aligning several pairs of ontologies becomes a challenge for these repositories. Chapter 6 projected this problem as one of batch alignment and showed how it may be approached using the distributed computing paradigm of MapReduce, thereby allowing ontology alignment to exploit the proliferating cloud computing paradigm. Importantly, this general approach offers significant speedup for batch alignment of ontology pairs using any alignment algorithm without modifying it. I contrast this approach with the alternative of parallelizing an alignment algorithm itself (referred to as intra-matcher parallelization). Speeding up batch alignment of ontology pairs will benefit ontology repositories such as NCBO that publish maps between all housed ontologies.

The uses of ontologies are popular in the biomedical domain, where a significant number of ontology repositories have been built covering different aspects of medical research. Biomedical data and literature are annotated using these ontologies to facilitate improved information

exchange. With the growth in ontology usage, reconciliation between those that overlap in scope gains importance. I created two novel benchmarks using the ontologies hosted at NCBO and OBO Foundry to facilitate the ontology alignment evaluations with large and complex biomedical ontologies. One consists of ontologies with sizes ranging from a few hundreds to tens of thousands of entities constitute the pairs specifically created to evaluate the scalability of ontology alignment algorithms. The second testbed contains 35 ontology pairs where every ontology has significant amount of complex concepts within it. It serves as a testbed to analyze the utility of complex concepts for ontology alignment. The details of formulation of these testbeds are explained in Chapter 7. Both of these testbeds are publicly available for the benefit of the ontology alignment community. I have used these ontologies to evaluate the algorithms previously mentioned and presented the results and analysis in Chapter 7.

I have implemented all the algorithms I presented above and inherited the outcome of my research into the alignment tool **Optima+**. The executable **Optima+** tool and its source code are publicly available for ontology alignment community. The improved implementation of **Optima** known as **Optima+** has participated in OAEI 2012 and ranked second in the important conference track. Altogether, this dissertation presented algorithms and insights for existing algorithms to improve their quality and efficiency of the alignments and scale up for very large ontologies. Also, it projected a key challenge of ontology repositories as batch alignment of ontology pairs and demonstrated that it can be approached using the MapReduce distributed paradigm. Finally, two novel biomedical ontology testbeds were also provided for experimentations and evaluations.

8.2 Future Work

While this dissertation has overcome some of the key challenges of ontology alignment towards complete, efficient and scalable alignment, there are many open avenues for future improvement. Specifically, the presented algorithms and insights can be further explored for improved performance. I outline some of those here.

Complex concept similarities may be utilized in the seed alignment as well. Furthermore, the seed alignment may be refined using inferences drawn from complex concept mapping. A more accurate seed improves the overall performance of the alignment algorithm. As future work, one may integrate complex concepts deeper within the alignment algorithms; say by generating heuristics that utilize the complex concepts to guide the search.

Our study on the utility of WordNet for ontology alignment presented in Chapter 3 could be enhanced by evaluating the utility of WordNet in the context of multiple alignment algorithms and more ways of using WordNet. Also, the rules of thumb provided for ontology alignment users to decide whether WordNet would be worthwhile for a given ontology pair may be automated. As pointed out in this study, existing semantic similarity measures suffer while evaluating similarity between phrases. Efficient and effective semantic similarity measures to evaluate similarity between phrases would help alignment algorithms to improve on their alignment quality.

Chapter 5 presented and analyzed one technique to improve the performances of iterative alignment algorithms. As a future direction more approximation techniques such as simulated annealing could be explored to improve the efficiency of ontology level matchers. Additionally, analyzing the sub-manifolds in the alignment space may help the ontology level matchers to improve both efficiency and quality. The MapReduce approach presented in Chapter 6 could be further explored by analyzing ways of minimizing the loss in alignment quality by exploring other approaches of partitioning. Moreover the partitioning itself may be implemented in parallel to minimize the overhead.

A worthy and important next step in this line of research would be creating an extensive, flexible and configurable MapReduce framework for ontology repositories to perform batch alignment. This framework may allow both element level and ontology level matchers to operate in parallel. This framework could efficiently carry out the batch alignment of ontologies using several algorithms in a distributed computing cluster. Additionally, this framework may provide Webservice APIs for repositories to notify updates and creation of ontologies. Note existing repositories such as NCBO already provide Webservice APIs to submit alignments. This parallel framework could use such APIs to automatically upload alignments back to repositories. It would be immensely

beneficial for ontology repositories and the ontology alignment community. Specifically, such a framework will benefit ontology repositories such as NCBO that publish maps between all housed ontologies.

BIBLIOGRAPHY

- [1] Zharko Aleksovski, Willem Robert Van Hage, and Antoine Isaac. A survey and categorization of ontology-matching cases, 2007.
- [2] Ghazvinian Amir, Noy Natalya, and Musen Mark. Creating mappings for ontologies in biomedicine: simple methods work. *AMIA*, pages 198–202, 2009.
- [3] S. Arimoto. An algorithm for computing the capacity of arbitrary discrete memoryless channels. *IEEE Transactions on Information Theory*, 18(1):14–20, 1972.
- [4] Barry Smith and Michael Ashburner, Cornelius Rosse and Jonathan Bard and William Bug and Werner Ceusters, Louis J. Goldberg, Karen Eilbeck, Amelia Ireland and Christopher J. Mungall and Neocles Leontis, Philippe Rocca-Serra, Alan Ruttenberg, Susanna-Assunta Sansone, Richard H. Scheuermann, Nigam Shah, Patricia L. Whetzel, and Suzanna Lewis. The obo foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature Biotechnology*, 25(11):1251–1255, 2007.
- [5] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock. Gene ontology: tool for the unification of biology. the gene ontology consortium. *Nature genetics*, 25(1):25–29, 2000.
- [6] Gómez-Pérez Asunción. Semantic evaluation at large scale - home, september 2012.
- [7] M. Ba and G. Diallo. Servomap and servomap-lt results for oaei 2012. In *Workshop on Ontology Matching at 11th International Semantic Web Conference (ISWC)*, 2012.

- [8] Franz Baader, Ian Horrocks, and Ulrike Sattler. Description logics as ontology languages for the semantic web. In *Lecture Notes in Artificial Intelligence*, pages 228–248. Springer-Verlag, 2003.
- [9] François Belleau, Marc-Alexandre Nolin, Nicole Tourigny, Philippe Rigault, and Jean Morissette. Bio2rdf: Towards a mashup to build bioinformatics knowledge systems. *Journal of Biomedical Informatics*, 41(5):706–716, October 2008.
- [10] Bethesda. Umls reference manual. <http://www.ncbi.nlm.nih.gov/books/NBK9676/>, 2009.
- [11] Richard E. Blahut. Computation of channel capacity and rate-distortion functions. *IEEE Transactions on Information Theory*, 18:460–473, 1972.
- [12] Jurgen Bock and Jan Hettenhausen. Discrete particle swarm optimisation for ontology alignment. *Information Sciences*, pages 1–22, 2010.
- [13] Olivier Bodenreider and Robert Stevens. Bio-ontologies: current trends and future directions. *Briefings in Bioinformatics*, 7:256–274, 2006.
- [14] Gong Cheng, Weiyi Ge, and Yuzhong Qu. Falcons: searching and browsing entities on the semantic web. In *Proceedings of the 17th international conference on World Wide Web*, pages 1101–1102, 2008.
- [15] Namyoun Choi, Il-Yeol Song, and Hyoil Han. A survey on ontology mapping. *SIGMOD Rec.*, 35(3):34–41, September 2006.
- [16] Isabel F. Cruz, Flavio Palandri Antonelli, and Cosmin Stroe. Agreementmaker: efficient matching for large real-world schemas and ontologies. *Proc. VLDB Endow.*, 2(2), 2009.
- [17] Isabel F. Cruz, Cosmin Stroe, and Matteo Palmonari. Interactive user feedback in ontology matching using signature vectors. In *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering, ICDE '12*, pages 1321–1324, 2012.

- [18] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.
- [19] Anhai Doan, Jayant Madhavan, Pedro Domingos, and Alon Halevy. Ontology matching: A machine learning approach. In *Handbook on Ontologies in Information Systems*, pages 397–416, 2003.
- [20] Prashant Doshi, Ravikanth Kolli, and Christopher Thomas. Inexact matching of ontology graphs using expectation-maximization. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(2):90–106, 2009.
- [21] William F. Dowling and Jean H. Gallier. Linear-time algorithms for testing the satisfiability of propositional horn formulae. *The Journal of Logic Programming*, 1(3):267–284, 1984.
- [22] Marc Ehrig and Steffen Staab. Qom quick ontology mapping. In *The Semantic Web ISWC 2004*, volume 3298 of *Lecture Notes in Computer Science*, pages 683–697, 2004.
- [23] Jérôme Euzenat. Ontology alignment evaluation initiative :: Home, April 2013.
- [24] Jérôme Euzenat, David Loup, Mohamed Touzani, and Petko Valtchev. Ontology alignment with ola. In *Proceedings of the 3rd EON Workshop, 3rd International Semantic Web Conference*, pages 59–68. CEUR-WS, 2004.
- [25] Jeffrey A. Fessler and Alfred O. Hero. Space-alternating generalized expectation-maximization algorithm. *IEEE Transactions on Signal Processing*, 42:2664–2677, 1994.
- [26] Jeffrey A. Fessler and Donghwan Kim. Axial block coordinate descent (abcd) algorithm for X-ray CT image reconstruction. In *Proceedings of Fully 3D Image Reconstruction in Radiology and Nuclear Medicine*, pages 262–265, 2011.
- [27] Fausto Giunchiglia, Pavel Shvaiko, Mikalai Yatskevich, Fausto Giunchiglia, Pavel Shvaiko, and Mikalai Yatskevich. S-match: an algorithm and an implementation of semantic matching. In *European Semantic Web Symposium*, pages 61–75, 2004.

- [28] GlenMazza. Hadoop wiki. <http://wiki.apache.org/hadoop/ProjectDescription>, 2012.
- [29] Semantic Web Company GmbH. Poolparty semantic information management. <http://www.poolparty.biz/>, 2013.
- [30] J. Golbeck, G. Fragoso, F. Hartel, J. Hendler, J. Oberthaler, and B. Parsia. The national cancer institutes thesaurus and ontology. *Journal of web semantics*, 1(1):75–80, 2003.
- [31] Anika Gross, Michael Hartung, Toralf Kirsten, and Erhard Rahm. On matching large life science ontologies in parallel. In *Proceedings of 7th international conference on data integration in the life sciences (DILS)*, volume 6254, pages 35–49, 2010.
- [32] Peter Haase, Holger Lewen, Rudi Studer, Michael Erdmann, and Ontoprise GmbH. The neon ontology engineering toolkit, 2009.
- [33] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November 2009.
- [34] Faycal Hamdi, Brigitte Safar, Chantal Reynaud, and Haifa Zargayouna. Alignment-based partitioning of large-scale ontologies. In *Advances in Knowledge Discovery and Management*, volume 292, pages 251–269, 2009.
- [35] Md’ Seddiqui Hanif and Masaki Aono. Anchor-flood: results for OAEI 2009. In *Workshop on Ontology Matching at 8th International Semantic Web Conference*, pages 127–134, 2009.
- [36] Jonathan Hayes and Claudio Gutierrez. Bipartite graphs as intermediate model for RDF. In *Proceedings of the 3rd International Semantic Web Conference (ISWC)*, Lecture Notes in Computer Science, pages 47–61. Springer Berlin / Heidelberg, 2004.
- [37] Alfred O. Hero and Jeffrey A. Fessler. Asymptotic convergence properties of em-type algorithms. Technical report, Department of EECS, Univ. of Michigan, Ann Arbor, MI, 1993.

- [38] Alfred Horn. On Sentences Which are True of Direct Unions of Algebras. *The Journal of Symbolic Logic*, 16(1):14–21, 1951.
- [39] Wei Hu, Ningsheng Jian, Yuzhong Qu, and Yanbing Wang. GMO: A graph matching for ontologies. In *K-Cap Workshop on Integrating Ontologies*, pages 43–50, 2005.
- [40] Wei Hu, Yuzhong Qu, and Gong Cheng. Matching large ontologies: A divide-and-conquer approach. *Data Knowl. Eng.*, 67(1):140–160, 2008.
- [41] Wei Hu, Yuanyuan Zhao, and Yuzhong Qu. Partition-based block matching of large class hierarchies. In *Proceedings of the 1st Asian Semantic Web Conference (ASWC)*, pages 72–83, 2006.
- [42] Todd C. Hughes and Benjamin C. Ashpole. The semantics of ontology alignment. In *Information Interpretation and Integration Conference (I3CON)*, 2004.
- [43] Robert Isele, Anja Jentzsch, and Christian Bizer. Silk server - adding missing links while consuming linked data. In *COLD*, pages 23–31, 2010.
- [44] Aminul Islam and Diana Inkpen. Semantic text similarity using corpus-based word similarity and string similarity. *ACM Trans. Knowl. Discov. Data*, 2:10:1–10:25, 2008.
- [45] Yves R. Jean-Mary, E. Patrick Shironoshita, and Mansur R. Kabuka. Ontology matching with semantic verification. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3):235–251, 2009.
- [46] Ningsheng Jian, Wei Hu, Gong Cheng, and Yuzhong Qu. Falcon-AO: Aligning ontologies with Falcon. In *K-Cap Workshop on Integrating Ontologies*, pages 87–93, 2005.
- [47] Ernesto Jiménez-Ruiz and Bernardo Cuenca Grau. LogMap: Logic-Based and Scalable Ontology Matching. In *Proc. of the 10th International Semantic Web Conference (ISWC’11)*, volume 7031, pages 273–288, 2011.

- [48] Toralf Kirsten, Anika Gross, Michael Hartung, and Erhard Rahm. Gomma: a component-based infrastructure for managing and analyzing life science ontologies and their evolution. *Journal of Biomedical Semantics*, 2:6, 2011.
- [49] P. Lambrix, H. Tan, V. Jakoniene, and L. Stromback. *Biological ontologies In Semantic Web: Revolutionizing Knowledge Discovery in the Life Sciences*, pages 85–99. Springer, 2007.
- [50] Holger Lewen. H.: Cupboard - a place to expose your ontologies to applications and the community. In *Proceedings of the ESWC 2009, Heraklion, Greece*, pages 913–918, 2009.
- [51] Yi Li, Juanzi Li, and Jie Tang. RiMOM: Ontology alignment with strategy selection. In *6th International and 2nd Asian Semantic Web Conference (ISWC2007+ASWC2007)*, pages 51–52, November 2007.
- [52] D. Lin. An information-theoretic definition of similarity. In *ICML*, pages 296–304, 1998.
- [53] F. Lin and K. Sandkuhl. A survey of exploiting wordnet in ontology matching. In M Bramer, editor, *Artificial Intelligence in Theory and Practice II*, volume 276, pages 341–350, 2008.
- [54] Jose Luis. Benchmark test library. <http://oaei.ontologymatching.org/2012/benchmarks/index.html>, 2012.
- [55] R. Mandala, T. Tokunaga, and H. Tanaka. Improving information retrieval system performance by combining different text-mining techniques. *Intelligent Data Analysis*, 4(6):489–511, 2000.
- [56] M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Comp. Linguistics*, 19(2):313–330, 1993.
- [57] B. McBride and R. Guha. Rdf vocabulary description language 1.0: Rdf schema. Technical report, W3C, 2004.
- [58] D. McGuinness and F. Harmelen. Owl web ontology language overview. Technical report, W3C, 2004.

- [59] D. McGuinness and F. Harmelen. Owl 2 web ontology language document overview. Technical report, W3C, 2009.
- [60] Christian Meilicke. *Alignment Incoherence in Ontology Matching*. PhD thesis, University of Mannheim, 2011.
- [61] Sergey Melnik, Hector Garcia-molina, and Erhard Rahm. Similarity flooding: A versatile graph matching algorithm. In *ICDE: Int. Conference on Data Engineering*, pages 117–128, 2002.
- [62] G. A. Miller. Wordnet: A lexical database for english. In *CACM*, pages 39–41, 1995.
- [63] Mark A. Musen, Natalya Fridman Noy, Nigam H. Shah, Patricia L. Whetzel, Christopher G. Chute, Margaret-Anne D. Storey, and Barry Smith. The national center for biomedical ontology. *JAMIA*, 19(2):190–195, 2012.
- [64] 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.
- [65] Yurii Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.
- [66] DuyHoa Ngo and Zohra Bellahsene. Yam++ : A multi-strategy based approach for ontology matching task. In *Knowledge Engineering and Knowledge Management*, volume 7603, pages 421–425, 2012.
- [67] Peter F. Patel-Schneider and Boris Motik. Owl 2 web ontology language mapping to rdf graphs. http://www.w3.org/2007/OWL/wiki/Mapping_to_RDF_Graphs. W3C OWL wiki.
- [68] Heiko Paulheim. On applying matching tools to large-scale ontologies. In *OM*, pages 214–218, 2008.

- [69] Ted Pedersen and Siddharth Patwardhan. Wordnet::similarity - measuring the relatedness of concepts. In *AAAI*, pages 1024–1025, 2004.
- [70] János D. Pintér. Yair censor and stavros a. zenios, parallel optimization – theory, algorithms, and applications. *Journal of Global Optimization*, 16:107–108, 2000.
- [71] Yuzhong Qu, Wei Hu, and Gong Cheng. Constructing virtual documents for ontology matching. In *Proceedings of the 15th international conference on World Wide Web, WWW '06*, pages 23–31, 2006.
- [72] Erhard Rahm. Towards large-scale schema and ontology matching. In *Schema Matching and Mapping*, pages 3–27. Springer, 2011.
- [73] Jinghai Rao and Xiaomeng Su. Toward the composition of semantic web services. In *GCC (2)*, pages 760–767, 2003.
- [74] Stuart J. Russell and Peter Norvig. *Artificial Intelligence - A Modern Approach (3. internat. ed.)*. Pearson Education, 2010.
- [75] Marta Sabou, Martin Dzbor, Claudio Baldassarre, Sofia Angeletou, and Enrico Motta. Watson: A gateway for the semantic web. In *Poster session of the European Semantic Web Conference, ESWC*, pages 11–15, 2007.
- [76] Ankan Saha and Ambuj Tewari. On the non-asymptotic convergence of cyclic coordinate descent methods. *SIAM Journal on Optimization*, ();, 2013. In press.
- [77] Md. Hanif Seddiqui and Masaki Aono. An efficient and scalable algorithm for segmented alignment of ontologies of arbitrary size. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7:344–356, 2009.
- [78] Pavel Shvaiko and Jérôme Euzenat. A Survey of Schema-Based Matching Approaches Journal on Data Semantics IV. *Journal on Data Semantics IV*, 3730:146–171, 2005.

- [79] Pavel Shvaiko, Jérôme Euzenat, Fausto Giunchiglia, Bin He, Ming Mao, Natalya Noy, and Heiner Stuckenschmidt, editors. *International Workshop on Ontology Matching*, volume 304. CEUR-WS.org, 2007.
- [80] Pavel Shvaiko, Jérôme Euzenat, Fausto Giunchiglia, and Heiner Stuckenschmidt, editors. *International Workshop on Ontology Matching*, volume 431. CEUR-WS.org, 2008.
- [81] Pavel Shvaiko, Jérôme Euzenat, Fausto Giunchiglia, Heiner Stuckenschmidt, Ming Mao, and Isabel F. Cruz, editors. *International Workshop on Ontology Matching*, volume 689. CEUR-WS.org, 2010.
- [82] Pavel Shvaiko, Jérôme Euzenat, Tom Heath, Christoph Quix, Ming Mao, and Isabel F. Cruz, editors. *International Workshop on Ontology Matching*, volume 551. CEUR-WS.org, 2009.
- [83] Pavel Shvaiko, Jérôme Euzenat, Tom Heath, Christoph Quix, Ming Mao, and Isabel F. Cruz, editors. *International Workshop on Ontology Matching*, volume 814. CEUR-WS.org, 2011.
- [84] Pavel Shvaiko, Jérôme Euzenat, Anastasios Kementsietsidis, Ming Mao, Natalya Noy, and Heiner Stuckenschmidt, editors. *International Workshop on Ontology Matching*, volume 946. CEUR-WS.org, 2012.
- [85] Pavel Shvaiko, Jérôme Euzenat, Natalya Fridman Noy, Heiner Stuckenschmidt, V. Richard Benjamins, and Michael Uschold, editors. *International Workshop on Ontology Matching*, volume 225. CEUR-WS.org, 2006.
- [86] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. In *JMB*, volume 147, pages 195–197, 1981.
- [87] Giorgos Stoilos, Giorgos Stamou, and Stefanos Kollias. A String Metric for Ontology Alignment. In *ISWC 2005*, pages 624–637, 2005.

- [88] Suzette K. Stoutenburg, Jugal Kalita, Kaily Ewing, and Lisa M. Hines. Scaling alignment of large ontologies. *International Journal of Bioinformatics Research and Applications*, 6:384–401, 2010.
- [89] Uthayasanker Thayasivam and Prashant Doshi. Optima results for oaei 2011. In *Proceedings of the Workshop on Ontology Matching at 10th International Semantic Web Conference (ISWC)*, pages 204–211, 2012.
- [90] Uthayasanker Thayasivam and Prashant Doshi. Optima+ results for oaei 2012. In *Proceedings of the Workshop on Ontology Matching at 11th International Semantic Web Conference (ISWC)*, pages 181–188, 2012.
- [91] P. Tseng. Convergence of block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, 109:475–494, 2001.
- [92] Kim Viljanen, Jouni Tuominen, Eetu Makela, and Eero Hyvonen. Normalized access to ontology repositories. In *Proceedings of the 2012 IEEE Sixth International Conference on Semantic Computing*, pages 109–116, 2012.
- [93] Peng Wang and Baowen Xu. Lily: Ontology alignment results for OAEI 2008. In *Workshop on Ontology Matching at 7th International Semantic Web Conference (ISWC)*, 2009.
- [94] Peng Wang, Yuming Zhou, and Baowen Xu. Matching large ontologies based on reduction anchors. In *22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2343–2348, 2011.
- [95] Tom White. *Hadoop: The Definitive Guide*. O’Reilly Media, Inc., 1st edition, 2009.
- [96] M. Yatskevich and F. Giunchiglia. Element level semantic matching using wordnet. Technical report, University of Trento, 2007.

- [97] Hang Zhang, Wei Hu, and Yuzhong Qu. Vdoc+: a virtual document based approach for matching large ontologies using mapreduce. *Journal of Zhejiang University - Science C*, 13(4):257–267, 2012.

Appendix A

ONTOLOGIES USED IN OUR EVALUATIONS

We used a comprehensive testbed of several ontology pairs – some of which are very large – spanning multiple domains in our performance evaluations and experiments. We used ontology pairs from the OAEI competition in its recent version, 2012, as the testbed for our evaluation. In addition to that we created couple of ontology alignment testbeds using ontologies from the NCBO.

Table A.1: Ontologies from OAEI’s benchmark and conference tracks participating in our evaluation and the number of named classes, complex concepts and properties in each.

Ontology	Named Classes	Complex Concepts	Properties
101	37	99	70
205	36	99	69
301	16	41	40
302	14	11	30
303	57	151	72
304	41	73	49
ekaw	74	27	33
sigkdd	50	15	28
iasted	150	128	41
cmt	30	11	59
edas	104	30	50
confOf	39	42	36
conference	60	33	64

Among the OAEI tracks, we focus on the test cases that involve real-world ontologies for which the reference (true) alignment was provided by OAEI. These ontologies were either acquired from the Web or created independently of each other and based on real-world resources. This includes all ontology pairs in the 300 range of the benchmark, which relate to *bibliography*, expressive ontologies in the *conference* track all of which structure knowledge related to conference organization,

and the *anatomy* track, which consists of large ontologies from life sciences, describing anatomy of adult mouse and human. We list the benchmark and conference track ontologies participating in our evaluation in Table A.1 and provide an indication of their sizes.

Similarly, the large ontologies from anatomy, library and large biomedical tracks are listed in Table A.2. Here, we also provide an indication of their sizes. These large ontologies are used in our performance evaluation on batch alignment of a set of ontology pairs using MapReduce. The details of this experiment and results are provided in Section 6.5.

Table A.2: Large ontologies from OAEI 2012 used in our evaluations and the number of named classes and properties in each of them.

Ontology	Named Classes	Properties
Life Science Domain		
mouse anatomy	2,744	2
human anatomy	3,304	3
Library Domain		
STW taxonomy	6,575	0
TheSoz taxonomy	8,376	0
Biomedical Domain		
Foundational Model of Anatomy (FMA)	78,989	186
National Cancer Institute Thesaurus (NCI)	66,724	200
SNOMED Clinical Terms (SNOMED)	122,464	41

Biomedical ontologies bring unique challenges to the ontology alignment problem. Moreover, there is an explicit interest for ontologies and ontology alignment in the domain of biomedicine. Consequently, we present a new biomedical ontology alignment testbed, which provides an important application context to the alignment research community. Due to the large sizes of biomedical ontologies, the testbed could serve as a comprehensive large ontology benchmark. Existing correspondences submitted to NCBO may serve as the reference alignments for the pairs, although our analysis reveals that these maps represent just a small fraction of the total alignment that is possible between two ontologies. Consequently, new correspondences that are discovered during benchmarking may be submitted to NCBO for curation and publication.

In order to create the testbed, we combed through more than 300 ontologies hosted at NCBO and OBO Foundry, and isolated a benchmark of 50 different biomedical ontology pairs. Thirty-

Table A.3: Selected ontologies from NCBO in the biomedical ontology alignment testbed 1 and the number of named classes and properties in each. Notice that this data set includes very large ontologies. NCBO abbreviations for these ontologies are also provided.

Ontology	Named Classes	Data Properties	Object Properties
Bilateria anatomy (BILA)	114	0	9
Common Anatomy Reference Ontology (CARO)	50	0	9
Plant Growth and Development Stage (PO_PSDA)	282	2	0
FlyBase Controlled Vocabulary (FBcv)	821	0	10
Spatial Ontology (BSPO)	129	0	9
Amphibian gross anatomy (AAO)	1603	0	9
Anatomical Entity Ontology (AEO)	238	0	6
Cereal plant gross anatomy (GR_CPGA)	1270	7	0
Plant Anatomy (PO_PAE)	1,270	6	0
Subcellular Anatomy Ontology (SAO)	821	0	85
Xenopus anatomy and development (XAO)	1,041	0	10
vertebrate Homologous Organ Groups (sHOG)	1,184	0	7
Hymenoptera Anatomy Ontology (HAO)	1,930	4	4
Teleost Anatomy Ontology (TAO)	3,039	0	9
Tick gross anatomy (TADS)	628	0	0
Zebrafish anatomy and development (ZFA)	2,788	5	0
Medaka fish anatomy and development (MFO)	4,358	0	6
BRENDA tissue / enzyme source (BTO)	5,139	4	9
Expressed Sequence Annotation for Humans (eVOC)	2274	0	7
Drosophila gross anatomy (FBbt)	7,797	0	10
Phenotypic quality (PATO)	2,281	24	0
Uber anatomy ontology (UBERON)	7,294	112	0
Fly taxonomy (FBsp)	6,599	0	0
Protein modification (MOD)	1,338	4	0
Human developmental anatomy (EHDA)	2,314	0	7
Human developmental anatomy timed version (EHDA)	8,340	0	7
Plant Ontology (PO)	1,585	7	0
NIF Cell (NIF_Cell)	2,703	73	5
Mouse adult gross anatomy (MA)	2,982	1	6
Mosquito gross anatomy (TGMA)	1,864	3	0
Ontology for Biomedical Investigations (OBI)	3,537	102	6
Chemical entities of biological interest (CHEBI)	31,470	9	0

two ontologies with sizes ranging from a few hundreds to tens of thousands of entities constitute the pairs, and are listed in Table A.3. We provide a snapshot of the benchmark in Table A.4. Our

primary criteria for including a pair in the benchmark was the presence of a sufficient amount of correspondences between the ontologies in the pair, as determined from NCBO’s BioPortal. We briefly describe the steps followed in creating the testbed:

1. We selected ontologies, which exist in either OWL or RDF models.
2. Next, we paired the ontologies and ordered the pairs by the percentage of available correspondences. This was calculated as the ratio of correspondences that exist in BioPortal for the pair of ontologies under consideration divided by the product of the number of entities in both the ontologies.
3. Top 100 ontology pairs were selected, followed by ordering the pairs based on their joint sizes.
4. We created 5 bins of equal sizes and randomly sampled each bin with a uniform distribution, to obtain the final 50 pairs.

Table A.4: The biomedical ontology pairs in our testbed 1 sorted in terms of $|V_1| \times |V_2|$. This metric is illustrative of the complexity of aligning the pair.

Biomedical Domain Testbed 1		
Ontology $\mathcal{O}_1$	Ontology $\mathcal{O}_2$	$V_1 \times V_2$
Bilateria anatomy	Human developmental anatomy	263796
Common Anatomy Reference Ontology	Human developmental anatomy	417000
Plant Growth and Development Stage	Plant Ontology	446970
Bilateria anatomy	Human developmental anatomy	950760
FlyBase Controlled Vocabulary	Cereal plant gross anatomy	1042670
Spatial Ontology	Human developmental anatomy	1075860
FlyBase Controlled Vocabulary	Plant Ontology	1301285
Continued on next page		

Ontology $\mathcal{O}_1$	Ontology $\mathcal{O}_1$	$V_1 \times V_2$
Amphibian gross anatomy	Xenopus anatomy and development	1668723
Anatomical Entity Ontology	Human developmental anatomy	1984920
Cereal plant gross anatomy	Plant Ontology	2012950
Plant Anatomy	Plant Ontology	2012950
SAO	NIF Cell	2219163
Xenopus anatomy and development	eVOC	2367234
vertebrate Homologous Organ Groups	eVOC	2692416
Xenopus anatomy and development	Zebrafish anatomy and development	2902308
Xenopus anatomy and development	Teleost Anatomy Ontology	3163599
vertebrate Homologous Organ Groups	Mouse adult gross anatomy	3530688
Hymenoptera Anatomy Ontology	Mosquito gross anatomy	3597520
Teleost Anatomy Ontology	vertebrate Homologous Organ Groups	3598176
Amphibian gross anatomy	eVOC	3645222
Amphibian gross anatomy	Zebrafish anatomy and development	4469164
Amphibian gross anatomy	Teleost Anatomy Ontology	4871517
Tick gross anatomy	Human developmental anatomy	5237520
Plant Anatomy	BRENDA tissue / enzyme source	6526530
Xenopus anatomy and development	Uber anatomy ontology	7593054
Zebrafish anatomy and development	Teleost Anatomy Ontology	8472732
vertebrate Homologous Organ Groups	Uber anatomy ontology	8636096
Xenopus anatomy and development	Human developmental anatomy	8681940
vertebrate Homologous Organ Groups	Human developmental anatomy	9874560
Medaka fish anatomy and development	eVOC	9910092
BRENDA tissue / enzyme source	eVOC	11686086
Continued on next page		

Ontology $\mathcal{O}_1$	Ontology $\mathcal{O}_1$	$V_1 \times V_2$
Amphibian gross anatomy	Uber anatomy ontology	11692282
Amphibian gross anatomy	Human developmental anatomy	13369020
Hymenoptera Anatomy Ontology	Uber anatomy ontology	14077420
Hymenoptera Anatomy Ontology	Drosophila gross anatomy	15048210
Hymenoptera Anatomy Ontology	Human developmental anatomy	16096200
eVOC	Uber anatomy ontology	16586556
Drosophila gross anatomy	eVOC	17730378
eVOC	Human developmental anatomy	18965160
Phenotypic quality	Human developmental anatomy	19023540
Zebrafish anatomy and development	Uber anatomy ontology	20335672
Uber anatomy ontology	Mouse adult gross anatomy	21750708
Zebrafish anatomy and development	Human developmental anatomy	23251920
Fly taxonomy	Ontology for Biomedical Investigations	23340663
Teleost Anatomy Ontology	Human developmental anatomy	25345260
Medaka fish anatomy and development	Human developmental anatomy	36345720
BRENDA tissue / enzyme source	Uber anatomy ontology	37483866
Drosophila gross anatomy	BRENDA tissue / enzyme source	40068783
Protein modification	Chemical entities of biological interest	42106860
BRENDA tissue / enzyme source	Human developmental anatomy	42859260

Complex concepts, which include restrictions and Boolean combinations in OWL ontologies, are often precluded by alignment algorithms. We introduced an approach for modeling different types of complex concepts by introducing axiomatic canonical forms and subsequently deriving

equivalent RDF graphs in canonical forms. Consequently, we created another novel testbed of ontology pairs to evaluate the impact of complex concept modeling in alignment algorithms.

Table A.5: Selected ontologies from NCBO in the biomedical ontology alignment testbed 2 and the number of named classes, anonymous classes and different type of properties in each. We also provide the NCBO abbreviations for these ontologies. Notice that this data set includes very large ontologies.

NCBO Acronym	Ontology	Total Classes	Anonymous Classes	Named Classes
AAO	Amphibian gross anatomy	2668	1059	1609
AEO	Anatomical Entity Ontology	357	101	256
AERO	Adverse Event Reporting ontology	445	162	283
BILA	Bilateria anatomy	160	40	120
BSPO	Spatial Ontology	333	198	135
CHEMINF	Chemical Information Ontology	1268	619	649
ERO	eagle-i research resource ontology	2919	521	2398
FBbt	Drosophila gross anatomy	16826	9023	7803
FBcv	FlyBase Controlled Vocabulary	880	53	827
FLU	Influenza Ontology	1485	741	744
HAO	Hymenoptera Anatomy Ontology	6576	4602	1974
IDO	Infectious Disease Ontology	1036	528	508
KiSAO	Kinetic Simulation Algorithm Ontology	691	466	225
MFO	Medaka fish anatomy and development	8597	4233	4364
OPL	Ontology for Parasite Life Cycle	801	463	338
PO_PAE	Plant Anatomy	2120	844	1276
ProPreO	Proteomics data and process provenance	604	204	400
SAO	Subcellular Anatomy Ontology	1026	263	763
SWO	Software Ontology	3606	2702	904
TADS	Tick gross anatomy	1297	663	634
vHOG	vertebrate Homologous Organ Groups	2371	1181	1190
XAO	Xenopus anatomy and development	2338	1291	1047
MA	Mouse adult gross anatomy	4782	1794	2988
ZFA	Zebrafish anatomy and development	10502	7708	2794
EHDA	Human developmental anatomy	16662	8316	8346
OBI	Ontology for Biomedical Investigations	8239	4690	3549
EHDA	Human developmental anatomy	4655	2335	2320
BTO	BRENDA tissue / enzyme source	7834	2479	5355
GRO_CPGA	Cereal plant gross anatomy	2120	844	1276
PO	Plant Ontology	2621	1030	1591
PATO	Phenotypic quality	2708	412	2296
TGMA	Mosquito gross anatomy	3822	1952	1870
NIF_Cell	NIF Cell	3205	436	2769

The second testbed contains 35 ontology pairs made out of 33 ontologies where each of them have significant amount of complex concepts. The ontologies were selected based on having 10% percent or more of complex concepts and a good amount of reference correspondences available in NCBO (10% or more of each ontology's concepts are present in the reference). I provide the list of ontologies participating in the benchmarks in Table A.5. The biomedical testbed presented in Table A.6 is also available for public use at http://thinc.cs.uga.edu/thinc/labwiki/index.php/Modeling_Complex_Concepts.

Table A.6: The 35 biomedical ontology pairs from our second testbed are listed above using their NCBO acronym. These ontologies contains significant amount of complex concepts within them.

Biomedical Domain Testbed 2	
Ontology $\mathcal{O}_1$	Ontology $\mathcal{O}_2$
AAO	MA
AAO	XAO
AAO	ZFA
AEO	EHDA
AERO	CHEMINF
AERO	FLU
AERO	OBI
BILA	EHDA
BILA	EHDA
BSPO	EHDA
CHEMINF	FLU
CHEMINF	OBI
ERO	OBI
FBbt	BTO
FBcv	GRO_CPGA
FBcv	PO
FLU	OBI
HAO	FBbt
HAO	TGMA
IDO	AERO
KiSAO	PATO
Continued on next page	

Ontology $\mathcal{O}_1$	Ontology $\mathcal{O}_2$
MFO	MA
OPL	AERO
OPL	ERO
OPL	IDO
OPL	OBI
PAO	BTO
PO.PAE	PO
ProPreO	OBI
SAO	NIF_Cell
SWO	AERO
SWO	OBI
TADS	EHDA
vHOG	MA
XAO	ZFA

Appendix B

ADDITIONAL RESULTS ON WORDNET UTILITY

Here, we report distinct trends in the performance of WordNet(WN)-based alignment in comparison with alignment that uses syntactic matching only as detailed in Section 3.3.1. We evaluated the recall and F-measure of the alignment generated by **Optima** when WordNet is integrated and that of the alignment when just the syntactic similarity is used. While we showed the results and discussed the trends for 6 representative ontology pairs out of 23 in Section 3.3.1, the results for the rest of the ontology pairs are given below for completeness.

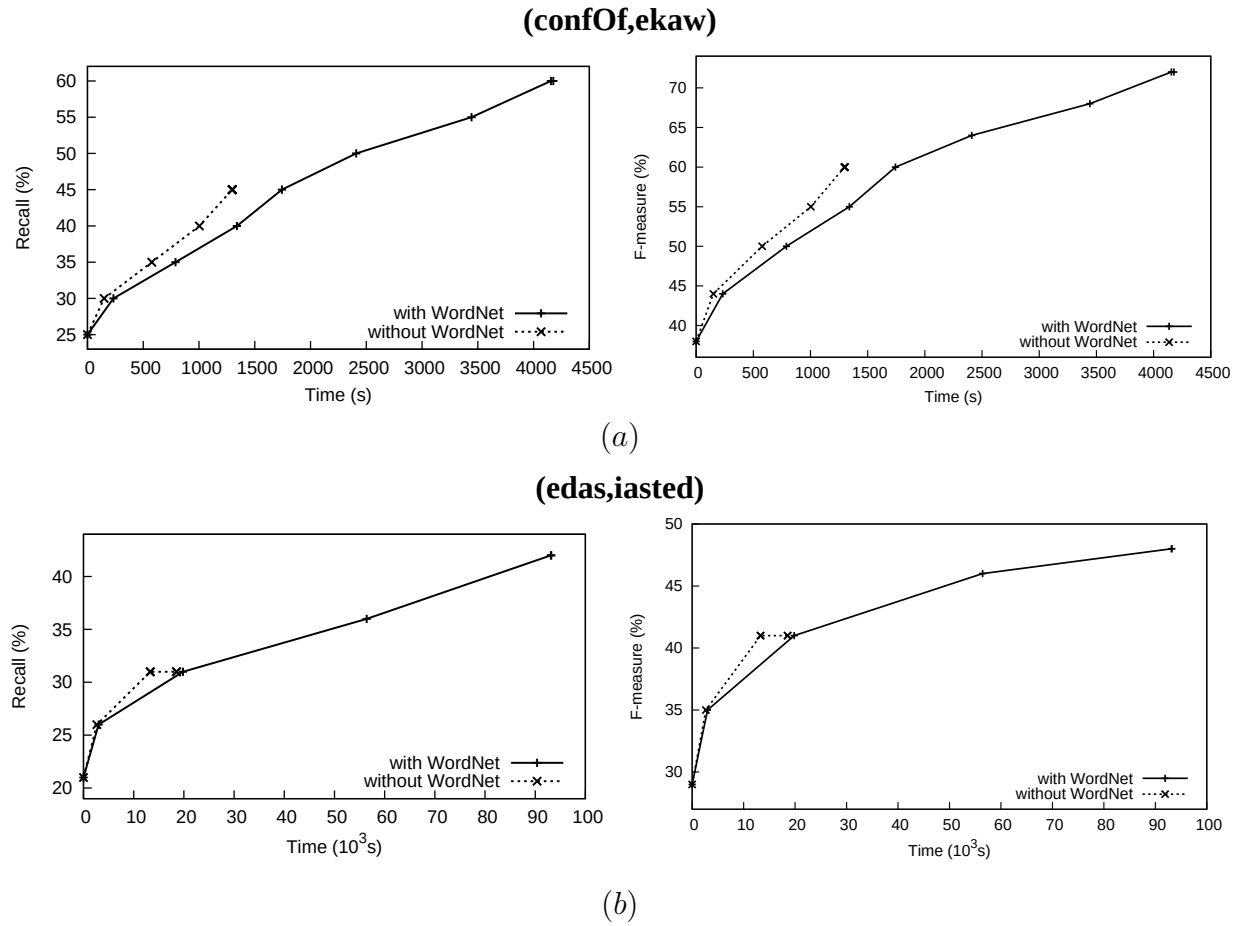


Figure B.1: Recall and F-measure for 2 ontology pairs of the same trend where the final recall and F-measure with WN integrated is higher than the recall and F-measure with just syntactic similarity.

We categorized the ontology pairs based on the trends that their recall and F-measure exhibited. In Fig. B.1, we show another 2 out of 7 of those pairs for which the final recall and F-measure due to WordNet improved considerably although, in some cases, the intermediate values of recall and F-measure were achieved by **Optimaw** without WordNet in less time.

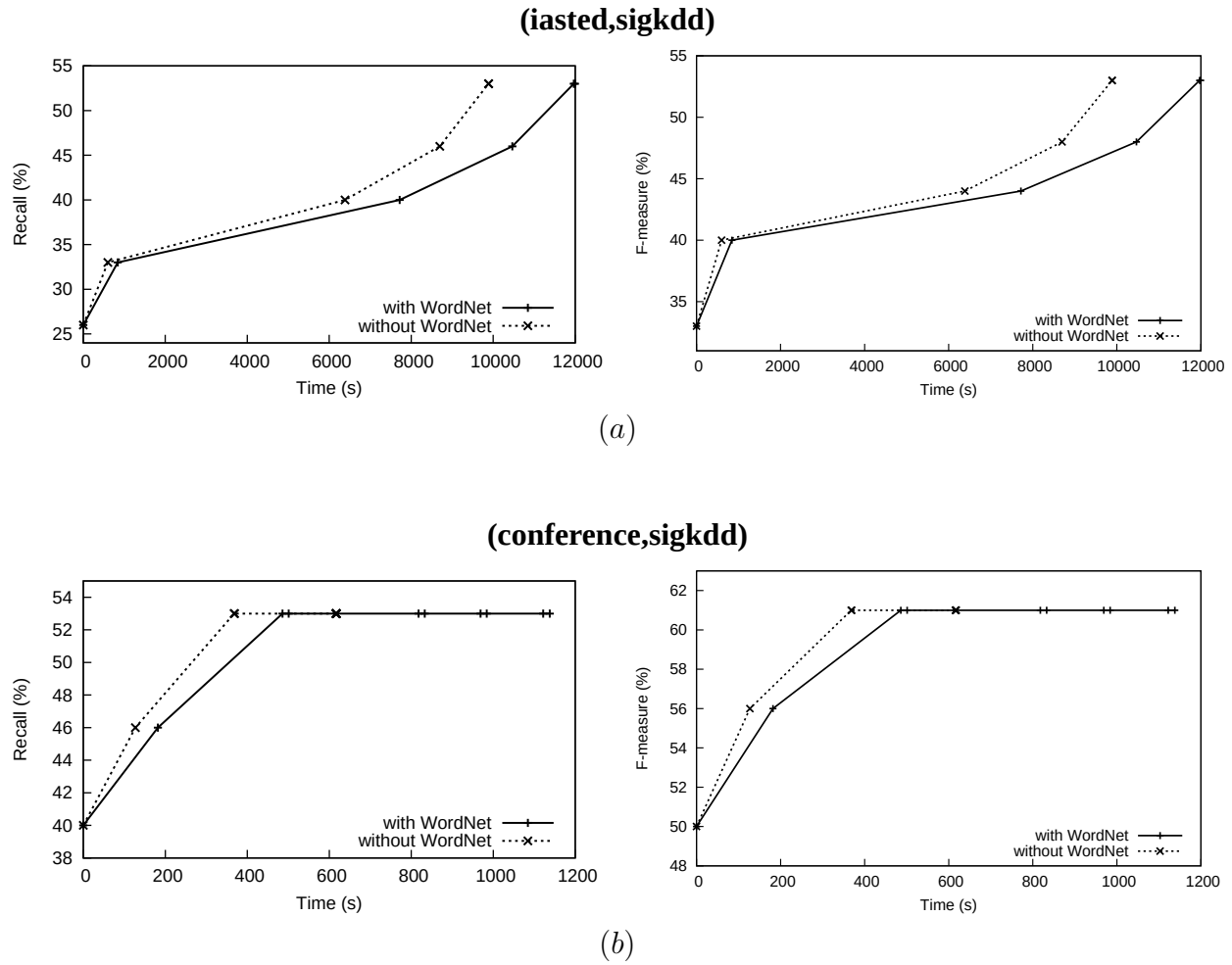


Figure B.2: Recall and F-measure for 2 ontology pairs of the same trend where the final recall and F-measure with WN integrated did not improve on the recall and F-measure without WN.

Next, we show pairs for which the alignment with WordNet showed similar recall and F-measure as achieved by aligning with just string similarity. Six ontology pairs exhibit this trend and we show 2 of them in Fig. B.2. Notice the increased execution time due to WordNet for similar recall and F-measure values.

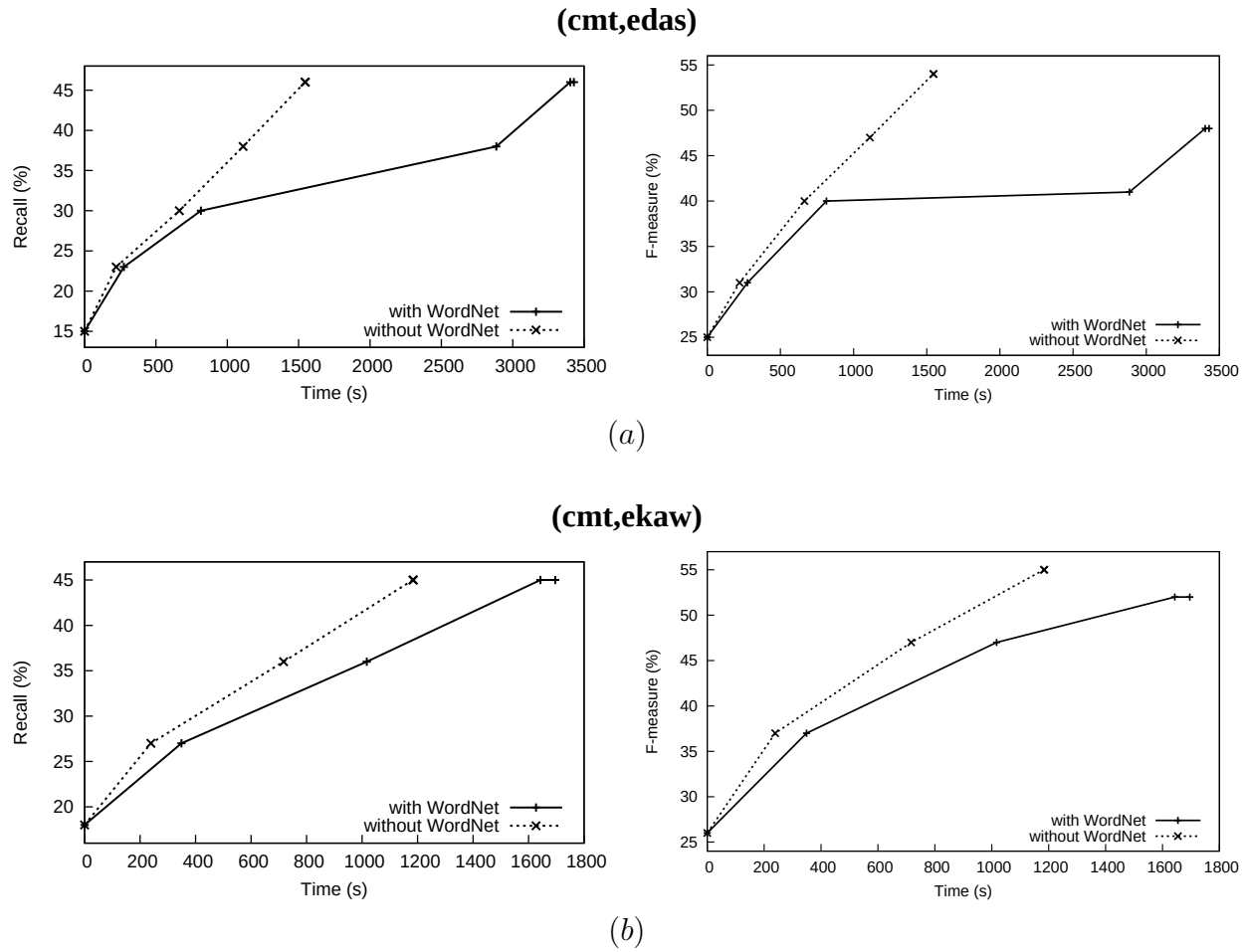


Figure B.3: Both the ontology pairs shown here exhibit a final recall with WordNet that is same as the recall without it. However, the F-measure with WordNet is less than the F-measure without WordNet.

Finally, 10 ontology pairs resulted in recall with WordNet that was similar to recall with just the syntactic string similarity, but poorer F-measure while aligning with WordNet due to reduced precision. When the additional execution time is taken into consideration, the utility of WordNet is questionable in these cases. We show 2 of these pairs in Fig. B.3.