Randomized Metric Induction and Evolutionary Conceptual Clustering for Semantic Knowledge Bases

Nicola Fanizzi, Claudia d'Amato, and Floriana Esposito
Dipartimento di Informatica – Università degli studi di Bari
Campus Universitario, Via Orabona 4, 70125 Bari, Italy
{fanizzi|claudia.damato|esposito}@di.uniba.it

ABSTRACT

We present an evolutionary clustering method which can be applied to multi-relational knowledge bases storing resource annotations expressed in the standard languages for the Semantic Web. The method exploits an effective and language-independent semi-distance measure defined for the space of individual resources, that is based on a finite number of dimensions corresponding to a committee of discriminating features (represented by concept descriptions). A maximally discriminating group of features can be obtained with the randomized optimization methods described in the paper. The clustering algorithm represents the possible clusterings as strings of central elements (medoids, w.r.t. the given metric) of variable length. Hence, the number of clusters is not required as a parameter since the method is able to find an optimal choice by means of the evolutionary operators and of a proper fitness function. We also show how to assign each cluster with a newly constructed intensional definition in the employed concept language. An experimentation with some ontologies proves the feasibility of our method and its effectiveness in terms of clustering validity indices.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Clustering; I.5.4 [**Pattern Recognition**]: Clustering

General Terms

Algorithms, Measurement

Keywords

Conceptual clustering, unsupervised learning, metric learning, genetic programming, evolutionary algorithms, description logics, randomized optimization

1. INTRODUCTION

In the inherently distributed applications related to the Semantic Web (henceforth SW) there is an extreme need of automatizing those activities which are more burdensome for the knowledge engineer, such as ontology construction, matching and evolution. These phases can be assisted by specific supervised [5, 18, 13] or unsupervised learning methods [16, 8] crafted for knowledge bases expressed in the standard representations of the field and complying with their semantics.

In this work, we investigate on unsupervised learning for knowledge bases expressed in such standard languages. In particular, we focus on the problem of conceptual clustering of semantically annotated resources. The benefits of *conceptual clustering* [24] in the context of semantically annotated knowledge bases are manifold. Clustering annotated resources enables the definition of new emerging concepts (*concept formation*) on the grounds of the concepts defined in a knowledge base; supervised methods can exploit these clusters to induce new concept definitions or to refining existing ones (*ontology evolution*); intensionally defined groupings may speed-up the task of search and *discovery* (see [5] for an application of dissimilarity measures to retrieval); a clustering may also suggest criteria for *ranking* the retrieved resources based on the distance from the centers.

Essentially, most of the clustering methods are based on the application of similarity (or density) measures defined over a fixed set of attributes of the domain objects. Classes of objects are taken as collections that exhibit low interclass similarity (density) and high intraclass similarity (density). These methods are rarely able to take into account some form of *background knowledge* that could characterize object configurations by means of global concepts and semantic relationships. This hinders the interpretation of the outcomes of these methods which is crucial in the SW perspective which enforces sharing and reusing the produced knowledge in order to enable forms of semantic interoperability across different knowledge bases and applications.

Conceptual clustering methods can answer these requirements since they have been specifically crafted for defining groups of objects through (simple) descriptions based on selected attributes [24]. In the perspective, the expressiveness of the language adopted for describing objects and clusters (concepts) is extremely important. Related approaches, specifically designed for terminological representations (*Description Logics* [1], henceforth DLs), have recently been introduced [16, 8]. They pursue logic-based methods for attacking the problem of clustering w.r.t. some specific DL languages. The main drawback of these methods is that they are language-dependent, which prevents them to scale to the standard SW representations that are mapped on

CIKM'07, November 6–8, 2007, Lisboa, Portugal.

complex DLs. Moreover, purely logic methods can hardly handle noisy data.

These problems motivate the investigation on similarity-based clustering methods which can be more noise-tolerant and language-independent. Specifically, the extension of distance-based techniques is proposed, which can cope with the standard SW representations and profit by the benefits of a randomized search for optimal clusterings. Indeed, the method is intended for grouping similar resources w.r.t. a notion of similarity, coded in a distance measure, which fully complies with the semantics knowledge bases expressed in DLs. The individuals are gathered around cluster centers according to their distance. The choice of the best centers (and their number) is performed through an evolutionary approach [9, 17].

From a technical viewpoint, upgrading existing distance-based algorithms to work on multi-relational representations, like the concept languages used in the SW (RDF through OWL), requires similarity measures that are suitable for such representations and their semantics. A theoretical problem is posed by the *Open World Assumption* (OWA) that is generally made on the language semantics, differently from the *Closed World Assumption* (CWA) which is standard in other contexts. Moreover, as pointed out in a seminal paper on similarity measures for DLs [3], most of the existing measures focus on the similarity of atomic concepts within hierarchies or simple ontologies.

Recently, dissimilarity measures have been proposed for some specific DLs [5]. Although they turned out to be quite effective for specific inductive tasks, they were still partly based on structural criteria which makes them fail to fully grasp the underlying semantics and hardly scale to more complex ontology languages. Moreover, they have been conceived for assessing *concept* similarity, whereas, for other tasks, a notion of similarity between *individuals* is required.

Therefore, we have devised a family of dissimilarity measures for semantically annotated resources, which can overcome the aforementioned limitations [7]. Following the criterion of semantic discernibility of individuals, a family of measures is derived that is suitable for a wide range of languages since it is merely based on the discernibility of the input individuals with respect to a fixed committee of features represented by a set of concept definitions. Hence, the new measures are not absolute, they rather depend on the knowledge base they are applied to. Thus, also the choice of good feature sets deserves a preliminary optimization phase, which can be performed by means of a randomized search procedures such as *simulated annealing* or *genetic programming*, defining mutation and crossover steps through the refinement operators recently proposed in the context of ontology evolution [18, 13].

In this setting, instead of the notion of *centroid* that characterizes the distance-based algorithms descending from K-MEANS [14], originally developed for numeric or ordinal features, we recur to the notion of *medoids* [15] as central individuals in a cluster. The proposed clustering algorithm employs genetic programming as a search schema. The evolutionary problem is modeled by considering populations made up of strings of medoids with different lengths. The medoids are computed according to the semantic measure induced with the methodology mentioned above. On each generation, the strings in the current population are evolved by mutation and cross-over operators, which are also able to change the number of medoids. Thus, this algorithm is also able to suggest autonomously a promising number of clusters. Accordingly, the fitness function is based both on the optimization of a cluster cohesion index and on the penalization of lengthy medoid strings.

The remainder of the paper is organized as follows. Sect. 2 presents the basics of the target representation and the semantic similarity measure adopted with the clustering algorithm. This algorithm is presented and discussed in Sect. 3. After Sect. 4 surveying the related work, we report in Sect. 5 an experiment aimed at assessing the validity of the method on some ontologies available in the Web. Conclusions and extensions are finally examined in Sect. 6.

2. SEMANTIC DISTANCE MEASURES

2.1 Preliminaries on the Representation

In the following, we assume that resources, concepts and their relationship may be defined in terms of a generic ontology language that may be mapped to some DL language with the standard model-theoretic semantics (see the DLs handbook [1] for a thorough reference). As mentioned in the previous section, one of the advantages of our method is that it does not depend on a specific language for semantic annotations.

In the intended framework setting, a *knowledge base* $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ contains a *TBox* $\mathcal{T}$ and an *ABox* $\mathcal{A}$. $\mathcal{T}$ is a set of concept definitions. The complexity of such definitions depends on the specific DL language constructors. $\mathcal{A}$ contains *assertions* (ground facts) on *individuals* (domain objects) concerning the current world state, namely:

class-membership $C(a)$, a is an instance of concept C

relations $R(a, b)$, a is R-related to b

The set of the individuals referenced in the assertions ABox $\mathcal{A}$ will be denoted with $\mathsf{Ind}(\mathcal{A})$. The *unique names assumption* can be made on the ABox individuals[1] therein.

As regards the required inference services, like all other instance-based methods, the measure proposed in this section requires performing *instance-checking*, which amounts to determining whether an individual, say a, belongs to a concept extension, i.e. whether $C(a)$ holds for a certain concept C. In the simplest cases (primitive concepts) this requires simple lookups, yet for defined concepts the reasoner may need to perform a number of inferences. Besides, differently from the standard DB settings, due to the OWA, the reasoner might be unable to provide a definite answer. Hence one has to cope with this form of uncertainty.

2.2 Semantic Similarity between Individuals

For our purposes, we need a function for measuring the similarity of individuals. It can be observed that individuals do not have a syntactic structure that can be compared. This has led to lifting them to the concept description level before comparing them (recurring to the approximation of the *most specific concept* of an individual w.r.t. the ABox) [5].

For clustering procedures, such as the one specified in Sect. 3, we have developed a new measure with a definition

[1]Each individual can be assumed to be identified by its own URI, however this is not bound to be a one-to-one mapping.

that totally depends on semantic aspects of the individuals in the knowledge base. On a semantic level, similar individuals should behave similarly with respect to the same concepts. We have introduced a novel measure for assessing the similarity of individuals in a knowledge base, which is based on the idea of comparing their semantics along a number of dimensions represented by a committee of concept descriptions.

Following some techniques for distance induction borrowed from ILP [23], we propose the definition of totally semantic distance measures for individuals in the context of a knowledge base which is also able to cope with the OWA.

The rationale of the new measure is to compare individuals on the grounds of their behavior w.r.t. a given set of features, that is a collection of concept descriptions, say $\mathsf{F} = \{F_1, F_2, \ldots, F_m\}$, which stands as a group of discriminating *features* expressed in the considered DL language.

A family of dissimilarity measures for individuals inspired to the Minkowski's distances (L_p) can be defined as follows [7]:

DEFINITION 2.1 (FAMILY OF DISSIMILARITY MEASURES). *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a knowledge base. Given set of concept descriptions $\mathsf{F} = \{F_1, F_2, \ldots, F_m\}$, a family of functions $\{d_p^{\mathsf{F}}\}_{p \in \mathbb{N}}$ with*

$$d_p^{\mathsf{F}} : \mathsf{Ind}(\mathcal{A}) \times \mathsf{Ind}(\mathcal{A}) \mapsto [0, 1]$$

is defined as follows:
$\forall a, b \in \mathsf{Ind}(\mathcal{A})$

$$d_p^{\mathsf{F}}(a, b) := \frac{L_p(\pi(a), \pi(b))}{m} = \frac{1}{m} \left(\sum_{i=1}^{m} \mid \pi_i(a) - \pi_i(b) \mid^p \right)^{\frac{1}{p}}$$

where $p > 0$ and the i-th projection function *π_i of vector π, $i \in \{1, \ldots, m\}$, is defined by:*
$\forall a \in \mathsf{Ind}(\mathcal{A})$

$$\pi_i(a) = \begin{cases} 1 & \mathcal{K} \models F_i(a) \\ 0 & \mathcal{K} \models \neg F_i(a) \\ 1/2 & \textit{otherwise} \end{cases}$$

The superscript F will be omitted when the set of features is fixed.

The case of $\pi_i(a) = 1/2$ corresponds to the case when a reasoner cannot give the truth value for a certain membership query. This is due to the OWA normally made in this context.

2.3 Discussion

Compared to other proposed distance (or dissimilarity) measures for individuals [5], the presented function does not depend on the constructors of a specific language, rather it requires only the instance-checking service used for deciding whether an individual is asserted in the knowledge base to belong to a concept extension or, alternatively, if this could be derived as a logical consequence.

It is easy to see that the functions $\{d_p^{\mathsf{F}}\}_{p \in \mathbb{N}}$ are dissimilarity measures. Even more so, the standard properties for semi-distances can be proven [7]:

PROPOSITION 2.1 (SEMI-DISTANCE). *For a fixed feature set and $p > 0$, given any three instances $a, b, c \in \mathsf{Ind}(\mathcal{A})$. it holds that:*

1. $d_p^{\mathsf{F}}(a, b) \geq 0$
2. $d_p^{\mathsf{F}}(a, b) = d_p^{\mathsf{F}}(b, a)$
3. $d_p^{\mathsf{F}}(a, c) \leq d_p^{\mathsf{F}}(a, b) + d_p^{\mathsf{F}}(b, c)$

The functions are not metrics because it cannot be proved that if $d_p^{\mathsf{F}}(a, b) = 0$ then $a = b$ (whereas the opposite implication easily holds). This is the case of *indiscernible* individuals with respect to the given set of features F. If the property were strictly required, a distance could be derived either by considering equivalence classes [25] or, if the *unique names assumption* were made, by introducing equality as a new meta-feature π_0.

Note that the projection functions for the individuals in the knowledge base can be computed in advance thus determining a speed-up in the actual computation of the measure. This is very important for the measure integration in algorithms which massively use this distance, such as all instance-based methods.

2.4 Optimizing the Feature Set

The underlying idea in the measure definition is that similar individuals should exhibit the same behavior w.r.t. the concepts in F. Here, we make the assumption that the feature-set F represents a sufficient number of (possibly redundant) features that are able to discriminate really different individuals.

Preliminary experiments, where the measure has been exploited for instance-based classification (*Nearest Neighbor* algorithm) and similarity search [25], demonstrated the effectiveness of the measure using the very set of both primitive and defined concepts found in the knowledge bases.

However, the choice of the concepts to be included in the committee F is crucial and may be the object of a preliminary learning problem to be solved (*feature selection for metric learning*).

Various optimizations of the feature set can be foreseen as concerns its definition. Among the possible sets of features we will prefer those that are able to discriminate the individuals in the ABox:

DEFINITION 2.2 (GOOD FEATURE SET). *Let F be a set of concept descriptions $\mathsf{F} = \{F_1, F_2, \ldots, F_m\}$. F is a* good feature set *for the knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ iff $\forall a, b \in \mathsf{Ind}(\mathcal{A}), a \neq b, \exists i \in \{1, \ldots, m\} : \pi_i(a) \neq \pi_i(b)$.*

Then, when the previously defined function is parameterized on a good feature set, it has the properties of a metric function.

Namely, since the function is strictly dependent on the committee of features F, two immediate heuristics arise:

- the *number* of concepts of the committee,
- their discriminating power in terms of a *discernibility factor*.

Finding optimal sets of discriminating features, should profit also by their composition employing the specific constructors made available by the representation language of choice.

These objectives can be accomplished by means of randomized optimization techniques, especially when knowledge bases with large sets of individuals are available. Namely, part of the entire data can be drawn in order to learn optimal F sets, in advance with respect to the successive usage for all other purposes.

```
FeatureSet GPOptimization(K, maxGenerations, fitnessThr)
input:
    K: current knowledge base
    maxGenerations: maximal number of generations
    fitnessThr: minimal required fitness threshold
output:
    FeatureSet: set of concept descriptions
static:
    currentFSs, formerFSs:
        array of current/previous feature sets
    currentBestFitness, formerBestFitness = 0:
        array of current/previous best fitness values
    offsprings: array of generated feature sets
    fitnessImproved: improvement flag
    generationNo = 0: number of current generation
begin
currentFSs = makeInitialFS(K,INIT_CARD)
formerFSs = currentFSs
repeat
    fitnessImproved = false
    currentBestFitness = bestFitness(currentFSs)
    while (currentBestFitness < fitnessThr) and
          (generationNo < maxGenerations)
        begin
        offsprings = generateOffsprings(currentFSs)
        currentFSs = selectFromPopulation(offsprings)
        currentBestFitness = bestFitness(currentFSs)
        ++generationNo
        end
    if  (currentBestFitness > formerBestFitness) and
        (currentBestFitness < fitnessThr) then
        begin
        formerFSs = currentFSs
        formerBestFitness = currentBestFitness
        currentFSs = extendFS(currentFSs)
        end
    else
        fitnessImproved = true
    end
until not fitnessImproved
return selectBest(formerFSs)
end
```

Figure 1: Feature set optimization algorithm based on genetic programming.

2.4.1 Optimization through Genetic Programming

A specific optimization algorithm founded in *genetic programming* has been devised to find optimal choices of discriminating concept committees. The resulting algorithm is depicted in Fig. 1. Essentially it searches the space of all possible feature committees starting from an initial guess (determined by the call to the makeInitialFS() procedure) based on the concepts (both primitive and defined) currently referenced in the knowledge base $\mathcal{K}$, starting with a committee of a given cardinality (INIT_CARD). This initial cardinality may be determined as a function of $\lceil \log_3(N) \rceil$, where $N = |\mathsf{Ind}(\mathcal{A})|$, as each feature projection can categorize the individuals in three sets.

The outer loop gradually augments the cardinality of the candidate committees. It is repeated until the threshold fitness is reached or the algorithm detects some fixpoint: employing larger feature committees would not yield a better feature set with respect to the best fitness recorded in the previous iteration (with fewer features). Otherwise, the extendFS() procedure extends the current for the next generations by including a newly generated random concept.

The inner while-loop is repeated for a number of generations until a stop criterion is met, based on the maximal number of generations maxGenerations or, alternatively, when a minimal fitness threshold fitnessThr is crossed by some feature set in the population, which can be returned.

As regards the bestFitness() routine, it computes the best fitness of the feature sets in the input vector. Fitness can be determined as the *discernibility factor* yielded by the feature set, as computed on the whole set of individuals or on a smaller sample. For instance, given the fixed set of individuals $IS \subseteq \mathsf{Ind}(\mathcal{A})$ the fitness function may be:

$$\text{DISCERNIBILITY}(\mathsf{F}) := \nu \sum_{(a,b)\in IS^2} \sum_{i=1}^{|\mathsf{F}|} \mid \pi_i(a) - \pi_i(b) \mid$$

where ν is a normalizing factor that can depend on the overall number of couples involved.

As concerns finding candidate sets of concepts to replace the current committee (the generateOffsprings() routine), the function was implemented by recurring to some transformations of the current best feature sets:

- choose $\mathsf{F} \in$ currentFSs;
- randomly select $F_i \in \mathsf{F}$;
  - replace F_i with $F_i' \in$ randomMutation(F_i) randomly generated, or
  - replace F_i with one of its refinements $F_i' \in \text{REF}(F_i)$

The possible refinements of concept description are language-specific. E.g. for the case of $\mathcal{ALC}$ logic, refinement operators have been proposed in [18, 13].

This is iterated till a suitable number of offsprings is generated. Then these offspring feature sets are evaluated and the best ones are included in the new version of the currentFSs array; the best fitness value for these feature sets is also computed.

When the while-loop is over the current best fitness is compared with the best one computed for the former feature set length; if an improvement is detected then the outer repeat-loop is continued, otherwise (one of) the former best feature set(s) is selected and returned as the result of the algorithm.

2.4.2 Optimization through Simulated Annealing

The randomized optimization algorithm based on genetic programming just described may suffer from being possibly caught in plateaux or local minima if a limited number of generations are explored before checking for an improvement. This is likely due to the extent of the search space, which, in turn, depends on the language of choice. Moreover. maintaining a single best genome for the next generation may slow down the search process.

To prevent such cases, different randomized search procedures which aim at global optimization should be adopted. Hence, a method based on *simulated annealing* [4] has also been proposed [7], whose algorithm is reported in Fig. 2.

The algorithm searches the space of feature sets starting from an initial guess (determined by makeInitialFS($\mathcal{K}$)) based on the concepts (both primitive and defined) currently referenced in the knowledge base, which can be freely combined to form new descriptions.

```
FeatureSet SAOptimization(𝒦, ΔT)
input:
        𝒦: knowledge base
        ΔT(): cooling function
output:
        FeatureSet: set of concept descriptions
static:
        currentFS: current Feature Set
        nextFS: new Feature Set
        time: time controlling variable
        ΔE: energy increment
        temp: temperature (probability of replacement)
begin
currentFS = makeInitialFS(𝒦)
for time = 1 to ∞ do
        temp = temp − ΔT(time)
        if (temp == 0)
            return currentFS
        nextFS = randomSuccessor(currentFS,𝒦)
        ΔE = fitness(nextFS) − fitness(currentFS)
        if (ΔE > 0)
            // replacement
            currentFS = nextFS
        else
            // conditional replacement with given probability
            currentFS = replace(nextFS, e^(ΔE/temp))
end
```

Figure 2: Feature Set optimization procedure based on simulated annealing.

The loop controlling the search is repeated for a number of times that depends on the temperature temp controlled by the cooing function $\Delta T()$ which gradually decays to 0, when the current feature committee can be returned. In this cycle, the current feature set is iteratively refined calling procedure randomSuccessor() which makes a step in the space by refining the current set. Then the fitness of the new feature set is compared to that of the current one determining the increment of energy ΔE. If this is positive then the candidate committee replaces the current one. Otherwise it will (less likely) be replaced with a probability that depends on ΔE and on the current temperature.

The energy increase ΔE is determined by the fitness() function applied to the new and current feature sets, which can be computed as the average *discernibility* factor, defined as above.

As concerns finding candidates to replace the current committee, randomSuccessor() can be implemented by recurring to simple transformations of the feature set:

- add (resp. removing) a concept C:
  $\mathsf{nextFS} \leftarrow \mathsf{currentFS} \cup \{C\}$
  (resp. $\mathsf{nextFS} \leftarrow \mathsf{currentFS} \setminus \{C\}$)
- randomly choose one of the current concepts from currentFS, say C;
  replace it with one of its refinements $C' \in \text{REF}(C)$

Note that these transformation may change the cardinality of the current feature set. As mentioned before, refining concept descriptions is language-dependent. Complete operators are to be preferred to ensure exploring the whole search space.

Given a suitable cooling schedule, the algorithm is known to find an optimal solution. To control the complexity of the process alternate schedules may be preferred that guarantee the construction of suboptimal solutions in polynomial time [4].

3. EVOLUTIONARY CONCEPTUAL CLUSTERING PROCEDURE

Many similarity-based clustering algorithms (see [14]) can be applied to semantically annotated resources stored in a knowledge base, exploiting the measures discussed in the previous section.

We focussed on the techniques based on evolutionary methods which are able to determine also an optimal number of clusters, instead of requiring it as a parameter (although the algorithm can be easily be modified to exploit this information that greatly reduces the search-space).

Conceptual clustering requires also to provide a definition for the detected groups, which may be the basis for the formation of new concepts inductively elicited from he knowledge base. Hence, the conceptual clustering procedure consists of two phases: one that detects the clusters in the data and the other that finds an intensional definition for the groups of individuals detected in the former phase.

3.1 The Evolutionary Clustering Algorithm

The first clustering phase implements a genetic programming learning scheme, where the designed representation for the competing genomes is made up of strings (lists) of individuals of different lengths, with each gene standing as prototypical for a cluster.

Specifically, each cluster will be represented by its prototype recurring to the notion of *medoid* [15, 14] on a categorical feature-space w.r.t. the distance measure previously defined. Namely, the medoid of a group of individuals is the individual that has the minimal distance w.r.t. the others. Formally. in this setting:

Definition 3.1 (medoid). *Given a cluster of individuals* $C = \{a_1, a_2, \ldots, a_n\} \subseteq \mathsf{Ind}(\mathcal{A})$*, the* medoid *of the cluster is defined:*

$$\text{medoid}(C) := \operatorname*{argmin}_{a \in C} \sum_{j=1}^{n} d(a, a_j)$$

In the proposed evolutionary algorithm, the population will be made up of genomes represented by a list of medoids $G = \{m_1, \ldots, m_k\}$ of variable lengths. The algorithm performs a search in the space of possible clusterings of the individuals, optimizing a fitness measure that maximizes the discernibility of the individuals of the different clusters (inter-cluster separation) and the intra-cluster similarity measured in terms of the d_p^{F} pseudo-metric.

On each generation those strings that are considered as best w.r.t. a fitness function are selected for passing to the next generation. Note that the algorithm does not prescribe a fixed length of the genomes (as, for instance in k-means and its extensions [14]), hence it searches a larger space aiming at determining an optimal number of clusters for the data at hand.

Fig. 3 reports a sketch of the algorithm, named ECM, *Evolutionary Clustering around Medoids*. After the call to the initialize() function returning (to currentPopulation) a randomly generated initial population of popLength medoid strings, it essentially consists of the typical generation loop

```
medoidVector ECM(maxGenerations)
input:
    maxGenerations: max number of iterations;
output:
    medoidVector: list of medoids
static:
    offsprings: vector of generated offsprings
    fitnessVector: ordered vector of fitness values
    generationNo: generation number
begin
INITIALIZE(currentPopulation,popLength)
generationNo = 0
while (generationNo < maxGenerations)
    begin
    offsprings = GENERATEOFFSPRINGS(currentPopulation)
    fitnessVector = COMPUTEFITNESS(offsprings)
    currentPopulation = SELECT(offsprings,fitnessVector)
    ++generationNo
    end
return currentPopulation[0] // fittest genome
end
```

Figure 3: ECM: the EVOLUTIONARY CLUSTERING AROUND MEDOIDS algorithm.

of genetic programming where a new population is computed and then evaluated for deciding on the best genomes to be selected for survival to the next generation.

On each iteration new offsprings of current best clusterings in currentPopulation are computed. This is performed by suitable genetic operators explained in the following. The fitnessVector recording the quality of the various offsprings (i.e. clusterings) is then updated, and then the best offsprings are selected for the next generation.

The fitness of a single genome $G = \{m_1, \ldots, m_k\}$ is computed by distributing all individuals among the clusters ideally formed around the medoids in that genome. For each medoid m_i $(i = 1, \ldots, k)$, let C_i be such a cluster. Then, the fitness is computed by the function:

$$\text{FITNESS}(G) = \left(\lambda(k) \sum_{i=1}^{k} \sum_{x \in C_i} d_p(x, m_i) \right)^{-1}$$

The factor $\lambda(k)$ is introduced in order to penalize those clusterings made up of too many clusters that could enforce the minimization in this way (e.g. by proliferating singletons). A suggested value may be $\lambda(k) = \sqrt{k+1}$ which was used in the experiments (see Sect. 5).

The loop condition is controlled by the maximal number of generation (the maxGenerations parameter) ensuring that eventually it may end even with a suboptimal solution to the problem. Besides other parameters can be introduced for controlling the loop based on the best fitness value obtained so far or on the gap between the fitness of best and of the worst selected genomes in currentPopulation. Eventually, the best genome of the vector (supposed to be sorted by fitness in descending order) is returned.

It remains to specify the nature of the GENERATEOFFSPRINGS procedure and the number of such offsprings, which may as well be another parameter of the ECM algorithm. Three mutation and one crossover operators are implemented:

DELETION(G) drop a randomly selected medoid:
$G := G \setminus \{m\}, m \in G$

INSERTION(G) select $m \in \mathsf{Ind}(\mathcal{A}) \setminus G$ that is added to G:
$G := G \cup \{m\}$

REPLACEMENTWITHNEIGHBOR(G) randomly select $m \in G$ and replace it with $m' \in \mathsf{Ind}(\mathcal{A}) \setminus G$ such that $\forall m'' \in \mathsf{Ind}(\mathcal{A}) \setminus G \; d(m, m') \leq d(m, m'')$:
$G' := (G \setminus \{m\}) \cup \{m'\}$

CROSSOVER(G_A, G_B) select subsets $S_A \subset G_A$ and $S_B \subset G_B$ and exchange them between the genomes:
$G_A := (G_A \setminus S_A) \cup S_B$ and $G_B := (G_B \setminus S_B) \cup S_A$

A (10+60) selection strategy has been implemented, with the numbers indicating, resp., the number of parents selected for survival and the number of their offsprings generated employing the mutation operators presented above.

The representation of centers by means of medoids has two advantages. First, it presents no limitations on attributes types, and, second, the choice of medoids is dictated by the location of a predominant fraction of points inside a cluster and, therefore, it is less sensitive to the presence of outliers. In K-MEANS case a cluster is represented by its centroid, which is a mean (usually weighted average) of points within a cluster. This works conveniently only with numerical attributes and can be negatively affected even by a single outlier.

An algorithm based on medoids has several favorable properties. Since it performs clustering with respect to any specified metric, it allows a flexible definition of similarity. Many clustering algorithms do not allow for a flexible definition of similarity, but allow only Euclidean distance in current implementations. In addition, medoids are robust representations of the cluster centers that are less sensitive to outliers than other cluster profiles, such as the cluster means of K-MEANS. This robustness is particularly important in the common context that many elements do not belong exactly to any cluster, which may be the case of the membership in DL knowledge bases, which may be not ascertained given the OWA.

3.2 The Supervised Learning Phase

The second phase is more language dependent. The various cluster can be considered as training examples for a supervised algorithm aimed at finding an intensional DL definition for one cluster against the counterexamples, represented by individuals in different clusters [16, 8].

Each cluster may be labeled with an intensional concept definition which characterizes the individuals in the given cluster while discriminating those in other clusters [16, 8]. Labeling clusters with concepts can be regarded as a number of supervised learning problems in the specific multi-relational representation targeted in our setting [13]. As such it deserves specific solutions that are suitable for the DL languages employed.

A straightforward solution may be found, for DLs that allow for the computation of (an approximation of) the *most specific concept* (msc) and *least common subsumer* (lcs) [1] (such as $\mathcal{ALC}$). The first operator, given the current knowledge base and an individual, provides (an approximation of) the most specific concept that has the individual as one of its instances. This would allow for lifting individuals to the concept level. The second operator computes minimal generalizations of the input concept descriptions.

Given these premises, the learning process can be described through the following steps:

let C_j be a cluster of individuals

- **for each** individual $a_i \in C_j$
  do compute $M_i := \mathsf{msc}(a_i)$ w.r.t. $\mathcal{A}$;
- **let** $\mathsf{mscs}_j := \{M_i \mid a_i \in C_j\}$;
- **return** $\mathsf{lcs}(\mathsf{mscs}_j)$

As an alternative, other algorithms for learning concept descriptions expressed in DLs may be employed [18, 13]. Indeed, concept formation can be cast as a supervised learning problem: once the two clusters at a certain level have been found, where the members of a cluster are considered as positive examples and the members of the dual cluster as negative ones. Then any concept learning method which can deal with this representation (and semantics) may be utilized for this new task.

4. RELATED WORK

The unsupervised learning procedure presented in this paper is mainly based on two factors: the semantic dissimilarity measure and the clustering method. To the best of our knowledge in the literature there are very few examples of similar clustering algorithms working on complex representations that are suitable for knowledge bases of semantically annotated resources. Thus, in this section, we briefly discuss sources of inspiration for our procedure and some related approaches.

4.1 Relational Similarity Measures

As previously mentioned, various attempts to define semantic similarity (or dissimilarity) measures for concept languages have been made, yet they have still a limited applicability to simple languages [3] or they are not completely semantic depending also on the structure of the descriptions [5]. Very few works deal with the comparison of individuals rather than concepts.

In the context of clausal logics, a metric was defined [21] for the Herbrand interpretations of logic clauses as induced from a distance defined on the space of ground atoms. This kind of measures may be employed to assess similarity in *deductive databases*. Although it represents a form of fully semantic measure, different assumptions are made with respect to those which are standard for knowledgeable bases in the SW perspective. Therefore the transposition to the context of interest is not straightforward.

Our measure is mainly based on Minkowski's measures [25] and on a method for distance induction developed by Sebag [23] in the context of *machine learning*, where *metric learning* is developing as an important subfield. In this work it is shown that the induced measure could be accurate when employed for classification tasks even though set of features to be used were not the optimal ones (or they were redundant). Indeed, differently from our unsupervised learning approach, the original method learns different versions of the same target concept, which are then employed in a voting procedure similar to the Nearest Neighbor approach for determining the classification of instances.

A source of inspiration was also *rough sets* theory [22] which aims at the formal definition of vague sets by means of their approximations determined by an indiscernibility relationship. Hopefully, these methods developed in this context will help solving the open points of our framework (see Sect. 6) and suggest new ways to treat uncertainty.

4.2 Clustering Procedures

Our algorithm adapts to the specific representations devised for the SW context a combination of evolutionary clustering and the distance-based approaches (see [14]). Specifically, in the methods derived from K-MEANS and K-MEDOIDS each cluster is represented by one of its points.

Early versions of this approach are represented by the algorithms PAM, CLARA [15], and CLARANS [20]. They implement iterative optimization methods that essentially cyclically relocate points between perspective clusters and recompute potential medoids. The leading principle for the process is the effect on an objective function. The whole dataset is assigned to resulting medoids, the objective function is computed, and the best system of medoids is retained. In CLARANS a graph is considered whose nodes are sets of k medoids and an edge connects two nodes if they differ by one medoid. While CLARA compares very few neighbors (a fixed small sample), CLARANS uses random search to generate neighbors by starting with an arbitrary node and randomly checking maxneighbor neighbors. If a neighbor represents a better partition, the process continues with this new node. Otherwise a local minimum is found, and the algorithm restarts until a certain number of local minima is found. The best node (i.e. a set of medoids) is returned for the formation of a resulting partition. Ester et al. [6] extended CLARANS to deal with very large spatial databases.

Our algorithm may be considered an extension of evolutionary clustering methods [11] which are also capable to determine a good estimate of the number of clusters [9]. Besides, we adopted the idea of representing clusterings (genomes) as strings of cluster centers [17] transposed to the case of medoids for the categorical search spaces of interest.

Other related recent approaches are represented by the UNC algorithm and its extension to the hierarchical clustering case H-UNC [19]. Essentially, UNC solves a multimodal function optimization problem seeking dense areas in the feature space. It is also able to determine their number. The algorithm is also demonstrated to be noise-tolerant and robust w.r.t. the presence of outliers. However, the applicability is limited to simpler representations w.r.t. those considered in this paper.

Further comparable clustering methods are those based on an *indiscernibility relationship* [12]. While in our method this idea is embedded in the semi-distance measure (and the choice of the committee of concepts), these algorithms are based on an iterative refinement of an equivalence relationship which eventually induces clusters as equivalence classes.

As mentioned in the introduction, the classic approaches to conceptual clustering [24] in complex (multi-relational) spaces are based on structure and logics. Kietz & Morik proposed a method for efficient construction of knowledge bases for the BACK representation language [16]. This method exploits the assertions concerning the roles available in the knowledge base, in order to assess, in the corresponding relationship, those subgroups of the domain and ranges which may be inductively deemed as disjoint. In the successive phase, supervised learning methods are used on the discovered disjoint subgroups to construct new concepts that account for them. A similar approach is followed in [8], where the supervised phase is performed as an iterative refinement step, exploiting suitable refinement operators for a different DL, namely $\mathcal{ALC}$.

5. EVALUATION

The feasibility of the clustering algorithm has been evaluated with an experimentation on knowledge bases selected from standard repositories. Note that for testing our algorithm we preferred using populated ontologies (which may be more difficult to find) rather than randomly generating assertions for artificial individuals, which might have biased the procedure.

5.1 Experimental Setup

A number of different populated knowledge bases represented in OWL were selected from various sources[2], namely: FSM, SURFACEWATERMODEL, TRANSPORTATION, NEWTESTAMENTNAMES, and FINANCIAL. Table 1 summarizes important details concerning the ontologies employed in the experimentation. Of course, the number of individuals gives only a partial indication of the number of assertions (RDF triples) concerning them which affects both the complexity of reasoning and distance assessment.

For each populated knowledge base, the experiments have been repeated for ten times. In the computation of the distances between individuals (the most time-consuming operation) all concepts in the knowledge base have been used for the committee of features, thus guaranteeing meaningful measures with high redundancy. The PELLET reasoner[3] (ver. 1.4) was employed to perform the inferences (instance-checking) that were necessary to compute the projections.

The experimentation consisted of 10 runs of the algorithm per knowledge base. The indexes which were chosen for the experimental evaluation were the following: the *generalized* R-Squared (modRS), the *generalized* Dunn's index, the average Silhouette index, and the number of clusters obtained. In the following explanation of these quality measures, we will consider a generic partition $P = \{C_1, \ldots, C_k\}$ of n individuals in k clusters.

The R-Squared index [10] is a measure of cluster separation, ranging in [0,1]. Instead of the cluster means, we generalize the measure by computing it w.r.t. their medoids, namely:

$$RS(P) := \frac{SS_b(P)}{SS_b(P) + SS_w(P)}$$

where SS_b is the *between clusters Sum of Squares* defined as follows:

$$SS_b(P) := \sum_{i=1}^{k} d(\overline{m}, m_i)^2$$

where $\overline{m}$ is the medoid of the whole dataset and SS_t is the *within cluster Sum of Squares* that is defined:

$$SS_w(P) := \sum_{i=1}^{k} \sum_{a \in C_i} d(a, m_i)^2$$

The generalized Dunn's index is a measure of both compactness (within clusters) and separation (between clusters). The original measure is defined for numerical feature vectors in terms of centroids and it is known to suffer from the presence of outliers. To overcome these limitations, we adopt a generalization of Dunn's index [2] that is modified to deal with medoids. The new index can be defined:

$$V_{GD}(P) := \min_{1 \le i \le k} \left\{ \min_{\substack{1 \le j \le k \\ i \neq j}} \left\{ \frac{\delta_p(C_i, C_j)}{\max_{1 \le h \le k} \{\Delta_p(C_h)\}} \right\} \right\}$$

where δ_p is the Hausdorff distance for clusters derived[4] from d_p, while the cluster diameter measure Δ_p is defined:

$$\Delta_p(C_h) := \frac{2}{|C_h|} \sum_{c \in C_h} d_p(c, m_h)$$

which is more noise-tolerant w.r.t. the original measure. Of course, this measure, ranging in $[0, +\infty[$, has to be maximized.

Conversely, the average Silhouette index [15] is a measure ranging in the interval [-1,1], thus suggesting an absolute best value for the validity of a clustering. For each individual x_i, $i \in \{1, \ldots, n\}$, the average distance to other individuals within the same cluster C_j, $j \in \{1, \ldots, k\}$, is computed:

$$a_i := \frac{1}{|C_j|} \sum_{x \in C_j} d_p(a_i, x)$$

Then the average distance to the individuals in other clusters is also computed:

$$b_i := \frac{1}{|C_j|} \sum_{x \in C_h}^{h \neq j} d_p(a_i, x)$$

Hence, the Silhouette value for the considered individual is obtained as follows:

$$s_i := \frac{(b_i - a_i)}{\max(a_i, b_i)}$$

Finally, the average Silhouette value s for the whole clustering is computed:

$$s := \frac{1}{k} \sum_{1=1}^{k} s_i$$

We also considered the average number of clusters resulting from the repetitions of the experiments on each knowledge base. A stable algorithm should return almost the same number of clusters on each repetition. It is also interesting to compare this number to the one of the primitive and defined concepts in each ontology (see Tab. 1), although this is not a hierarchical clustering method.

5.2 Results

As mentioned, the experiment consisted in 10 runs of the evolutionary clustering procedure with an optimized feature set (computed in advance). each run took from a few minutes to 41 mins on a 2.5GhZ (512Mb RAM) Linux Machine. Note that these timings include the pre-processing phase, that was needed to compute the distance values between all couples of individuals. Indeed, the elapsed time for the core clustering algorithm is actually very short (max 3 minutes).

The outcomes of the experiments are reported in Tab. 2. For each for each knowledge base and index, the average values observed along the various repetitions is considered.

[2]See the Protégé library: `http://protege.stanford.edu/plugins/owl/owl-library` and the website: `http://www.cs.put.poznan.pl/alawrynowicz/financial.owl`

[3]`http://pellet.owldl.com`

[4]δ_p is defined $\delta_p(C_i, C_j) := \max\{d_p(C_i, C_j), d_p(C_j, C_i)\}$, where $d_p(C_i, C_j) := \max_{a \in C_i}\{\min_{b \in C_j}\{d_p(a, b)\}\}$.

Table 1: Ontologies employed in the experiments.

Ontology	DL lang.	# concepts	# object prop.	# data prop.	#individuals
FSM	$\mathcal{SOF}(D)$	20	10	7	37
SurfaceWaterModel	$\mathcal{ALCOF}(D)$	19	9	1	115
Transportation	$\mathcal{ALC}$	44	7	0	331
NewTestamentNames	$\mathcal{SHIF}(D)$	47	27	8	676
Financial	$\mathcal{ALCIF}$	60	16	0	1000

Table 2: Results of the experiments: for each index, average value (±standard deviation) and [min,max] interval of values are reported.

Ontology	R-Squared	Dunn's	Silhouette	# clusters
FSM	.39 (±.07) [.33,.52]	.72 (±.10) [.69,1.0]	.77 (±.01) [.74,.78]	4 (±.00) [4,4]
SurfaceWaterModel	.45 (±.15) [.28,.66]	.99 (±.03) [.9,1.0]	.999 (±.000) [.999,.999]	12.9 (±.32) [12,13]
Transportation	.33 (±.04) [.26,.40]	.67 (±.00) [.67,.67]	.975 (±.004) [.963,.976]	3 (±.00) [3,3]
NewTestamentNames	.46 (±.08) [.35,.59]	.79 (±.17) [.5,1.0]	.985 (±.008) [.968,.996]	29.2 (±2.9) [25,32]
Financial	.37 (±.06) [.29,.45]	.88 (±1.16) [.57,1.0]	.91 (±.03) [.87,.94]	8.7 (±.95) [8,10]

Moreover, the standard deviation and the range of minimum and maximum values are also reported.

The R-Squared index values denotes an acceptable degree of separation between the various clusters. We may interpret the outcomes observing that clusters present a higher degree of compactness (measured by the SS_w component). It should also pointed out that flat clustering penalizes separation as the concepts in the knowledge base are not necessarily disjoint. Rather, they naturally tend to form subsumption hierarchies. Observe also that the variation among the various runs is very limited.

Dunn's index measures both compactness and separation; the rule in this case is *the larger the better*. Results are good for the various bases. These outcomes may serve for further comparisons to the performance of other clustering algorithms. Again, note that the variation among the various runs is very limited, so the algorithm was quite stable, despite its inherent randomized nature.

It can be observed that for the average Silhouette measure, that has a precise range of values, the performance of our algorithm is generally very good with a degradation with the increase of individuals taken into account. Besides, note that the largest knowledge base (in terms of its population) is also the one with the maximal number of concepts which provided the features for the metric. Thus in the resulting search space there is more freedom in the choice of the ways to make one individual discernible from the others. Surprisingly, the number of clusters is limited w.r.t. the number of concepts in the KB, suggesting that many individuals gather around a restricted subset of the concepts, while the others are only complementary (they can be used to discern the various individuals). Such subgroups may be detected extending our method to perform hierarchical clustering.

As regards the overall stability of the clustering procedure, we may observe that the main indices (and the number of clusters) show very little variations along the repetitions (see the standard deviation values), which suggests that the algorithm tends to converge towards clusterings of comparable quality with generally the same number of clusters. As such, the optimization procedure does not seem to suffer from being caught in local minima. However, the case needs a further investigation. Indeed, the optimization performed by the clustering procedure is two-fold: it does not optimize the choice of the clustering medoids but also their number, which is normally considered as a fixed parameter for other algorithms (see Sect. 4).

Other experiments (whose outcomes are not reported here) showed that sometimes the initial genome length may have an impact to the resulting clustering, thus suggesting the employment of different randomized search procedures (e.g. again simulated annealing or tabu search) which may guarantee a better exploration of the search space.

6. CONCLUSIONS AND FUTURE WORK

This work has presented a framework for evolutionary conceptual clustering that can be applied to standard relational representations for knowledge bases in the SW context. Its intended usage is for discovering interesting groupings of semantically annotated resources and can be applied to a wide range of concept languages. Besides, the induction of new concepts may follow from such clusters, which allows for accounting for them from an intensional viewpoint.

The method exploits a dissimilarity measure, that is based on the undelying resource semantics w.r.t. a number of dimensions corresponding to a committee of features represented by a group of concept descriptions in the language of choice. A preliminary learning phase, based on randomized search, can be exploited to optimize the choice of the most discriminating features.

The evolutionary clustering algorithm is an extension of distance-based clustering procedures employing medoids as

cluster prototypes so to deal with complex representations of the target context. Variable-length strings of medoids yielding different partitions are searched guided by a fitness function based on cluster separation. As such the algorithm can also determine the length of the list, i.e. an optimal number of clusters.

As for the metric induction part, a promising research line, for extensions to matchmaking, retrieval and classification, is *retrieval by analogy* [5]: a search query may be issued by means of prototypical resources; answers may be retrieved based on local models (intensional concept descriptions) for the prototype constructed (on the fly) based on the most similar resources (w.r.t. some similarity measure). The presented algorithm may be the basis for the model construction activity. The distance measure may also serve as a ranking criterion.

The natural extensions of the clustering algorithm that may be foreseen are towards incrementality and hierarchical clustering. The former may be easily achieved by assigning new resources to their most similar clusters, and restarting the whole algorithm when some validity measure crosses a given threshold. The latter may be performed by wrapping the algorithm within a level-wise procedure starting with the whole dataset and recursively applying the partitive method until a criterion based on quality indices determines the stop. This may produce more meaningful concepts during the next supervised phase.

Better fitness functions may be also investigated for both distance optimization and clustering. For instance, some clustering validity indices can be exploited in the algorithm as measures of compactness and separation.

7. ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for making useful points for the improvement of the paper and for providing suggestions for further investigations.

This work was partially supported by the regional interest projects DIPIS (*Distributed Production as Innovative System*) and DDTA (*Distretto Digitale Tessile Abbigliamento*) in the context of the semantic web service discovery.

8. REFERENCES

[1] Baader, F., Calvanese, D., McGuinness, D., Nardi, D., and Patel-Schneider, P., Eds. *The Description Logic Handbook*. Cambridge University Press, 2003.

[2] Bezdek, J., and Pal, N. Some new indexes of cluster validity. *IEEE Transactions on Systems, Man, and Cybernetics 28*, 3 (1998), 301–315.

[3] Borgida, A., Walsh, T., and Hirsh, H. Towards measuring similarity in description logics. In *Working Notes of the International Description Logics Workshop* (Edinburgh, UK, 2005), I. Horrocks, U. Sattler, and F. Wolter, Eds., vol. 147 of *CEUR Workshop Proceedings*.

[4] Burke, E., and Kendall, G., Eds. *Search Methodologies*. Springer, 2005, ch. 7. Simulated Annealing, pp. 187–210.

[5] d'Amato, C., Fanizzi, N., and Esposito, F. Reasoning by analogy in description logics through instance-based learning. In *Proceedings of Semantic Web Applications and Perspectives, 3rd Italian Semantic Web Workshop, SWAP2006* (Pisa, Italy, 2006), G. Tummarello, P. Bouquet, and O. Signore, Eds., vol. 201 of *CEUR Workshop Proceedings*.

[6] Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. A density-based algorithm for discovering clusters in large spatial databases. In *Proceedings of the 2nd Conference of ACM SIGKDD* (1996), pp. 226–231.

[7] Fanizzi, N., d'Amato, C., and Esposito, F. Induction of optimal semi-distances for individuals based on feature sets. In *Working Notes of the 20th International Description Logics Workshop, DL2007* (Bressanone, Italy, 2007), D. Calvanese, E. Franconi, V. Haarslev, D. Lembo, B. Motik, A.-Y. Turhan, and S. Tessaris, Eds., vol. 250 of *CEUR Workshop Proceedings*.

[8] Fanizzi, N., Iannone, L., Palmisano, I., and Semeraro, G. Concept formation in expressive Description Logics. In *Proceedings of the 15th European Conference on Machine Learning, ECML2004* (2004), J.-F. Boulicaut, F. Esposito, F. Giannotti, and D. Pedreschi, Eds., vol. 3201 of *LNAI*, Springer, pp. 99–113.

[9] Ghozeil, A., and Fogel, D. Discovering patterns in spatial data using evolutionary programming. In *Genetic Programming 1996: Proceedings of the First Annual Conference* (Stanford University, CA, USA, 1996), J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, Eds., MIT Press, pp. 521–527.

[10] Halkidi, M., Batistakis, Y., and Vazirgiannis, M. On clustering validation techniques. *Journal of Intelligent Information Systems 17*, 2-3 (2001), 107–145.

[11] Hall, L. O., Özyurt, I. B., and Bezdek, J. C. Clustering with a genetically optimized approach. *IEEE Trans. Evolutionary Computation 3*, 2 (1999), 103–112.

[12] Hirano, S., and Tsumoto, S. An indiscernibility-based clustering method. In *2005 IEEE International Conference on Granular Computing* (2005), X. Hu, Q. Liu, A. Skowron, T. Y. Lin, R. Yager, and B. Zhang, Eds., IEEE, pp. 468–473.

[13] Iannone, L., Palmisano, I., and Fanizzi, N. An algorithm based on counterfactuals for concept learning in the semantic web. *Applied Intelligence 26*, 2 (2007), 139–159.

[14] Jain, A., Murty, M., and Flynn, P. Data clustering: A review. *ACM Computing Surveys 31*, 3 (1999), 264–323.

[15] Kaufman, L., and Rousseeuw, P. *Finding Groups in Data: an Introduction to Cluster Analysis*. John Wiley & Sons, 1990.

[16] Kietz, J.-U., and Morik, K. A polynomial approach to the constructive induction of structural knowledge. *Machine Learning 14*, 2 (1994), 193–218.

[17] Lee, C.-Y., and Antonsson, E. K. Variable length genomes for evolutionary algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO00* (2000), L. Whitley, D. Goldberg, E. Cantú-Paz, L. Spector, I. Parmee, and H.-G. Beyer, Eds., Morgan Kaufmann, p. 806.

[18] Lehmann, J. Concept learning in description logics. Master's thesis, Dresden University of Technology, 2006.

[19] Nasraoui, O., and Krishnapuram, R. One step evolutionary mining of context sensitive associations and web navigation patterns. In *Proceedings of the SIAM conference on Data Mining* (Arlington, VA, 2002), pp. 531–547.

[20] Ng, R., and Han, J. Efficient and effective clustering method for spatial data mining. In *Proceedings of the 20th Conference on Very Large Databases, VLDB94* (1994), pp. 144–155.

[21] Nienhuys-Cheng, S.-H. Distances and limits on herbrand interpretations. In *Proceedings of the 8th International Workshop on Inductive Logic Programming, ILP98* (1998), D. Page, Ed., vol. 1446 of *LNAI*, Springer, pp. 250–260.

[22] Pawlak, Z. *Rough Sets: Theoretical Aspects of Reasoning About Data*. Kluwer Academic Publishers, 1991.

[23] Sebag, M. Distance induction in first order logic. In *Proceedings of the 7th International Workshop on Inductive Logic Programming, ILP97* (1997), S. Džeroski and N. Lavrač, Eds., vol. 1297 of *LNAI*, Springer, pp. 264–272.

[24] Stepp, R. E., and Michalski, R. S. Conceptual clustering of structured objects: A goal-oriented approach. *Artificial Intelligence 28*, 1 (Feb. 1986), 43–69.

[25] Zezula, P., Amati, G., Dohnal, V., and Batko, M. *Similarity Search – The Metric Space Approach*. Advances in database Systems. Springer, 2007.