Discrete Particle Swarm Optimisation for Ontology Alignment

Jürgen Bock*,[a], Jan Hettenhausen[b]

[a]FZI Research Center for Information Technology, Haid-und-Neu-Straße 10-14, 76131 Karlsruhe, Germany
[b]Institute for Integrated and Intelligent Systems, Griffith University, Nathan Campus, Brisbane, Australia

Abstract

Particle Swarm Optimisation (PSO) is a biologically-inspired, population-based optimisation technique that has been successfully applied to various problems in science and engineering. In the context of semantic technologies, optimisation problems also occur but have rarely been considered as such. This work addresses the problem of ontology alignment, which is the identification of overlaps in heterogeneous knowledge bases backing semantic applications. To this end, the ontology alignment problem is revisited as an optimisation problem. A discrete particle swarm optimisation algorithm is designed in order to solve this optimisation problem and compute an alignment of two ontologies. A number of characteristics of traditional PSO algorithms are partially relaxed in this article, such as fixed dimensionality of particles. A complex fitness function based on similarity measures of ontological entities, as well as a tailored particle update procedure are presented. This approach brings several benefits for solving the ontology alignment problem, such as inherent parallelisation, anytime behaviour, and flexibility according to the characteristics of particular ontologies. The presented algorithm has been implemented under the name MapPSO (Ontology **Map**ping using **P**article **S**warm **O**ptimisation). Experiments demonstrate that applying PSO in the context of ontology alignment is a feasible approach.

Key words: discrete particle swarm optimization, ontology alignment, ontology mapping, ontology matching, semantic integration, semantic web

1. Introduction

Particle Swarm Optimisation (PSO) [17, 27] is a biologically-inspired optimisation meta-heuristic which has continuously gained momentum in recent

*Corresponding author
Email addresses: bock@fzi.de (Jürgen Bock), j.hettenhausen@griffith.edu.au (Jan Hettenhausen)

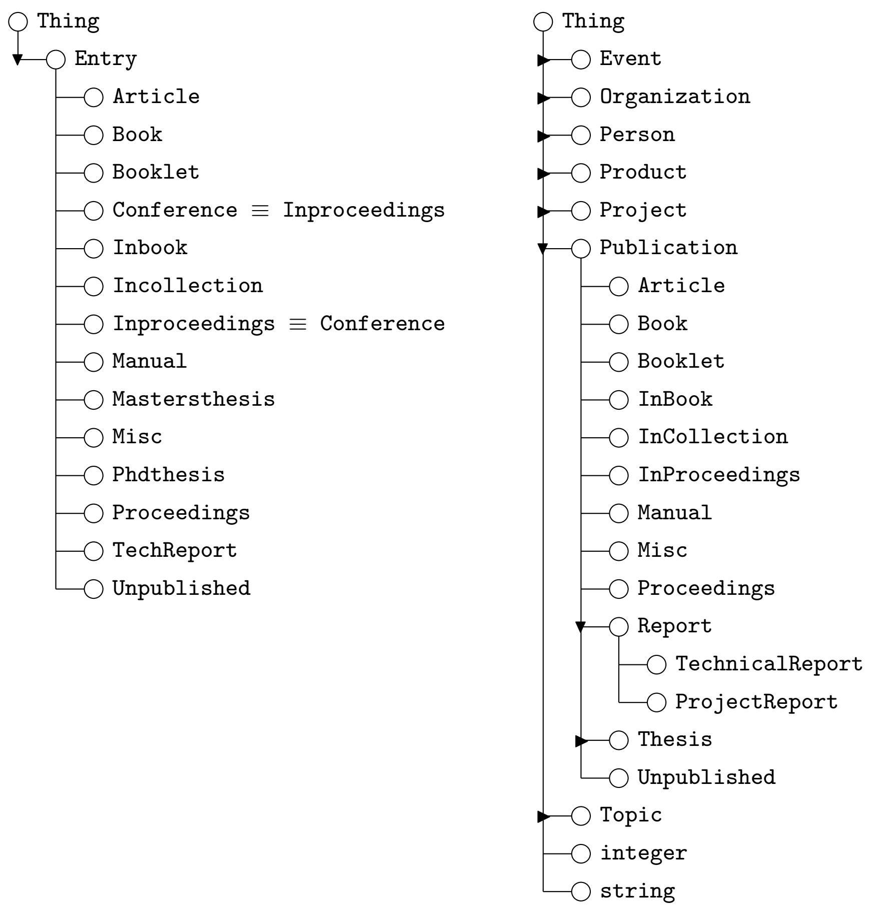
years. It is generally applicable for problems, where the global optimum of an objective function is to be found in a multi-dimensional search space. Although originally developed for continuous optimisation problems, a number of modifications have been proposed to make PSO also applicable to discrete optimisation problems [17, 2, 3]. Convergence and run-time behaviour of PSO cannot be trivially determined, however, increasingly numerous case studies and applications show that it usually performs better than non-heuristic optimisers[1].

The emergence of intelligent information systems bears the need for information and knowledge to be represented in machine readable form. Ontologies in the context of computer science have been formally described by Gruber [11] in 1993 as "an explicit specification of a conceptualization"—a description which has frequently been refined and reinterpreted since. In particular the attribute "formality" has been added to the definition, since formality is a key feature for ontologies in order to be machine processable and hence usable in automated systems. Making real-world concepts explicit in a machine readable form now allows for the modelling of knowledge bases in order to make *knowledge* processable by intelligent systems. Informally, ontologies formalise concepts, individuals, and relations among them, in order to describe real-world entities in a certain domain of interest.

From a number of approaches to standardise a language for ontologies, the Web Ontology Language (OWL) [13] became widely accepted and is standardised by the W3C. Large extents of OWL are based on Description Logics [1]. This logical underpinning enriches OWL ontologies with a formal semantics allowing them for being used in so-called *semantic applications* or the Semantic Web.

One of the key benefits of using ontologies in semantic applications apart from the ability to infer implicit information, is the automated processing of knowledge that is formally described in ontologies. This allows for *e.g.* integrating knowledge to be used in semantic application or the Semantic Web from different sources, *i.e.* ontologies. This problem is tackled in the discipline of *ontology alignment*. It is based on the observation, that ontologies to be integrated are heterogeneous models, often representing a similar or equal domain of interest, and hence have a certain overlap. As an example consider the two ontologies about bibliography presented in Fig. 1. The ontologies originate from different providers (MIT and University of Karlsruhe) and an intelligent information system might want to refer to literature which is annotated by either of the two ontologies in an integrated application. As one can easily see, there is a significant overlap in these two ontologies, which needs to be identified by sophisticated ontology alignment systems. Other examples for the need of ontology alignment can be found in the context of information sharing among peers in distributed environments, such as peer-to-peer systems or grid environments [25, 15, 30]. A use case for ontology alignment would also

[1]For some variations of PSO such analyses were performed. They are, however, not trivially transferable to the proposed DPSO algorithm.



(a) MIT BibTEX ontology (b) Karlsruhe BibTEX ontology

Figure 1: Two example ontologies about the domain of bibliography. The figures show the class hierarchy of each ontology, *i.e.* indented classes are subclasses of their parent, which denotes an *is-a* relationship between sub- and superclasses. Note that there is only a partial overlap between the two ontologies, since ontology 1b covers a wider domain.

become apparent in a medical information system which needs to incorporate knowledge from a disease ontology, as well as from an ontology about human anatomy. In order to retrieve information about which body parts are affected by a certain disease, both ontologies need to be consulted. Since the disease ontology contains information about anatomy in much less detail, a small overlap between the two ontologies exists. Again an ontology alignment system is required to identify this overlap and provide an alignment, which can be used by the medical information system.

An alignment is defined as a set of correspondences between ontological entities, *i.e.* classes, properties, and individuals, from two ontologies. It is agreed that finding a unique best alignment of two ontologies is an inherently difficult task, which is hard or even impossible to accomplish. This does not only apply to automatic ontology alignment systems, but often also for humans. Hence the optimal alignment is often unknown, since there is no gold standard as a reference. Another problem is the increasing size of ontologies, *e.g.* in the medical domain, in library use cases, or in other large scale thesauri, where ontologies with several ten thousand concepts are common. This article regards the ontology alignment problem as an optimisation problem and then applies an adapted discrete PSO (DPSO) algorithm to find an optimal alignment.

For ontology alignment problems the application of PSO provides several potential benefits. Firstly, the approach can easily process very large inputs, since an iterative traversal of large ontologies is avoided. Secondly, it serves as a general framework, where the objective function, *i.e.* the evaluation of an alignment can easily be adjusted and refined towards particular alignment scenarios and domains. Furthermore, the strategy for particles to search for an optimal alignment can easily be refined. Thirdly, due to the independent computation of each particle, the algorithm is highly parallelisable, which is a crucial feature to keep up with recent developments in computer architecture. A final advantage is the inherent anytime behaviour of PSO, where the algorithm can be interrupted at any time or after a certain number of iterations, and will provide the best alignment found so far. This is important for time critical tasks, where quality can be traded for speed.

PSO is adapted for ontology alignment to achieve the following two goals:

1. Identify the most reasonable alignment
2. Maximise the number of correspondences in the alignment

The first objective is what is most widely understood as the ontology alignment problem. The second objective arises out of the observation, that there is most likely not a complete overlap of ontologies to be aligned, *i.e.* not all ontological entities of either ontology will be involved in the desired alignment. However, one still does not want to miss any of the valid correspondences, so the goal is to maximise the number of correspondences contained in the alignment.

The presented solution is based on a DPSO by Correa *et al.* [2, 3], who applied this optimisation strategy for selecting the optimal set of attributes for a classifier. The authors compared their DPSO with a more traditional version, the binary PSO, and demonstrated that DPSO performs better for this task.

Since the selection of correspondences for an alignment is a similar task, this previous work is very relevant.

The work presented in this article has been implemented under the name MapPSO (Ontology **Map**ping using **P**article **S**warm **O**ptimisation), and experiments demonstrating the feasibility of the approach have been conducted.

In the remainder of this article, the approach of adapting PSO to the ontology alignment problem is formally described and the implementation, called MapPSO, is introduced. The following Section provides an overview of related work and the basics behind PSO are recalled. Furthermore a relevant variant of DPSO is explained. In Section 3 the ontology alignment problem will be revisited as an optimisation problem. A DPSO algorithm for solving this optimisation problem is presented in Section 4. Section 5 presents a fitness function in terms of so-called base matchers, which are used to evaluate a candidate alignment. Finally, the MapPSO system as a prototypical implementation is presented in Section 6 and experimental results are discussed in Section 7. Section 8 summarises this article and provides an outlook on future work.

2. Related Work

This section gives an overview of scientific work that is related to this research. Since this work relates to different fields, relevant work in PSO, ontology alignment, and previous efforts to combine nature inspired optimisation techniques with semantic web research are discussed separately.

2.1. Particle Swarm Optimisation

Particle Swarm Optimisation (PSO) is a non-deterministic, population-based optimisation technique originally developed by Kennedy and Eberhart [16] and later refined by Shi and Eberhart [27]. It is commonly classified under the paradigm of computational swarm intelligence [7]. The concepts of PSO are derived from behavioural models of swarming animals such as swarms of bees and fish schools.

Although it is relatively new in comparison to other common optimisation techniques, such as Genetic Algorithms, PSO has quickly gained momentum and has been successfully applied to a wide variety of optimisation problems since. The performance of PSO in terms of speed of convergence and ability to overcome local minima is on a similar good level as the performance of, *e.g.* Genetic Algorithms. Yu *et al.* [32] even state that for the majority of optimisation problems the convergence of PSO is superior to the one of Genetic Algorithms as a result of the social information sharing that PSO incorporates.

For a single-objective optimisation problem with an n-dimensional parameter space, PSO models a fixed number of particles that can freely swarm within the parameter space such that each particle's position is an input vector to the objective function. In every iteration the current position of each particle is evaluated against the objective function in order to acquire a fitness value for the corresponding particle. Every particle also keeps a memory of the position with the highest fitness value it has discovered so far, denoted as the *cognitive component*, and the swarm or neighbourhoods of particles within the swarm preserve a memory of the best position any particle in the swarm has visited up to that iteration, denoted as the *social component.* The movements of the particles are based on velocity vectors which for continuous problems comprise a social component, a cognitive component, and an inertia consisting of the weighted previous velocity. These cognitive and social components are also weighted with a constant factor and an additional random value in a certain range to allow over- and undershooting of the personal and global best positions. The velocity for each particle is updated in every iteration based on the updated social and cognitive component from the previous iteration and new positions for all particles are determined based on the velocity and their previous positions.

While PSO is widely applicable to continuous problems, it is in its basic form not applicable to discrete problems. For this, however, a series of adapted PSO algorithms have been developed to fill this gap. Binary PSO has been proposed by Kennedy and Eberhart [17]. It uses a discrete parameter space, where its position in each dimension can be either 0 or 1. Binary PSO is used for instance by Zhang *et al.* to tackle the problem of overlapping coalition formation in multi-agent systems, particularly agents participating in multiple virtual organisations [33]. More recently there has been work on discrete PSO, which uses a more relaxed notion of dimensionality. The method of Correa *et al.* [2, 3] for instance uses variable dimensionality for each particle. The authors propose a novel DPSO algorithm for attribute selection in classification tasks. The objectives in this case are to maximise the accuracy of the classifier and to find the smallest subset of attributes necessary to achieve high accuracy. The varying dimensionality of the particles entail a more natural adaptation to the task of attribute selection. To this end, each particle can represent a different number and selection of attributes. In the context of ontology alignment, where an alignment is a set of correspondences between ontological entities, a similar scenario is to be addressed.

2.2. Nature Inspired Optimisation Techniques for the Semantic Web

The idea of using nature inspired optimisation techniques in the semantic web has recently been attracting increasing interest. For this purpose a workshop "Nature inspired Reasoning for the Semantic Web (NatuReS)" has been set up at the International Semantic Web Conference (ISWC) in 2008 to attract awareness of this possibility. A recent work by Oren *et al.* [23] addresses the problem of query answering in RDF (Resource Description Framework) [20] by utilising evolutionary algorithms. The intention is to achieve anytime behaviour in this time critical task. However, the benefit gained in the domain of RDF query answering is negligible, since highly optimised query engines are hard to compete with.

In the context of ontology alignment, however, application of nature inspired optimisation techniques is more promising. Martinez-Gil *et al.* introduce the GOAL system [18], where a genetic algorithm is used to determine the optimal weight configuration for a weighted average aggregation of various base matchers. The system does not directly treat the ontology alignment problem as an optimisation problem, but rather serves as a meta optimisation. It takes several iterative runs of a matching algorithm to come up with an optimal weight configuration for a given alignment task. The algorithm requires a reference alignment in order to evaluate its fitness function and can hence only provide an optimal weight configuration for alignment problems, where the optimal solution is already known. For alignment problems with unknown solutions the system can only serve as a heuristic, if the optimal weight configuration has been determined for a similar problem with a known solution. Due to the different objective of the GOAL system, it is not directly comparable to this proposed PSO-based approach.

In contrast to GOAL, the GAOM system developed by Wang *et al.* [31] tackles the ontology alignment problem as an optimisation problem, similar to this approach. GAOM utilises a genetic algorithm, where each population member (chromosome) represents an alignment of two ontologies. Each chromosome is evaluated by a fitness function. To this end the authors distinguish between extensional and intentional features of ontological concepts which are compared using certain mapping rules to validate the alignment represented by each chromosome. The system as described by the authors has several restrictions, which are overcome by this PSO-based approach. Firstly, in GAOM a *complete* alignment is represented by each chromosome, *i.e. all* concepts from the first ontology participate in the alignment. In situations where there is only a partial overlap of ontologies, the algorithm cannot reduce the number of correspondences accordingly. Secondly, it is unclear, how structural similarities w.r.t. the concept hierarchy can be exploited. The system respects relations between concepts by comparing relations among them lexically, which is feasible for object properties, but does not account for subsumption relations defining the concept hierarchy. Thirdly, the system only considers classes in the ontologies, disregarding properties. These limitations do not effect the algorithm's performance on the Ontology Alignment Evaluation Initiative's benchmark test cases, but will bear problems for other more realistic ontology alignment scenarios. The GAOM system is neither available for download, nor it has officially participated in the Ontology Alignment Evaluation Initiative, which precludes an experimental comparison with this PSO-based approach.

2.3. Ontology Alignment

The field of ontology alignment has been attracting more and more interest in recent years. A comprehensive overview of state-of-the-art work in ontology alignment is given by Euzenat and Shvaiko [9]. The most recent advances in the field are reflected in the Ontology Matching Workshop series, such as in the OM-2008 Workshop on Ontology Matching [28]. Many of the systems build a large matrix of all possible combinations of entity matches and calculate their similarities. This can become rather cumbersome, in particular when the ontologies are very large. The PSO method does not face this problem, since it starts from random though valid candidate alignments. Similar to traditional systems, such as Similarity Flooding [19], or FOAM [6] it converges to a stable optimum in an iterative process. Similarity Flooding performs an iterative fix-point computation to approach this optimum, while FOAM performs a repeated similarity computation and aggregation. Direct formulation of the ontology alignment problem as an optimisation problem has also previously been stated [4, 31].

In the context of the Ontology Matching Workshop series, an Ontology Alignment Evaluation Initiative (OAEI)[2] has been set up in order to compare different alignment systems. The OAEI has several tracks in order to validate systems in different alignment scenarios. Core part of the OAEI is the benchmark track, which provides a number of test cases with reference alignments. This allows alignment systems to be directly comparable in terms of precision and recall w.r.t. the reference alignments. Tests in the benchmark track alter a given ontology systematically by omitting several language features or information. This requires alignment systems to consider different information contained in the ontologies in order to come up with reasonable correspondences between entities.

In order to evaluate alignment systems for large ontologies, one track in the OAEI 2008 campaign was the *vlcr*[3] track, where the focus was on scalability in ontology alignment tasks. The DSSim system [21] was the only tool that participated in this track. DSSim applies parallel computation techniques in order to process the large thesauri. To this end it splits the ontologies into several chunks which are processed in parallel. A similar technique has been proposed by Paulheim [24]. However, the method of splitting large ontologies inherently decreases the precision value of the alignment unless sophisticated partitioning and combination methods are applied for the different chunks.

3. Ontology Mapping as Optimisation Problem

This section firstly formally defines an ontology alignment, and then formulates the problem of finding a best alignment as an optimisation problem.

For the purpose of this article, the following simplified, informal notion of an ontology is sufficient. An ontology $\mathcal{O}$ has a set of classes C, a set of properties P and subsumption relations on the elements of C and P resp. For two ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$, the set of all possible pairs of entities allowed for an alignment is defined as

$$\mathcal{C} = (C_1 \times C_2) \ \cup \ (P_1 \times P_2) \tag{1}$$

with C_1 and P_1 the sets of classes and properties of $\mathcal{O}_1$, and C_2 and P_2 the sets of classes and properties of $\mathcal{O}_2$ resp. An (equality) *correspondence*

$$c \in \mathcal{C} \tag{2}$$

[2]`http://oaei.ontologymatching.org/2008/`

[3]*very large crosslingual resources*

denotes a pair of entities (e_1, e_2) meaning that e_1 and e_2 refer to the same real-world entity. An *alignment*

$$A \subseteq \mathcal{C} \tag{3}$$

is a set of correspondences between entities of the two ontologies. In this approach only 1:1 alignments are considered, *i.e.* A denotes a bijective relation. In other words, each entity $e_1 \in C_1 \cup P_1$ corresponds to exactly one entity $e_2 \in C_2 \cup P_2$ and vice versa.

For a correspondence c, a *fitness function*

$$f(c) = \Gamma\left(\vec{h}(c), \vec{\omega}\right) \tag{4}$$

evaluates c, according to an *evaluation strategy* Γ, a vector of *rating functions* $\vec{h}$, and a vector of weights $\vec{\omega}$, where the weights determine the influence of each rating function. In a simple example, vectors $\vec{h}$ and $\vec{\omega}$ consist of a single element each, where $h(c) = h_1(c) = lev(c)$ the Levenshtein distance of the natural language labels of corresponding entities in c, $\omega = \omega_1 = 1$, and the strategy $\Gamma = \text{id}$. Thus $f(c) = lev(c)$. The fitness of an alignment A

$$F(A) = \frac{\sum_{i=1}^{|A|} f(c_i)}{|A|} \qquad c_i \in A \tag{5}$$

is the average fitness of its correspondences, where $|A|$ denotes the size, *i.e.* the number of correspondences in A.

Since distance measures are used to evaluate correspondences, lower evaluation values denote better correspondences / alignments. Apart from this, the second objective is to maximise the size, *i.e.* the number of correspondences in an alignment. Hence the goal is for all possible candidate alignments A to identify

$$A^* = \text{argmin } F(A) \tag{6}$$

which is the alignment A which causes $F(A)$ to be minimal, and at the same time

$$A^* = \text{argmax } |A| \tag{7}$$

which is the alignment A which causes $|A|$ to be maximal. The two objective functions are aggregated to a single fitness value, representing the overall evaluation of the alignment, that also respects the size of the alignment. For this purpose the following weighted average formula is used:

$$\bar{F}(A) = \eta(\min\{|C_1 \cup P_1|, |C_2 \cup P_2|\} - |A|) + (1 - \eta)F(A) \tag{8}$$

The first part of the sum weighted by η, accounts for the maximisation of the number of correspondences by calculating the number of entities of the smaller ontology that are not part of the alignment A. The second part represents the evaluation of A as in (5). Hence the best alignment A^* can be determined as

$$A^* = \text{argmin } \bar{F}(A) \tag{9}$$

Since the number of possible candidate alignments is typically very large, a sophisticated search strategy is needed for the PSO algorithm, which will be discussed in the following section.

4. A Particle Swarm Algorithm for Ontology Alignment

This section introduces an ontology alignment algorithm based on DPSO. Firstly important notions and computation procedures are formally defined, which will then be illustrated by an example. Finally the algorithm is sketched in a condensed presentation.

4.1. Formal Definitions

In the proposed solution for ontology alignment by DPSO a particle swarm consists of a number N of particles, the so-called *population*, which evolves in a number I of *iterations*. In traditional PSO, in each iteration, each particle evolves using a so-called *velocity* vector, which determines its new position in the parameter space. This evolution happens via a guided, randomised re-initialisation of each particle. Since this approach uses a modified *discrete* PSO, this idea is partially relaxed, and particles and velocities are defined following the approach of Correa *et al.* [2].

Each particle represents a *candidate alignment*. Particles can have different dimensionality, *i.e.* the number of correspondences in the alignment it currently represents, and hence differ from traditional PSO, where each particle has the same dimensionality. A dimensionality of zero means that a particle represents the empty alignment. For $p \in \{1, \ldots, N\}$, a particle is defined as a vector[4]

$$\vec{X}_p = \left\{c_{(p,1)}, c_{(p,2)}, \ldots, c_{(p,k)}\right\} \tag{10}$$

where for each $j \in \{1, \ldots, k\}$, $c_{(p,j)}$ is a correspondence as in (2). This set of correspondences is also called a *configuration* of the particle. Since only 1:1 alignments are considered so far, there can be at most $n = \min\{|C_1|, |C_2|\} + \min\{|P_1|, |P_2|\}$ correspondences in an alignment. The (variable) dimensionality of a particle is $k \in \{0, \ldots, n\}$. According to (5), the fitness of a particle $\vec{X}_p$ is

$$F(\vec{X}_p) \tag{11}$$

Each particle maintains the configuration of the best alignment it has ever represented w.r.t. F. This *personal best* (*pBest*) alignment of dimensionality $l \in \{0, \ldots, n\}$ is denoted by

$$\vec{B}_p = \left\{d_{(p,1)}, d_{(p,2)}, \ldots, d_{(p,l)}\right\} \tag{12}$$

where for each $j \in \{1, \ldots, l\}$, $d_{(p,j)}$ is a correspondence. Note that the number of correspondences can change during the iteration of the swarm (see Section 4.3). Hence the dimensionality l of the *pBest* configuration of a particle does not need to coincide with the dimensionality k of its current configuration.

[4]Note that the exact mathematical notation is violated and the vector is denoted with curly braces, as it can also be seen as a set.

The *global best* (*gBest*), *i.e.* the best performing parameter configuration any particle in the swarm has ever represented w.r.t. F is denoted by

$$\vec{G} = \{d_1,\, d_2,\, \dots,\, d_m\} \tag{13}$$

where for each $j \in \{1,\, \dots,\, m\}$, d_j is a correspondence. Its dimensionality is $m \in \{0,\, \dots,\, n\}$.

To ensure a guided convergence towards an optimal alignment during the iterations, the influence of arbitrary random re-initialisation of each particle has to be restricted. To this end, the likelihood is raised that those correspondences in a particle are preserved, which *(i)* are evaluated best, and *(ii)* are also present in the global (13) or personal (12) best alignment.

The *fitness vector* of a particle is denoted by a 2-by-k array

$$\vec{F}_p = \begin{pmatrix} f_{(p,1)} & f_{(p,2)} & \cdots & f_{(p,k)} \\ c_{(p,j_1)} & c_{(p,j_2)} & \cdots & c_{(p,j_k)} \end{pmatrix} \tag{14}$$

mapping a fitness $f_{(p,\mu)}$ to each correspondence $c_{(p,j_\mu)}$. (See (4) and Section 5.) Note that the vector is ordered by its fitness values.

A *velocity vector* is defined as another 2-by-k array

$$\vec{V}_p = \begin{pmatrix} v_{(p,1)} & v_{(p,2)} & \cdots & v_{(p,k)} \\ c_{(p,l_1)} & c_{(p,l_2)} & \cdots & c_{(p,l_k)} \end{pmatrix} \tag{15}$$

mapping a proportional likelihood $v_{(p,\mu)}$ to each correspondence $c_{(p,l_\mu)}$. Note that the vector is ordered by its proportional likelihoods. Proportional likelihoods are used to raise the probability of those correspondences to be preserved in a particle, that are also present in the personal and global best alignments. Initially, for each $c_{(p,l_\mu)}$, $v_{(p,\mu)}$ is set to 1. This initialisation is also done for new correspondences joining the particle during its evolution. The update of the proportional likelihoods is then done in two steps, using two parameters $\beta \in \mathbb{R}^+$ and $\gamma \in \mathbb{R}^+$. Firstly, if $c_{(p,l_\mu)}$ is present in $\vec{B}_p$, add β to $v_{(p,\mu)}$. If it is present in $\vec{G}$, add γ to $v_{(p,\mu)}$. These two parameters control the influence of the fact that a correspondence is also present in the personal best (β) or the global best (γ) alignment resp. After this, each $v_{(p,\mu)}$ is multiplied by a uniform random number $\phi_\mu \in (0,\, 1)$.

To calculate a *keep-set*, which will *not* be replaced by a random re-initialisation during an iteration, two sets are defined as

$$\begin{aligned} F_{(p,\kappa)} &= \Big\{ c_{(p,j_\mu)} \mid \mu \in \{1,\, \dots,\, \kappa \cdot k\},\ j_\mu \text{ a reordering as in } \vec{F}_p \Big\} && (16) \\ V_{(p,\kappa)} &= \Big\{ c_{(p,l_\mu)} \mid \mu \in \{1,\, \dots,\, \kappa \cdot k\},\ l_\mu \text{ a reordering as in } \vec{V}_p \Big\} && (17) \end{aligned}$$

with a parameter $\kappa \in (0,\, 1)$ to control the size of the keep-set. The sets $F_{(p,\kappa)}$ and $V_{(p,\kappa)}$ hence contain those correspondences of a particle, which are the $\kappa \cdot k$ best evaluated, and highest ranked according to their proportional likelihood resp. The *keep-set* is now defined as

$$K_{(p,\kappa)} = F_{(p,\kappa)} \cap V_{(p,\kappa)} \tag{18}$$

containing those correspondences, which are part of both sets $F_{(p,\kappa)}$ and $V_{(p,\kappa)}$.

For a more stringent convergence towards an optimum alignment, an additional *safe-set* is introduced as

$$S_{(p,\sigma)} = \left\{ c_{(p,j_\mu)} \mid f_{(p,\mu)} < \sigma \right\} \tag{19}$$

a set of correspondences, which will never be replaced in this particle. Here $\sigma \in (0,\, 1)$ is the fitness threshold for correspondences to be included in the safe-set. Since there is the chance of getting stuck in a local optimum for the alignment, one would typically choose a very small value for σ. In each iteration, the update algorithm firstly computes a new particle length k' according to a self-adaptation process as discussed later in this section. Secondly, the particle keeps the set $S_{(p,\sigma)} \cup K_{(p,\kappa)}$ and replaces the remaining $k' - |S_{(p,\sigma)} \cup K_{(p,\kappa)}|$ correspondences with new random ones. This behaviour ensures a convergence of each particle towards an optimum according to (5), since the keep-set will steadily increase, and the fluctuation due to random re-initialisation will become less drastic as the swarm evolves.

The presented DPSO differs from the approach by Correa *et al.* mainly in two aspects. Firstly, the size, *i.e.* dimensionality of each particle is updated in each iteration, where in the approach of Correa *et al.* each particle is given a randomly chosen size, which does not change throughout the iterations. In their approach this is reasonable seeing that in their experiment [2] the authors used a population size, which is much larger than the number of possible particle lengths. For the problem of ontology alignment the number of possible particle lengths can be much larger, since it depends on the size of the input ontologies, *i.e.* their number of classes and properties. It might thus become difficult to increase the population size accordingly, which makes it necessary to dynamically adjust the particle lengths in order to find the optimal size of an alignment. The second aspect in which this approach differs from the one of Correa *et al.* is the particle update procedure. In this approach, the change of a particle's configuration does not only depend on the configuration of the personal best and global best, but also on the evaluation of the single correspondences. This is not possible in the use case of attribute selection for a classifier, as attributes cannot be evaluated independently.

4.2. Example

In order to illustrate the theoretical procedure from Section 4.1, one iteration is run through in this example, updating a particle $\vec{X}_p$. Consider an alignment of the two example ontologies presented in Fig. 1. Suppose, $\vec{X}_p$ represents an alignment consisting of $k = 5$ correspondences

$$\vec{X}_p = \left\{ c_{(p,1)},\, c_{(p,2)},\, c_{(p,3)},\, c_{(p,4)},\, c_{(p,5)} \right\}$$

which are allocated as in Table 1. Suppose the fitness values of the single correspondences have been determined and are represented as follows

$$\vec{F}_p = \begin{pmatrix} 0.04 & 1.23 & 1.55 & 3.65 & 7.54 \\ c_{(p,3)} & c_{(p,2)} & c_{(p,5)} & c_{(p,1)} & c_{(p,4)} \end{pmatrix}$$

Table 1: Example set of correspondences and assigned fitness values for a candidate alignment represented by particle p of the two example ontologies presented in Fig. 1.

Correspondence	Fitness value
$c_{(p,1)} = (\texttt{Unpublished}, \texttt{Publication})$	3.65
$c_{(p,2)} = (\texttt{Incollection}, \texttt{InCollection})$	1.23
$c_{(p,3)} = (\texttt{Book}, \texttt{Book})$	0.04
$c_{(p,4)} = (\texttt{Mastersthesis}, \texttt{Event})$	7.54
$c_{(p,5)} = (\texttt{TechReport}, \texttt{TechnicalReport})$	1.55

Note, that the array is sorted by its fitness values in ascending order, as lower values mean a better evaluation.

The velocity vector $\vec{V}_p$ has been initialised with all proportional likelihoods set to 1:

$$\vec{V}_p = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ c_{(p,1)} & c_{(p,2)} & c_{(p,3)} & c_{(p,4)} & c_{(p,5)} \end{pmatrix}$$

Now suppose, correspondences $c_{(p,2)}$, $c_{(p,3)}$, and $c_{(p,5)}$, are also present in $\vec{B}_p$, and $c_{(p,3)}$ is also present in $\vec{G}$. Parameters β and γ are added accordingly (*e.g.* $\beta = 0.4$ and $\gamma = 0.5$):

$$\vec{V}_p = \begin{pmatrix} 1 & (1+\beta) & (1+\beta+\gamma) & 1 & (1+\beta) \\ c_{(p,1)} & c_{(p,2)} & c_{(p,3)} & c_{(p,4)} & c_{(p,5)} \end{pmatrix}$$

After adding the parameters, each proportional likelihood is multiplied by a uniform random number $\phi_j \in (0, 1)$, $\forall j \in \{1, \ldots, 5\}$. The array will then be sorted by its proportional likelihoods in descending order, as higher values mean a higher likelihood. This might result in something like

$$\vec{V}_p = \begin{pmatrix} 1.34 & 1.12 & 0.88 & 0.76 & 0.32 \\ c_{(p,2)} & c_{(p,5)} & c_{(p,4)} & c_{(p,3)} & c_{(p,1)} \end{pmatrix}$$

Suppose $\kappa = 0.6$ is chosen, so the *keep-set* $K_{(p,\kappa)}$ is built as the intersection of the first $\kappa \cdot k = 3$ correspondences of the arrays $\vec{F}_p$ and $\vec{V}_p$, which results in

$$K_{(p,\kappa)} = \left\{c_{(p,2)},\, c_{(p,5)}\right\}$$

Assume that $\sigma = 0.1$ is chosen and thus the *safe-set* $S_{(p,\sigma)}$ determined as

$$S_{(p,\sigma)} = \left\{c_{(p,3)}\right\}$$

The update algorithm will now keep the set

$$S_{(p,\sigma)} \cup K_{(p,\kappa)} = \left\{c_{(p,2)},\, c_{(p,3)},\, c_{(p,5)}\right\}$$

and replaces the remaining two correspondences with random new ones.

4.3. Self-Adaptation of Particle Length

A general problem when aligning two ontologies is that the actual number of correspondences is not known upfront. This method approaches this by assigning each particle a random number of correspondences during the initialisation. As usual, no knowledge about the actual number of correspondences exist, the initial guesses are uniformly distributed between zero and the maximum number of possible correspondences between the two ontologies considering only 1:1 alignments. Assuming that the chances for a particle to receive a good fitness value are higher if its number of correspondences is close to the actual number of correspondences, the heuristic below attempts to adjust the number of correspondences for each particle and in each iteration based on the current global best particle. Let k_p be the number of correspondences represented by particle p, and let k_{gBest} be the number of correspondences represented by the *gBest*. (Note that class and property correspondences are computed separately.) Each particle adjusts its number of class / property correspondences if the following expression becomes true

$$\begin{cases} r_1 \geq \tau_i & \text{if } k_{gBest} > k_p \\ r_1,\, r_2 \geq \tau_i & \text{if } k_{gBest} < k_p \\ \text{false} & \text{else} \end{cases} \tag{20}$$

where r_1 and r_2 denote random values and τ_i an iteration dependent threshold value defined as

$$\tau_i = \lambda \left(\frac{i}{i_{\max}} \right)^2 \tag{21}$$

where λ is a constant weighting factor and i and $i_{\max}$ denote the current and maximum iteration resp. The probability for a change therefore increases with the number of iterations which prevents very rapid changes of the number of correspondences at the beginning of the process where the prediction of the actual number of correspondences is less accurate than later in the optimisation. Furthermore the probability for decrease is always significantly lower than for an increase. The underlying assumption behind this is that more correspondences are generally more desirable and in the majority of tests this scheme has proven to be successful.

For each of the particles whose number of correspondences are changed, the maximum range of this change is determined by

$$\Delta k_p^{max} = \begin{cases} w_{inc} \cdot (k_{gBest} - k_p) & \text{if } k_{gBest} > k_p \\ w_{dec} \cdot (k_p - k_{gBest}) & \text{if } k_{gBest} < k_p \end{cases} \tag{22}$$

where w_{inc} and w_{dec} denote constant weighting factors for the size of the interval. The extended range ensures, similar to the velocity update in continuous PSO, that the interval exceeds the distance between the two values and allows a new value to either under- or overshoot the reference value, *i.e.* the number of correspondences of the global best particle. The new number of correspondences of each type for a particle p is then adjusted by a random value Δk_p from the interval $\left[0,\, \Delta k_p^{max}\right]$. The new number of correspondences k'_p of particle p can be computed as

$$k'_p = \begin{cases} k_p + \Delta k_p & \text{if } k_{gBest} > k_p \\ k_p - \Delta k_p & \text{if } k_{gBest} < k_p \end{cases} \tag{23}$$

In the case of an increase the algorithm attempts to adopt these from the keep-set of the global best particle. If more new correspondences are needed than can be added this way the remaining correspondences are randomly created. In either case only valid correspondences are added, *i.e.* the new correspondences cannot violate constraints such as the restriction to 1:1 alignments, *etc.* When on the other hand the number of correspondences decreases, a fitness ranking of all correspondences is performed and the worst performing elements are removed.

4.4. Algorithm

This subsection presents an algorithm that computes an ontology alignment following the method presented in Section 4.1. In this presentation the algorithm is split into three parts, an initialisation step, the swarm iteration, and an update procedure to determine the new configuration of each particle.

The computation of an alignment starts with an initialisation, encoded in algorithm 1. In this initialisation step, each particle is initialised with a random number of correspondences. It also encompasses evaluation, *i.e.* computation of the fitness value of each correspondence and the initial assertion of the personal best alignment.

The execution of the algorithm is an iterative, guided evolution of the particle swarm as outlined in algorithm 2. In each iteration, the globally best alignment is updated, if a new best performing particle is seen. The guided evolution of particles behaves according to the update procedure denoted by algorithm 3. Note, that each particle can be evaluated and updated in parallel, which is one of the main advantages of this approach.

The particle update procedure in algorithm 3 states the formal definitions of Section 4.1 in a sequential manner. The single steps are explained in detail there.

5. Fitness Scores by Correspondence Evaluation

As formally discussed in Section 3, the fitness of an alignment as denoted in (5) calculates from the fitness values of each single correspondence in the alignment as denoted in (4). To this end, each correspondence is evaluated according to some rating functions which are weighted and aggregated to a single fitness value. In the context of ontology alignment, a rating function is widely known as *base matcher*, which computes a single similarity or distance measure of ontological entities. The algorithm presented in Section 4 is adjustable in term of which base matchers are to be used, *i.e.* how the fitness values are computed. This section presents some base matchers, which are currently used as rating

Algorithm 1 Initialisation of particles

```
Require: $N$ the number of particles,
  $n_C = \min\{|C_1|, |C_2|\}$,
  $n_P = \min\{|P_1|, |P_2|\}$,
  $rand_C \in \{1, \ldots, n_C\}$, a uniform random number,
  $rand_P \in \{1, \ldots, n_P\}$, a uniform random number
  for $i = 1$ to $N$ do
    $k_C \Leftarrow rand_C$
    for $j = 1$ to $k_C$ do
      Randomly select classes $e_1 \in C_1$ and $e_2 \in C_2$ that have not already been
      selected, and create correspondence $c_j = (e_1, e_2)$
      Compute $f(c_j)$ according to (4)
      $\vec{X}_i \Leftarrow \vec{X}_i \cup \{c_j\}$
    end for
    $k_P \Leftarrow rand_P$
    for $j = 1$ to $k_P$ do
      Randomly select properties $e_1 \in P_1$ and $e_2 \in P_2$ that have not already
      been selected, and create correspondence $c_j = (e_1, e_2)$
      Compute $f(c_j)$ according to (4)
      $\vec{X}_i \Leftarrow \vec{X}_i \cup \{c_j\}$
    end for
    Build $\vec{F}_i$ according to (14)
    Compute $F(\vec{X}_i)$ according to (5)
    $\vec{B}_i \Leftarrow \vec{X}_i$
  end for
```

Algorithm 2 Swarm evolution

```
Require: $N$ the number of particles,
  $I$ the number of iterations
  for $i = 1$ to $I$ do
    for $j = 1$ to $N$ do
      Update particle according to algorithm 3
      if $F(\vec{X}_j) < F(\vec{G})$ then
        $G \Leftarrow \vec{X}_j$
      end if
    end for
  end for
```

Algorithm 3 Update of particle $\vec{X}_i$

```
Require: $k$ the number of correspondences in this particle
  $\beta$, $\gamma$, $\kappa$, $\sigma$ parameters (see Section 4.1)
  $\vec{V}_i$ the proportional likelihood vector
  $\vec{F}_i$ the evaluation vector

  $k' \Leftarrow$ *updateParticleLength*$(k)$ {this also modifies $\vec{X}_i$ by adding or removing
  correspondences according to the length adjustment}
  for $\mu = 1$ to $k'$ do
    if $c_{(i,l_\mu)} \in \vec{B}_i$ then
      $v_{(i,\mu)} \Leftarrow v_{(i,\mu)} + \beta$
    end if
    if $c_{(i,l_\mu)} \in \vec{G}$ then
      $v_{(i,\mu)} \Leftarrow v_{(i,\mu)} + \gamma$
    end if
    $v_{(i,\mu)} \Leftarrow v_{(i,\mu)} \cdot \phi_\mu, \quad \phi_\mu \in (0,\, 1)$ a uniform random number
  end for
  Sort $\vec{V}_i$ by $v_i$ in descending order
  Sort $\vec{F}_i$ by $f_i$ in ascending order
  Compute $K_{(i,\kappa)}$ according to (16), (17), and (18)
  Compute $S_{(i,\sigma)}$ according to (19)
  Replace correspondences $\vec{X}_i - (S_{(i,\sigma)} \cup K_{(i,\kappa)})$ by the same number of randomly
  generated new ones.
  For each newly generated correspondence $c_{(i,j)}$ compute $f(c_{(i,j)})$ according
  to (4)
```

functions. This is by no means an exhaustive collection. In fact the quality of the computed alignment depends to a large extent on the base matchers which are used for evaluating the correspondences.

5.1. Base Matchers

The output of a base matcher is a distance between two ontological entities, represented as a value between 0 and 1. Hereby, 0 is an exact match according to the respective base matcher, and 1 is the clearest mismatch. Base matchers can be categorised in lexical, linguistic, and structural base matchers.

5.1.1. Lexical Base Matchers

Lexical base matchers compute a string distance between named entity identifiers. In the ontologies that are considered by this approach each entity is identified by a URI and can optionally have a natural language label. Currently this approach utilises a string distance, designed for ontology matching [29], the so-called *smoa* distance.

Entity Name Distance. Names of entities are encoded as a fragment in their URI. In many cases, these URI fragments are based on natural language terms. However, also other identifiers, such as numeric codes or other proprietary IDs are common. This URI fragment is used by this base matcher.

Label Distance. Entities can be annotated by a label, which is typically a natural language term. In the typical RDF/XML representation of ontologies these labels occur as `rdfs:label` annotations. The lexical *label distance* base matcher compares these labels in the same way the lexical *entity name distance* base matcher compares URI fragments.

5.1.2. Linguistic Base Matchers

Linguistic base matchers compute a similarity between entity names/labels, based on synonymy, hypernymy, and other linguistic relations. Here WordNet[5] is used in order to compute a linguistic distance. The interface is extended by a cache, since due to the (parallel) nature of the PSO algorithm, distances of entities are computed more than once[6]. This caching effectively speeds up the algorithm, since WordNet access is rather slow.

Entity Name Distance. Similar to the lexical *entity name distance* base matcher, this version compares the same entity identifiers, but computes a WordNet distance.

[5] `http://wordnet.princeton.edu/`

[6] The current implementation of this algorithm does not yet utilise a distributed hardware architecture, *i.e.* all particles are computed in multiple threads on a single machine, where a single WordNet base matcher instance is used for all particles. However, efficient caching limits the impact of this bottleneck.

Label Distance. Similar to the lexical *label distance* base matcher, this version compares the same entity labels, but computes a WordNet distance.

Comment Distance. Entities can include an `rdfs:comment` annotation. Comments may contain sentences or phrases to describe an entity in natural language. A distance measure based on the vector space model presented by Salton *et al.* [26] is used here to compare comments. To this end, let d_i be an `rdfs:comment` of an ontological entity e_i. Let $T_i = \{t_{i_1}, t_{i_2}, \ldots, t_{i_n}\}$ be a set of distinct terms occurring in d_i with n being the number of distinct terms in d_i. For two comments d_1 and d_2 to be compared, a *bag of words* is considered as the union $T = T_1 \cup T_2$. This set representation contains all terms occurring in d_1 or d_2 with no duplicates and can now be denoted as a vector $\vec{T} = (t_1, t_2, \ldots, t_m)$. Let $\vec{U}_i = (u_{i_1}, u_{i_2}, \ldots, u_{i_m})$ be a vector representation of d_i, where each u_j is the number of occurrences of t_j in d_i. The similarity of two comments d_1 and d_2 is now the cosine angle between $\vec{U}_1$ and $\vec{U}_2$

$$\cos\phi = \frac{\vec{U}_1 \cdot \vec{U}_2}{|\vec{U}_1|\,|\vec{U}_2|} \tag{24}$$

and hence the comment distance of two entities e_1 and e_2 computes as

$$h_{\text{comment}}(e_1, e_2) = 1 - \cos\phi \tag{25}$$

5.1.3. Structural Base Matchers

Structural base matchers compute a similarity between ontological entities based on their structural properties according to the "ontology graph". A class e_1 in an ontology is said to be subsumed by a class e_2, denoted by $e_1 \sqsubseteq e_2$, if any individual that is an instance of e_1 is also an instance of e_2. e_1 is called a subclass of e_2, and e_2 is a superclass of e_1 analogously. There is also a subsumption relation for properties with the notions of sub- and superproperties resp. The subsumption relation can build a subsumption hierarchy for classes and properties in the ontology. Properties can further be assigned domain and range restrictions. Properties are relating individuals of their "domain" to individuals of their "range". A property assertion $r(i_1, i_2)$ states that individual i_1 is related to individual i_2 via property r. Domains and ranges can be restricted to (possibly complex) class descriptions by stating that $\text{dom}(r) = e_D$ and $\text{ran}(r) = e_R$, which means that each individual in the domain of r must be a member of class e_D, and each individual in the range of r must be a member of class e_R.

Current implementations of base matchers account for similarities arising out of the subsumption hierarchy as well as from the domain and range restrictions on properties.

Hierarchy Distance. According to the semantics of the subsumption relation in ontologies, a correspondence between two classes/properties inherits a similarity of the correspondence between their respective superclasses/superproperties, in case this correspondence is present in the alignment the particle represents. More formally, let e_1 and e_2 be classes of different ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$. If $e_1 \sqsubseteq f_1$ in $\mathcal{O}_1$ and $e_2 \sqsubseteq f_2$ in $\mathcal{O}_2$, and there is a correspondence $c = (f_1, f_2)$ with an evaluation $f(c)$, then

$$h_{\text{hierarchy}}(e_1, e_2) = f(c) \tag{26}$$

Property Domain/Range Distance. Intuitively, two properties can be regarded as similar, if their domain and range restrictions are similar class descriptions. A current implementation accounts for this in the case of named classes as domain and range restrictions of properties. More formally, let s_1 and s_2 be (object) properties of different ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$. Let D_1 and D_2 be the sets of domain class descriptions of s_1 and s_2 resp. Let R_1 and R_2 be the sets of range class descriptions of s_1 and s_2 resp. Let A be the alignment represented by the particle, whose fitness is to be computed. Let

$$C_D = \{(f_1, f_2) \in A \quad | \quad f_1 \in D_1 \text{ and } f_2 \in D_2\} \tag{27}$$

$$C_R = \{(f_1, f_2) \in A \quad | \quad f_1 \in R_1 \text{ and } f_2 \in R_2\} \tag{28}$$

be the sets of correspondences of the domain and range class descriptions, which are also present in the alignment to be evaluated. The derived domain class distance is

$$d_{\text{der}} = \begin{cases} \frac{\sum_{c \in C_D} f(c)}{|C_D|} & \text{if } |C_D| \neq 0 \\ 1 & \text{else} \end{cases} \tag{29}$$

which averages the distances of corresponding domain classes, if there are any. In case the number of domain class correspondences present in A is close to the maximum number possible (which is the smaller number of domain classes of s_1 or s_2 resp.) this is also an indicator of similarity between s_1 and s_2. This is defined as

$$d_{\text{num}} = \frac{|C_D|}{\min\{|D_1|, |D_2|\}} \tag{30}$$

In order to also account for the number of potential domain class correspondences, regardless of their presence in A, another distance can be approximated by

$$d_{\text{pot}} = \frac{1}{\min\{|D_1|, |D_2|\} + 1} \tag{31}$$

The distance derived from the domain classes s_1 and s_2 now computes as

$$d = w_{\text{der}} d_{\text{der}} + w_{\text{num}} d_{\text{num}} + w_{\text{pot}} d_{\text{pot}} \tag{32}$$

where w_{der}, w_{num}, and w_{pot} are weighting factors (summing up to 1) in order to account for the influence of each similarity indicator.

Analogously the values are computed for range class distances, where

$$r_{\text{der}} = \begin{cases} \frac{\sum_{c \in C_R} f(c)}{|C_R|} & \text{if } |C_R| \neq 0 \\ 1 & \text{else} \end{cases} \tag{33}$$

$$r_{\text{num}} = \frac{|C_R|}{\min\{|R_1|,\, |R_2|\}} \tag{34}$$

$$r_{\text{pot}} = \frac{1}{\min\{|R_1|,\, |R_2|\} + 1} \tag{35}$$

and accordingly

$$r = w_{\text{der}} r_{\text{der}} + w_{\text{num}} r_{\text{num}} + w_{\text{pot}} r_{\text{pot}} \tag{36}$$

The total property domain/range distance computes as

$$h_{\text{propDomRan}}(s_1,\, s_2) = \left(\frac{d + r}{2}\right)^3 \tag{37}$$

which is exponentiated in order to lower the distances for bad scores that arise due to the linear calculation of the domain and range class distances.

Class as Domain/Range Distance. Intuitively, two classes can be regarded as similar, if they are domain/range classes of similar properties. Let e_1 and e_2 be classes of different ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$. Let D_1 and R_1 be the sets of properties, which have e_1 as domain or range class resp. Let D_2 and R_2 the sets of properties, which have e_2 as domain or range class resp. For datatype properties, only D_1 and D_2 are considered. Let A be the alignment represented by the particle, whose fitness is to be computed. Let

$$C_D = \{(f_1,\, f_2) \in A \quad | \quad f_1 \in D_1 \text{ and } f_2 \in D_2\} \tag{38}$$

$$C_R = \{(f_1,\, f_2) \in A \quad | \quad f_1 \in R_1 \text{ and } f_2 \in R_2\} \tag{39}$$

be the sets of correspondences between properties, which have the classes of interest (e_1 and e_2) as domain or range resp. The computation of the base distance happens the same way as in equations (29), (30), (31), (32), (33), (34), (35), (36), with the computation of $h_{\text{clsDomRan}}(e_1,\, e_2)$ as in (37).

5.2. Evaluation Strategies

In order to come up with a single evaluation of a correspondence, the various base distances need to be aggregated using an evaluation strategy Γ, defined as

$$\Gamma : \; \vec{h}(c) \times \vec{\omega} \;\rightarrow\; (0,\, 1) \tag{40}$$

where $\sum_{i=1}^{n} \omega_i = 1$. There are three possible evaluation strategies implemented, out of which one can be selected through system configuration.

Minimum Aggregation. This simple evaluation strategy does not utilise the vector of weights, and just takes the minimum, *i.e.* the best result of all base matchers.

$$\Gamma\left(\vec{h}(c), \vec{\omega}\right) = \min\{h_1(c), h_2(c), \ldots h_n(c)\} \tag{41}$$

Weighted Average Aggregation. This evaluation strategy computes the standard weighted average of all base distances.

$$\Gamma\left(\vec{h}(c), \vec{\omega}\right) = \sum_{i=1}^{n} \omega_i h_i(c) \tag{42}$$

Ordered Weighted Average Aggregation. The ordered weighted average evaluation strategy is similar to the weighted average aggregation, but applies a constant vector of weights to a reordering of the base distances, such that the base distances are in ascending order. It has successfully been applied to ontology alignment by Ji *et al.* [14].

$$\Gamma\left(\vec{h}(c), \vec{\omega}\right) = \sum_{i=1}^{n} \omega_i h_{k_i}(c) \tag{43}$$

where for $i \in \{1, \ldots n\}$, k_i is a reordering, such that $h_{k_i}(c) < h_{k_j}(c)$ for $i < j$.

6. Implementation

The algorithm for ontology alignment by DPSO as presented in Section 4 has been implemented under the name MapPSO[7] (Ontology **Map**ping using **P**article **S**warm **O**ptimisation). The prototypical Java$^{\text{TM}}$ implementation is based on the Alignment API[8] [8], which serves as an interface to ontologies and alignments. Furthermore it provides basic base matching algorithms, such as the *smoa* distance [29] and the WordNet distance, which are also used in MapPSO. The API ships with a command line processor and a server implementation, and allows for the use of MapPSO as a black box algorithm from within any semantic application, that uses the Alignment API. Within this framework a generic DPSO alignment method has been built, which can be tuned and adjusted in several ways. In particular there are two critical components, namely the *evaluation* method and the *update* method. Both implement the algorithms presented in Section 4.

The MapPSO system can be configured by a parameter file, which determines size of the population, number of iterations, the base matchers and evaluation strategy to be used along with a set of weights. Furthermore, the parameters κ and σ to determine the size of the *keep-set* and *safe-set* resp., β and γ to control the influence of personal and global best particle resp. as well

[7] `http://mappso.sourceforge.net`
[8] `http://alignapi.gforge.inria.fr/`

as parameters to regulate self-adaptive behaviour can be configured. Table 2 shows the list of parameters that can be adjusted.

Particles in MapPSO are evaluated and updated in parallel running threads. A flowchart of the system is illustrated in Fig. 2.

7. Experiments

This section demonstrates experiences drawn from initial experiments using the MapPSO algorithm. The MapPSO system is evaluated by conducting two kinds of experiments: alignment quality testing and scalability testing.

7.1. Alignment Quality

To demonstrate the correct operation of this approach the implementation was tested on the datasets of the *benchmark* track of the 2008 Ontology Alignment Evaluation Initiative (OAEI)[9]. The benchmark consists of a number of tests, where an ontology is aligned to different derivatives of itself, which are altered by removing information that can be exploited by alignment tools. These alternations change or remove natural language labels, comments, structural information, *etc.*, in order to figure out the strengths and weaknesses of different alignment systems.

In the OAEI, alignment systems are compared using *precision* and *recall* metrics [9], which are well-known from information retrieval. For an alignment A and a reference alignment R,

$$P(A, R) = \frac{|A \cap R|}{|A|} \tag{44}$$

$$R(A, R) = \frac{|A \cap R|}{|R|} \tag{45}$$

define precision and recall respectively. These classic metrics, however, bear the problem, that correspondences either exactly match the reference or not. In an ontology alignment however, this yes/no evaluation is infeasible, as correspondences can be close to the reference or completely unacceptable. If for instance an ontology alignment is used for query expansion, *i.e.* an additional data source (ontology) is attached to retrieve additional results, correspondences of an alignment might be interpreted as pointers to those classes of the additional ontology, whose instances are to be added to the results. If the correspondence determined by an alignment system does not exactly map to the correct class, but to a sub- or superclass thereof, the result quality of the expanded query would only slightly decrease. A classical precision and recall evaluation of that correspondence would not account for this closeness but would simply count the correspondence as a miss. Motivated by this observation, a relaxed precision and recall metric for ontology alignment was introduced [5, 9] in order to

[9]`http://oaei.ontologymatching.org/2008/benchmarks/`

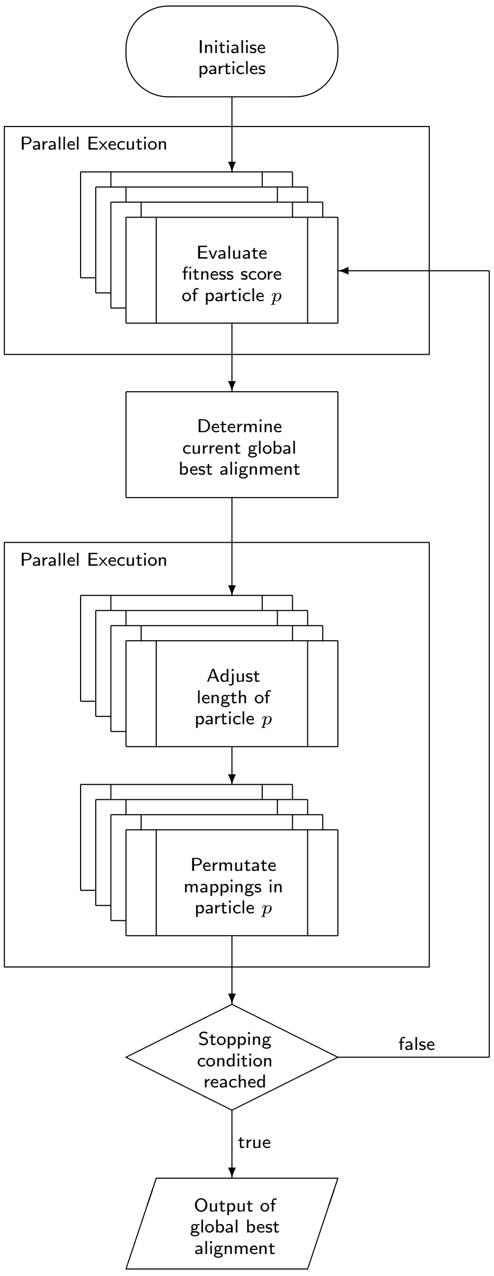

Initialise particles
Parallel Execution
Evaluate fitness score of particle p
Determine current global best alignment
Parallel Execution
Adjust length of particle p
Permutate mappings in particle p
Stopping condition reached
false
true
Output of global best alignment

Figure 2: Flowchart of the MapPSO system.

respect the closeness of correspondences to their references. Relaxed precision and recall are defined as

$$P_\omega(A, R) = \frac{\omega(A, R)}{|A|} \tag{46}$$

$$R_\omega(A, R) = \frac{\omega(A, R)}{|R|} \tag{47}$$

where ω is a proximity function, such that

$$|A \cap R| \leq \omega(A, R) \leq \min(|A|, |R|) \tag{48}$$

and thus P_ω and R_ω denote real generalisations of the traditional metrics.

7.1.1. Setup

In these tests, a swarm population of 50 particles and 200 iterations were used. The algorithm uses a 2:3 ratio for weighting number maximisation to alignment evaluation, as denoted in (8). The MapPSO algorithm is highly adjustable via its parameter file and can be tuned to perform well on specific problems, as well as to perform well in terms of precision or recall. To obtain the results presented in Table 3, a compromise parameter configuration was used as listed in Table 2.

7.1.2. Experiments

Table 3 shows the results of the experiments with the alignment benchmarks. The numbers in the table refer to the globally best alignment found by the swarm after the given number of iterations. The table shows classical, as well as relaxed (symmetric [5]) precision and recall measures w.r.t. a given reference alignment.

For **tests 101-104** MapPSO achieves precision values of around 90 % and recall values of 100 %. Test 102 with a totally irrelevant ontology, however, still determines a number of wrong correspondences.

As for **tests 201-210** results are not as positive, as the quality of the alignment decreases with the number of linguistic features to exploit. For test case 202 where all names and comments are unavailable, MapPSO performs worst in this group of tests. Regarding the relaxed metrics, results are much more stable and constant around 90 % symmetric precision and 100 % symmetric recall.

In **tests 221-247**, where the structure of the ontologies varies, the results are similar to the 10x tests. The reason is that the main focus of the current implementation of MapPSO's base matchers is on linguistic features, such as string distance and WordNet distance. In these tests the relaxed precision and recall metrics roughly coincide with the classical metrics.

The **tests 248-266** combine linguistic and structural problems. As the results show, the quality of the alignments is decreasing with the decreasing number of features available in the ontologies. However, as Table 3, and Figs. 3 and 4 demonstrate, relaxed precision and recall values are much higher and more stable throughout the tests.

For the real-life cases, **tests 301-304**, no uniform results can be derived as the algorithm's precision and recall values vary between 0 and 60 %.

Table 2: Parameter configuration as used for the OAEI 2008 campaign. Parameters are labelled as in the MapPSO parameter file. Symbols in parentheses denote the corresponding variables used throughout this article.

Parameter	Value
population (N)	50
iterations (I)	200
numberProportion (η)	0.4
propLikeLocalInc (β)	0.4
propLikeGlobalInc (γ)	0.5
kappa (κ)	0.7
sigma (σ)	0.1
changeCorrProbFactor (λ)	0.6
increaseCorrWeightStart ($\rightarrow w_{inc}$)	2.5
decreaseCorrWeightStart ($\rightarrow w_{dec}$)	1.5
increaseCorrWeightEnd ($\rightarrow w_{inc}$)	1.5
decreaseCorrWeightEnd ($\rightarrow w_{dec}$)	1.1
changeCorrWeightMethod ($\rightarrow w_{inc}, w_{dec}$)	linear
baseDistances ($\vec{h}()$)	entityNameDistance entityLabelDistance entityNameWNDistance entityLabelWNDistance hierarchyDistance classPropertyDistance propertyDomainRangeDistance
weights ($\vec{\omega}$)	0.3 0.2 0.15 0.15 0.1 0.05 0.05
aggregationFunction ($\Gamma()$)	owaOperator

Table 3: Experimental results for OAEI 2008 benchmark datasets[a].

Test	Prec.	Recall	Sym. Prec.	Sym. Recall	Test	Prec.	Recall	Sym. Prec.	Sym Recall
101	0.9	1	0.9	1	251-2	0.76	0.8	0.88	0.91
102	0	NaN	n/a	n/a	251-4	0.47	0.53	0.87	0.98
103	0.94	1	0.94	1	251-6	0.28	0.3	0.88	0.95
104	0.92	1	0.92	1	251-8	0.22	0.24	086	0.95
201	0.12	0.13	0.89	1	252	0.06	0.06	0.91	0.99
201-2	0.79	0.88	0.89	0.99	252-2	0.62	0.7	0.9	1
201-4	0.66	0.7	0.89	0.95	252-4	0.63	0.71	0.89	1
201-6	0.5	0.56	0.89	1	252-6	0.63	0.69	0.89	0.98
201-8	0.28	0.31	0.9	1	252-8	0.63	0.71	0.9	1
202	0.05	0.05	0.89	1	253	0.06	0.07	0.89	1
202-2	0.72	0.81	0.9	1	253-2	0.75	0.71	0.9	0.86
202-4	0.55	0.6	0.91	0.99	253-4	0.5	0.56	0.9	1
202-6	0.34	0.37	0.91	1	253-6	0.38	0.42	0.9	1
202-8	0.2	0.23	0.9	1	253-8	0.17	0.19	0.89	1
203	0.95	0.94	0.95	0.94	254	0	0	0	0
204	0.85	0.93	0.9	0.98	254-2	0.85	0.7	0.89	0.73
205	0.3	0.33	0.91	1	254-4	0.83	0.45	0.83	0.45
206	0.35	0.38	0.9	0.99	254-6	0.37	0.39	0.74	0.79
207	0.35	0.39	0.9	1	254-8	0.71	0.15	0.71	0.15
208	0.78	0.88	0.89	1	257	0.05	0.06	0.74	0.97
209	0.22	0.25	0.91	1	257-2	0.91	0.61	0.91	0.61
210	0.18	0.2	0.9	0.99	257-4	0.53	0.61	0.76	0.88
221	0.9	1	0.9	1	257-6	0.4	0.52	0.76	0.97
222	0.91	1	0.91	1	257-8	0.23	0.27	0.75	0.91
223	0.96	0.89	0.96	0.89	258	0.08	0.09	0.86	0.98
224	0.9	1	0.9	1	258-2	0.74	0.74	0.9	0.9
225	0.9	1	0.9	1	258-4	0.49	0.53	0.86	0.94
228	0.8	1	0.8	1	258-6	0.34	0.39	0.87	0.98
230	0.86	1	0.86	1	258-8	0.2	0.23	0.87	0.99
231	0.92	1	0.92	1	259	0.01	0.01	0.89	1
232	0.94	1	0.94	1	259-2	0.68	0.76	0.89	1
233	0.79	1	0.79	1	259-4	0.64	0.72	0.89	1
236	0.8	1	0.8	1	259-6	0.66	0.74	0.89	1
237	0.93	1	0.93	1	259-8	0.66	0.73	0.9	0.99
238	0.9	0.95	0.91	0.96	260	0.03	0.03	0.63	0.86
239	0.89	0.86	0.89	0.86	260-2	0.67	0.76	0.79	0.9
240	0.71	0.82	0.74	0.85	260-4	0.53	0.72	0.65	0.9
241	0.79	1	0.79	1	260-6	0.64	0.31	0.64	0.31
246	0.81	1	0.81	1	260-8	0.21	0.28	0.63	0.83
247	0.73	0.82	0.76	0.85	261	0.04	0.06	0.73	1
248	0.04	0.04	0.89	1	261-2	0.86	0.36	0.86	0.36
248-2	0.75	0.79	0.91	0.96	261-4	0.82	0.27	0.82	0.27
248-4	0.48	0.54	0.89	0.99	261-6	0.75	0.45	0.85	0.52
248-6	0.36	0.4	0.9	1	261-8	0.68	0.79	0.74	0.85
248-8	0.16	0.18	0.89	0.98	262	0.07	0.09	0.74	0.97
249	0.06	0.07	0.9	1	262-2	0.86	0.76	0.9	0.79
249-2	0.73	0.82	0.9	1	262-4	0.5	0.55	0.75	0.82
249-4	0.53	0.59	0.9	0.99	262-6	0.79	0.33	0.79	0.33
249-6	0.34	0.38	0.9	1	262-8	0.16	0.21	0.74	0.97
249-8	0.16	0.18	0.9	1	265	0.03	0.03	0.65	0.9
250	0.07	0.09	0.74	0.94	266	0.02	0.03	0.73	1
250-2	0.78	0.85	0.81	0.88	301	NaN	0	1	0
250-4	0.67	0.48	0.67	0.48	302	0.22	0.21	0.51	0.48
250-6	0.38	0.48	0.74	0.94	303	NaN	0	1	0
250-8	0.21	0.27	0.74	0.97	304	0.65	0.64	n/a	n/a
251	0.07	0.08	0.86	0.97					

[a]Classical precision and recall values are drawn from the official OAEI 2008 results published at `http://oaei.ontologymatching.org/2008/results/benchmarks/`; symmetric precision and recall values are computed using the `ExtGroupEval` evaluation class shipped with the Alignment API [8].

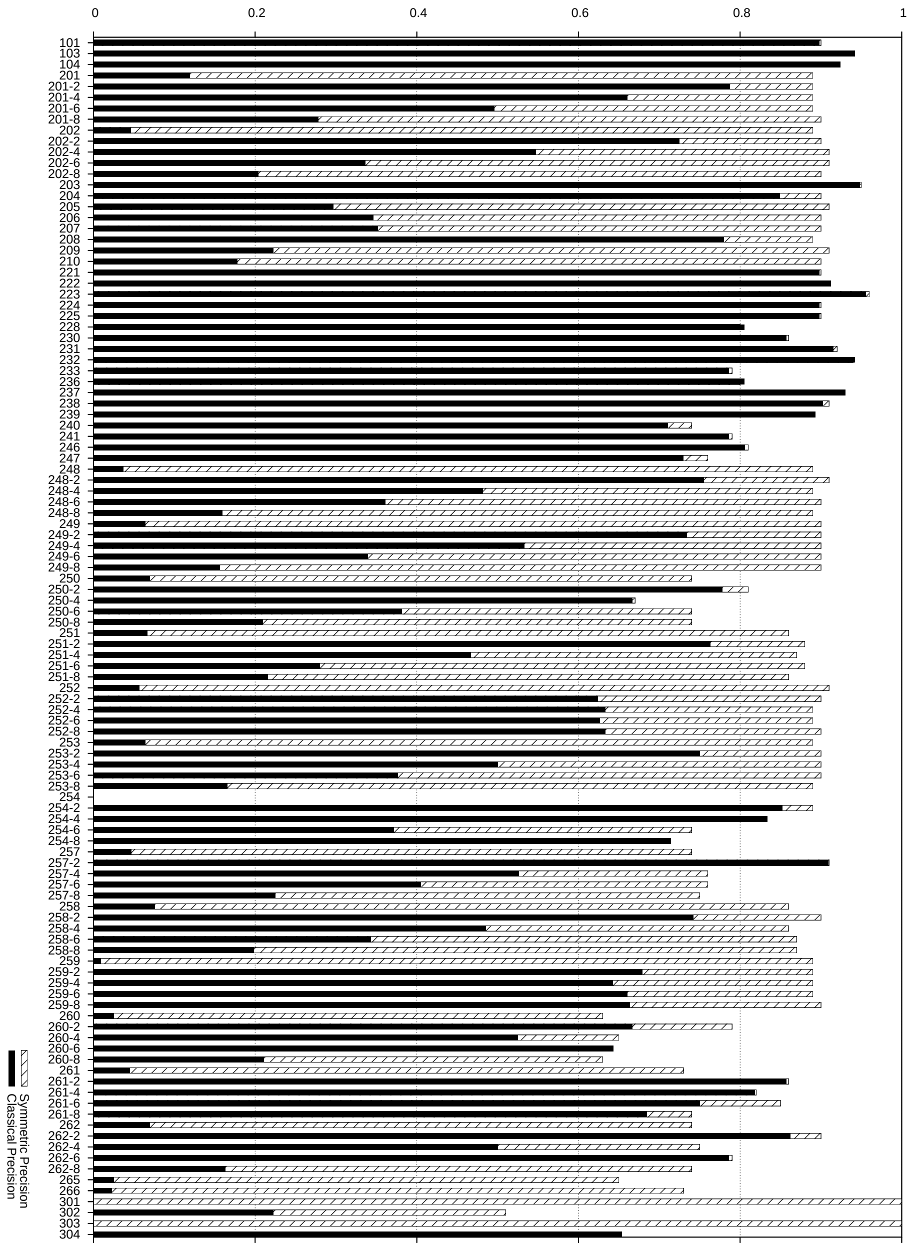


Figure 3: Classical and symmetric *precision* results of MapPSO for the OAEI 2008 benchmark datasets. Symmetric precision cannot be less than classical precision by (44), (46), and (48).

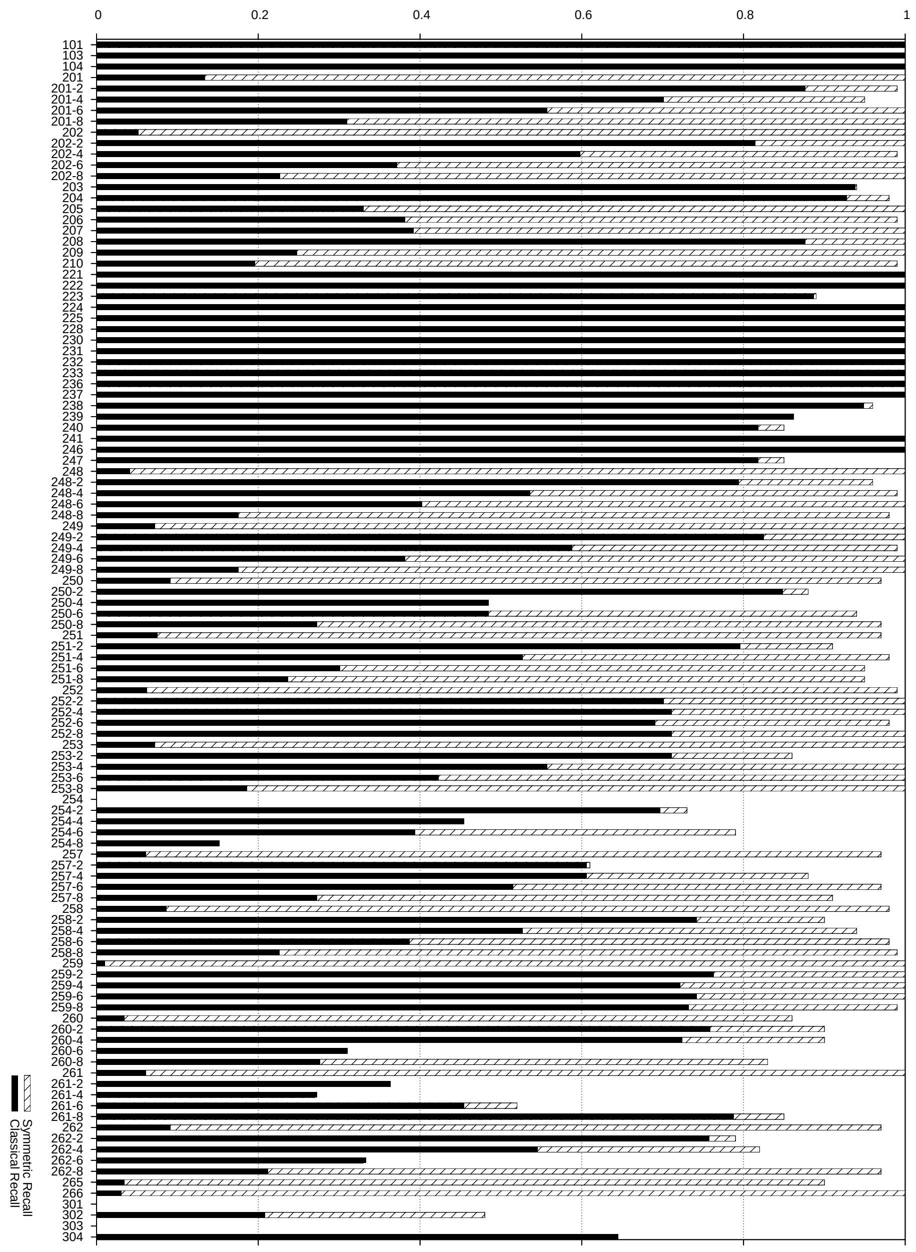


Figure 4: Classical and symmetric *recall* results of MapPSO for the OAEI 2008 benchmark datasets. Symmetric recall cannot be less than classical recall by (45), (47), and (48).

7.1.3. Discussion

Since MapPSO is a heuristic search algorithm, it is non-deterministic and therefore on a set of independent runs the quality of the results and the number of correspondences in the alignments will be subject to slight fluctuations.

For many of the test cases in the benchmark dataset the current implementation of MapPSO could provide reasonably good solutions. However, particularly alignments which are largely based on structural criteria currently impose a problem on the algorithm if the goal is to maximise classical precision and recall, and thus exactly match the reference. This behaviour is particularly reflected in test cases, where lexical and linguistic information is omitted, such as in 201 and 202. However, the relaxed precision and recall results indicate, that the alignments discovered by MapPSO are not completely different from the reference, but rather close. Obviously, in the current implementation the particle swarm just slightly fails to find the global optimum. The results, however, do show the feasibility of the approach but also the need for further development such as the addition of appropriate base matchers that more accurately evaluate the fitness scores of correspondences in the case that only structural features are available.

The submitted results were all acquired using an identical configuration file with a non-optimised and rather general set of parameters. For individual alignment problems, the quality of fitness values and thereby to some extent the efficiency of the algorithm can be improved by limiting the selection of base matchers to those that are most likely to provide useful ratings for the involved ontologies.

Figures 5 and 6 illustrate the convergence of the swarm towards the global optimum. The plots show the fitness of the globally best performing particle in each iteration and the size of the alignment it represents. Lower values thereby indicate shorter distances and therefore represent better solutions. The graphs show a representative run on test cases *101* and *102* of the OAEI 2008 benchmark track. These two special test cases represent the tasks to align an ontology with itself (*101*) or with a completely irrelevant one (*102*). The parameter configuration is the same as listed in Table 2, however, the algorithm ran 200 iterations with a population of 100 particles and a *numberProportion* of 0.2. As Fig. 5 illustrates, the algorithm converges quickly towards the global minimum on the *101* test case and reaches a stable state after 120 iterations, where no further improvements on the situation occur. The convergence on test case *102*, as illustrated in Fig. 6, is significantly slower and the algorithm does not reach a fitness value as good as on test case *101*. This can be explained by the low value for *numberProportion* which causes the fitness of the particle to mainly depend on the evaluation of its correspondences. The fitness values in this test yield bad scores since the ontologies are completely unrelated, resulting in the overall fitness of the best particle strongly correlating with the number of its correspondences. However, the convergence of the swarm also stagnates after about 120 iterations. Since in this test case, the desired alignment is empty, the convergence of the number of correspondences can be observed to approach zero

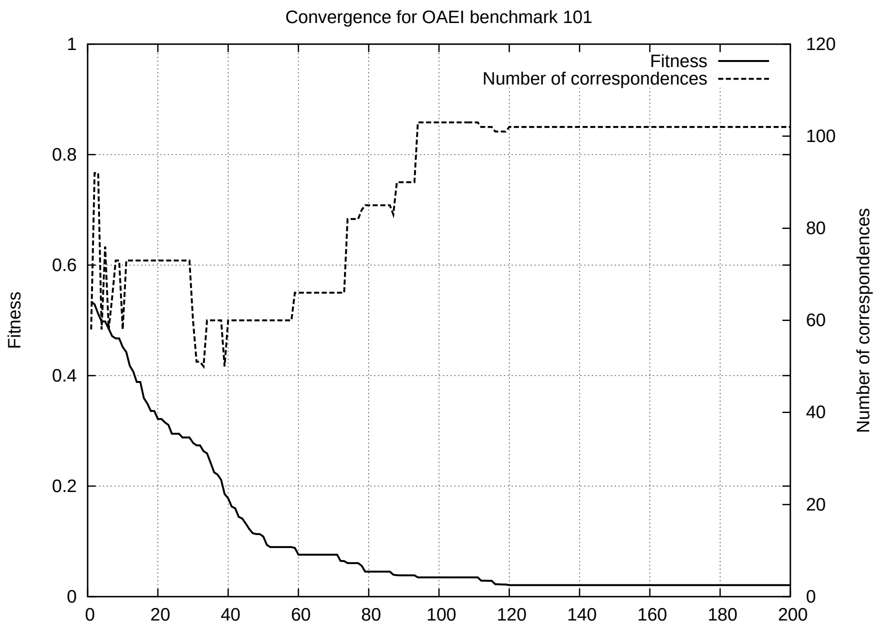


Figure 5: Convergence of the global optimum and the number of its correspondences during 200 iterations with 100 particles in the OAEI 2008 benchmark test case *101*, self-alignment.

during the run of the algorithm.

In the experiments the computational time was not taken into consideration since the prototypical implementation was not deployed to a large parallel computational system. Generally, however, parallel scalability is a strong advantage of population-based algorithms like PSO. A faster distributed parallel implementation is currently under development and will be provided in the near future. A significant speed-up on parallel architectures is expected from this implementation.

7.2. Scalability

Aligning large ontologies is a challenge, which imposes problems for most state-of-the-art mapping systems. In order to demonstrate scalability of the MapPSO approach, a proof-of-concept experiment has been conducted, where a single particle was used to process the alignment computation of two large ontologies in one iteration. To this end, two large medical ontologies have been used, namely the Foundational Model of Anatomy (FMA)[10], with 78,988

[10]`http://bioportal.bioontology.org/ontologies/4513`

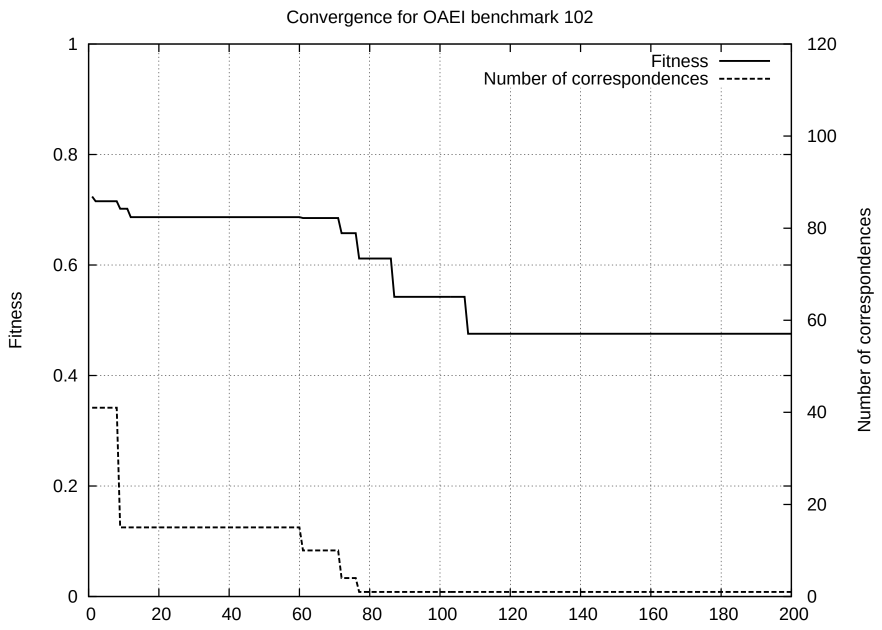


Figure 6: Convergence of the global optimum and the number of its correspondences during 200 iterations with 100 particles in the OAEI 2008 benchmark test case *102*, irrelevant ontology.

named classes, and an OWL version of the International Classification of Diseases (ICD10)[11] with 11,290 named classes[12]. The experiment solely demonstrates that MapPSO is able to process large ontologies successfully. In order to provide high quality results, more particles would need to run through more iterations, which has not been conducted yet.

8. Conclusion

In this article a novel ontology alignment algorithm based on Particle Swarm Optimisation was presented. In order to allow the application of a PSO-based method to discrete ontology alignment problems, where correspondences between ontological entities are either present or not present in an alignment, a variety of changes and adaptations to the PSO model were developed. Furthermore, to apply a heuristic optimiser to the ontology alignment problem, ontology alignment was formulated as an optimisation problem with two objectives: *(i)* identify a set of correct correspondences, and *(ii)* maximise the number of correspondences in this alignment. The DPSO algorithm by Correa *et al.* [2] has been adapted to have a population of particles search for the optimal alignment w.r.t. the two objectives. In the approach each particle represents a candidate alignment of the two ontologies. The convergence of the swarm is guided by proportional likelihood values assigned to each correspondence, based on the best alignments that the swarm and each particle individually have so far discovered. Initial experiments with a prototypical implementation showed promising results in terms of alignment quality on benchmark datasets taken from the Ontology Alignment Evaluation Initiative (OAEI) 2008. Furthermore, the possibility to process extremely large ontologies was demonstrated using two medical ontologies in a separate experiment.

The presented method is highly parallelisable, which, deployed on a highly parallel architecture, potentially allows for very complex computations of similarities in each particle without causing performance or memory issues. As opposed to many state-of-the-art alignment tools, the population-based DPSO approach does not require the computation of large similarity matrices making it more scalable on parallel architectures.

Future work on MapPSO will focus on developing and adding additional base matchers and methods to dynamically adapt the aggregating weight function. To this end, a number of additional base matchers can potentially be adapted from existing alignment systems. For instance, TBox axioms and descriptions of entities can be taken into account (in the case of ontologies of higher expressivity), as demonstrated *e.g.* in the OLA approach [10]. Concepts from the Anchor-PROMPT approach [22], where non-local contexts are considered,

[11] `http://www.dimdi.de/static/en/klassi/diagnosen/icd10/index.htm`

[12] The rational behind aligning anatomy and disease ontologies arises out of a real world use case, where details about anatomical structures are requested according to their relevance to certain diseases. This use case is also an example of the necessity to identify only partial overlaps between ontologies.

are promising candidates. Another possibility to further improve the algorithm is the tuning of input parameters, such as size of the population, number of iterations and potentially more intelligent termination criteria. Furthermore, adjustment to the β, γ, κ, and σ parameters will be made as their current values are mere estimates.

Automatic ontology alignment systems often utilise sophisticated background knowledge bases to identify alignments. Semi-automatic systems typically serve as assistants for humans, *e.g.* by working interactively. Integrating a human decision making process appears promising to find alignments for particularly difficult ontologies and could represent a viable compromise between poor results from automated systems and very labour intensive manual approaches. A solution for an interactive component for multi-objective PSO algorithms was investigated by Hettenhausen [12]. This work will be adapted to DPSO and the proposed ontology alignment algorithm in particular.

On the implementation side adaptions will be made to allow execution on static and ad-hoc distributed systems such as grids, clouds, or clusters to reveal the full potential of the parallel nature of the algorithm for large-scale problems.

Acknowledgement

The presented research was partially funded by the German Federal Ministry of Economics (BMWi) under the project "Theseus" (number 01MQ07019). We would furthermore like to extend our gratitude to Florian Berghoff for his contributions to the implementation of the method.

Authors

Jürgen Bock is a PhD candidate at the department of Information Process Engineering (IPE) at the FZI Research Center for Information Technology. In 2006 he graduated from Griffith University as Bachelor of Information Technology with Honours and in 2007 from the University of Ulm as "Diplom Informatiker". His main research interests are in the area of semantic technologies. In particular he is interested in the application of parallel computation techniques for automated processing of ontologies, such as ontology reasoning and ontology alignment.

Jan Hettenhausen is a PhD candidate at the Institute for Integrated and Intelligent Systems at Griffith University. He did his undergraduate studies at the Technical University of Hamburg and received his Master of Information and Communication Technology at Griffith University in 2006. His research interests are Multi-Objective Optimisation and Interactive Optimisation techniques with applications in bioinformatics and engineering. He is currently exploring ways to combine machine learning strategies with optimisation heuristics to facilitate the use of domain knowledge in optimisation processes.

References

[1] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications.* Cambridge University Press, New York, NY, USA, 2003.

[2] Elon S. Correa, Alex A. Freitas, and Colin G. Johnson. 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)*, pages 35–42, New York, NY, USA, 2006. ACM.

[3] Elon S. Correa, Alex A. Freitas, and Colin G. Johnson. Particle Swarm and Bayesian Networks Applied to Attribute Selection for Protein Functional Classification. In *Proceedings of the 9th Genetic and Evolutionary Computation Conference (GECCO-2007)*, pages 2651–2658, New York, NY, USA, 2007. ACM.

[4] Prashant Doshi and Christopher Thomas. Inexact Matching of Ontology Graphs Using Expectation-Maximization. In *Proceedings of the 21st National Conference on Artificial Intelligence and the 18th Innovative Applications of Artificial Intelligence Conference.* AAAI Press, July 2006.

[5] Marc Ehrig and Jérôme Euzenat. Relaxed Precision and Recall for Ontology Matching. In Benjamin Ashpole, Marc Ehrig, Jérôme Euzenat, and Heiner Stuckenschmidt, editors, *Proceedings of the K-CAP 2005 Workshop on Integrating Ontologies*, volume 156 of *CEUR Workshop Proceedings*, pages 25–32. CEUR-WS.org, October 2005.

[6] Marc Ehrig and York Sure. FOAM - Framework for Ontology Alignment and Mapping; Results of the Ontology Alignment Initiative. In Benjamin Ashpole, Marc Ehrig, Jérôme Euzenat, and Heiner Stuckenschmidt, editors, *Proceedings of the K-CAP 2005 Workshop on Integrating Ontologies*, volume 156 of *CEUR Workshop Proceedings*, pages 72–76. CEUR-WS.org, October 2005.

[7] Andries P. Engelbrecht. *Fundamentals of Computational Swarm Intelligence.* Wiley, 2005.

[8] Jérôme Euzenat. An API for Ontology Alignment. In Sheila A. McIlraith, Dimitris Plexousakis, and Frank van Harmelen, editors, *Proceedings of the 3rd International Semantic Web Conference*, volume 3298 of *LNCS*, pages 698–712, Berlin, November 2004. Springer.

[9] Jérôme Euzenat and Pavel Shvaiko. *Ontology Matching.* Springer, 2007.

[10] Jérôme Euzenat and Petko Valtchev. Similarity-based Ontology Alignment in OWL-Lite. In Ramon Lpez de Mntaras and Lorenza Saitta, editors, *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI)*,

volume 110 of *Frontiers in Artificial Intelligence and Applications*, pages 333–337, Amsterdam, The Netherlands, August 2004. IOS Press.

[11] Thomas R. Gruber. Toward Principles for the Design of Ontologies Used for Knowledge Sharing. In Nicola Guarino and Roberto Poli, editors, *Formal Ontology in Conceptual Analysis and Knowledge Representation*, Dordrecht, The Netherlands, 1993. Kluwer Academic Publishers.

[12] Jan Hettenhausen. Interactive Multi-Objective Particle Swarm Optimisation with Heatmap Visualisation based User Interface. Master's thesis, Griffith University, 2007.

[13] Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F. Patel-Schneider, and Sebastian Rudolph. OWL 2 Web Ontology Language: Primer. W3C recommendation, W3C, June 2009. `http://www.w3.org/TR/2009/WD-owl2-primer-20090611/`.

[14] Qiu Ji, Peter Haase, and Guilin Qi. Combination of Similarity Measures in Ontology Matching using the OWA Operator. In *Proceedings of the 12th International Conference on Information Processing and Management of Uncertainty in Knowledge-Base Systems (IPMU'08)*, 2008.

[15] Jason J. Jung. Reusing Ontology Mappings for Query Routing in Semantic Peer-to-Peer Environment. *Information Sciences*, In Press, Uncorrected Proof, 2010.

[16] James Kennedy and Russell C. Eberhart. Particle Swarm Optimization. *Proceedings of the IEEE International Conference on Neural Networks*, 4, 1995.

[17] James Kennedy and Russell C. Eberhart. A Discrete Binary Version of the Particle Swarm Algorithm. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, volume 5, pages 4104–4108, Washington, DC, USA, October 1997. IEEE Computer Society.

[18] Jorge Martinez-Gil, Enrique Alba, and Jose F. Aldana Montes. Optimizing Ontology Alignments by Using Genetic Algorithms. In Christophe Guéret, Pascal Hitzler, and Stefan Schlobach, editors, *Nature inspired Reasoning for the Semantic Web (NatuReS)*, volume 419 of *CEUR Workshop Proceedings*. CEUR-WS.org, October 2008.

[19] Sergey Melnik, Hector Garcia-Molina, and Erhard Rahm. Similarity Flooding: A Versatile Graph Matching Algorithm and Its Application to Schema Matching. In *Proceedings of the 18th International Conference on Data Engineering (ICDE '02)*, Washington, DC, USA, February 2002. IEEE Computer Society.

[20] Eric Miller and Frank Manola. RDF Primer. W3C recommendation, W3C, February 2004. `http://www.w3.org/TR/2004/REC-rdf-primer-20040210/`.

[21] Miklos Nagy, Maria Vargas-Vera, and Enrico Motta. DSSim Results for OAEI 2008. In Pavel Shvaiko, Jérôme Euzenat, and Fausto Giunchiglia Heiner Stuckenschmidt, editors, *Proceedings of the 3rd International Workshop on Ontology Matching (OM-2008)*, volume 431 of *CEUR Workshop Proceedings*. CEUR-WS.org, November 2008.

[22] Natalya F. Noy and Mark A. Musen. Anchor-PROMPT: Using Non-Local Context for Semantic Matching. In *Workshop on Ontologies and Information Sharing at the 17th International Joint Conference on Artificial Intelligence (IJCAI-2001)*, Seattle, WA, 2001.

[23] Eyal Oren, Christophe Guéret, and Stefan Schlobach. Anytime Query Answering in RDF through Evolutionary Algorithms. In *Proceedings of the 7th International Semantic Web Conference*, volume 5318 of *LNCS*, pages 98–113, Berlin, October 2008. Springer.

[24] Heiko Paulheim. On Applying Matching Tools to Large-scale Ontologies. In Pavel Shvaiko, Jérôme Euzenat, and Fausto Giunchiglia Heiner Stuckenschmidt, editors, *Proceedings of the 3rd International Workshop on Ontology Matching (OM-2008)*, volume 431 of *CEUR Workshop Proceedings*. CEUR-WS.org, November 2008.

[25] Biao Qin, Shan Wang, Xiaoyong Du, Qiming Chen, and Qiuyue Wang. Graph-based Query Rewriting for Knowledge Sharing between Peer Ontologies. *Information Sciences*, 178(18):3525–3542, September 2008.

[26] G. Salton, A. Wong, and C. S. Yang. A Vector Space Model for Automatic Indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[27] Y. Shi and R. Eberhart. A Modified Particle Swarm Optimizer. *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, pages 69–73, 1998.

[28] Pavel Shvaiko, Jérôme Euzenat, Fausto Giunchiglia, and Heiner Stuckenschmidt, editors. *OM-2008 Ontology Matching*, volume 431 of *CEUR Workshop Proceedings*. CEUR-WS, October 2008.

[29] Giorgos Stoilos, Giorgos Stamou, and Stefanos Kollias. A String Metric For Ontology Alignment. In Yolanda Gil, Enrico Motta, V. Richard Benjamins, and Mark A. Musen, editors, *Proceedings of the 4rd International Semantic Web Conference (ISWC)*, volume 3729 of *LNCS*, pages 624–637, Berlin, November 2005. Springer.

[30] A.R. Tawil, M. Montebello, R. Bahsoon, W.A. Gray, and N.J. Fiddian. Interschema Correspondence Establishment in a Cooperative OWL-based Multi-Information Server Grid Environment. *Information Sciences*, 178(4):1011–1031, 2008.

[31] Junli Wang, Zhijun Ding, and Changjun Jiang. GAOM: Genetic Algorithm Based Ontology Matching. In *Proceedings of the IEEE Asia-Pacific Conference on Services Computing (APSCC)*, pages 617–620, Washington, DC, USA, 2006. IEEE Computer Society.

[32] Xin-Mei Yu, Xin-Yin Xiong, and Yao-Wu Wu. A PSO-based Approach to Optimal Capacitor Placement with Harmonic Distortion Consideration. *Electric Power Systems Research*, 71(1):27–33, 2004.

[33] Guofu Zhang, Jianguo Jiang, Zhaopin Su, Meibin Qi, and Hua Fang. Searching for Overlapping Coalitions in Multiple Virtual Organizations. *Information Sciences*, In Press, Accepted Manuscript, 2010.