Towards an Automatic Parameterization of Ontology Matching Tools based on Example Mappings

Dominique Ritze[1] and Heiko Paulheim[2]

[1] Mannheim University Library
dominique.ritze@bib.uni-mannheim.de
[2] Technische Universität Darmstadt
Knowledge Engineering Group
paulheim@ke.tu-darmstadt.de

Abstract. With a growing number of ontologies and datasets using those ontologies, ontology mappings become an essential building block of the Semantic Web. In the last years, a larger number of sophisticated ontology matching tools for generating such mappings has been developed. The quality of the mappings provided by those tools typically depends on the settings of the tools' parameters. As this is a non-trivial task for an end user, we propose the *ECOMatch* approach, which asks the user to provide example mappings instead of parameter settings, and automatically determines a suitable parameter setting based on those examples. We show how the preliminary result quality of ontology mappings can be improved by applying automatic, example-based configuration of ontology matching tools.

1 Introduction

Ontologies formally describe the concepts used in a domain. While in an ideal scenario, there is one ontology which is shared throughout a whole domain, reality often faces the parallel use of different ontologies, which have been developed independently from each other. *Ontology matching* [8] is used for creating mappings between ontologies.

During the past years, a lot of research has been devoted to developing highly sophisticated tools for performing ontology matching automatically [6, 7]. Those tools are able to produce high-quality mappings between ontologies, given that their parameters (such as weights and thresholds used to compute the mappings) are tuned well. Such a tuning, however, is often complicated, since it involves the setting of many parameters and requires a lot of detail knowledge about the underlying algorithms and implementations. For example, the state of the art matching tool *Falcon-AO* [12] has 33 different parameters that can be manipulated, which makes it hard to guess an optimal parameter set without a planned approach. Furthermore, there are often no universally optimal settings: a configuration that performs well on one pair of ontologies may produce bad results on another one. Therefore, an automatic configuration of matching tools has been named as one of the top ten challenges for ontology matching [22].

In this paper, we introduce the *ECOMatch*[3] approach for automatic configuration of ontology matching tools. Instead of letting the user directly manipulate the parameters

[3] **E**xample-based **C**onfiguration of **O**ntology **Match**ing tools

(which he often does not understand), we ask her to provide a set of example mappings (a task which can be done by a domain expert in a reasonable amount of time). We use those example mappings to test a number of different configurations and determine a good or even optimal parameter setting. That setting is then used to match the input ontologies and provides the final mapping.

The rest of this paper is structured as follows. In Sect. 2, we lay out the theoretical considerations for our work, and in Sect. 3, we discuss the implementation of ECO-Match. This implementation is the basis of various experimental evaluations, which are discussed in Sect. 4. We conclude with a summary and an outlook on future work.

2 Approach

A mapping between two ontologies which has been created by a domain expert user is called a *reference alignment*[4]. The goal of ontology matching tools is to produce a mapping which gets as close to a reference alignment as possible, i.e., which is as good as if a human expert would have created the mapping manually to achieve semantic interoperability.

For automatic tuning of ontology matching tools, we assume that the user is able to provide a set of example mappings. We call that set of examples a *partial reference alignment* between the target ontologies. We use this partial reference alignment to evaluate several configurations of the target matching tool. The configuration which has been evaluated best based on the partial reference alignment is then used to produce the final mapping.

To determine which of the tested configurations is the best one, we use the output produced by the matching tool when applying the respective configuration, and compute the result quality on the partial reference alignment, i.e., how well the partial reference alignment is reproduced by the matcher. Our assumption is that a configuration which reproduces the partial reference alignment well will also produce a high-quality overall ontology mapping.

For computing the result quality, we introduce the following measures for computing recall, precision, and f-measure on a partial mapping. Following Euzenat and Shvaiko [8], a mapping between two ontologies O_1 and O_2 can be defined as a set of 5-tuples of the form $\langle id, e_1, e_2, r, n\rangle$, where $e_1 \in O_1$ and $e_2 \in O_2$, and where r defines a type of relation (such as equality, subclass, etc.), and n depicts a confidence level provided by a matching tool. Therefore, given a reference alignment R, a partial reference alignment $R' \subseteq R$, and an alignment A computed by a matching tool, we define the partial alignment $A' \subseteq A$ as the subset of A which contains all elements in A which share at least one entity with an element in R':

Definition 1 (Partial Alignment).

$$\begin{aligned} A' := &\{\langle id, e_1, e_2, r, n\rangle \in A | \exists id', e_1', n' : \langle id', e_1', e_2, r, n'\rangle \in R'\} \\ \cup\ &\{\langle id, e_1, e_2, r, n\rangle \in A | \exists id', e_2', n' : \langle id', e_1, e_2', r, n'\rangle \in R'\} \end{aligned}$$

[4] The terms *mapping* and *alignment* are often used synonymously.

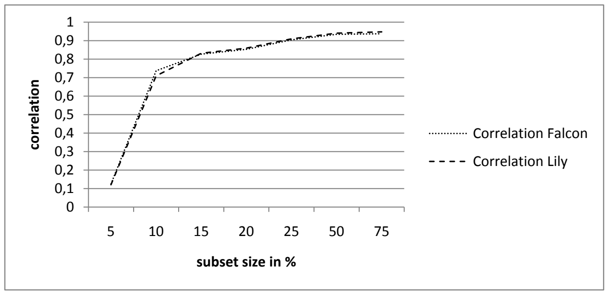

Fig. 1. Correlation between f-measure values of partial reference alignments with the f-measure value of the full reference alignment

	5	10	15	20	25	50	75
Falcon-AO	0.006	0.031	0.024	0.012	0.007	0.003	0.003
Lily	0.060	0.094	0.042	0.019	0.028	0.000	0.000

Table 1. T-test on correlations between f-measure values of partial and full reference alignments

Based on that definition, we can define a set of quality measures for an alignment A produced by a matching tool, which can be evaluated using a partial reference alignment:

Definition 2 (Precision on Partial Reference Alignment).

$$P'(A, R') := P(A', R') = \frac{|R' \cap A'|}{|A'|} \in [0, 1]$$

Definition 3 (Recall on Partial Reference Alignment).

$$R'(A, R') := R(A', R') = \frac{|R' \cap A'|}{|R'|} \in [0, 1]$$

Definition 4 (F-measure on Partial Reference Alignment).

$$F'(A, R') := M_{0.5}(A', R') = \frac{2 * P'(A, R') * R'(A, R')}{P'(A, R') + R'(A, R')} \in [0, 1]$$

It is particularly noteworthy that P', R', and F' can be computed only from the output of a matching tool and the partial reference alignment, without the need to know the complete reference alignment. To show that those measures are valid for assessing the result quality of a matching tool, we have analyzed the correlation of F and F', i.e., the f-measure computed on the full and on the partial reference alignment. This correlation is an indicator for the precision of the prediction of F by the means of F'.

To that end, we have used the two matching tools that we also used later on in our prototype (see Sect. 3), Falcon-AO and Lily, and 21 pairs of ontologies with reference alignments. We have run each tool with 100 random configurations and determined the f-measures F and F' for different sizes of partial reference alignments. The results are shown in Fig. 1. The key observation is that the results do not differ much between the individual matchers, and that there is a positive correlation, which becomes considerably high for partial reference alignments covering 10% or more of the reference alignment. Table 1 shows the confidence levels of the correlation, using a two-sample paired t-test. It reveals that for Falcon-AO, all the results are statistically significant, while for Lily, the correlation is only significant for partial reference alignment sizes of at least 15%.

These figures show that using partial reference alignments to score the configurations is reasonable: since a matcher configuration achieves a high f-measure value on a partial reference alignment it will most likely achieve a high f-measure value on the full reference alignment as well. With those considerations in mind, we have implemented a prototype to further analyze the potential of improving ontology matching by automatically configuring matchers based on example mappings.

3 Implementation

We have implemented our approach using a variety of algorithms for parameter optimization, as well as different matching tools. Fig. 2 shows the basic architecture of the system, based on the "classic" matching architecture proposed by Euzenat and Shvaiko [8]. The system takes as input two ontologies and a partial reference alignment, i.e., a set of known mappings.

The ECOMatch system itself treats the matcher it uses as a black box. For creating an optimized mapping, it performs the following steps:

1. Create an initial configuration for the matcher.
2. Run the matcher with that configuration on the pair of input ontologies and observe the mapping created.
3. Compute the f-measure F' based on the partial reference alignment.
4. Based on that resulting F', decide for a new configuration to try out.

Steps 2 to 4 are repeated until a stopping criterion – e.g., a fixed number of matcher runs or a time limit – is reached, or until the configuration generator decides not to proceed any further, e.g., because a search algorithm has reached a local optimum, or the whole search space has been covered. At that point, the mapping achieved with the configuration yielding the maximum F' is returned.

3.1 Configuration Generators

Besides a baseline algorithm which generates random configurations, we have included five metaheuristic algorithms for determining configurations from the large variety of parameter tuning approaches (see, e.g., [2] for a survey). As parameter optimization can also be regarded as a machine learning problem, we have furthermore implemented two machine learning algorithms.

Metaheuristics Metaheuristics are a family of stochastic optimization algorithms. They are especially practicable for problems for which it is hard to describe how to find an optimal solution, but which allow for easily assessing the quality of a given solution [17]. Metaheuristics subsequently create solutions and rate them according to a quality

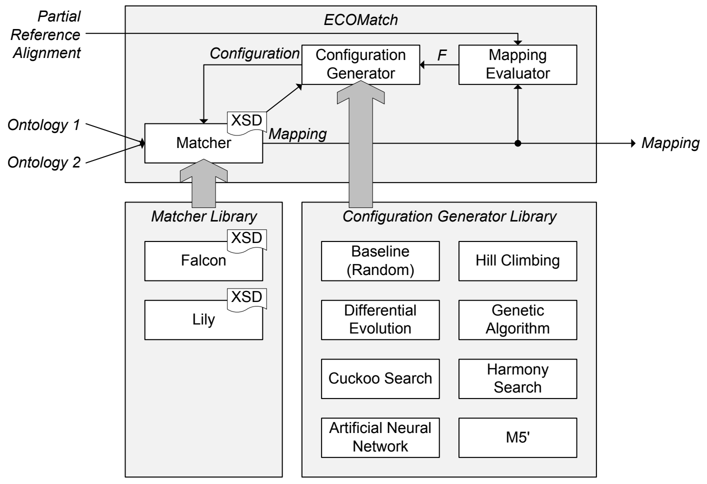

Fig. 2. Prototype architecture

function (in our case: the f-measure F' computed on the partial reference alignment). The rating determines which solution to examine next. Given that not every possible solution is examined, the optimal solution may not be found because it has not been examined at all. Thus, metaheuristics are usually used to solve optimization problem with a huge search space where it is hardly feasible to examine every possible solution [21]. In the prototype of *ECOMatch*, we have implemented the following metaheuristics: hill climbing [21], genetic algorithm [11], differential evolution [23], harmony search [10] and cuckoo search [26].

Machine Learning Techniques Machine learning methods are very widespread in various fields, e.g. in search engines, natural language processing or recommendation systems. While metaheuristics only examine a fixed set of candidate solutions, machine learning techniques can be used to train an explicit model, e.g., a decision tree or an artificial neural network, of the parameter space. This model can be used to predict the performance of other parameter combinations, which have not examined before. In *ECOMatch*, a small set of random parameter configurations is created and serves as training data. The employed methods M5', based on modeltrees [20], and artificial neural networks [9] each build a model which is used to predict the f-measure values F', called $F'_{predicted}$, for unseen random examples. For the configurations with best $F'_{predicted}$, the matching tool is run to determine the exact value F' and in turn the configuration with best F' is used to create the final mapping.

While determining the exact value F' using the matching tool is a costly operation, which takes minutes or even up to hours, calculating the value $F'_{predicted}$ using a learned model can be performed within milliseconds. Thus, a much larger amount of configurations can be examined using the trained model. To avoid negative effects due to wrong predictions, the predicted best configurations are double-checked using the matcher before taking them for producing a mapping.

3.2 Matching Systems

Besides different algorithms for parameter configuration, matching tools may be plugged into the *ECOMatch* framework. We have tested the framework with two matchers that performed very well in the last OAEI evaluations: Falcon-AO[5] [12] and Lily[6] [25]. While it is possible to run *ECOMatch* with every matching tool that provides a means to be run from the command line in batch mode, we have chosen those matchers because of their popularity and their performance in evaluations, as well as the provision of a sufficient size of the parameter set. Falcon-AO has 33 parameters in total, while Lily has eight. For our experiments with Falcon-AO, we have manually reduced the size of the parameter set to 26, discarding all parameters that do not have an effect on the result quality (e.g., parameters that only have an effect when processing large scale ontologies).

For each matching tool, we store a configuration description as an XML Schema Definition (XSD) file, which contains a list of the tool's parameters, as well as their respective types and ranges of values. This definition file represents the parameter space of the matching tool and is used by the configuration generator for creating the configurations.

4 Evaluation

To examine the impact of automatic parameter configuration on the result quality of ontology matching tools, we have conducted experiments on different data sets.

4.1 Evaluation Setup

For our evaluation, we have used all possible combinations of configuration generation algorithms and matchers, as described in Section 2.

We have used three different datasets for our evaluation:

- A subset of the OAEI benchmark dataset[7], which consists of the four non-artificial ontologies with reference alignments to one common ontology
- The OAEI conference dataset[8], consisting of seven ontologies and 21 reference alignments
- The six pairs of ontologies delivered with the FOAM ontology matching tool[9]

Altogether, we have tested the tool with 31 different pairs of ontologies and corresponding reference alignments.

To allow for comparison between the parameter optimization algorithms, we let each algorithm run the matching tool a certain amount of times depending on their

[5] Using the 2008 version, which can be downloaded from http://ws.nju.edu.cn/falcon-ao/index.jsp
[6] http://code.google.com/p/lilyontmap/
[7] http://oaei.ontologymatching.org/2010/benchmarks/index.html
[8] http://oaei.ontologymatching.org/2010/conference/index.html
[9] http://people.aifb.kit.edu/meh/foam/ontologies.htm

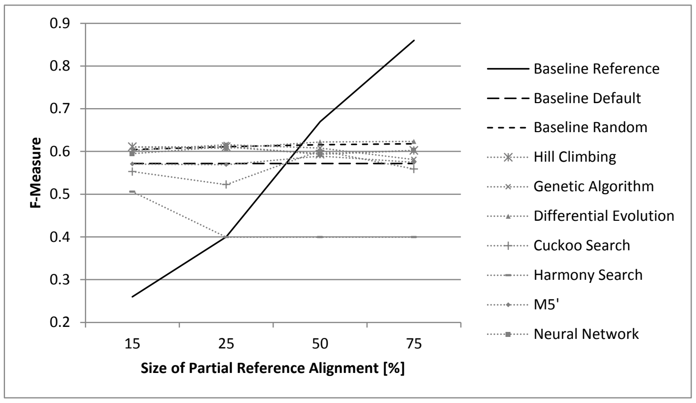

Fig. 3. Evaluations with Falcon-AO on the FOAM datasets

runtime, i.e. Falcon-AO 250 times and Lily 50 times (since running the matching tool is the most time consuming step). The machine learning approaches used 200 (Falcon-AO) resp. 40 (Lily) random configurations for training the respective models, 10,000 randomly generated configurations were rated by the trained model. Out of those, the 50 resp. 10 best-rated configurations were re-evaluated by running the matching tool with them. For each experiment, the average of three runs has been taken into the evaluation to reduce the impact of random outliers.

In order to show the value of our approach, we have tested against three baselines. The first baseline is the result quality achieved with default configuration of each tool. The second baseline is to choose the best out of 250 resp. 50 random configurations (since we let each algorithm perform 250 resp. 50 runs of the matcher). The third baseline is the result quality which would be achieved if the partial reference alignment would be returned as is, without running any matcher. For showing that parameter optimization based on example mappings has any value, it should at least perform better than the default configuration and the use of the partial reference alignment as is. A sophisticated approach for parameter optimization should outperform all three baselines, including the use of randomly generated configurations.

4.2 Evaluation Results

Running the evaluations leads to mixed results. For Falcon-AO, the default configuration proves to be rather robust and yield better results on most data sets than configurations found with any other approach. Figure 3 shows the average results for the FOAM datasets, which were the only ones where an optimized configuration could perform better than the default configuration.

The results reveal that most approaches, except for differential evolution, perform slightly worse than the random baseline. This is most likely due to the large size and dimensionality of the search space, combined with the restriction of the number of runs of the matching tool, which in the end make the search algorithms terminate early (and possibly too early to search the parameter space far enough for yielding good results).

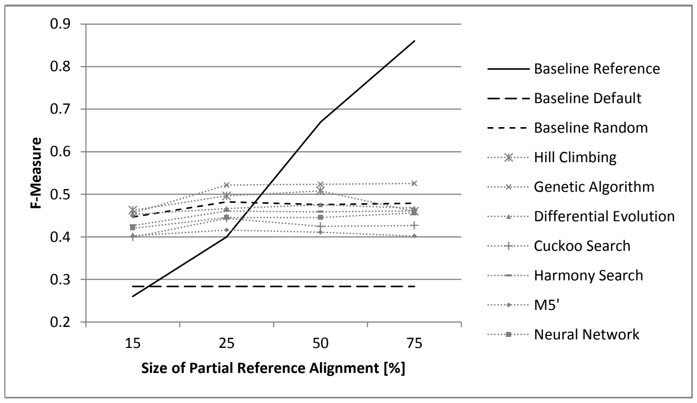

Fig. 4. Evaluations with Lily on the conference datasets

With Lily, the improvements that could be achieved over the standard configuration were more significant. Figure 4 depicts the average results achieved on the conference dataset. Again, a very good performance of the random baseline can be observed, with evolutionary algorithms performing slightly better. As for Falcon-AO, a possible explanation is the large search space which has to be explored with a low number of matcher runs for the optimization.

In all cases, the results achieved with partial reference alignments of up to 25% outperformed the baseline of using the partial reference alignment as is. Thus, the value gained from the partial reference alignment is higher than the effort that has to be used for producing that alignment. Partial reference alignments larger than 25% serve better as final mappings, rather than using them as input for parameter optimization.

The results achieved with both matchers also reveal that the size of the partial reference alignment does not significantly influence the result quality. This is a very encouraging result, as the provision of example mappings may be a crucial bottleneck for the *ECOMatch* approach. The observation that the result quality already significantly increases with a partial reference alignment of 15% (and probably even with smaller sizes) demonstrates that the *ECOMatch* approach is not only feasible in theory, but may also be practically applied.

When looking more closely at the parameter configurations that are created by the different approaches, there are several observations that can be made. We have compared those configurations both with each other as well as with the respective tool's default configuration. The first observation is that approaches perform better if they create more different configurations. This also explains the good performance of the random baseline, which ensures a maximum entropy of configurations. The second observation is that the best configurations that are found by the algorithms are rather close (although not identical) to the default configuration.

In summary, some optimization approaches, such as cuckoo search or harmony search, do not perform very well on that problem, while significant improvements of the resulting mapping quality are possible with a suitable approach, even when the set of examples is not too large.

5 Related Work

Despite the large body of work in the field of ontology matching and its predecessor, schema matching, there is little work done which is focused on automatic or semi-automatic parameterization of matching tools.

eTuner [16] is a tool which is directed at the parametrization for schema matching techniques and their combination. For evaluating different settings, eTuner generates synthetic pairs of schemes from the input schemes using predefined rules (similar to our approach using the result quality on a partial reference alignment as an approximation).

A problem closely related to parameter tuning of matching tools is selecting a suitable matcher – or a suitable combination of matchers – for a specific matching task. Such a combination can be described by a set of weights for each matcher, as done, e.g., with the matching framework *CROSI* [14] and can be seen as a special case of the parameter tuning problem. One work uses meta-level learning in order to find the best ensemble of matchers, as discussed by Eckert et al. [3]. Other strategies are based on Bayesian Networks [24] or Support Vector Machines [13]. Mochol et al. [18] propose an approach to find the best matcher to match two specified ontologies from a set of matchers.

Many matching tools like the systems QOM [5] or PROMPT [19] support semi-automatic matching processes, e.g. by presenting the user suggestions or potential problems. They typically use the examples provided by the user as anchors for searching for new mappings, rather than for tuning the parameters of the underlying matching algorithms. A very related idea which also takes user support into account is *APFEL* [4]. It tries to improve the alignment by involving the user into the matching process. It first creates potential correspondences based on initial settings, presents them to the user, the user marks the correct ones which serve as training set for the machine learning approach. The whole process is repeated to gradually improve the alignment.

In our approach, we have used partial reference alignments provided by a domain expert to optimize different matching tools. The matching tool *SAMBO* [15] also uses partial reference alignments for various purposes. They are used as anchors to give hints for partitioning larger ontologies in a preprocessing step, as well as for filtering potential incorrect results in a post-processing step. A schema matching system exploiting examples and applying machine learning methods is *LSD* [1]. The approach defines the matching problem as a classification problem, where the class to predict for an element of a source schema is the corresponding target schema element. Thus, a machine learning algorithm is trained using correspondences provided by the user.

The key differences between those approaches and *ECOMatch* are that existing approaches typically work in dialog mode and require constant user attention, while our approach works in batch mode once the user examples are given. Futhermore, *ECOMatch* is not restricted to a particular matching algorithm, but can be used with any existing matching tool.

6 Conclusion and Outlook

In this paper, we have presented the *ECOMatch* approach for automatically parameterizing ontology matching tools based on example mappings. Since we could observe a significant correlation of the result quality, it is possible to test different matcher configurations against a set of example mappings, and thereby find a good or even an optimal matcher configuration. ECOMatch treats the matcher it uses as a black box; thus, our approach can be easily adopted by developers of other matching tools or used as a generic framework for optimizing ontology mappings.

Since it is often not feasible to test every possible combination of parameter settings, we have used a number of heuristics for determining good, near-optimal parameter settings. We have tested our approach with two state-of-the-art matching tools and a set of benchmark ontologies. The results show that parameter optimization based on example mappings can help providing significant better results than using the default configuration of matching tools, and is a good alternative to manually trying to find a good configuration, since no understanding of the parameters is required.

The two main bottlenecks of our approach are the provisioning of examples (which typically means manual work for a domain expert), and the running of matching tools (which is a costly operation that drastically increases the overall processing time). These two bottlenecks point at the main opportunities for improving the ECOMatch system: reducing the amount of examples required, and reducing the number of required matcher runs.

In our experiments, we have used a random subset of a gold standard mapping for emulating a domain expert creating example mappings. This may not be entirely correct, as such an expert may first determine a set of *obvious* mappings, which may not be a representative random sample of the full mapping. Thus, we want to further evaluate how the selection of the mapping subset influences our approach. To reduce the workload for the domain expert providing the partial mapping, we would also like to explore how the size of the partial mapping can be minimized, e.g., in an interactive mode where the tool interrogates a domain expert and tries to determine mappings which efficiently divide the search space. As negative examples may be retrieved with less work than positive ones, we also want to explore how a combination of positive and negative examples can be used to find an optimal mapping.

Most matchers are tuned for achieving a good f-measure, i.e., a good trade-off between precision and recall. However, it is possible to develop matchers with high precision at the cost of worse recall. Such a matcher could be used as a generator for the example mappings, so that the whole tuning process of the target matcher could even be fully automatized in the future. In the context of ontology matching for linked open data, other mechanisms for obtaining the example mappings are also possible, such as a community creating a set of example mappings (the same as they are creating links between instances) [27], or guessing example mappings for classes of different ontologies that share a lot of common instances.

So far, we have only investigated the quality of the generated configurations *after* a fixed number of runs of the matching tool. In order to speed up the whole process, it would be interesting to look at the gradient, i.e., the time it takes for a certain optimization approach to find a good configuration. This would allow for a more sophisticated comparison of the individual strategies for finding an optimum configuration. Furthermore, our experiments have shown that most of the good configurations found by *ECOMatch* are similar, yet not identical, to the default configuration. This insight may also help improving the process of searching for a good configuration, e.g., when creating a starting population for a genetic algorithm.

In summary, the work presented in this paper has addressed one of the top ten challenges in ontology matching. We have proposed a solution that is suitable for domain experts with low skills in ontology matching, since we only rely on a set of example mappings provided by a domain expert. The results show that matcher configurations can be automatically improved based on example mappings, which makes this approach a promising research direction for future ontology matching tools.

Acknowledgements

The work presented in this paper has been partly funded by the German Federal Ministry of Education and Research under grant no. 01ISO7009. Dominqiue Ritze is supported by a Karl-Steinbuch scholarship of the MFG foundation Baden-Württemberg. The authors would like to thank Heiner Stuckenschmidt, Christian Meilicke, and Johannes Fürnkranz for their valuable feedback.

References

1. A. Doan, P. Domingos, and A. Halevy. Reconciling schemas of disparate data sources: A machine-learning approach. In *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, pages 509–520, 2001.
2. Felix Dobslaw. Automatic parameter tuning for metaheuristics, 2010. Accessed June 17th, 2011.
3. Kai Eckert, Christian Meilicke, and Heiner Stuckenschmidt. Improving ontology matching using meta-level learning. In *Proceedings of the 6th European Semantic Web Conference (ESWC)*, pages 158–172, 2009.
4. M. Ehrig, S. Staab, and Y. Sure. Bootstrapping ontology alignment methods with apfel. In *Proceedings of the 4th International Semantic Web Conference (ISWC)*, pages 186–200, 2005.
5. Marc Ehrig and Steffen Staab. Qom - quick ontology mapping. In *The Semantic Web - ISWC 2004*, volume 3298, pages 683–697. Springer, 2004.
6. Jérôme Euzenat, Alfio Ferrara, Laura Hollink, Antoine Isaac, Cliff Joslyn, Véronique Malaisé, Christian Meilicke, Andriy Nikolov, Juan Pane, Marta Sabou, François Scharffe, Pavel Shvaiko, Vassilis Spiliopoulos, Heiner Stuckenschmidt, Ondrej Sváb-Zamazal, Vojtech Svátek, Cássia Trojahn dos Santos, George A. Vouros, and Shenghui Wang. Results of the Ontology Alignment Evaluation Initiative 2009. In *Proceedings of the 4th International Workshop on Ontology Matching (OM-2009)*, volume 551 of *CEUR-WS*, 2009.
7. Jérôme Euzenat, Alfio Ferrara, Christian Meilicke, Juan Pane, François Scharffe, Pavel Shvaiko, Heiner Stuckenschmidt, Ondrej Sváb-Zamazal, Vojtech Svátek, and Cássia Trojahn. Results of the Ontology Alignment Evaluation Initiative 2010. In *Proceedings of the Fifth International Workshop on Ontology Matching (OM-2010)*, volume 689 of *CEUR-WS*, 2010.

8. Jérôme Euzenat and Pavel Shvaiko. *Ontology Matching*. Springer, Berlin, Heidelberg, New York, 2007.
9. M.W. Gardner and S.R. Dorling. Artificial neural networks (the multilayer perceptron) - a review of applications in the atmospheric sciences. *Atmospheric Environment*, 32(14–15):2627–2636, 1998.
10. Zong Woo Geem, Joong-Hoon Kim, and G. V. Loganathan. A new heuristic optimization algorithm: Harmony search. *Simulation*, 76(2):60–68, 2001.
11. John H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.
12. Wei Hu and Yuzhong Qu. Falcon-AO: A practical ontology matching system. *Journal of Web Semantics*, 6(3), 2008.
13. Ryutaro Ichise. Machine learning approach for ontology mapping using multiple concept similarity measures. In *Proceedings of the 7th International Conference on Computer and Information Science (ICIS)*, pages 340–346, 2008.
14. Yannis Kalfoglou and Bo Hu. CROSI Mapping System (CMS) - Result of the 2005 Ontology Alignment Contest. In *Integrating Ontologies '05, Proceedings of the K-CAP 2005 Workshop on Integrating Ontologies, Banff, Canada, October 2, 2005*, 2005.
15. Patrick Lambrix and Qiang Liu. Using partial reference alignments to align ontologies. In *Proceedings of the 6th European Semantic Web Conference (ESWC)*, pages 188–202, 2009.
16. Yoonkyong Lee, Mayssam Sayyadian, AnHai Doan, and Arnon S. Rosenthal. eTuner: tuning schema matching software using synthetic scenarios. *VLDB Journal: Very Large Data Bases*, 16(1):97–122, 2007.
17. Sean Luke. *Essentials of Metaheuristics*. Lulu, 2009.
18. Malgorzata Mochol, Anja Jentzsch, and Jérôme Euzenat. Applying an analytic method for matching approach selection. In *Proceedings of the 1st International Workshop on Ontology Matching (OM-2006)*, 2006.
19. Natalya F. Noy and Mark A. Musen. The PROMPT suite: interactive tools for ontology merging and mapping. *International Journal of Human-Computer Studies*, 59(6):983–1024, 2003.
20. J. R. Quinlan. Learning with continuous classes. In *Proceedings of the 5th Australian Joint Conference on Artificial Intelligence*, pages 343–348, 1992.
21. S. Russel and P. Norvig. *Artificial Intelligence: a Modern Approach*. Prentice-Hall, 1995.
22. Pavel Shvaiko and Jérôme Euzenat. Ten Challenges for Ontology Matching. In *On the Move to Meaningful Internet Systems: OTM 2008*, volume 5332 of *LNCS*, pages 1164–182. Springer, 2008.
23. R. Storn and K. Price. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11:341–359, 1997.
24. Ondrej Sváb and Vojtech Svátek. Combining ontology mapping methods using bayesian networks. In *Proceedings of the 1st International Workshop on Ontology Matching (OM)*, volume 225, pages 206–210, 2006.
25. Peng Wang and Baowen Xu. Lily: Ontology Alignment Results for OAEI 2009. In *Proceedings of the 4th International Workshop on Ontology Matching (OM-2009)*, 2009.
26. X.-S. Yang and S. Deb. Cuckoo search via lévy flights. In *Proceedings of World Congress on Nature and Biologically Inspired Computing (NaBIC 2009)*, pages 210–214, 2009.
27. Anna V. Zhdanova and Pavel Shvaiko. Community-driven ontology matching. In *Proceedings of the 3rd European Semantic Web Conference (ESWC)*, pages 34–49, 2006.