Journal of Artificial Intelligence Research 50 (2014) 805-845 Submitted 02/14; published 08/14

Speeding Up Iterative Ontology Alignment using Block-Coordinate Descent

Uthayasanker Thayasivam UTHAYASA@CS.UGA.EDU
Prashant Doshi PDOSHI@CS.UGA.EDU
THINC Lab, Department of Computer Science,
University of Georgia, Athens, GA 30602, USA

Abstract

In domains such as biomedicine, ontologies are prominently utilized for annotating data. Consequently, aligning ontologies facilitates integrating data. Several algorithms exist for automatically aligning ontologies with diverse levels of performance. As alignment applications evolve and exhibit online run time constraints, performing the alignment in a reasonable amount of time without compromising the quality of the alignment is a crucial challenge. A large class of alignment algorithms is iterative and often consumes more time than others in delivering solutions of high quality. We present a novel and general approach for speeding up the multivariable optimization process utilized by these algorithms. Specifically, we use the technique of block-coordinate descent (BCD), which exploits the subdimensions of the alignment problem identified using a partitioning scheme. We integrate this approach into multiple well-known alignment algorithms and show that the enhanced algorithms generate similar or improved alignments in significantly less time on a comprehensive testbed of ontology pairs. Because BCD does not overly constrain how we partition or order the parts, we vary the partitioning and ordering schemes in order to empirically determine the best schemes for each of the selected algorithms. As biomedicine represents a key application domain for ontologies, we introduce a comprehensive biomedical ontology testbed for the community in order to evaluate alignment algorithms. Because biomedical ontologies tend to be large, default iterative techniques find it difficult to produce a good quality alignment within a reasonable amount of time. We align a significant number of ontology pairs from this testbed using BCD-enhanced algorithms. Our contributions represent an important step toward making a significant class of alignment techniques computationally feasible.

1. Introduction

Recent advances in Web-based ontologies provide a needed alternative to conventional schemas allowing descriptive annotations of data sets. As an example, the National Center for Biomedical Ontology (NCBO) hosts more than 370 curated biomedical ontologies in its BioPortal including those in high use such as SNOMED-CT, and whose concepts participate in more than 2 billion data annotations (Musen et al., 2012). Therefore, the present day challenge toward data integration and to manage the multitude of ontologies is to build bridges between ontologies that have overlapping scope – a problem often referred to as that of ontology matching which produces an *alignment* (Euzenat & Shvaiko, 2007). We illustrate a partial alignment between biomedical ontologies in Fig. 1.

Consequently, several algorithms exist for automatically aligning ontologies using various techniques (Euzenat, Loup, Touzani, & Valtchev, 2004; Jian, Hu, Cheng, & Qu, 2005; Li, Li, & Tang, 2007; Jean-Mary, Shironoshita, & Kabuka, 2009; Doshi, Kolli, & Thomas, 2009; Wang & Xu, 2009; Hanif & Aono, 2009; Bock & Hettenhausen, 2010; Jiménez-Ruiz & Grau, 2011; Shvaiko &

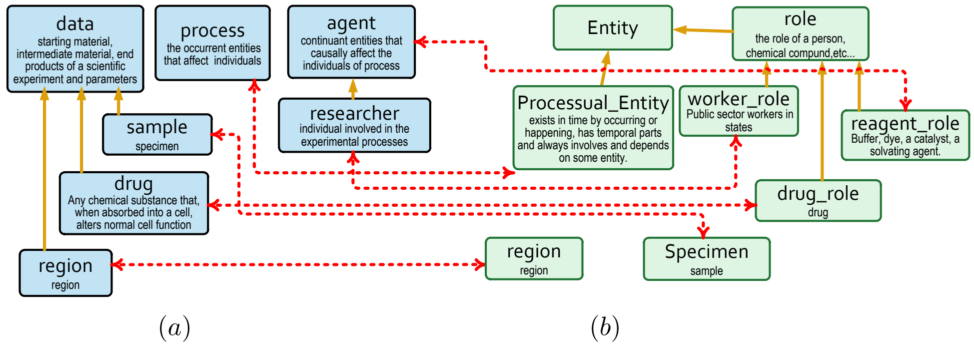


Figure 1: Biomedicine is an important application domain for ontologies. Alignment (shown in dashed red) between portions of, (a) the Parasite Experiment Ontology (PEO) and, (b) the Ontology of Biomedical Investigations (OBI) as discovered by an automated algorithm called AgreementMaker (Cruz et al., 2012). Both these ontologies are available at NCBO. Each identified map in the alignment signifies an equivalence between the concepts.

Euzenat, 2013), 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, only 6 alignment algorithms (not including their variants) out of the 21 that participated in the 2012 and 2013 instances of the annual ontology alignment evaluation initiative (OAEI) competition (Shvaiko et al., 2012, 2013) generated results in an acceptable amount of time for aligning large biomedical 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 (2004) 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 (Shvaiko et al., 2011) began reporting the execution times of the participating alignment algorithms as well. As ontologies become larger, efficiency and scalability become key properties of alignment algorithms.

A large class of algorithms that performs automated alignment is *iterative* in nature (Melnik, Garcia-molina, & Rahm, 2002; Euzenat et al., 2004; Jian et al., 2005; Li et al., 2007; Doshi et al., 2009; Wang & Xu, 2009; Hanif & Aono, 2009; Bock & Hettenhausen, 2010). 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 may run until convergence, which means that the solution cannot be improved further because it is a, possibly local, optimum. However, in practice, the runs 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. For example, algorithms performing among the top three in OAEI 2012 in terms of alignment quality such as YAM++ (Ngo & Bellahsene, 2012), which ranked first in the *conference* track, Optima+, ranked third in the *conference* track, and GOMMA (Kirsten et al., 2011), which ranked first in *anatomy* and *library* tracks, are iterative. [1] On the other hand, YAM++ consumed an excessive amount of time in completing the *conference* track (greater than 5 hours) and Optima+ 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 have delivered ways of scaling the algorithms to larger ontologies, such as through ontology partitioning (Hu, Zhao, & Qu, 2006; Seddiqui & Aono, 2009; Stoutenburg, Kalita, Ewing, & Hines, 2010; Rahm, 2011) and the use of inverted indices (Jiménez-Ruiz & Grau, 2011), we seek to speed up the alignment process of multiple algorithms. We think that considerations of space and time go hand in hand in the context of usability.

Our primary contribution in this article is a general approach and its comprehensive evaluation for significantly speeding up the convergence of iterative ontology alignment techniques. Thayasivam and Doshi (2012a) provide a preliminary introduction to this approach. Objective functions that measure the solution quality are typically multidimensional. Instead of the traditional approach of modifying the values of a large number of variables in each iteration, we decompose the problem into optimization subproblems in which the objective 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 (Tseng, 2001). While it forms a standard candidate tool for multidimensional optimization in statistics, and has been applied in contexts such as image reconstruction (Pintér, 2000; Fessler & Kim, 2011) and channel capacity computation (Blahut, 1972; Arimoto, 1972), this article presents its use in ontology alignment.

We extensively evaluate this approach by integrating it into multiple ontology alignment algorithms. We selected Falcon-AO (Jian et al., 2005), MapPSO (Bock & Hettenhausen, 2010), OLA (Euzenat & Valtchev, 2004) and Optima (Doshi et al., 2009) as representative algorithms. These algorithms have participated in OAEI competitions in the past, and some of them have ranked in the top tier. Consequently, these algorithms in their default forms exhibit favorable alignment performance. Furthermore, their implementations and source codes that are needed for our approach are freely accessible.

Using a comprehensive testbed of several ontology pairs – some of which are large – spanning multiple domains, we show a significant reduction in the execution times of the alignment processes thereby converging faster. 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 if possible in contrast to a predefined ad hoc number of iterations, which possibly leads to the similar or improved alignments. This is useful in the context of techniques that are guaranteed to converge.

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, we may order the blocks for consideration in any manner within a cycle. Consequently, our second contribution is an empirical study of 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 grouped entities, we consider reversing this ordering, and a third approach in which we sample the

1. GOMMA utilizes multiple matching strategies some of which may not be iterative, and these partly contributed toward its performance in OAEI as well.

blocks based on a probability distribution that represents the estimated likelihood of finding a large alignment in a block. In the context of partitioning, we additionally consider grouping alignment variables such that the entities are divided in a breadth-first search based partition. While our default approach partitions one of the ontologies in a pair, we also consider the impact of partitioning both. Performances of the iterative algorithms are impacted differently by various ways of formulating the blocks and ordering them. Notably, the quality of the alignment may be adversely impacted.

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, we contribute a novel biomedical ontology alignment testbed. In addition to being an important application domain, aligning biomedical ontologies has its own unique challenges. We selected biomedical ontologies published in NCBO for our testbed, which also provides a primarily UMLS-sourced but incomplete reference alignment. Thirty-two different biomedical ontologies form the 50 pairs in our testbed with about half of these having 3,000+ named classes.

The rest of this article is organized as follows. In the next section, we briefly explain iterative ontology alignment and introduce the four representative iterative algorithms. Additionally, we briefly review the technical approach of BCD. We show how BCD may be integrated into iterative ontology alignment algorithms in Section 3. In Section 4, we empirically evaluate the performances of the BCD enhanced algorithms using a comprehensive data set. Then, in Section 5, we explore other ways of ordering the blocks and partitioning the alignment variables. Thereafter, in Section 6, we detail a new biomedical ontology benchmark and report the performances of the BCD enhanced and optimized iterative techniques on this benchmark. We discuss the impact of BCD along with its limitations in Section 7, and conclude this article in Section 8. Appendix A outlines the representative algorithms and their modifications to utilize BCD, followed by details on the biomedical ontology alignment testbed in Appendix B.

2. Background

We provide a brief overview of the ontology alignment problem in the next subsection. This is followed by brief descriptions of the four algorithms that are representative of iterative alignment approaches. Finally, we describe the technique of BCD in general.

2.1 Overview of Ontology Alignment

An ontology is a specification of knowledge pertaining to a domain of interest formalized into entities and relationships between the entities. Contemporary ontologies utilize description logics (Baader, Horrocks, & Sattler, 2003) such as the Web Ontology Language (OWL) (McGuinness & Harmelen, 2004) 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.

The ontology alignment problem 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 (Hayes & Gutierrez, 2004) and OLA utilizes an OL-graph (Euzenat et al., 2004). 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 uniquely 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, $m_{a\alpha}$, between two entities, $x_a \in \mathcal{O}_1$ and $y_\alpha \in \mathcal{O}_2$, consists of the relation, $r \in \{=, \subseteq, \supseteq\}$, and confidence, $c \in \mathbb{R}$. However, the alignment algorithms that we use 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 $|V_1| \times |V_2|$ matrix that represents the correspondence between the two ontologies, $\mathcal{O}_1 = \langle V_1, E_1, L_1 \rangle$ and $\mathcal{O}_2 = \langle V_2, E_2, L_2 \rangle$:

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

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 relationship labels as well. In order to facilitate matching relationships, alignment techniques, including some that we use transform the edge-labeled graphs into unlabeled bipartite 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.

2.2 Iterative Ontology Alignment

A large class of alignment algorithms is iterative in nature (Melnik et al., 2002; Euzenat et al., 2004; Jian et al., 2005; Li et al., 2007; Doshi et al., 2009; Wang & Xu, 2009; Hanif & Aono, 2009; Bock & Hettenhausen, 2010; Ngo & Bellahsene, 2012). Iterative algorithms utilize a seed matrix, M^0, which is iteratively improved until it converges. The seed matrix is either input by the user or generated automatically often using fast string matching and other lexical matching.

Two types of iterative techniques are predominant. These differ in how the next match matrix, M, is obtained from the previous iteration's match matrix at each step. The first type of iterative algorithms improve the real-valued similarity matrix from the previous iteration, M^{i-1}, by directly updating it:

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

where, U is a function that updates the similarities. This type of algorithms often converges to a fixed point, M_*, such that, $M_* = U(M_*)$. [2]

The second type of iterative algorithms repeatedly and explicitly search over the space of match matrices, denoted as $\mathcal{M}$. The goal is to find the alignment that optimizes an objective function,

2. Convergence is predicated on U, and a fixed point may not exist for some techniques. However, convergence is a desirable property for iterative alignment algorithms; in its absence the stop criteria is often ad hoc.

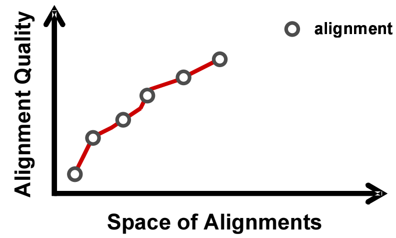


Figure 2: Both iterative update and search jump from one alignment to another improving on the previous one. The two differ in how they obtain the next alignment in each iteration and in the qualitative metric used for assessing it. An alignment that cannot be improved further signifies convergence.

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,

$$M_*^i = \underset{M \in \mathcal{M}}{\arg\max}\, Q(M, M_*^{i-1}) \tag{2}$$

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 iterations reach a point, M_*, which cannot be improved on searching for an alignment matrix, $M \in \mathcal{M}$, such that $Q(M, M_*) > Q(M_*, M_*)$. Equations 1 and 2 help solve a multidimensional optimization problem iteratively with $m_{a\alpha}$ in M as the variables. We abstractly illustrate the iterative approaches in Fig. 2.

In Fig. 3, we show the abstract algorithms for the two types of iterative approaches. In the iterative update of Fig. $3(a)$, we 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 $\eta \rightarrow 0$ we get closer to the fixed point and obtain the fixed point in the limit. Iterative search in Fig. $3(b)$ often requires a seed map (line 3) to obtain M^0, which is typically generated using fast lexical matching.

Next, we briefly review *four* ontology alignment algorithms that optimize iteratively. The selection of these algorithms is based on their accessibility and competitive performance in previous OAEI competitions, and is meant to be representative of iteration-based alignment algorithms. [3]

2.2.1 FALCON-AO

Falcon-AO (Jian et al., 2005) is a well-known automated ontology alignment system combining output from multiple components including a linguistic matcher, an iterative structural graph matching algorithm called GMO (Hu, Jian, Qu, & Wang, 2005), and a method for partitioning large ontologies and focusing on some of the parts.

3. We sought to include YAM++ as well in our evaluation, which was the top performer in the *conference* track of OAEI 2012 and 2013. However, its source code is not freely available and we could not access it.

ITERATIVE UPDATE ($\mathcal{O}_1$, $\mathcal{O}_2$, η)

Initialize:
1. Iteration counter $i \leftarrow 0$
2. Calculate similarity between the entities in $\mathcal{O}_1$ and $\mathcal{O}_2$ using a measure
3. Populate the real-valued matrix, M^0, with initial similarity values
4. $M_* \leftarrow M^0$

Iterate:
5. **Do**
6. $\quad i \leftarrow i + 1$
7. $\quad M^i = U(M^{i-1})$
8. $\quad \delta \leftarrow Dist(M^i, M_*)$
9. $\quad M_* \leftarrow M^i$
10. **While** $\delta \geq \eta$
11. Extract an alignment from M_*

(a)

ITERATIVE SEARCH ($\mathcal{O}_1$, $\mathcal{O}_2$)

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. $M_* \leftarrow M^0$

Iterate:
5. **Do**
6. $\quad i \leftarrow i + 1$
7. $\quad$ Search $M^i \leftarrow \arg\max_{M \in \mathcal{M}} Q(M, M^{i-1})$
8. $\quad M_* \leftarrow M^i$
9. **While** $M_* \neq M^{i-1}$
10. Extract an alignment from M_*

(b)

Figure 3: 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.

GMO measures the structural similarity between the ontologies that are modeled as bipartite graphs (Hayes & Gutierrez, 2004). Matrix M in GMO is real-valued and this similarity matrix is iteratively updated (Eq. 1) by updating each variable, $m_{a\alpha}$, with the average of its neighborhood similarities until M stops changing significantly. GMO takes external input, typically obtained from lexical matching, as the seed. Equation 1 manifests in GMO as a series of matrix operations:

$$M^i = G_1 M^{i-1} G_2^T + G_1^T M^{i-1} G_2 \tag{3}$$

Here, G_1 and G_2 are the adjacency matrices of the bipartite 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. Iterations terminate when the cosine similarity between successive matrices, M^i and M^{i-1}, is less than a parameter, η. The iterative update algorithm manifests in Falcon-AO as shown in Fig. 17(a) in Appendix A.

2.2.2 MAPPSO

MapPSO (Bock & Hettenhausen, 2010) utilizes discrete particle swarms to perform the optimization. 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 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:

$$Q(M_k^i) = \frac{\sum_{a=1}^{|V_1|} \sum_{\alpha=1}^{|V_2|} m_{a\alpha} \times f(x_a, y_\alpha)}{|V_1||V_2|} \tag{4}$$

where, $m_{a\alpha}$ is a correspondence in M_k^i and f represents a weighted combination of multiple syntactic and possibly semantic 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 a particle, and replacing others based on the correspondences in the previous best particle. Iterations terminate when the increment in Q due to a new alignment matrix is lower than a parameter, η. Iterative search of Eq. 2 manifests in MapPSO as shown in the algorithm in Fig. 18(a).

2.2.3 OWL-LITE ALIGNMENT

OWL-Lite alignment (OLA) (Euzenat et al., 2004) is limited to aligning ontologies expressed in OWL with an emphasis on its most restricted dialect called OWL-Lite. 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 properties in individuals.

OLA computes the similarity between a pair of entities from two ontologies as a weighted aggregation of the similarities between respective neighborhood entities. 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, solved iteratively for the fixed point.

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

$$Sim(x_a, y_\alpha) = \sum_{\mathcal{F} \in \mathcal{N}(x_a, y_\alpha)} w_{\mathcal{F}}^{a\alpha} SetSim(\mathcal{F}(x_a), \mathcal{F}(y_\alpha)) \tag{5}$$

where, $\mathcal{N}(x_a, y_\alpha)$ is the set of all edge types in which x_a, y_α participate. Weight, $w_{\mathcal{F}}^{a\alpha}$, for an entity pair, x_a, y_α, and edge type, $\mathcal{F}$, is normalized, i.e., $\sum_{\mathcal{F} \in \mathcal{N}(x_a, y_\alpha)} w_{\mathcal{F}}^{a\alpha} = 1$. Function, $SetSim$, evaluates the similarity between sets, $\mathcal{F}(x_a)$ and $\mathcal{F}(y_\alpha)$, 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, x_a and y_α are of the same type, then $m_{a\alpha}$ is updated using Eq. 5, otherwise the value is 0. Iterative update of Eq. 1 is realized by OLA as in Fig. 19(a) in Appendix A.

2.2.4 OPTIMA

Optima (Doshi et al., 2009) 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 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:

$$Q(M^i|M^{i-1}) = \sum_{a=1}^{|V_1|}\sum_{\alpha=1}^{|V_2|} Pr(y_\alpha|x_a, M^{i-1}) \times \; logPr(x_a|y_\alpha, M^i)\pi_\alpha^i \tag{6}$$

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 involves finding a matrix, M_*^i, that improves on the previous one:

$$M_*^i = M^i \in \mathcal{M} : \; Q(M^i|M_*^{i-1}) \geq Q(M_*^{i-1}|M_*^{i-1}) \tag{7}$$

We show the iterative alignment algorithm of Optima in Fig. $20(a)$.

Altogether, the four alignment algorithms that we describe in this subsection 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 competitions, with Falcon-AO and Optima demonstrating strong performances with respect to the comparative quality of the generated alignment. For example, Falcon-AO often ranked 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+) ranked second in the *conference* track (F2-measure and recall) in the 2012 edition of the OAEI competition (Thayasivam & Doshi, 2012b). Consequently, these representative algorithms exhibit strong alignment performances. On the other hand, MapPSO's performance is comparatively poor but it's particle-swarm based iterative approach motivates its selection in our representative set.

2.3 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) (Tseng, 2001) 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*. In our application setting, recall that the coordinates are the alignment variables in the match matrix, M.

Let S denote a block of coordinates, which is a non-empty subset of $\{1, 2, \ldots, N\}$. Define a set of such blocks as, $B = \{S_0, S_1, \ldots, S_C\}$, which is a set of subsets each representing a coordinate block with the constraint that, $S_0 \cup S_1 \cup \ldots \cup S_C = \{1, 2, \ldots, N\}$. B could have a single block or be a partition of the coordinates although this is not required and the blocks may intersect. We also define the complement of a coordinate block, S_c, where $c \in \{0, 1, \ldots, C\}$, 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, $M = \{m_1, m_2, m_3, \ldots, m_{10}\}$, where each element is a variable. We 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 $S_1 = \{m_1, m_3, m_4, m_6, m_7, m_9, m_{10}\}$. Finally, $\tilde{S}_0$ denotes the block, S_1.

BCD converges to a fixed point such as a local or a global optimum of the objective function under relaxed conditions such as pseudoconvexity of the function and requires the function to have bounded level sets (Tseng, 2001). 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, we must satisfy the following rule, which ensures that each coordinate is chosen sufficiently often (Tseng, 2001).

Definition 1 (Cyclic rule) *There exists a constant, $T \geq C$ and $C > 0$, 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 we must continue iterating until each block converges to the fixed point.

Recently, Saha and Tewari (2013) show that the nonasymptotic convergence rate [4] of BCD under the cyclic rule is faster than that of gradient descent (GC) if they both start from the same point, under the conditions that the objective function, Q, has a Lipschitz continuous gradient (it is differentiable everywhere and has a bounded derivative) or is strongly convex, and $I - \frac{\nabla Q}{L}$ is isotonic, where I is the identity function and L is the Lipschitz constant. Starting from the same initial map, $M^0_{BCD} = M^0_{GC}$, let M^i_{BCD} and M^i_{GC} 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 continuous and isotonic, $\forall i \geq 1$, $Q(M^i_{BCD}) \leq Q(M^i_{GC})$. The nonasymptotic convergence rate of BCD under the cyclic rule for objective functions with the previous properties is, $\mathcal{O}(1/i)$, where i is the iteration count.

3. Integrating BCD into Iterative Alignment

As we 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, we may speed up the convergence rate using BCD as we describe below.

3.1 General Approach

In Section 2.2, we identified 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. Of course, existing algorithms may be viewed as having a single block of variables and therefore trivially utilizing BCD.

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

Though a matrix may be partitioned using one of several ways, we adopt an approach that is well supported 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 (Doan, Madhavan, Domingos, & Halevy, 2003). 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. Subsequently, the alignment in blocks with less height (containing the child entities) is optimized first followed by those with increasing height (containing the parent entities). In determining this height, we utilize the tree or graph model of the ontology that is built internally by the respective ontology alignment algorithm. These include property nodes and may differ between algorithms.

Let the partition of M into coordinate blocks be $\{M_{S_0}, M_{S_1}, \ldots, M_{S_C}\}$, 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 Optima, 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, we choose a block, M_{S_c}, at iterations, $i = c + qC$ where $q \in \{0, 1, 2, \ldots\}$. We point out that BCD is applicable to both types of iterative alignment techniques outlined in Section 2.2. Alignment algorithms which update the similarity matrix iteratively as in Eq. 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:

$$\begin{array}{l} M^i_{S_c} = U_{S_c}(M^{i-1}) \\ M^i_{\tilde{S}} = M^{i-1}_{\tilde{S}} \qquad \forall \tilde{S} \in \tilde{S}_c \end{array} \tag{8}$$

where $\tilde{S}_c$ is the complement of S_c in B. Note that $M^i_{S_c}$ combined with $M^i_{\tilde{S}}$ for all $\tilde{S} \in \tilde{S}_c$ forms M^i. Update function, U_{S_c}, modifies U in Eq. 1 to update just a block of the coordinates.

Analogously, iterative alignment which searches for the candidate alignment that maximizes the objective function as in Eq. 2, will now choose a block, M_{S_c}, at each iteration. It 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,

$$\begin{array}{l} M^i_{S_c,*} = \underset{M_{S_c} \in \mathcal{M}_{S_c}}{\arg\max}\; Q_S\left(M_{S_c}, M^{i-1}_*\right) \\ M^i_{\tilde{S},*} = M^{i-1}_{\tilde{S},*} \qquad \forall \tilde{S} \in \tilde{S}_c \end{array} \tag{9}$$

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.

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 we may perform more iterations as we cycle

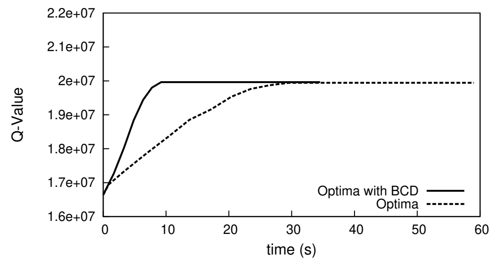


Figure 4: BCD facilitates faster convergence in aligning ontologies *iasted* and *sigkdd* both related to conference organization.

through the blocks, the use of partially updated matrices from the previous iteration in evaluating the next block facilitates faster convergence. We illustrate the impact of BCD on iterative search as performed by Optima on an example ontology pair in Fig. 4. Alignment using BCD shows the faster convergence rate.

ITERATIVE UPDATE WITH BCD ($\mathcal{O}_1$, $\mathcal{O}_2$, η)

Initialize:
1. Iteration counter $i \leftarrow 0$
2. Calculate similarity between the entities in $\mathcal{O}_1$ and $\mathcal{O}_2$ using a measure
3. Populate the real-valued matrix, M^0, with initial similarity values
4. Create a partition of M: $\{M_{S_0}, M_{S_1}, \ldots, M_{S_C}\}$
5. $M_* \leftarrow M^0$

Iterate:
6. **Do**
7. $c \leftarrow i \,\%\, (C+1), i \leftarrow i+1$
8. $M^i_{S_c} \leftarrow U_{S_c}(M^{i-1})$
9. $M^i_{\tilde{S}} \leftarrow M^{i-1}_{\tilde{S}} \quad \forall \tilde{S} \in \tilde{S}_c$
10. **If** $c = C$ **then**
11. $\delta \leftarrow Dist(M^i, M_*)$
    else
12. δ is a high value
13. $M_* \leftarrow M^i$
14. **While** $\delta \geq \eta$
15. Extract an alignment from M_*

(a)

ITERATIVE SEARCH WITH BCD ($\mathcal{O}_1$, $\mathcal{O}_2$)

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. Create a partition of M: $\{M_{S_0}, M_{S_1}, \ldots, M_{S_C}\}$
5. $M_* \leftarrow M^0$

Iterate:
6. **Do**
7. $c \leftarrow i \,\%\, (C+1), i \leftarrow i+1$
8. Search $M^i_{S_c,*} \leftarrow \underset{M_{S_c} \in \mathcal{M}_{S_c}}{\arg\max}\; Q_S\left(M_{S_c}, M^{i-1}_*\right)$
9. $M^i_{\tilde{S},*} \leftarrow M^{i-1}_{\tilde{S},*} \quad \forall \tilde{S} \in \tilde{S}_c$
10. **If** $c = C$ **then**
11. $changed \leftarrow M^i_* \neq M^{i-1}_*$?
    else
12. $changed \leftarrow$ true
13. **While** $changed$
14. Extract an alignment from M^i_*

(b)

Figure 5: General iterative algorithms of Fig. 3 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.

Algorithms in Fig. 5 revise the iterative update and search algorithms of Fig. 3 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 others fixed (lines 7-9). On completing a cycle through all coordinate blocks as determined by the check in line 10, we evaluate whether the new alignment matrix differs from the one in the previous iteration, and continue the iterations if it does (lines 11-13). Observe that the regular iterations improving the full match matrix are now replaced with "mini-iterations" updating the blocks.

Given the general modifications brought about by BCD, we describe how these manifest in the four iterative alignment systems that form our representative set. The modifications are based on the type of iterative technique and are uniform within each group. They do not change the core alignment approach of each algorithm given the input as we see next.

3.2 BCD Enhanced Falcon-AO

We enhance Falcon-AO by modifying GMO to utilize BCD as it iterates. As depicted in Fig. 17(b), we 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. 3 becomes:

$$\begin{array}{l} M^i_{S_c} = G_{1,S_c} M^{i-1} G_2^T + G^T_{1,S_c} M^{i-1} G_2 \\ M^i_{\tilde{S}} = M^{i-1}_{\tilde{S}} \quad \forall \tilde{S} \in \tilde{S}_c \end{array} \tag{10}$$

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^T_{1,S_c} 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(a) may be realized in Falcon-AO as in the algorithm of Fig. 17(b) in Appendix A.

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 4 is modified to measure the quality of the correspondences in just the coordinate block S_c, in the k^{th} particle by taking the average:

$$Q_S(M^i_k) = \frac{\sum_{a=1}^{|V_{1,c}|} \sum_{\alpha=1}^{|V_2|} m_{a\alpha} \times f(x_a, y_\alpha)}{|V_{1,c}||V_2|} \tag{11}$$

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, we retain the best particle(s) based on this measure and improve on the alignment in a coordinate block, M^i_{k,S_c}, 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. 18(b).

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. 8. Specifically, the similarity values of the coordinates of the chosen block, S_c, will be updated using the similarity computations (Eq. 5). The remaining blocks, $M^i_{\tilde{S}_c}$, are kept unchanged.

$$\begin{aligned} m^i_{a\alpha} &= \begin{cases} Sim(x_a, y_\alpha) & \text{if types of } x_a \text{ and } y_\alpha \text{ are the same} \\ 0 & \text{otherwise} \end{cases}, \forall_{m^i_{a\alpha} \in M^i_{Sc}} \\ M^i_{\tilde{S}} &= M^{i-1}_{\tilde{S}} \quad \forall \tilde{S} \in \tilde{S}_c \end{aligned} \tag{12}$$

3.5 BCD Enhanced Optima

As we mentioned previously, Optima utilizes generalized expectation-maximization to iteratively improve the likelihood of candidate alignments. Jeffery and Alfred (1994) discuss a BCD inspired 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. [5] In each iteration, Optima enhanced using SAGE chooses a block of the match matrix, $M^i_{S_c}$, and its expected log likelihood is estimated. As in previous techniques, we choose the blocks in a sequential manner such that all the blocks are iterated in order.

Equation 6 changes to estimate the expected log likelihood of a block of a candidate alignment:

$$Q_S(M^i_{S_c}|M^{i-1}) = \sum_{a=1}^{|V_{1,c}|} \sum_{\alpha=1}^{|V_2|} Pr(y_\alpha|x_a, M^{i-1}) \times \ logPr(x_a|y_\alpha, M^i_{S_c})\ \pi^i_{\alpha,c} \tag{13}$$

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, $\pi^i_{\alpha,c}$, is modified as well to utilize just $V_{1,c}$ in its calculations.

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

$$M^i_{S_c,*} = M^i_{S_c} \in \mathcal{M}_{S_c} : Q_S(M^i_{S_c,*}|M^{i-1}_*) \geq Q_S(M^{i-1}_{S_c,*}|M^{i-1}_*) \tag{14}$$

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

At iteration i, the best alignment matrix M^i_*, is formed by combining the block $M^i_{S_c,*}$, which improves Q_S as defined in Eq. 14 with the remaining blocks from the previous iteration, $M^{i-1}_{\tilde{S},*}$, in the complement of S_c, unchanged.

5. Furthermore, the convergence rate may be improved by choosing the hidden space with less Fisher information (Hero & Fessler, 1993).

The algorithm in Fig. 20(b) shows how Optima may be enhanced with BCD. We expect significant savings in time because of the search over a reduced space of alignments focused on a block, $\mathcal{M}_{S_c}$, in each iteration. Additionally, both the objective function, Q_S, and the prior operate on a single coordinate block in reduced time. Finally, using aligned blocks in the next iteration improves the convergence rate.

4. Empirical Analysis

While the use of BCD is expected to make the iterative approaches exhibit a greater rate of improvement, and if the approach converges, reach the fixed point faster, we seek to empirically determine:

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 optimum.

Ontology	**Named Classes**	**Properties**
Conference domain		
ekaw	74	33
sigkdd	49	28
iasted	150	41
cmt	36	59
edas	104	50
confOf	38	36
conference	60	64
Life Sciences		
mouse anatomy	2,744	2
human anatomy	3,304	3

Table 1: Ontologies from OAEI 2012 used in our evaluation. We show the number of named classes and properties in each as an estimate of their size. Notice that our evaluation includes some large ontologies from different domains as well. Additionally, Thayasivam and Doshi (2012a) present evaluations on the four pairs in the 300 range of the *bibliography* benchmark competition.

We use a comprehensive testbed of several ontology pairs – some of which are large – spanning two domains. We used ontology pairs from the OAEI competition in its 2012 version as the testbed for our evaluation (Shvaiko et al., 2012). 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 pairs of the 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 mid-sized ontologies from the life sciences describing the anatomy of an adult mouse and human. We list the ontologies from OAEI participating in our evaluation in Table 1 and provide an indication of their sizes. Additionally, Thayasivam and Doshi (2012a) evaluate Falcon-AO, MapPSO and Optima with BCD on the four pairs in the 300 range of the *bibliography* benchmark competition. Ontology pairs in the 100 and 200 ranges of the bibliography benchmark were not utilized as the participating ontologies are very small with just 33 classes and 64 properties. Subsequently, our representative iterative techniques align these very quickly in the order of milliseconds leaving no significant room for improvement.

We align ontology pairs using the four representative 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 we 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.

We 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, we 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, we tested the data for normality and used Student's paired t-test if it exhibits normality. Otherwise, we employed the Wilcoxon signed-rank test. We utilized the 1% level ($p \leq 0.01$) to deem significance.

As Thayasivam and Doshi (2012a) did not previously evaluate OLA on the bibliography domain ontology pairs, we discuss its performance in this article for completeness. Similar to the other algorithms, the introduction of BCD in OLA reduced its execution time on all four pairs by a total of 1.3 seconds compared to the original time of 27.3 seconds. 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 ontologies in the *conference* domain vary widely in their size and structure. As shown in Fig. 6, 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, we 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, we observed a total reduction of 50 seconds for Falcon-AO to 3 minutes, 1 minute and 37 seconds for MapPSO, 31 seconds for OLA to a total of 1 minute and 37 seconds, and by 29 minutes and 20 seconds for Optima to 4 minutes and 53 seconds.

Falcon-AO shows no change due to BCD in its alignment, holding its precision at 25% and recall at 66%. Optima shows a 4% improvement in average precision from 56% to 60% but average recall reduced from 70% to 68%. Nevertheless, this causes a 2% improvement in average F-measure to 64%. MapPSO with BCD resulted in a significant improvement in final precision from 9% to 43% on average, although the difference in recall was not significant. The precision and recall for OLA remained unchanged.

The mid-sized *anatomy* ontologies for mouse and human were not successfully aligned by MapPSO and OLA despite the use of BCD. However, BCD reduced Falcon-AO's average execution time for aligning this single ontology pair by 6.2 seconds to 2.6 minutes, and drastically reduced Optima's average execution time to 4.4 minutes from 62.7 minutes. The alignment gen-

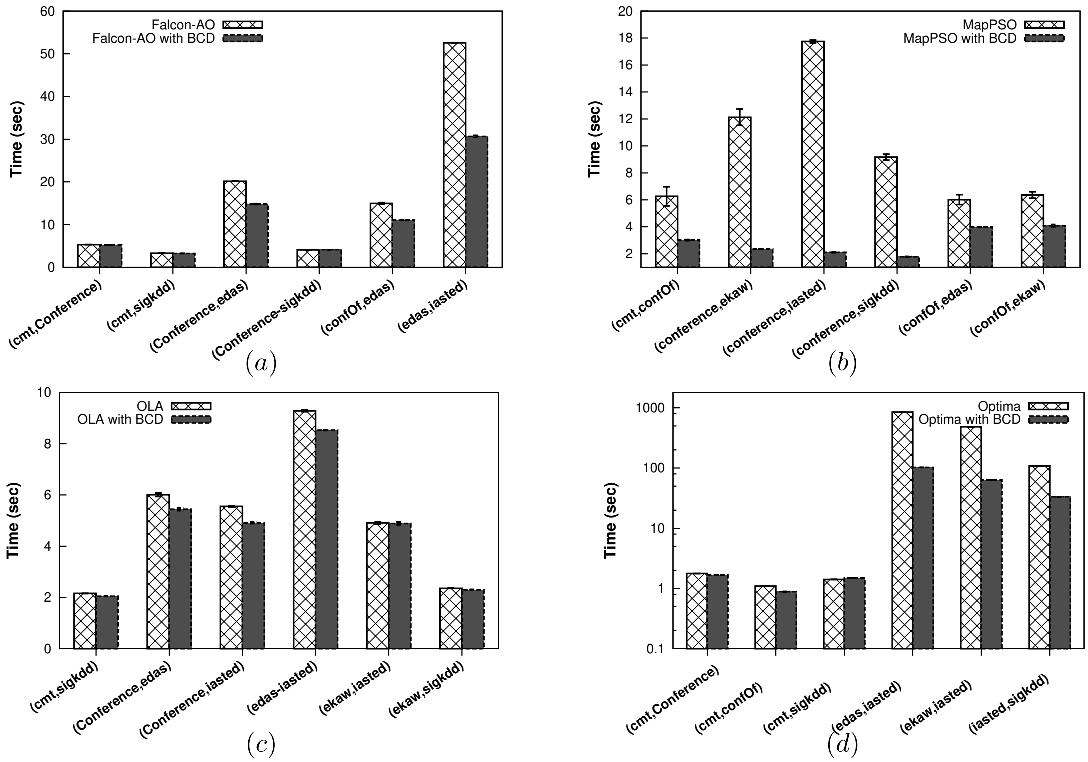


Figure 6: Average execution time consumed by, (a) Falcon-AO, (b) MapPSO, (c) OLA, and (d) Optima in their original form and with BCD, for 6 of the 21 ontology pairs from *conference* domain. We ran the algorithms for *all the pairs*, and selected ontology pairs which exhibited the three highest and the 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.

erated by Falcon-AO with BCD remained unchanged at 76.1% precision and 80% recall while the alignment from Optima with BCD improved to a precision of 96% and recall of 74.2%. Both Falcon-AO and Optima automatically utilized their ontology partitioning methods in order to align these mid-sized pairs.

In summary, the introduction of BCD led to significant reductions in convergence time for all four iterative algorithms on several 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, we did not observe a change in the F-measure for many of the pairs. Therefore, our empirical observations indicate that BCD does not have a significant adverse impact on the quality of the alignment.

5. Optimizing BCD using Ordering and Partitioning Schemes

As we 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, we explore other ways of ordering the blocks and partitioning the alignment variables in the context of the representative algorithms. These include:

1. *Ordered from roots to leaves*: Cycle over blocks of decreasing height starting with the block containing entities with the largest height.

2. *Ordered by similarity distribution*: Obtain an aggregate measure of the lexical similarity between the ontology entities participating in each block. The normalized distribution of similarities provides the likelihood of picking the next block.

3. *Both ontologies partitioned*: A block contains participating entities from each ontology that are at the same height.

4. *Subtree-based partitioning*: Transform the ontology into a tree and form a block of variables such that the participating entities are a part of a subtree of a predefined size.

5. *Random partitioning*: Form a block by randomly selecting alignment variables for inclusion.

While the partitioning and ordering utilized in the previous section are intuitive, our objective is to discover if other ways may further improve the run time performances of the algorithms. In subsequent experimentation, we exclude MapPSO from our representative set due to the randomness in its algorithm, which leads to comparatively high variability in its run times.

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 as illustrated in Fig. 7. Other ordering schemes could improve performance:

- We 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.

- We 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, we 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 (Nesterov, 2012).

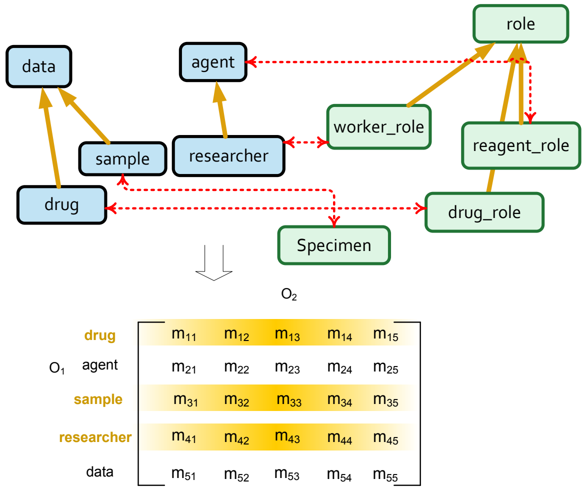


Figure 7: Presence or absence of correspondences between entities of two ontologies is represented in the match matrix. Concepts *drug*, *sample*, and *researcher* are all leaves and correspondences with these may be grouped into a block (highlighted). We may process this block first followed by the block containing *data* and *agent*. Alternately, we may reverse this ordering for optimizing blocks.

We 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. 8. 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.

Our second alternate ordering scheme involves determining the aggregate lexical similarity between the entities participating in a block. The distribution of similarities is normalized and the next block to consider is sampled from this distribution. Notice from Fig. 9 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*. We select 6 pairs, which exhibit the highest and lowest differences in average execution times to show in Fig. 9. 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

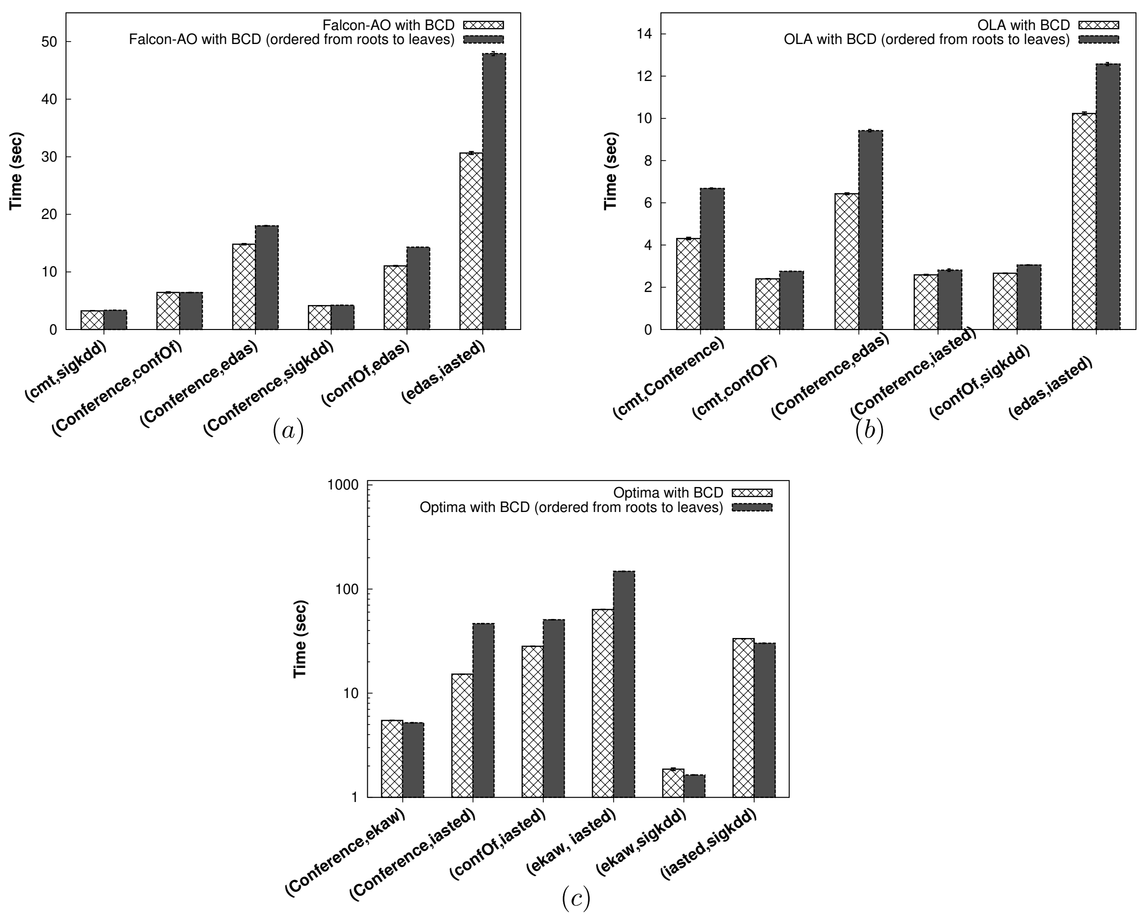


Figure 8: Average execution times of, (a) Falcon-AO, (b) OLA, and (c) Optima, with BCD that uses the initial ordering scheme and with BCD ordering the blocks from root(s) to leaves, for 6 of the 21 ontology pairs from the *conference* domain. While we ran the algorithms for all the pairs, we selected ontology pairs which exhibited the highest and lowest differences in average execution times. This alternate ordering increases the run times to convergence and we did not observe significant improvements in the F-measures.

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 we see in the context of Optima.

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:

- An extension of the initial scheme (Fig. 10(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 we illustrate in Fig. 10(b). Note that

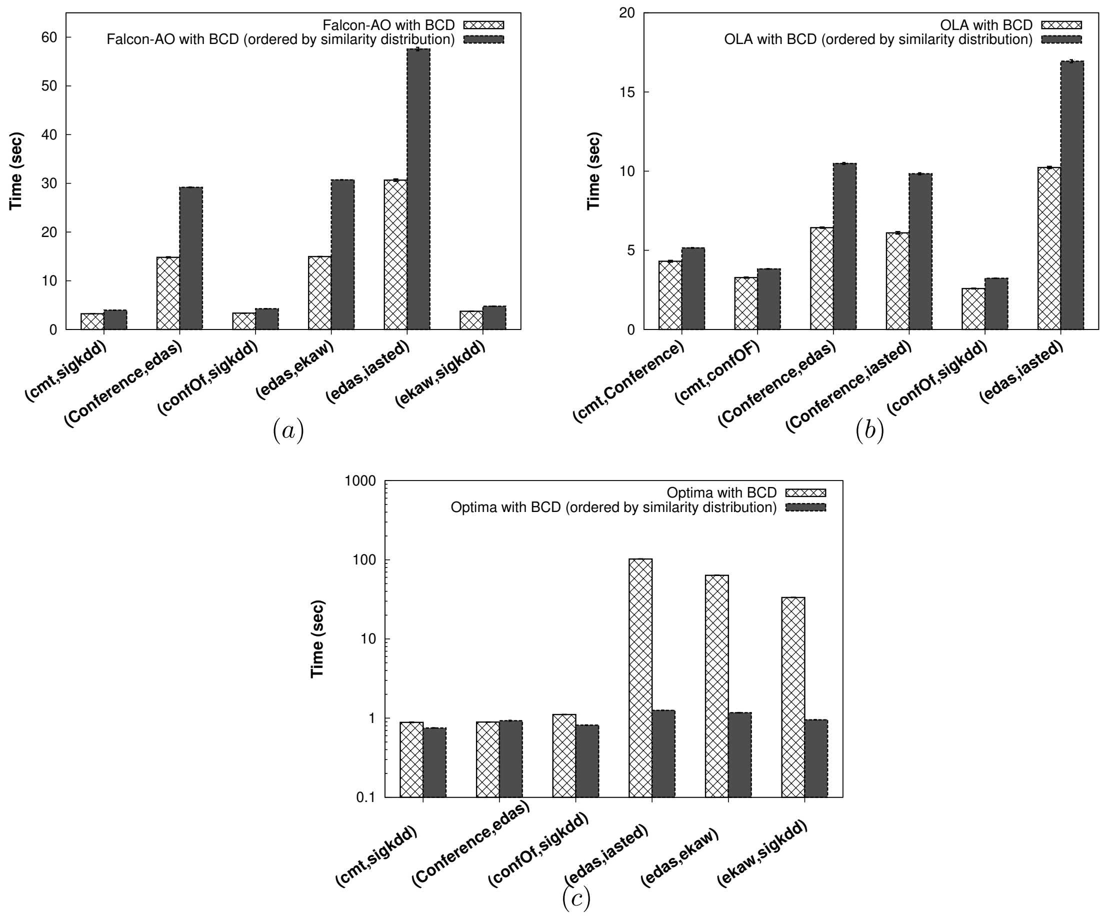


Figure 9: Average execution time consumed by, (a) Falcon-AO, (b) OLA and (c) Optima with BCD utilizing the previous ordering scheme and with BCD ordering the blocks by similarity distribution, for 6 of the 21 ontology pairs from *conference* domain. Although we ran the algorithms for all the pairs, we show 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.

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. We may utilize any graph search technique that handles repeated nodes, such as breadth-first search for graphs (Russell & Norvig, 2010), to obtain the tree. If the ontology has isolated graphs leading to separate trees, we use the owl:thing node to combine them into a single tree. Subsequently, we group those variables such that participating entities from each ontology are part of a subtree of a

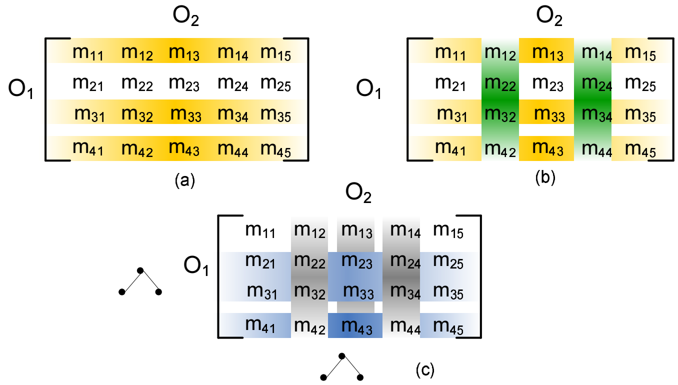


Figure 10: Matrices representing an intermediate alignment between entities of $\mathcal{O}_1$ and $\mathcal{O}_2$. (a) Identically shaded rows form a block of variables because the corresponding entities of $\mathcal{O}_1$ are at the same height. (b) Identically shaded rows and columns correspond to entities at the same heights 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. A fourth approach is to randomly select variables for inclusion into a block.

predefined size (Fig. 10(c)). We 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.

- A simple point for comparison would be a scheme that randomly selects alignment variables for inclusion in a block. With no clear way to determine how many variables to include in a block, we randomly inserted variables into 5 blocks.

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. We begin with the blocks of smaller height and proceed to those with increasing height. For Optima, we sample the blocks using a distribution based on the lexical similarity between participating entities.

As illustrated in Fig. 11, 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 12 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

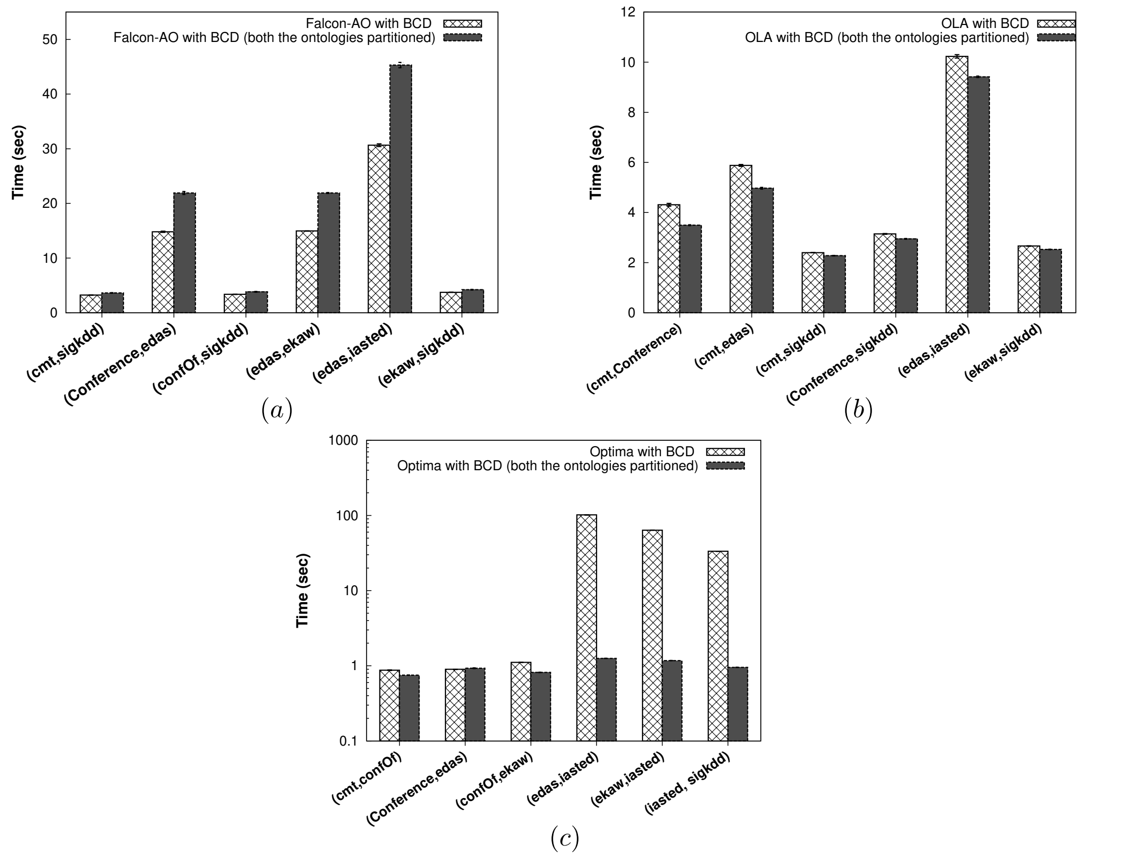


Figure 11: Execution times consumed by, (a) Falcon-AO, (b) OLA, and (c) Optima with BCD that uses blocks obtained by partitioning a single ontology and with BCD that utilizes partitions of both the ontologies, for 6 of the 21 ontology pairs from *conference* domain. Although we ran the algorithms for all the pairs, we 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.

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 to Optima.

Finally, in Fig. 13 we explore the impact of randomly partitioning the variables into blocks on all three alignment algorithms. Both Falcon-AO and OLA showed significant increases in execution time ($p < 0.01$) on the conference pairs. While Falcon-AO's precision improved by less than 1%, its recall dropped by 2% with an overall reduction in F-measure of 1%. OLA exhibited a minor increase in precision of 0.2% while the recall remained unchanged resulting in an increase of F-measure by 0.2%. Optima demonstrated mixed results as shown in Fig. 13(c) with the execution

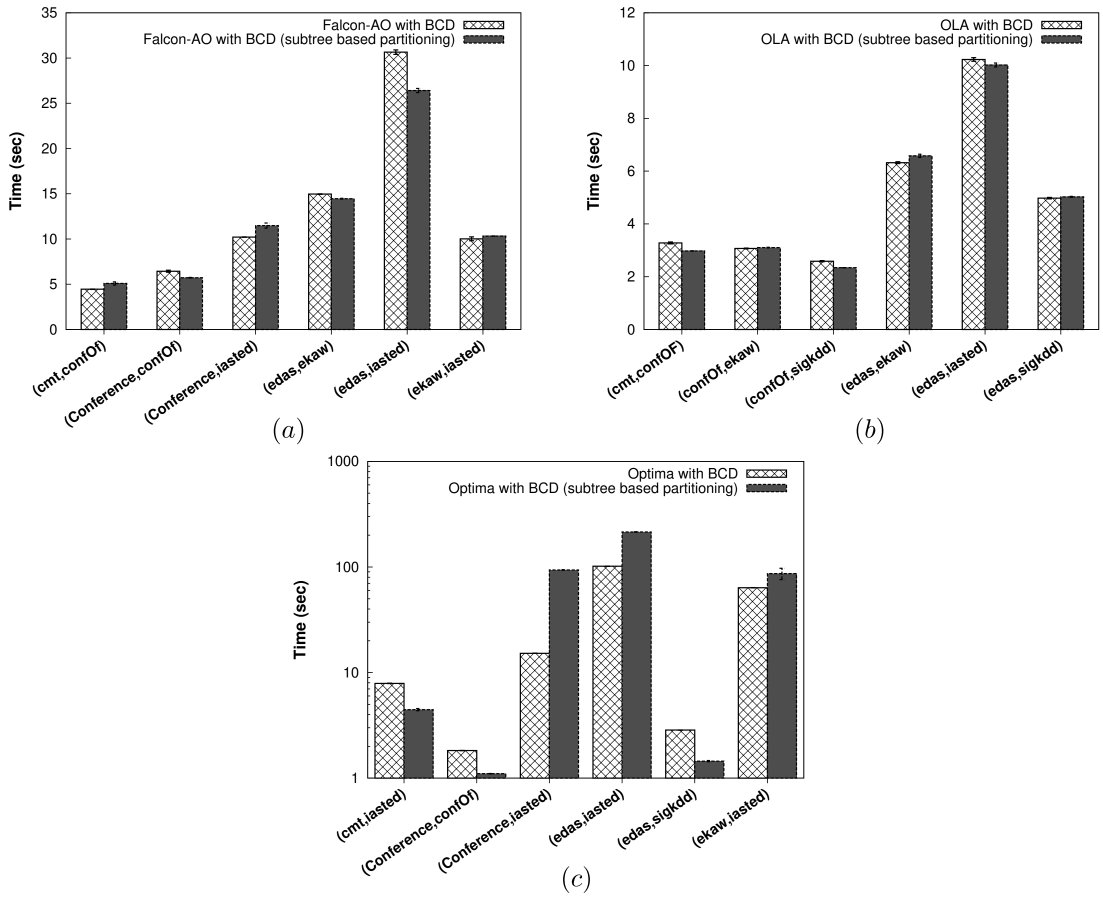


Figure 12: Execution times consumed by, (a) Falcon-AO, (b) OLA, and (c) Optima, with BCD that uses the default partitioning approach and with BCD that uses subtree-based partitioning, for 6 of the 21 ontology pairs from *conference* domain. We ran the algorithms for all the pairs of which we selected 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.

time increasing for some pairs while reducing for others. On the whole, we did not observe a statistically significant difference in execution times. Furthermore, BCD due to random partitioning did not improve beyond the seed alignment for many of the pairs, with an overall decrease in F-measure of 1% across all pairs.

In summary, a side-by-side comparison of the various block ordering and partitioning techniques discussed previously is presented in Fig. 14 for all three alignment algorithms on a single ontology pair, *(edas, iasted)*. We do not include the random partitioning as its alignment performance in terms of recall and precision was poor on many of the ontology pairs making it illsuited as a candidate. Differences in run time performance of the algorithms on *(edas, iasted)* is representative of their

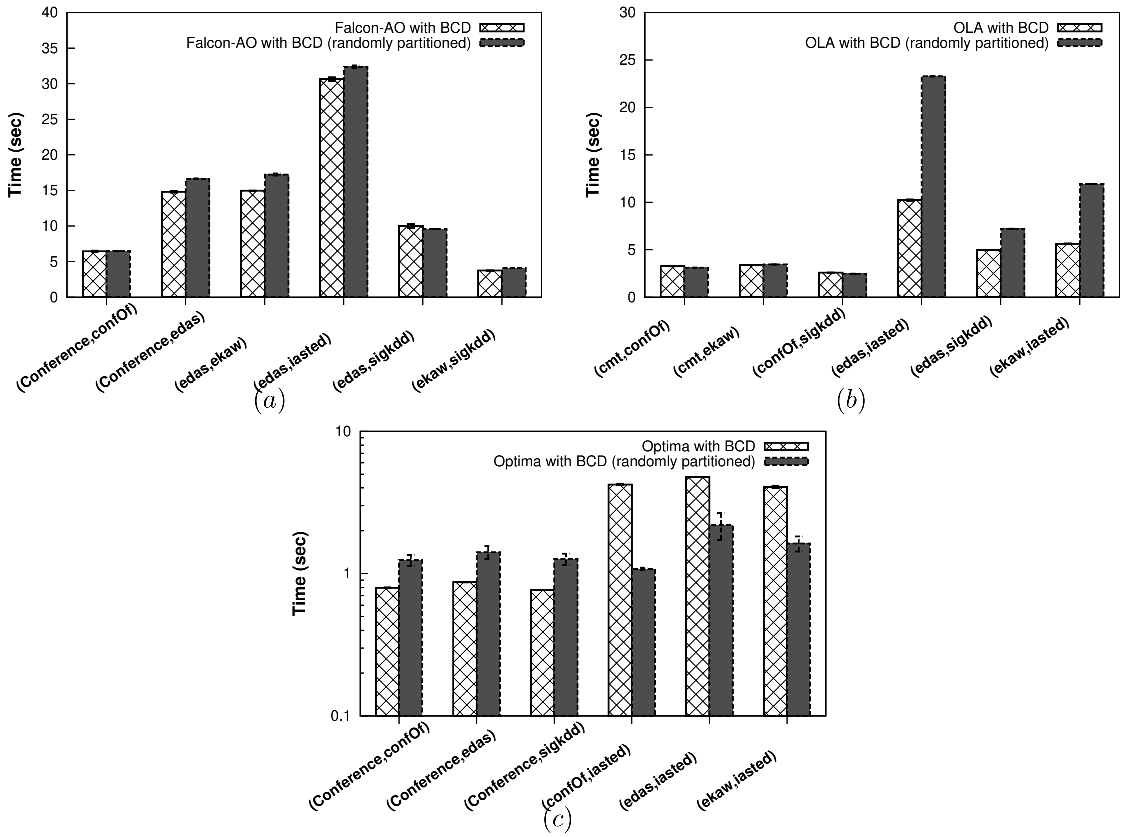


Figure 13: Execution times consumed by, (a) Falcon-AO, (b) OLA, and (c) Optima, with BCD that uses the default partitioning approach and with BCD that uses random partitioning. We show 6 of the 21 ontology pairs from *conference* domain. We ran the algorithms for all the pairs of which we selected ontology pairs that exhibited the highest and lowest differences in execution times. The total execution time of Falcon-AO for the complete conference track increases by 19.5 secs due to the random partitioning. OLA takes an additional 28 secs in total execution time while Optima saves 8.5 seconds over all the pairs at the expense of alignment quality.

performances on the larger data set in general. In particular, Falcon-AO's run time reduces on using subtree-based partitioning to obtain the blocks. OLA's run time reduces the most when both ontologies in the pair are partitioned using entity height, while Optima benefits from ordering blocks based on a preliminary measure of the similarity of the participating entities and forming blocks by partitioning both ontologies.

6. Aligning Large Biomedical Ontologies

Ontologies are becoming increasingly critical in the life sciences (Bodenreider & Stevens, 2006; Lambrix, Tan, Jakoniene, & Stromback, 2007) with multiple repositories such as Bio2RDF (Belleau et al., 2008), OBO Foundry (Smith et al., 2007) and NCBO's BioPortal (Musen et al., 2012)

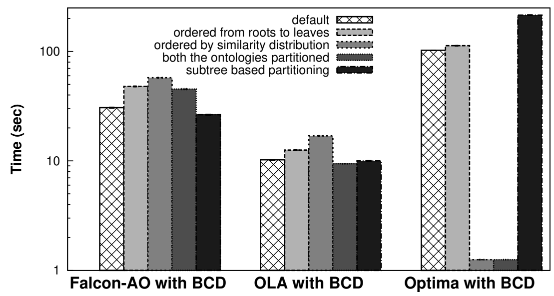


Figure 14: A side-by-side comparison of the performances of the three iterative algorithms using various block ordering and formation techniques. A single moderately large ontology pair, *(edas, iasted)*, is aligned. The "default" represents the iterative alignment algorithm with BCD where the blocks are ordered based on the height of the participating entities from the leaves to the root and a single ontology is partitioned to form the blocks. Differences in run times are indicative of the performance in general.

publishing a growing number of biomedical ontologies from different domains such as anatomy and molecular biology. For example, BioPortal hosts more than 370 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 (Shvaiko et al., 2012). In addition to multiple tracks with real-world test cases, the competition emphasizes on benchmark comparison tracks that use test pairs that are modifications of a single ontology pair in order to systematically identify the strengths and weaknesses of the alignment algorithms. One of the tracks on real-world ontology pairs involves aligning the ontology on the adult mouse anatomy with the human anatomy portion of NCI thesaurus (Golbeck et al., 2003), 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 (Ashburner et al., 2000) 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, we combed through more than 370 ontologies hosted at NCBO (Musen et al., 2012) and OBO Foundry (Smith et al., 2007), and isolated a community benchmark of

50 different biomedical ontology pairs. Thirty-two ontologies with sizes ranging from a few hundred to tens of thousands of entities constitute the pairs. We provide the list of ontologies participating in the benchmark and the ontology pairs in Appendix B. This new benchmark guides comparative evaluation of alignment algorithms to the context of a key application domain of biomedicine.

Our 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, we calculated the ratio of the correspondences posted in BioPortal for each ontology pair to the largest number of possible correspondences that could exist. We selected the 50 pairs with the largest such ratio. Existing correspondences will serve in the reference alignment. These include maps from the UMLS Metathesaurus and those that are crowd sourced. Nevertheless, our analysis reveals that the existing correspondences constitute just a small fraction of the total alignment that is possible between two ontologies.

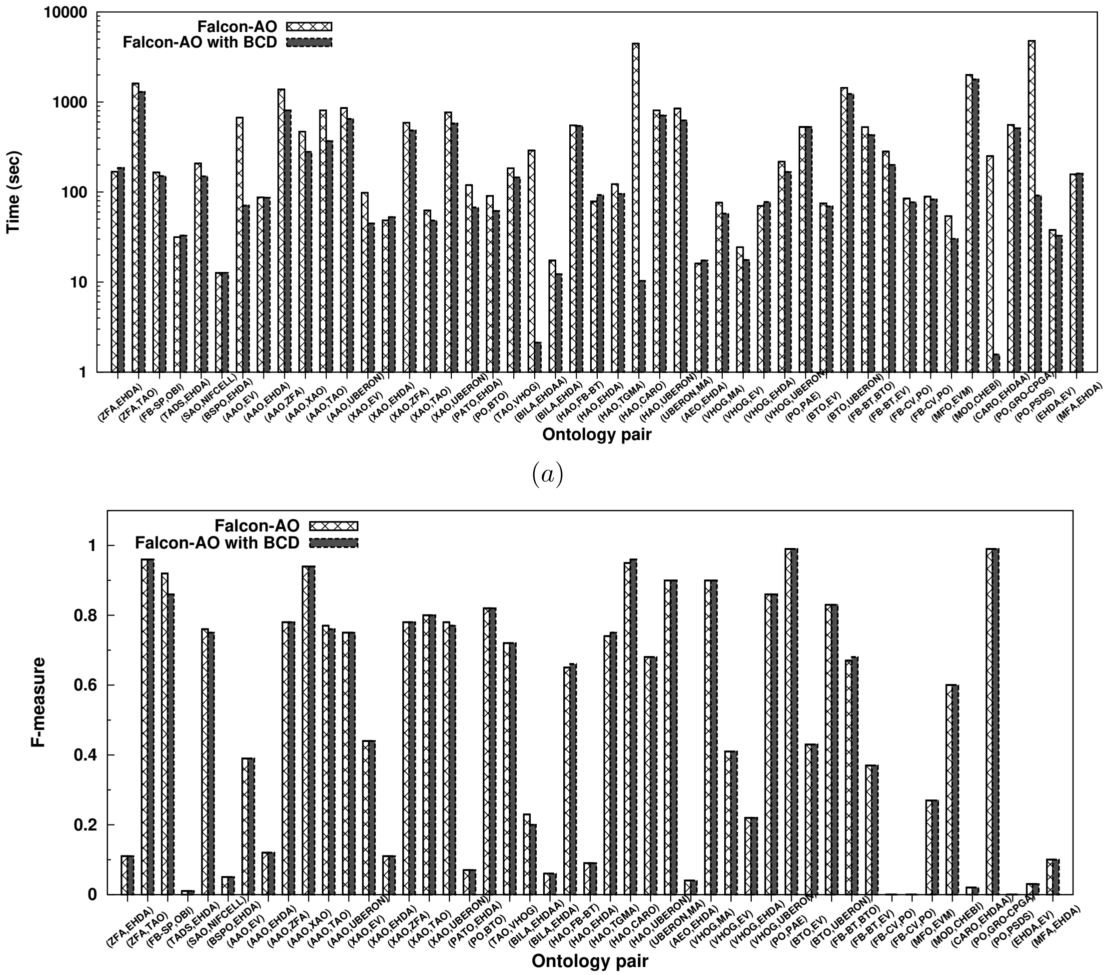
We sought to align the pairs in our new biomedical ontology alignment testbed using the BCD-enhanced representative algorithms. The obtained alignments are evaluated using the existing correspondences previously present in BioPortal; the reference alignments between the pairs are likely incomplete. A 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, 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 execution times and F-measure for all the pairs successfully aligned within an arbitrary window of 5 hours per pair by the BCD-enhanced algorithms are shown in Figs. 15 and 16. We point out that BCD speeds up the algorithms 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. Both Falcon-AO enhanced with BCD and without aligned 47 pairs within the time window. Falcon-AO was unable to parse one or both the ontologies in the remaining 3 pairs due to which no results are available for these. Falcon-AO with BCD aligned the pairs taking 3.7 hours less time in total than the original which consumed about 7.5 hours for all the pairs. We show the time for each pair in Fig. 15(a). A closer look reveals that Falcon-AO with BCD exhibited time greater than the default on 9 of the 47 pairs. Time on these few pairs did not exceed by more than 16 seconds and is due to performing the subtree-based partitioning of the variables for forming the blocks in BCD. The corresponding F-measure did not change significantly due to the use of BCD over all the pairs with the F-measure over all the pairs being 54.7%.

Optima enhanced with BCD aligned 42 pairs each within the time window compared to 30 pairs without BCD. Optima was unable to parse one or both ontologies in the remaining 8 pairs due to which no results are available for these. Focusing on the 30 pairs that were aligned by both within the time window (Fig. 16), Optima with BCD aligned these pairs in 2.3 hours taking 11.4 hours less time compared to the original algorithm. Simultaneously, it found an additional 269 correct correspondences across all the pairs with an increase in F-measure of about 2%.

LogMap, a fast non-iterative algorithm that targets biomedical ontologies returned alignments for all 50 pairs in 20 minutes of total time. It produced a precision and recall of 23.5% and 39.5% (F-measure = 29.5%), respectively over all the pairs. These are significantly less than those of



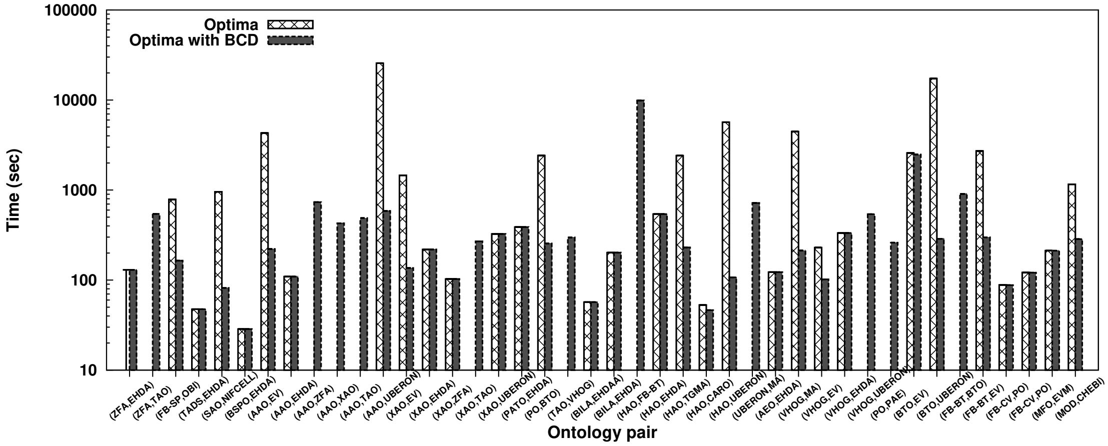
(a)

(b)

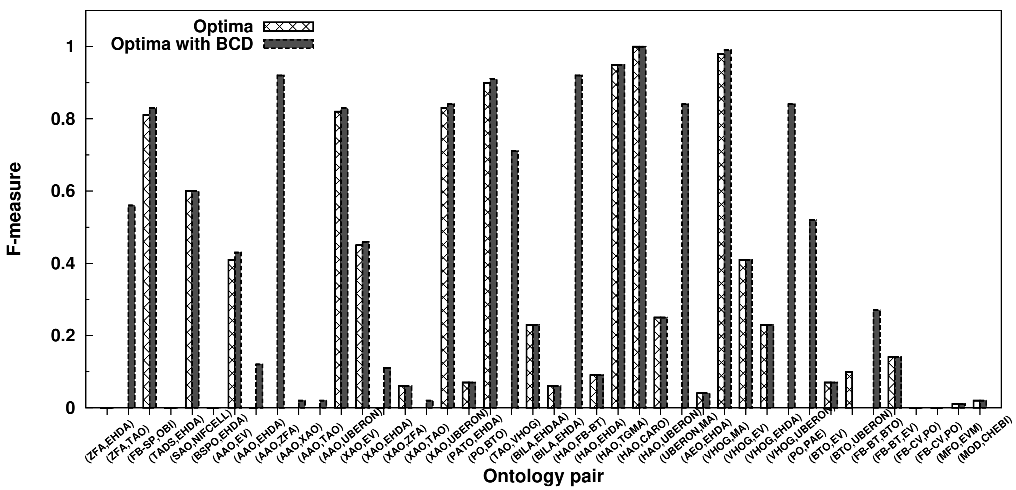
Figure 15: (a) Time consumed, and (b) F-measure attained by the original Falcon-AO and that with optimized BCD for 47 pairs of our large biomedical ontology testbed, respectively. Note that the time axis is in log scale. Ontology names are NCBO abbreviations. The alignment was performed on a Red Hat machine with Intel Xeon Core 2, processor speed of about 3 GHz with 8GB of memory.

Falcon-AO, which exhibited a precision and recall of 80.9% and 41.3% respectively, for the pairs it aligned. Optima with BCD exhibited a precision of 76.1% and a recall of 35.8% with an overall F-measure of 48.7%. While the recall is less than LogMap, the F-measure is significantly better due to the improved precision.

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



(a)



(b)

Figure 16: (a) Time consumed, and (b) F-measure attained by the original Optima and that with optimized BCD, for 42 pairs of our biomedical ontology testbed, respectively. Note that the time axis is in log scale.

7. Discussion

Performances of the iterative update and search techniques are impacted differently by various ways of formulating the blocks and the order of processing them. Importantly, the quality of the alignment may be adversely impacted. Nevertheless, the approach of grouping alignment variables into blocks based on the height of the participating entities in the ontologies is motivated by a recognized heuristic and leads to competitive performance with no observed negative impact on the precision and recall of the alignments. 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.

Given the BCD-based enhancement and optimization, 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 question, we compare with the performances of 18 algorithms that participated in the *conference* track of OAEI 2012 (Shvaiko et al., 2012). 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 that were tied or within 2% of each other. Optima+ ranked second after YAM++ when the alignment is evaluated using F2 measure due to its comparatively high recall.

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, we ran them separately on the 120 pairs on our 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 with OLA in the top half, and Optima+ and OLA in group 1 with respect to alignment quality. [6] While 7 competing algorithms completed the evaluation faster, 5 of these exhibit alignment quality that is substantially worse than that of the representative algorithms. In the absence of BCD, the representative algorithms would have ranked among the bottom third or exceeded the 5 hour cut off. Performance on the *anatomy* pair due to BCD would place both Falcon-AO and Optima+ in the top half of the 14 algorithms that participated in terms of run time and F-measure. Previously, Optima without BCD ranked in the bottom quarter.

The reductions in convergence time and the observed increases in precision of the alignment due to BCD is, in part, because of the optimized correspondences found for the previous coordinate block, which influence the selection of the correspondences for the current coordinate block. Furthermore, as we mentioned previously, limiting the randomly generated correspondences in MapPSO to the block instead of the whole ontology makes the search more guided. This is representative of the effect that BCD has on iterative search in general. Focusing on a single block significantly reduces the space of alignments over which iterative techniques must search thereby arriving at an optimum quicker. However, a greater number of these smaller optimization subproblems must be solved but as our results imply the smaller optimization problem offsets this expense.

Given that on integrating BCD the iterative algorithms converged to different values of the Q function during iterative search or different match matrices, M_*, during iterative update, which often produced better quality alignments, we 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. While this insight is not new (Euzenat et al., 2004), it is significant because it further reinforces the presence of local optima in the alignment space of these algorithms. This may limit the efficacy of iterative alignment techniques.

Falcon-AO and Optima+'s comparatively better performance measured using F-measure against the fast, non-iterative algorithm, LogMap, on the biomedical ontology alignment testbed indicates that iterative techniques continue to be among the best in the quality of the obtained alignment including on large ontology pairs. This motivates ways of making them efficient, such as BCD, and more scalable.

6. Note that MapPSO with BCD would have placed in the bottom third.

8. Conclusion and Future Work

While techniques for scaling automated alignment to large ontologies have been previously proposed, we presented a novel approach based on BCD to *speed up* the alignment process of an important class of algorithms. These algorithms are iterative and anytime demonstrating high quality alignments while often consuming more time than non-iterative algorithms. We 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. We 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, utilizing our default scheme of grouping those alignment variables such that the participating entities from one ontology in a block have the same height and optimizing the blocks in the order of increasing height, we observe an improvement in the precision of the alignments generated by some of the algorithms with no significant change in the recall.

While it is possible to improve on the run time performance of the default partitioning and ordering scheme by utilizing other schemes, we note that some of these may negatively impact the alignment quality. Subsequently, the default scheme is generally recommended for existing and new iterative alignment techniques that seek to utilize BCD.

The ability to improve quickly allows an iterative alignment algorithm to run until convergence if possible, 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. We observe that BCD does not promote scalability to large ontologies.

Finally, we 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. Our future work is to continue to focus on methods that would allow general and principled alignment approaches such as Falcon-AO and Optima to perform better on this testbed by producing better quality alignment for more pairs in less time, and on aligning other large biomedical ontologies that are in popular use such as SNOMED-CT and NCI. Consequently, we believe that our community benchmark could potentially drive future research toward pragmatic ontology alignment.

9. Acknowledgment

This research is supported in part by grant number R01HL087795 from the National Heart, Lung, And Blood Institute. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Heart, Lung, And Blood Institute or the National Institutes of Health. The authors thank Todd Minning and Rick Tarleton from the Center for Tropical and Emerging Diseases at the University of Georgia and Amit Sheth at Wright State University for useful discussions. The authors also thank the anonymous reviewers for feedback that benefited the article greatly.

Appendix A. Representative Iterative Algorithms Enhanced with BCD

We chose four representative iterative alignment algorithms, Falcon-AO, MapPSO, OLA and Optima in order to illustrate how iterative algorithms could be enhanced with BCD. In this section, we present each alignment algorithm in its original form and enhanced with BCD, to facilitate a direct comparison and a quick identification of the needed modifications.

FALCON-AO/GMO ($\mathcal{O}_1$, $\mathcal{O}_2$, η)
Initialize:
1. Iteration counter $i \leftarrow 0$
2. $G_1 \leftarrow$ AdjacencyMatrix ($\mathcal{O}_1$)
3. $G_2 \leftarrow$ AdjacencyMatrix ($\mathcal{O}_2$)
4. **For each** $m_{a\alpha} \in M^0$ **do**
5. $\quad m_{a\alpha} \leftarrow 1$
6. $M_* \leftarrow M^0$

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

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

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

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

Figure 17: (a) Iterative update in the structural matcher, GMO, in Falcon-AO. (b) Iterative update in GMO modified to perform BCD.

In Fig. 17, we show the iterative algorithm of the GMO component of Falcon-AO and its enhancement due to the use of BCD. AdjacencyMatrix ($\mathcal{O}_1$) (line 2 in Fig. 17(a)) 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. Notice that in each iteration of Fig. 17(b), just a block of variables, $M^i_{S_c}$, are updated using Eq. 10 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.

MapPSO's iterative search algorithm that performs particle swarm optimization and its modification due to BCD are shown in Fig. 18. The algorithm takes as input the number of particles, K, MAPPSO ($\mathcal{O}_1$, $\mathcal{O}_2$, K, η)

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: $P = \{M_1^0, M_2^0, \ldots, M_K^0\}$
5. Search $M_*^0 \leftarrow \arg\max_{M_k^0 \in P} Q(M_k^0)$

Iterate:

6. **Do**
7. $\quad i \leftarrow i + 1$
8. $\quad$ **For** $k \leftarrow 1, 2, \ldots, K$ **do**
9. $\qquad M_k^i \leftarrow UpdateParticle(M_k^i, M_*^{i-1})$
10. $\quad$ 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

(a)

MAPPSO-BCD ($\mathcal{O}_1$, $\mathcal{O}_2$, K, η)

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: $P = \{M_1^0, M_2^0, \ldots, M_K^0\}$
5. Create a partition of M: $\{M_{S_0}, M_{S_1}, \ldots, M_{S_C}\}$
6. Search $M_*^0 \leftarrow \arg\max_{M_k^0 \in P} Q(M_k^0)$

Iterate:

7. **Do**
8. $\quad c \leftarrow i \,\%\, (C + 1), i \leftarrow i + 1$
9. $\quad$ **For** $k \leftarrow 1, 2, \ldots, K$ **do**
10. $\qquad M_{k,S_c}^i \leftarrow UpdateBlock(M_{k,S_c}^i, M_*^{i-1})$
11. $\qquad M_{k,\tilde{S}}^i \leftarrow M_{k,\tilde{S}}^{i-1} \quad \forall \tilde{S} \in \tilde{S}_c$
12. $\quad$ Search $M_*^i \leftarrow \arg\max_{M_k^i \in P} Q_S(M_k^i)$
13. $\quad$ **If** $c = C$ **then**
14. $\qquad changed \leftarrow |Q(M_*^i) - Q(M_*^{i-1})| \geq \eta$?

    $\quad$ **else**
15. $\qquad changed \leftarrow$ true
16. **While** changed
17. Extract an alignment from M_*^i

(b)

Figure 18: (a) Iterative search in MapPSO. Objective function, Q, is as given in Eq. 4. (b) MapPSO's particle swarm based iterative algorithm enhanced with BCD.

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. The objective function, Q, is modified to Q_S in Fig. 18(b), such that it is calculated for the coordinate block of interest. A coordinate block in each particle, k, 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. 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.

OLA's iterative algorithm is shown in Fig. 19 (a), and its enhancement due to the use of BCD in Fig. 19(b). 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. Observe that we cycle through the blocks in the BCD enhanced algorithm in Fig. 19 (b) and only the coordinates belonging to the current block, $M_{S_c}^i$, are updated in lines 8-11.

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

Initialize:

1. Iteration counter $i \leftarrow 0$
2. Fill real-valued matrix, M^0, with lexical similarity
3. $M_* \leftarrow M^0$

Iterate:

4. **Do**
5. $\quad i \leftarrow i+1$
6. $\quad$ **for each** $m_{a\alpha} \in M^i$
7. $\quad\quad$ **if** the types of x_a and y_α are the same **then**
8. $\quad\quad\quad m_{a\alpha} \leftarrow \sum_{\mathcal{F} \in \mathcal{N}(x_a, y_\alpha)} w_{\mathcal{F}}^{a\alpha} SetSim(\mathcal{F}(x_a), \mathcal{F}(y_\alpha))$
9. $\quad\quad$ **else**
10. $\quad\quad\quad m_{a\alpha} \leftarrow 0$
11. $\delta \leftarrow Dist(M^i, M_*)$
12. $M_* \leftarrow M^i$
13. **While** $\delta \geq \eta$
14. Extract an alignment from M_*

(a)

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: $\{M_{S_0}, M_{S_1}, \ldots, M_{S_C}\}$
4. $M_* \leftarrow M^0$

Iterate:

5. **Do**
6. $\quad c \leftarrow i \,\%\, (C+1), i \leftarrow i+1$
7. $\quad$ **for each** $m_{a\alpha} \in M^i_{S_c}$
8. $\quad\quad$ **if** the types of x_a and y_α are the same **then**
9. $\quad\quad\quad m_{a\alpha} \leftarrow \sum_{\mathcal{F} \in \mathcal{N}(a, \alpha)} w_{\mathcal{F}}^{a\alpha} SetSim(\mathcal{F}(a), \mathcal{F}(\alpha))$
10. $\quad\quad$ **else**
11. $\quad\quad\quad m_{a\alpha} \leftarrow 0$
12. $M^i_{\tilde{S}} = M^{i-1}_{\tilde{S}} \quad \forall \tilde{S} \in \tilde{S}_c$
13. **If** $c = C$ **then**
14. $\quad \delta \leftarrow Dist(M^i, M_*)$

    else
15. $\quad \delta$ is a high value
16. $M_* \leftarrow M^i$
17. **While** $\delta \geq \eta$
18. Extract an alignment from M_*

(b)

Figure 19: (a) OLA iteratively updates the alignment matrix using a combination of neighboring similarity values. (b) OLA's BCD-integrated iterative ontology alignment algorithm.

Finally, in Fig. 20, we outline the iterative search undertaken by Optima and its modification due to BCD. Optima's expectation-maximization based iterative search uses binary matrix, M^i, to represent an alignment. Objective function, Q, is defined in Eq. 6. The search for an improved alignment in line 8 is implemented using the two steps of expectation and maximization. Iterations terminate when no sample $M^i \in \mathcal{M}$, which improves the objective function, Q further, is available. The search is modified in Fig. 20 (b) to explore a reduced search space, $\mathcal{M}_{S_c}$, as we cycle through the blocks. Both the objective function, Q_S, and the prior operate on a single coordinate block.

Appendix B. Biomedical Ontology Alignment Benchmark

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

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

Initialize:

1. Iteration counter $i \leftarrow 0$
2. **For all** $\alpha \in \{1, 2, \ldots, |V_2|\}$ **do**
3. $\quad \pi_\alpha^0 \leftarrow \frac{1}{|V_2|}$
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. $\quad i \leftarrow i + 1$
8. $\quad$ Search $M_*^i \leftarrow \underset{M \in \mathcal{M}}{\arg\max}\, Q(M|M_*^{i-1})$
9. $\quad \pi_\alpha^i \leftarrow \frac{1}{|V_1|} \sum_{a=1}^{|V_1|} Pr(y_\alpha | x_a, M_*^{i-1})$
10. **While** $M_*^i \neq M_*^{i-1}$
11. Extract an alignment from M_*^i

(a)

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

Initialize:

1. Iteration counter $i \leftarrow 0$
2. **For all** $\alpha \in \{1, 2, \ldots, |V_2|\}$ **do**
3. $\quad \pi_\alpha^0 \leftarrow \frac{1}{|V_2|}$
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: $\{M_{S_0}, M_{S_1}, \ldots, M_{S_C}\}$

Iterate:

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

(b)

Figure 20: (a) Optima's expectation-maximization based iterative search algorithm. (b) Expectation-maximization based iterative ontology alignment of Optima with BCD.

ontologies, the testbed could serve as a comprehensive large ontology benchmark. Existing correspondences in 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 370 ontologies hosted at NCBO and OBO Foundry, and isolated a benchmark of 50 different biomedical ontology pairs. Thirty-two ontologies with sizes ranging from a few hundred to tens of thousands of entities constitute the pairs, and are listed in Table 2. We provide a snapshot of the full benchmark in Table 3. The testbed with reference alignments is available for download at `http://tinyurl.com/n4t2ns3`. 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 in creating the testbed:

1. We selected ontologies, which exist in either OWL or RDF models.

2. We paired the ontologies and ordered the pairs by the percentage of available correspondences. This is 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 based on the above ratio are 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.

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 (EHDAA)	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

Table 2: Selected ontologies from NCBO in the biomedical ontology alignment testbed and the number of named classes and properties in each. Notice that this data set includes large ontologies. NCBO abbreviations for these ontologies are also provided.

Biomedical ontology alignment testbed		
Ontology $\mathcal{O}_1$	**Ontology $\mathcal{O}_2$**	$\lvert V_1\rvert \times \lvert V_2\rvert$
Common Anatomy Reference Ontology (CARO)	Human developmental anatomy (EHDAA)	115,700
Bilateria anatomy (BILA)	Human developmental anatomy (EHDAA)	263,796
Bilateria anatomy (BILA)	Human developmental anatomy (EHDAA)	263,796
Spatial Ontology (BSPO)	Human developmental anatomy (EHDAA)	298,506
Plant Growth and Development Stage (PO_PSDA)	Plant Ontology (PO)	446,970
Anatomical Entity Ontology (AEO)	Human developmental anatomy (EHDAA)	550,732
FlyBase Controlled Vocabulary (FBcv)	Cereal plant gross anatomy (GR_CPGA)	1,042,670
FlyBase Controlled Vocabulary (FBcv)	Plant Ontology (PO)	1,301,285
Tick gross anatomy (TADS)	Human developmental anatomy (EHDAA)	1,453,192
Amphibian gross anatomy (AAO)	Xenopus anatomy and development (XAO)	1,668,723
Cereal plant gross anatomy (GR_CPGA)	Plant Ontology (PO)	2,012,950
Plant Anatomy (PO_PAE)	Plant Ontology (PO)	2,012,950
Subcellular Anatomy Ontology (SAO)	NIF Cell (NIF_Cell)	2,219,163
Xenopus anatomy and development (XAO)	Expressed Sequence Annotation for Humans (eVOC)	2,367,234
Xenopus anatomy and development (XAO)	Human developmental anatomy (EHDAA)	2,408,874
vertebrate Homologous Organ Groups (sHOG)	Expressed Sequence Annotation for Humans (eVOC)	2,692,416
vertebrate Homologous Organ Groups (sHOG)	Human developmental anatomy (EHDAA)	2,739,776
Xenopus anatomy and development (XAO)	Zebrafish anatomy and development (ZFA)	2,902,308
Xenopus anatomy and development (XAO)	Teleost Anatomy Ontology (TAO)	3,163,599
vertebrate Homologous Organ Groups (sHOG)	Mouse adult gross anatomy (MA)	3,530,688
Hymenoptera Anatomy Ontology (HAO)	Mosquito gross anatomy (TGMA)	3,597,520
Teleost Anatomy Ontology (TAO)	vertebrate Homologous Organ Groups (sHOG)	3,598,176
Amphibian gross anatomy (AAO)	Expressed Sequence Annotation for Humans (eVOC)	3,645,222
Amphibian gross anatomy (AAO)	Human developmental anatomy (EHDAA)	3,709,342
Hymenoptera Anatomy Ontology (HAO)	Human developmental anatomy (EHDAA)	4,466,020
Amphibian gross anatomy (AAO)	Zebrafish anatomy and development (ZFA)	4,469,164
Amphibian gross anatomy (AAO)	Teleost Anatomy Ontology (TAO)	4,871,517
Expressed Sequence Annotation for Humans (eVOC)	Human developmental anatomy (EHDAA)	5,262,036
Phenotypic quality (PATO)	Human developmental anatomy (EHDAA)	5,278,234
Zebrafish anatomy and development (ZFA)	Human developmental anatomy (EHDAA)	6,451,432
Plant Anatomy (PO_PAE)	BRENDA tissue / enzyme source (BTO)	6,526,530
Teleost Anatomy Ontology (TAO)	Human developmental anatomy (EHDAA)	7,032,246
Xenopus anatomy and development (XAO)	Uber anatomy ontology (UBERON)	7,593,054
Zebrafish anatomy and development (ZFA)	Teleost Anatomy Ontology (TAO)	8,472,732
		Continued on next page

Ontology 1	**Ontology 2**	$\|V_1\| \times \|V_2\|$
vertebrate Homologous Organ Groups (sHOG)	Uber anatomy ontology (UBERON)	8,636,096
Medaka fish anatomy and development (MFO)	Expressed Sequence Annotation for Humans (eVOC)	9,910,092
Medaka fish anatomy and development (MFO)	Human developmental anatomy (EHDAA)	10,084,412
BRENDA tissue / enzyme source (BTO)	Expressed Sequence Annotation for Humans (eVOC)	11,686,086
Amphibian gross anatomy (AAO)	Uber anatomy ontology (UBERON)	11,692,282
BRENDA tissue / enzyme source (BTO)	Human developmental anatomy (EHDAA)	11,891,646
Hymenoptera Anatomy Ontology (HAO)	Uber anatomy ontology (UBERON)	14,077,420
Hymenoptera Anatomy Ontology (HAO)	Drosophila gross anatomy (FBbt)	15,048,210
Expressed Sequence Annotation for Humans (eVOC)	Uber anatomy ontology (UBERON)	16,586,556
Drosophila gross anatomy (FBbt)	Expressed Sequence Annotation for Humans (eVOC)	17,730,378
Zebrafish anatomy and development (ZFA)	Uber anatomy ontology (UBERON)	20,335,672
Uber anatomy ontology (UBERON)	Mouse adult gross anatomy (MA)	21,750,708
Fly taxonomy (FBsp)	Ontology for Biomedical Investigations (OBI)	23,340,663
BRENDA tissue / enzyme source (BTO)	Uber anatomy ontology (UBERON)	37,483,866
Drosophila gross anatomy (FBbt)	BRENDA tissue / enzyme source (BTO)	40,068,783
Protein modification (MOD)	Chemical entities of biological interest (CHEBI)	42,106,860

Table 3: The biomedical ontology pairs in our testbed sorted in terms of $|V_1| \times |V_2|$. This metric is illustrative of the complexity of aligning the pair.

References

Arimoto, S. (1972). An algorithm for computing the capacity of arbitrary discrete memoryless channels. *IEEE Transactions on Information Theory*, *18*(1), 14–20.

Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D., Butler, H., Cherry, J. M., Davis, A. P., Dolinski, K., Dwight, S. S., Eppig, J. T., Harris, M. A., Hill, D. P., Issel-Tarver, L., Kasarskis, A., Lewis, S., Matese, J. C., Richardson, J. E., Ringwald, M., Rubin, G. M., & Sherlock, G. (2000). Gene ontology: tool for the unification of biology. the gene ontology consortium.. *Nature genetics*, *25*(1), 25–29.

Baader, F., Horrocks, I., & Sattler, U. (2003). Description logics as ontology languages for the semantic web. In *Lecture Notes in Artificial Intelligence*, pp. 228–248. Springer-Verlag.

Belleau, F., Nolin, M.-A., Tourigny, N., Rigault, P., & Morissette, J. (2008). (bio2rdf): Towards a mashup to build bioinformatics knowledge systems. *Journal of Biomedical Informatics*, *41*(5), 706–716.

Blahut, R. E. (1972). Computation of channel capacity and rate-distortion functions. *IEEE Transactions on Information Theory*, *18*, 460–473.

Bock, J., & Hettenhausen, J. (2010). Discrete particle swarm optimisation for ontology alignment. *Information Sciences*, *192*, 1–22.

Bodenreider, O., & Stevens, R. (2006). Bio-ontologies: current trends and future directions. *Brief Bioinform*, *7*, 256–274.

Cruz, I. F., Stroe, C., & Palmonari, M. (2012). Interactive user feedback in ontology matching using signature vectors. In *IEEE 28th International Conference on Data Engineering*, pp. 1321–1324. IEEE Computer Society.

Doan, A., Madhavan, J., Domingos, P., & Halevy, A. (2003). Ontology matching: A machine learning approach. In *Handbook on Ontologies in Information Systems*, pp. 397–416. Springer.

Doshi, P., Kolli, R., & Thomas, C. (2009). Inexact matching of ontology graphs using expectation-maximization. *Web Semantics: Science, Services and Agents on the World Wide Web*, *7*(2), 90–106.

Euzenat, J., Loup, D., Touzani, M., & Valtchev, P. (2004). Ontology alignment with OLA. In *In Proceedings of the 3rd EON Workshop, 3rd International Semantic Web Conference*, pp. 59–68. CEUR-WS.

Euzenat, J., & Valtchev, P. (2004). Similarity-based ontology alignment in OWL-lite. In *European Conference on Artificial Intelligence (ECAI)*, pp. 333–337.

Euzenat, J., & Shvaiko, P. (2007). *Ontology Matching*. Springer.

Fessler, J. A., & Hero, A. O. (1994). Space-alternating generalized expectation-maximization algorithm. *IEEE Transactions on Signal Processing*, *42*, 2664–2677.

Fessler, J. A., & Kim, D. (2011). Axial block coordinate descent (abcd) algorithm for X-ray CT image reconstruction. In *Proceedings of Fully 3D Image Reconstruction in Radiology and Nuclear Medicine*, pp. 262–265.

Golbeck, J., Fragoso, G., Hartel, F., Hendler, J., Oberthaler, J., & Parsia, B. (2003). The national cancer institutes thesaurus and ontology. *Journal of web semantics*, *1*(1), 75–80.

Hanif, M. S., & Aono, M. (2009). Anchor-flood: results for OAEI 2009. In *Proceedings of the Workshop on Ontology Matching at 8th International Semantic Web Conference*, pp. 127–134.

Hayes, J., & Gutierrez, C. (2004). Bipartite graphs as intermediate model for RDF. In *Proceedings of the 3rd International Semantic Web Conference (ISWC)*, Lecture Notes in Computer Science, pp. 47–61. Springer Berlin / Heidelberg.

Hero, A. O., & Fessler, J. A. (1993). Asymptotic convergence properties of (em)-type algorithms. Tech. rep., Department of EECS, Univ. of Michigan, Ann Arbor, MI.

Hu, W., Jian, N., Qu, Y., & Wang, Y. (2005). GMO: A graph matching for ontologies. In *K-Cap Workshop on Integrating Ontologies*, pp. 43–50.

Hu, W., Zhao, Y., & Qu, Y. (2006). Partition-based block matching of large class hierarchies. In *Proceedings of the 1st Asian Semantic Web Conference (ASWC)*, pp. 72–83.

Hughes, T. C., & Ashpole, B. C. (2004). The semantics of ontology alignment. In *Information Interpretation and Integration Conference (I3CON)*.

Jean-Mary, Y. R., Shironoshita, E. P., & Kabuka, M. R. (2009). Ontology matching with semantic verification. *Web Semantics: Science, Services and Agents on the World Wide Web*, *7*(3), 235–251.

Jian, N., Hu, W., Cheng, G., & Qu, Y. (2005). Falcon-AO: Aligning ontologies with Falcon. In *K-Cap Workshop on Integrating Ontologies*, pp. 87–93.

Jiménez-Ruiz, E., & Grau, B. C. (2011). LogMap: Logic-based and scalable ontology matching. In *International Semantic Web Conference*, pp. 273–288.

Kirsten, T., Gross, A., Hartung, M., & Rahm, E. (2011). GOMMA: a component-based infrastructure for managing and analyzing life science ontologies and their evolution. *Journal of Biomedical Semantics*, *2*, 6.

Lambrix, P., Tan, H., Jakoniene, V., & Stromback, L. (2007). *Biological ontologies In: Semantic Web: Revolutionizing Knowledge Discovery in the Life Sciences*, pp. 85–99. Springer.

Li, Y., Li, J., & Tang, J. (2007). RiMOM: Ontology alignment with strategy selection. In *Proceedings of the 6th International and 2nd Asian Semantic Web Conference (ISWC2007+ASWC2007)*, pp. 51–52.

McGuinness, D., & Harmelen, F. (2004). Owl web ontology language overview. Tech. rep., W3C.

Melnik, S., Garcia-molina, H., & Rahm, E. (2002). Similarity flooding: A versatile graph matching algorithm. In *ICDE: Int. Conference on Data Engineering*, pp. 117–128.

Musen, M. A., Noy, N. F., Shah, N. H., Whetzel, P. L., Chute, C. G., Storey, M.-A. D., & Smith, B. (2012). The national center for biomedical ontology. *JAMIA*, *19*(2), 190–195.

Nesterov, Y. (2012). Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, *22*(2), 341–362.

Ngo, D., & Bellahsene, Z. (2012). YAM++ : A multi-strategy based approach for ontology matching task. In *International Conference on Knowledge Engineering and Knowledge Management*, pp. 421–425.

Pintér, J. D. (2000). Yair censor and stavros a. zenios, parallel optimization – theory, algorithms, and applications. *Journal of Global Optimization*, *16*, 107–108.

Rahm, E. (2011). Towards large-scale schema and ontology matching. In Bellahsene, Z., Bonifati, A., & Rahm, E. (Eds.), *Schema Matching and Mapping*, pp. 3–27. Springer.

Russell, S. J., & Norvig, P. (2010). *Artificial Intelligence - A Modern Approach (3rd edition)*. Pearson Education.

Saha, A., & Tewari, A. (2013). On the non-asymptotic convergence of cyclic coordinate descent methods. *SIAM Journal on Optimization*, *23*(1), 576–601.

Seddiqui, M. H., & Aono, M. (2009). 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.

Shvaiko, P., & Euzenat, J. (2013). Ontology matching: State of the art and future challenges. *IEEE Transactions on Knowledge and Data Engineering*, *25*(1), 158–176.

Shvaiko, P., Euzenat, J., Heath, T., Quix, C., Mao, M., & Cruz, I. F. (Eds.). (2011). *Proceedings of the 6th International Workshop on Ontology Matching*, Vol. 814 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Shvaiko, P., Euzenat, J., Kementsietsidis, A., Mao, M., Noy, N., & Stuckenschmidt, H. (Eds.). (2012). *Results of the Ontology Alignment Evaluation Initiative (OAEI) 2012*, Vol. 946 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Shvaiko, P., Euzenat, J., Srinivas, K., Mao, M., & Jiménez-Ruiz, E. (Eds.). (2013). *Preliminary Results of the Ontology Alignment Evaluation Initiative (OAEI) 2013*, Vol. 1111 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Smith, B., Ashburner, M., Rosse, C., Bard, J., Bug, W., Ceusters, W., Goldberg, L. J., Eilbeck, K., Ireland, A., Mungall, C. J., Leontis, N., Rocca-Serra, P., Ruttenberg, A., Sansone, S.-A., Scheuermann, R. H., Shah, N., Whetzel, P. L., & Lewis, S. (2007). The OBO foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature Biotechnology*, *25*(11), 1251–1255.

Stoutenburg, S. K., Kalita, J., Ewing, K., & Hines, L. M. (2010). Scaling alignment of large ontologies. *International Journal of Bioinformatics Research and Applications*, *6*, 384–401.

Thayasivam, U., & Doshi, P. (2012a). Improved convergence of iterative ontology alignment using block-coordinate descent. In *Twenty-Sixth Conference on Artificial Intelligence (AAAI)*, pp. 150–156.

Thayasivam, U., & Doshi, P. (2012b). Optima+ results for OAEI 2012. In *Workshop on Ontology Matching at 11th International Semantic Web Conference (ISWC)*. Vol. 946 of CEUR-WS.org.

Tseng, P. (2001). Convergence of block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, *109*, 475–494.

Wang, P., & Xu, B. (2009). Lily: Ontology alignment results for OAEI 2008. In *Proceedings of the Workshop on Ontology Matching at 7th International Semantic Web Conference (ISWC).*