Ontology Alignment in the Cloud

Jürgen Bock, Alexander Lenk, and Carsten Dänschel

FZI Research Center for Information Technology, Karlsruhe, Germany
{bock,lenk,daenschel}@fzi.de

Abstract. The problem of ontology alignment is prominent for applications that operate on integrated semantic data. With ontologies becoming numerous and increasingly large in size, scalability is an important issue for alignment tools. This work introduces a novel approach for computing ontology alignments using cloud infrastructures. An alignment algorithm based on particle swarm optimisation is deployed on a cloud infrastructure, taking advantage of its ability to harness parallel computation resources. The deployment is done with a focus on parallel efficiency, taking into account both communication latency and computational inhomogeneity among parallel execution units. Complementing previous experiments showing the effectiveness of the alignment algorithm, this paper contributes an experiment executed "in the cloud", which demonstrates the scalability of the approach by aligning two large ontologies from the biomedical domain.

1 Introduction

Ontology alignment is a problem prominent in semantic applications, the semantic web, and the linked data web. It is based on the observation that ontologies[1] are heterogeneous models, often representing a similar or equal domain of interest, and hence have a certain overlap. Accordingly an ontology alignment is defined as a set of correspondences between ontological entities, *i.e.* classes, properties, and individuals, of two ontologies.

Two examples of where ontology alignment plays an important role are the linked data web [2] and the biomedical domain [21]. These examples also demonstrate the necessity for alignment tools to be able to deal with very large ontologies, which currently constitutes a problem for most alignment algorithms. Furthermore, it can be observed that ontologies evolve gradually, *i.e.* changes typically occur by adding / removing / modifying single entities while the largest part of the ontology remains unchanged. Thus, due to the typically slow changes, alignments do not need to be recomputed completely each time, but rather incrementally maintained and adjusted according to the evolving ontologies. Furthermore, in typical information systems that utilise alignments, the refresh period

[1] There is some disagreement among semantic web and linked data web communities about whether data sources in the linked data web can be called ontologies. In this paper the term *ontology* is used to cover all types of semantic data sources.

for alignments is not time-critical, *i.e.* alignments do not need to be recomputed ad-hoc, but can be adjusted offline, *e.g.* in nightly alignment jobs.

This paper addresses the scalability problem of ontology alignment by utilising cloud computing as a massively parallel computation infrastructure. The algorithm to be deployed in the cloud is an alignment algorithm based on particle swarm optimisation (PSO) [4]. A PSO-based approach to the ontology alignment problem has several characteristics that meet the aforementioned observations:

1. The algorithm provides a meta-heuristic, which is independent of the objective function to be optimised. Hence it is straightforward to exchange or adapt the objective function according to the alignment scenario at hand.
2. The algorithm works incrementally, which has two beneficial effects: Firstly, the algorithm can be interrupted at any time providing the best alignment that has been discovered so far. Secondly, the algorithm can be provided with an initial (partial) alignment as a start configuration to be refined. This in particular allows for alignment evolution of gradually changing ontologies.
3. The algorithm is inherently parallelisable allowing for efficient execution on parallel computing infrastructures.

The first two issues are inherent to the application of the PSO meta-heuristic. However, the usability of the approach depends on an efficient deployment on parallel computing infrastructures. Computation and adjustment of alignments on an irregular basis usually does not justify the acquisition of large parallel hardware infrastructures.

Cloud computing infrastructures are scalable and can be used for data intensive parallelisable jobs, as it has successfully been shown *e.g.* in the field of bio-informatics [20]. Recently, cloud computing has also been recognised as a promising technique to provide scalability for web data processing [18]. The use of cloud infrastructures provides the possibility to access a large number of computing resources in a convenient way without the need of operating one's own, expensive data centre. Many offerings by cloud providers such as Amazon Web Services™ (AWS)[2] are based on a pay-per-use pricing model with an hourly rate. This allows for flexible use of computation resources, since accessing large amounts of computation power for a short period of time comes at the same costs as using just a few resources for a long period of time.

The remainder of this paper is structured as follows. In Sect. 2 relevant background knowledge and related work is presented. Section 3 introduces the approach of ontology alignment by PSO and demonstrates its parallel scalability. Justified by these insights, Sect. 4 discusses the design and implementation of a cloud-based deployment of the formerly introduced approach. Experimental results are presented in Sect. 5 demonstrating both the effectiveness by referring to previous benchmarks, and the scalability by using a real-world biomedical alignment scenario. Section 6 summarises the contributions and provides an outlook on future work.

[2] `http://aws.amazon.com`

2 Foundations

This section introduces ontology alignment, particle swarm optimisation (PSO), and cloud computing in more details.

2.1 Ontology Alignment

An ontology alignment is defined as a set of correspondences between ontological entities [10], *i.e.* classes, properties, and individuals, of two ontologies. In this approach, an entity can correspond to at most one entity of the other ontology, which denotes what Euzenat and Shvaiko call an ?:? alignment [10]. Ontology alignment detection can be perceived as an optimisation problem [4], where the task is to find the optimal alignment of two ontologies w.r.t. the criteria denoted in terms of an evaluation function. These criteria can be based on several basic matching techniques [10, Chap. 4] or other measures, respecting domain specific and modelling language specific characteristics of the ontologies to be aligned.

A representative overview of the state-of-the-art in ontology alignment is given by Euzenat and Shvaiko [10, Chap. 6] and by the yearly Ontology Alignment Evaluation Initiative (OAEI) [9]. Most systems focus on the improvement of alignment quality, whereas the scalability problem has attracted interest only recently. As the OAEI participation shows, only few systems are capable of dealing with large ontologies [9, Sect. 10].

2.2 Particle Swarm Optimisation (PSO)

PSO is a biologically and socioculturally-inspired meta-heuristic [12, 13, 8], originally proposed in 1995 by Kennedy and Eberhart [12]. It has become a major research field since then.

PSO algorithms use a population of particles to find the optimal parameter configuration with respect to one or more objective functions. Each particle represents a candidate solution. A particle's *fitness* is evaluated using objective function(s). During initialisation, each particle in the swarm is assigned a random position in the parameter space. A PSO algorithm runs iteratively, where in each iteration, each particle adjusts its position in the parameter space by adding a *velocity* vector to its current position. These particle updates can be performed in parallel. The velocity vector for a particle is determined by the particle's previous velocity (*inertia*), the best position that has been visited by any neighbour of the particle so far (*social component*), and the best position, the particle itself has visited so far (*cognitive component*). Regarding the social component, several models of defining particle neighbourhoods have been analysed. Such models are called social network structures or *topologies*. A PSO algorithm is called *gBest* (global best) PSO, if the neighbourhood of a particle consists of the whole swarm. Thus the *gBest* PSO is a special case of the *lBest* (local best) PSO, where the neighbourhood of a particle comprises only a subset of the swarm [8, Chap. 12].

It is important to note, that the particle swarm search heuristic works without any knowledge about the objective function(s). Hence an objective function is replaceable and adjustable to arbitrary optimisation problems.

2.3 Cloud Computing

Cloud computing is a new paradigm that has been evolving over the last few years. Most definitions of cloud computing have in common that cloud computing offerings can be categorised using the "Everything as a Service" (XaaS) model. A more detailed view of this model is the "Cloud Computing Stack" [16]. According to the XaaS model the three main service classes are "Software as a Service" (SaaS), "Platform as a Service" (PaaS), and "Infrastructure as a Service" (IaaS). While SaaS offerings usually provide an interface directly to the end user by providing a Web service interface or a graphical user interface (GUI) the PaaS and IaaS offerings can be used by software architects to build new SaaS services on top of them. PaaS offerings usually provide a platform where the software developer can deploy the new services [16].

The IaaS offerings at the "Basic Infrastructure Services" level, give the developer full control over the servers that are running his software. At this level the user can deploy new machines using Web service technologies. This offers the power and flexibility to work on a machine level without running one's own data center and so having convenient access to new resources. Offerings such as Amazon EC2 give users the opportunity to automatically deploy hundreds of virtual machines within minutes. Thus, it is possible to build highly scalable applications that can scale up and down in short periods. One famous example of a successful cloud offering is Animoto [1]. The Animoto application transforms music files and photos into small video slide shows. After offering their service to users on a social network the demand of virtual machines went from about 40 to 3500 [15]. This scalability was only possible by designing the Animoto software as a distributed algorithm deployed on Amazon EC2.

Another interesting feature of cloud offerings is the typical pay-as-you-go pricing model. This means that users only pay for the resources they are really using. Having such a pricing model it makes no difference if one single server is running for 10 hours or if 10 servers are running for just one hour. The New York Times used this pricing scheme when they built their "TimesMaschine". By using Amazon EC2 they were able to convert their whole archive (4 TB of scanned TIFF files), spanning the years 1851-1922, to web documents within 24 hours and total costs of US$ 890 [11].

In the context of semantic technologies and the semantic web, cloud computing technologies have been successfully applied, mainly for RDF storage [22, 19, 18], querying [18], and materialisation of RDF/OWL knowledge bases [24, 23].

3 Ontology Alignment by Particle Swarm Optimisation

Considering ontology alignment as an optimisation problem, a novel discrete PSO algorithm (DPSO) has been developed to find an optimal alignment of two ontologies [4]. The algorithm has been implemented under the name MapPSO[3] and is based on a DPSO algorithm by Correa *et al.* [6], which implements efficient attribute selection in data mining classification tasks.

[3] `http://mappso.sourceforge.net`

3.1 Algorithm

In the case of ontology alignment, each particle represents a valid candidate alignment. With a *swarm* of particles being initialised randomly, in each iteration, each particle evaluates the alignment it represents via an objective function comprising various basic matching techniques [10, Chap. 4] as well as the size of an alignment. Typically there is only a partial overlap of ontologies, so the alignment algorithm strives for finding the largest alignment, *i.e.* maximum number of correspondences, of high quality. *Base matchers* can evaluate a correspondence according to different characteristics, such as lexical or linguistic similarity of entity labels or comments. Moreover, correspondences do not necessarily have to be evaluated in isolation, *e.g.* a correspondence of two classes can get a better evaluation if a correspondence of their superclasses is also part of the alignment represented by this particle. As discussed in Sect. 2.2 the objective function and thus the base matchers are replaceable and hence can be adapted to particular alignment scenarios. For instance in biomedical ontologies specific OBO[4] [21] annotations can be exploited to evaluate the similarity between entities.

Similar to the previous work by Correa *et al.* [6] and the original binary PSO algorithm by Kennedy and Eberhart [14], particle movements are determined by *proportional likelihoods*, since the traditional notion of velocity vectors is not applicable for binary search spaces, as it is the case for ontology alignment. The likelihood for a correspondence to be contained in a particle in the subsequent iteration is increased, if the correspondence exists in the global best (social component) or personal best (cognitive component) particle configuration.

This update mechanism allows for a guided convergence of the swarm to a global optimum, which represents the best alignment with respect to the basic matching techniques specified as objective function. Moreover, the size of each particle varies during the execution of the algorithm influenced by the size of the global best alignment. This causes the algorithm to search for the optimum size of the alignment, *i.e.* partial overlap, as well.

3.2 Parallel Efficiency

A detailed theoretical discussion of the correlation between population size and number of iterations with respect to the swarm convergence goes beyond the scope of this paper. Figure 1 illustrates an empirical analysis of the convergence towards the optimum[5] when aligning small ontologies of the OAEI benchmark dataset. The figure shows clearly that a larger number of particles results in faster convergence by reducing the number of required iterations and thus reducing wall-clock runtime. Where it takes about 225 iterations for a swarm of 8 particles to reach an alignment of high quality, the same result is achieved in only 90 iterations by a swarm of 64 particles. Using only 2 particles does not reach an equally good result within 300 iterations.

[4] Open Biomedical Ontologies

[5] Note that the optimum depends on the chosen base matchers and thus not necessarily has to be 0.

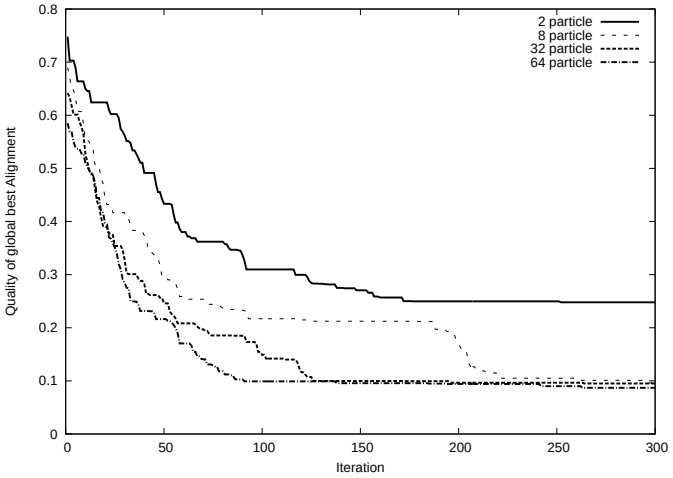

Fig. 1. Convergence of the PSO algorithm for different population sizes using test case *101* of the OAEI benchmarks. Alignment quality is represented as the fitness of the global best particle, where a smaller value represents better alignment quality.

4 Deployment in the Cloud

To obtain the required scalability, the PSO-based alignment algorithm has been ported to Amazon Web Services™ (AWS), the IaaS cloud service of Amazon®. AWS is one of the biggest IaaS providers with a large and active community, and is representative for any other IaaS provider in the context of this work.

The deployment has been realised using a server-worker pattern. Several *workers* evaluate and update several local particles each, while they are managed by a central *server*. Each worker determines the local best alignment among the results of its particles and sends it to the server. The server determines the global best alignment, then broadcasts it and synchronises the workers. Exchange of information between server and workers is realised by the Amazon Simple Queue Service (SQS)[6] or via TCP/IP. The exchange of concrete particle states is realised by the Amazon Simple Storage Service (S3)[7] for reasons of scalability, reliability, and parallel access.

4.1 Challenges

Deploying the algorithm on virtual machines connected by a network bears two main challenges. Firstly, the communication latency is much higher when communicating via network than via main memory. Hence finding and broadcasting the global best particle leads to a higher communication overhead, which slows down the algorithm and creates unwarranted costs.

[6] `https://aws.amazon.com/sqs/`
[7] `https://s3.amazonaws.com/`

Secondly, the computation times of workers in a particular iteration differ. This difference occurs mainly for two reasons: The unpredictable performance of the virtual environment, and the varying particle sizes. The performance of the virtual machine depends on its mapping to real hardware. For example a "noisy" neighbour, *i.e.* another program sharing the same real hardware via a different virtual machine, can slow down network communication. In turn, a virtual machine can utilise more computing power if there are no or only inactive neighbours. This results in unbalanced performance of the virtual machines. The random initialisation of particles and their different sizes add to this discrepancy in computation time. A small particle needs less computation time than a big particle, therefore a worker with smaller particles will require less runtime per iteration. The random initialisation of particles with different sizes is necessary to search for the optimal alignment size and thus be able to identify *partial* overlaps of ontologies. This computation time discrepancy causes fast workers to idle while waiting for the slower workers and thus decreases parallel efficiency.

4.2 Increasing Parallel Efficiency

The challenges of deploying the algorithm to a cloud-based infrastructre identified have been addressed and solutions are proposed as follows.

Addressing Latency. Amazon SQS was used as a means for communication between server and workers. Amazon advertises SQS as reliable, scalable, and simple. However, it turned out that SQS has high latency. For reducing the network latency, direct communication has been implemented using the TCP/IP protocol. Apart from reduced latency, this also resulted in a reduction of the additional communication overhead caused by multiple workers.

The latency is reduced further by only sending particle states when necessary, *i.e.* when a better particle state has been found. To achieve this a worker sends the fitness value of its local best particle to the server and writes the particle state itself into the S3 database. The server then broadcasts the global best fitness and the according database location to all worker instances.

The number of read operations is minimal, since in the PSO topology used, each particle must know about the global best. In case the global best does not change, no worker has to read a particle state. The (unlikely) worst case in terms of communication latency happens if every worker finds a new best alignment and thus writes its particle state before it has to read a new one found by another worker in the same iteration.

Addressing Runtime Discrepancy. The effect that workers require different runtimes for particle fitness computation can be minimised by *particle pooling*, *i.e.* having each worker instance computing multiple particles. Using few particles per worker results in high runtime variance between workers caused by different particle sizes and the resulting uneven workload. Using more particles per worker, *i.e.* the workload for each worker being the sum of workloads contributed by each

of its particles, averages the runtime because a rather uniform distribution of small and big particles per worker is expected. Using a multi-particle approach also increases parallel efficiency due to the increased overall runtime required to evaluate and update several particles. By increasing the runtime the proportion of time used for communication decreases and thus parallel efficiency is increased.

Using *asynchronous* particle updates is another way to compensate the runtime discrepancy. When using synchronous particle updates every worker has to wait for the slowest worker and thus is wasting computation time. In the asynchronous communication mode workers keep on evaluating and updating their particles until they find a particle state that is better than the global best they know about. They send this particle state to the server and continue. The server broadcasts the new particle state if it is better than the global best known by the server. Preventing workers to idle drastically increases parallel efficiency.

Introducing an asynchronous particle update strategy has an effect on the underlying particle swarm meta-heuristic. Having not all particles exchanging information at the same time step has a similar effect than changing the social network structure of the PSO from a star topology to a cluster topology[8] [8, Chap. 12], which results in fast communication between particles on the same worker compared to the communication between workers themselves. A clustered (*lBest*) topology in general results in slower convergence but better robustness compared to the star (*gBest*) topology [8, Chap. 12].

Furthermore, a loss of information can be observed compared to the synchronous update mechanism. This is due to the fact, that particles compute iterations with possibly outdated information about the global best, which might be available on a worker that is still computing another particle. This problem has been analysed by Lewis *et al.* [17]. Their investigations revealed that the information loss can be compensated by the increased number of computations that are possible due to the reduced waiting time.

The beneficial effect on the runtime for an alignment is reflected in Fig. 2, showing runtime behaviour for two medium sized ontologies for both synchronous and asynchronous particle updates. As expected, using an asynchronous communication mode does not have any effect if only a single worker is used. However, for 16 workers that need to communicate their local best results in each iteration, a clear runtime improvement of almost 50 % can be observed.

One further effect resulting from the asynchronous particle updates is that workers hosting mainly small particles can complete an iteration more quickly than those hosting mainly large particles. Therefore small particles tend to compute more iterations and thus influence the swarm stronger than it would be the case in the synchronous mode. This is beneficial to the overall runtime of the algorithm, because the average size of particles is smaller and therefore iterations are faster.

[8] While in a star topology, every particle shares information with every other particle, the cluster topology allows only groups of particles to communicate with each other, while the groups themselves exchange information only via dedicated particles, which are part of two clusters.

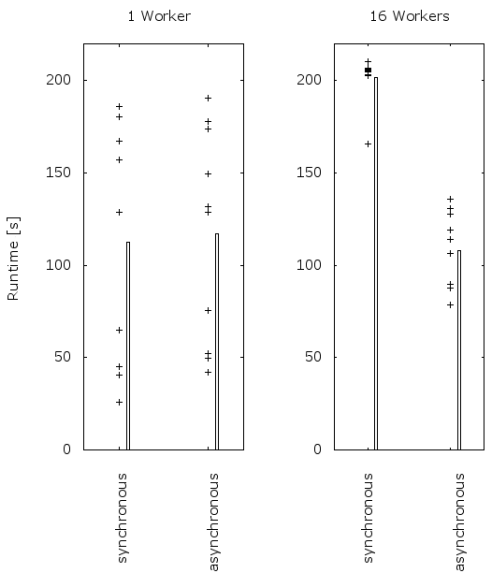

Fig. 2. Synchronous vs. asynchronous particle updates using 1 and 16 workers with 1 particle per worker. Bars denote an average of 10 runs (individual runs denoted left of the bars). Depicted is the total runtime for an alignment of the mouse ontology from the OAEI anatomy track with itself.

5 Evaluation

Since for large ontologies there are no reference alignments available, effectiveness and scalability of the approach need to be evaluated separately. While it can be shown for small benchmark ontologies, that the algorithm produces correct results using a problem specific parameter configuration, a separate experiment has been conducted evaluating scalability using a cloud infrastructure.

5.1 Effectiveness

Previous evaluations of the MapPSO system at the Ontology Alignment Evaluation Initiative (OAEI) campaigns 2008 and 2009 [3, 5] have shown the effectiveness of the approach. In particular with respect to relaxed precision and recall metrics [7] the OAEI 2009 organisers report that "[...] MapPSO has significantly better symmetric precision and recall than classical precision and recall, to the point that it is at the level of the best systems." [9]. The reason for this difference to classical metrics is most likely the rather generic parameter configuration that has been used for all benchmark tests, without adaptation to the single alignment scenarios. This generic parameter configuration is a restriction that is of minor importance in real-world alignment scenarios.

5.2 Scalability

In order to demonstrate the scalability of this approach, an experiment was conducted aligning two large ontologies from the biomedical domain. The chosen ontologies were the Gene Ontology (GO)[9] with 31,650 classes, and the Medical Subject Headings (MESH) Ontology[10] with 15,343 classes, both converted to the OWL format. Specific base matchers were used that take advantage of the class annotations resulting from the conversion from OBO to OWL.

In the experiment a population of 128 particles was used distributed on 16 Amazon EC2 instances (workers) and thus utilising particle pooling by computing 8 particles per worker. The EC2 instances were of the type "High-CPU Extra Large Instance, 7 GB of memory, 20 EC2 Compute Units (8 virtual cores with 2.5 EC2 Compute Units each), [...] 64-bit platform, [where] one EC2 Compute Unit (ECU) provides the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor"[11]. The setup was chosen in a way, that the number of particles on each instance matches the number of virtual cores, thus enabling efficient multi-threading of particle computation.

The experiment was run for 3.5 h, where due to the asynchronous particle updates, workers computed between 50 and 75 iterations. The difference in the number of iterations illustrates the different computational efforts of each worker. Workers that achieved a lower number of iterations than other workers at the same time, are most likely computing smaller particles on average. This illustrates the runtime discrepancy addressed in Sect. 4.1 and the increased parallel efficiency gained by using an asynchronous approach. By using a synchronous approach the slowest particle constitutes an upper bound and thus a maximum of 50 iterations could have been computed in 3,5 h. The asynchronous approach, however, computes an average of 62.875 iterations, which is an increase of 25 %.

Since there is no reference alignment available for GO and MESH as there is none for any other large ontologies, no statement about the result quality can be made. The experiment, however, demonstrates the scalability of the proposed approach, operating on large ontologies and converging to an alignment.

6 Conclusion

The problem of large-scale ontology alignment detection has been addressed by a PSO-based approach. PSO has the characteristics of (*i*) being independent of its objective function(s) and thus easily adaptable to various ontology alignment scenarios, (*ii*) being incremental, *i.e.* able to refine an alignment when input ontologies evolve, and (*iii*) being inherently parallelisable. The latter aspect has been exploited in this paper by deploying the algorithm in the AWS cloud. This deployment is making use of the emerging cloud computing paradigm, which

[9] `http://www.geneontology.org/`
[10] `http://www.nlm.nih.gov/mesh/meshhome.html`
[11] `http://aws.amazon.com/ec2/#instance`, accessed 2010/06/19

provides an on-demand infrastructure for dynamic computational needs, as it is the case for alignment refinements of gradually evolving ontologies.

When utilising parallel computation infrastructures on a pay-per-use basis, such as cloud offerings, it is crucial to maximise parallel efficiency. Due to the variable particle sizes used in this approach, different computation times for each particle in each iteration can be observed. Thus particle pooling, *i.e.* multiple particles per cloud instance, as well as asynchronous particle updates have been introduced to the algorithm in order to reduce the time wasted by idling particles.

Previous experiments in the course of OAEI participations have shown the effectiveness of the PSO-based approach. Complementing these results, the scalability via parallelisation has been shown for two large biomedical ontologies.

Having realised the successful deployment of an ontology alignment algorithm in the cloud using an IaaS provider, it is a straightforward extension to provide ontology alignment itself as a web service (SaaS) based on a dynamic and scalable infrastructure. This enables large-scale ontology alignment for every user without bothering about hardware requirements. Together with the anytime behaviour of the PSO-based algorithm, business models comprising the dimensions computation time, price, and alignment quality can be created.

Acknowledgement

The presented research was partially funded by the German Federal Ministry of Economics (BMWi) under the project Theseus (number 01MQ07019).

References

1. Animoto Productions: animoto - the end of slideshows:. online, `http://www.animoto.com`, last seen: 2010/06/16
2. Bizer, C., Heath, T., Berners-Lee, T.: Linked Data – The Story So Far. International Journal on Semantic Web and Information Systems 5(3), 1–22 (2009)
3. Bock, J., Hettenhausen, J.: MapPSO Results for OAEI 2008. In: Proceedings of the 3rd International Workshop on Ontology Matching (OM-2008). vol. 431. CEUR Workshop Proceedings, http://ceur-ws.org (2008)
4. Bock, J., Hettenhausen, J.: Discrete Particle Swarm Optimisation for Ontology Alignment. Information Sciences (Special Issue on Swarm Intellligence and Applications) (2010), article in press
5. Bock, J., Liu, P., Hettenhausen, J.: MapPSO Results for OAEI 2009. In: Proceedings of the 4th International Workshop on Ontology Matching (OM-2009). vol. 551, pp. 193–199. CEUR Workshop Proceedings, http://ceur-ws.org (2009)
6. Correa, E.S., Freitas, A.A., Johnson, C.G.: A New Discrete Particle Swarm Algorithm Applied to Attribute Selection in a Bioinformatics Data Set. In: Proceedings of the 8th Genetic and Evolutionary Computation Conference (GECCO-2006). pp. 35–42. ACM (2006)
7. Ehrig, M., Euzenat, J.: Relaxed Precision and Recall for Ontology Matching. In: Proceedings of the K-CAP 2005 Workshop on Integrating Ontologies. vol. 156, pp. 25–32. CEUR Workshop Proceedings, http://ceur-ws.org (2005)

8. Engelbrecht, A.P.: Fundamentals of Computational Swarm Intelligence. Wiley (2007)
9. Euzenat, J., Ferrara, A., Hollink, L., Isaac, A., Joslyn, C., Malaisé, V., Meilicke, C., Nikolov, A., Pane, J., Sabou, M., Scharffe, F., Shvaiko, P., Spiliopoulos, V., Stuckenschmidt, H., Šváb-Zamazal, O., Svátek, V., dos Santos, C.T., Vouros, G., Wang, S.: Results of the Ontology Alignment Evaluation Initiative 2009. In: Proceedings of the 4th International Workshop on Ontology Matching (OM-2009). vol. 551. CEUR Workshop Proceedings, http://ceur-ws.org (2009)
10. Euzenat, J., Shvaiko, P.: Ontology Matching. Springer (2007)
11. Hilley, D.: Cloud Computing: A Taxonomy of Platform and Infrastructure-level Offerings. Tech. rep., College of Computing, Georgia Institute of Technology (2009)
12. Kennedy, J., Eberhart, R.C.: Particle Swarm Optimization. In: Proceedings of IEEE International Conference on Neural Networks. vol. 4, pp. 1942–1948. IEEE Computer Society (1995)
13. Kennedy, J., Eberhart, R.C.: Swarm Intelligence. Morgan Kaufmann (2001)
14. Kennedy, J., Eberhart, R.C.: A Discrete Binary Version of the Particle Swarm Algorithm. In: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics. vol. 5, pp. 4104–4108. IEEE Computer Society (1997)
15. Killalea, T.: Building Scalable Web Services. Queue 6(6), 10–13 (2008)
16. Lenk, A., Klems, M., Nimis, J., Tai, S., Sandholm, T.: What's inside the Cloud? An architectural map of the Cloud landscape. In: Proceedings of the ICSE Workshop on Software Engineering Challenges of Cloud Computing (CLOUD). pp. 23–31. IEEE Computer Society (2009)
17. Lewis, A., Mostaghim, S., Scriven, I.: Asynchronous Multi-Objective Optimisation in Unreliable Distributed Environments, Studies in Computational Intelligence, vol. 210, pp. 51–78. Springer (2009)
18. Mika, P., Tummarello, G.: Web Semantics in the Clouds. IEEE Intelligent Systems 23(5), 82–87 (2008)
19. Newman, A., Li, Y.F., Hunter, J.: Scalable Semantics – the Silver Lining of Cloud Computing. In: Proceedings of the IEEE Fourth International Conference on eScience. pp. 111–118. IEEE Computer Society (2008)
20. Qiu, X., Ekanayake, J., Beason, S., Gunarathne, T., Fox, G., Barga, R., Gannon, D.: Cloud Technologies for Bioinformatics Applications. In: Proceedings of the 2nd Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS). pp. 1–10. ACM (2009)
21. 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.: The OBO Foundry: Coordinated Evolution of Ontologies to Support Biomedical Data Integration. Nature Biotechnology 25(11), 1251–1255 (2007)
22. Stein, R., Zacharias, V.: RDF On Cloud Number Nine. In: Proceedings of the 4th Workshop on New Forms of Reasoning for the Semantic Web: Scalable & Dynamic. pp. 11–23. CEUR Workshop Proceedings, http://ceur-ws.org (2010)
23. Urbani, J., Kotoulas, S., Maassen, J., van Harmelen, F., Bal, H.: OWL Reasoning with WebPIE: Calculating the Closure of 100 Billion Triples. In: Proceedings of the 7th Extended Semantic Web Conference (ESWC). LNCS, vol. 6088, pp. 213–227. Springer (2010)
24. Urbani, J., Kotoulas, S., Oren, E., van Harmelen, F.: Scalable Distributed Reasoning using MapReduce. In: Proceedings of the 8th International Semantic Web Conference (ISWC). LNCS, vol. 5823, pp. 634–649. Springer (2009)