An Evolutionary Algorithm for Discovering Multi-Relational Association Rules in the Semantic Web

Minh Tran Duc, Claudia d'Amato, Binh Thnanh Nguyen, Andrea Tettamanzi

An Evolutionary Algorithm for Discovering Multi-Relational Association Rules in the Semantic Web

Minh Duc Tran
Université Côte d'Azur, CNRS, Inria, I3S, France
tdminh2110@yahoo.com

Claudia d'Amato
University of Bari, Italy
claudia.damato@uniba.it

Binh Thanh Nguyen
The University of Danang – University of Science and Technology, Vietnam
ntbinh@dut.udn.vn

Andrea G. B. Tettamanzi
Université Côte d'Azur, CNRS, Inria, I3S, France
andrea.tettamanzi@unice.fr

ABSTRACT

In the Semantic Web context, OWL ontologies represent the conceptualization of domains of interest while the corresponding assertional knowledge is given by RDF data referring to them. Because of its open, distributed, and collaborative nature, such knowledge can be incomplete, noisy, and sometimes inconsistent. By exploiting the evidence coming from the assertional data, we aim at discovering hidden knowledge patterns in the form of multi-relational association rules while taking advantage of the intensional knowledge available in ontological knowledge bases. An evolutionary search method applied to populated ontological knowledge bases is proposed for finding rules with a high inductive power. The proposed method, EDMAR, uses problem-aware genetic operators, echoing the refinement operators of ILP, and takes the intensional knowledge into account, which allows it to restrict and guide the search. Discovered rules are coded in SWRL, and as such they can be straightforwardly integrated within the ontology, thus enriching its expressive power and augmenting the assertional knowledge that can be derived. Additionally, discovered rules may also suggest new axioms to be added to the ontology. We performed experiments on publicly available ontologies, validating the performances of our approach and comparing them with the main state-of-the-art systems.

CCS CONCEPTS

•**Computing methodologies** → **Ontology engineering; Logical and relational learning; Genetic algorithms;** *Randomized search;*

KEYWORDS

Evolutionary Algorithms; Description Logics; Pattern Discovery

ACM Reference format:
Minh Duc Tran, Claudia d'Amato, Binh Thanh Nguyen, and Andrea G. B. Tettamanzi. 2017. An Evolutionary Algorithm for Discovering Multi-Relational Association Rules in the Semantic Web. In *Proceedings of GECCO '17, Berlin, Germany, July 15-19, 2017,* 8 pages.
DOI: http://dx.doi.org/10.1145/3071178.3079196

GECCO '17, Berlin, Germany

1 INTRODUCTION

The Semantic Web (SW) [3] is a new vision of the Web aiming at making Web contents machine-readable. Web resources are semantically annotated with metadata referring to ontologies, which are formal conceptualizations of domains of interest acting as shared vocabularies where the meaning of the annotations is formally defined. As such, annotated web resources represent the assertional knowledge given the intensional definitions provided by ontologies. Assertional and intensional ontological knowledge will be referred to as ontological knowledge base. In the SW view, data, information, and knowledge are connected following best practices and exploiting standard Web technologies like HTTP, RDF, and URIs. This allows to share and link information that can be read automatically by computers meanwhile creating a global space of semantically described resources. The description of data/resources in terms of ontologies represents a key aspect in the SW. Interestingly, ontologies are also equipped with powerful deductive reasoning capabilities. However, due to the SW heterogeneous and distributed nature, ontological knowledge bases (KBs)[1] may turn out to be incomplete and noisy w.r.t. the domain of interest. Specifically, an ontology is incomplete when it is logically consistent (i.e., it contains no contradiction) but it lacks information (e.g., assertions, disjointness axioms, etc.) w.r.t. the reference domain; while it is noisy when it is logically consistent but it contains invalid information w.r.t. the reference domain. These situations may prevent the inference of relevant information or cause incorrect information to be derived.

By exploiting the evidence coming from (assertional) knowledge, data mining techniques could be exploited for discovering hidden knowledge patterns from ontological KBs, to be used for enriching an ontology both at terminological (schema) and assertional (facts) level, even in presence of incompleteness and/or noise. We present EDMAR,[2] an evolutionary algorithm (EA) for discovering hidden knowledge patterns in the form of multi-relational association rules (ARs) coded in SWRL [11], which can be added to the ontology enriching its expressive power and increasing the assertional knowledge that can be derived. Additionally, discovered rules may suggest new axioms to be added to the ontology, such as transitivity and symmetry of a role, and/or concept/role inclusion axioms. Related work focussing on similar goals can be found both in the

[1] By *ontological knowledge base*, we refer to a populated ontology, namely an ontology where both the schema and instance level are specified. This expression will be interchangeably used with the term ontology.
[2] EDMAR is for "**E**volutionary **D**iscovery of **M**ulti-relational **A**ssociation **R**ules".

inductive logic programming (ILP) [6, 15, 16], and in the SW community [9, 12, 13, 20]. However, none of those approaches takes advantage of the exploration capabilities of EAs jointly with the reasoning capabilities of ontologies. Recently, a first attempt in this novel direction to discover hidden knowledge patterns in ontological KBs has been made [5]. However, that proposal suffers from some limitations, which our solution manages to overcome. In this paper we show how EDMAR is able to return a number of rules with high inductive (predictive) capabilities while pruning redundant rules.

In the next section, basics are provided. EDMAR is presented in Section 3 and its main characteristics and added value w.r.t. the state of the art are discussed in Section 4. Section 5 provides an experimental evaluation and comparison to the main state-of-the-art systems. Conclusions and future directions are drawn in Section 6.

2 BASICS

We refer to ontological KBs described in Description Logics (DLs) [2] (representing the theoretical foundation of OWL[3] that is the standard representation language in the SW), without restricting ourselves to any specific DL. As usual in DLs, we refer to a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ consisting of a TBox $\mathcal{T}$ containing the terminological axioms and an ABox $\mathcal{A}$ containing the assertional axioms. The formal meaning of the axioms is given in terms of model-theoretic semantics. As for reasoning services, we exploit *instance checking*, which assesses if an individual is an instance of a given concept, and *concept subsumption*, which checks whether a concept (role) is subsumed by another concept (role). It should be recalled that DLs adopt the *open-world assumption* (OWA), which has consequences on answering class-membership queries. Specifically, for an individual that cannot be proved to be an instance of a certain concept, it cannot be intended as a counterexample for it. Rather, it should be interpreted as a case of insufficient (incomplete) knowledge for possibly proving the assertion. For more details concerning DLs see [2].

In the following, the definition of relational AR for an ontological KB is given. Hence, the addressed problem is formally defined.

Definition 2.1 (Relational Association Rule). Given a populated ontological KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, a *relational association rule* r for $\mathcal{K}$ is a Horn-like clause of the form: $body \rightarrow head$, where: (a) $body$ is a generalization of a set of assertions in $\mathcal{K}$ co-occurring together; (b) $head$ is a consequent that is induced from $\mathcal{K}$ and $body$

Definition 2.2 (Problem Definition).

Given:
- a populated ontological knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$;
- a minimum "frequency threshold", θ_f;
- a minimum "fitness threshold", θ_{fit};

Discover: all frequent and fit hidden patterns w.r.t θ_f and θ_{fit}, in the form of relational ARs, that may induce new assertions for $\mathcal{K}$.

Intuitively, a *frequent hidden pattern* is a generalization of a set of concept/role assertions co-occurring reasonably often (w.r.t. a fixed frequency threshold) together, showing an underlying form of correlation that is exploited for obtaining new assertions.

[3] https://www.w3.org/OWL/

The rules to be discovered (following Def. 2.2), will be represented in the Semantic Web Rule Language (SWRL) [11], which extends the set of OWL axioms with Horn-like rules.[4]

Definition 2.3 (SWRL Rule). Given a KB $\mathcal{K}$, a SWRL rule is an implication between an antecedent (body) and a consequent (head) of the form: $B_1 \wedge B_2 \wedge \ldots B_n \rightarrow H_1 \wedge \cdots \wedge H_m$,
where $B_1 \wedge \cdots \wedge B_n$ is the rule body and $H_1 \wedge \cdots \wedge H_m$ is the rule head. Each $B_1, \ldots, B_n, H_1, \ldots H_m$ is called *atom*.

An *atom* is a unary or binary predicate of the form $C(s)$, $R(s_1, s_2)$, $\mathsf{sameAs}(s_1, s_2)$ or $\mathsf{differentFrom}(s_1, s_2)$, where the predicate symbol C is a concept name in $\mathcal{K}$, R is a role name in $\mathcal{K}$, s, s_1, s_2 are *terms*. A *term* is either a variable (denoted by x, y, z) or a constant (denoted by a, b, c) standing for an individual name or data value.

The discovered rules can be generally called *multi-relational* rules since multiple binary predicates $R(s_1, s_2)$ with different role names of $\mathcal{K}$ could appear in a rule. The intended meaning of a rule is: whenever the conditions in the antecedent hold, the conditions in the consequent must also hold. A rule having more than one atom in the head can be equivalently transformed, due to the *safety condition* (see Def. 2.5), into multiple rules, each one having the same body and a single atom in the head. We will consider, w.l.o.g., only SWRL rules (hereafter just "rules") with one atom in the head.

Example 2.4 (SWRL rule). Given a SWRL rule of the form $\mathsf{fatherOf}(y, x) \wedge \mathsf{Male}(x) \rightarrow \mathsf{sonOf}(x, y)$ where $\mathsf{sonOf}(x, y)$ is the rule head; $\mathsf{fatherOf}(y, x) \wedge \mathsf{Male}(x)$ is the rule body; $\mathsf{fatherOf}$, Male, sonOf are atoms and x, y are variables.

2.1 Fixing the Language Bias

A *language bias* is a set of constraints giving a tight specification of the patterns worth considering, thus allowing to reduce the search space. We are interested in rules having only atomic concepts and/or role names of $\mathcal{K}$ as predicate symbols, and individual names as constants. Only *connected* [9] and *non-redundant* [12] rules satisfying the *safety condition* [10] are considered.[5] In the following, notations and formal definitions for the listed properties are reported.

Given an atom A, let $T(A)$ denote the set of all the terms occurring in A and let $V(A) \subseteq T(A)$ denote the set of all the variables occurring in A e.g. $V(C(x)) = \{x\}$ and $V(R(x, y)) = \{x, y\}$. Such notation may be extended to rules straightforwardly.

Definition 2.5 (Safety Condition). Given a KB $\mathcal{K}$ and a rule $r = B_1 \wedge B_2 \wedge \ldots B_n \rightarrow H$, r satisfies the *safety condition* if all variables appearing in the rule head also appear in the rule body; formally if: $V(H) \subseteq \bigcup_{i=1}^{n} V(B_i)$,

Definition 2.6 (Connected Rule). Given a KB $\mathcal{K}$ and a rule $r = B_1 \wedge B_2 \wedge \ldots B_n \rightarrow H$, r is *connected* iff every atom in r is transitively connected to every other atom in r.

Two atoms B_i and B_j in r, with $i \neq j$, are *connected* if they share at least a variable or a constant i.e. if $T(B_i) \cap T(B_j) \neq \emptyset$. Two atoms B_1 and B_k in r are *transitively connected* if there exist

[4] The result is a KB with an enriched expressive power. More complex relationships than subsumption can be expressed. For details see [10].

[5] To guarantee decidability, only *DL-safe rules* are sought for [14], i.e., rules interpreted under the DL-safety condition, whose variables are bound only to explicitly named individuals in $\mathcal{K}$. When added to an ontology, DL-safe rules are decidable and generate sound, but not necessarily complete, results.

in r, atoms $B_2, \ldots, B_{k-1}$, with $k \leq n$, such that, for all $i, j \in \{1, \ldots, k\}$ with $i \neq j$, $T(B_i) \cap T(B_j) \neq \emptyset$.

Example 2.7 (Disconnected rule). The rule $\mathsf{wifeOf}(y, x) \wedge \mathsf{siblingOf}(z, w) \rightarrow \mathsf{spouseOf}(x, y)$ is disconnected, since the atom $\mathsf{siblingOf}(z, w)$ does not share any variable with the other atoms.

Definition 2.8 (Closed Rule). Given a KB $\mathcal{K}$ and a rule $r = B_1 \wedge B_2 \wedge \ldots B_n \rightarrow H$, r is *closed* iff every variable in r is closed.
Each variable $v_j \in \bigcup_{i=1}^{n} V(B_i), j \in \{1, \ldots, k\}$, with $k \leq n$, is closed if it appears at least twice in r.

Example 2.9 (Open rule). Rule $\mathsf{sonOf}(z, x) \rightarrow \mathsf{spouseOf}(x, y)$ is not closed, since variables z and y are not closed.

Definition 2.10 (Non-redundant Rule). Given a KB $\mathcal{K}$ and a rule $r = B_1 \wedge B_2 \wedge \ldots B_n \rightarrow H$, r is a non-redundant rule if no atom in r is entailed by other atoms in r with respect to $\mathcal{K}$, i.e., if, $\forall i \in \{0, 1, \ldots, n\}$, with $B_0 = H$, results: $\bigwedge_{j \neq i} B_j \not\models_{\mathcal{K}} B_i$,

Example 2.11 (Redundant Rule). Given $\mathcal{K}$ with $\mathcal{T} = \{\mathsf{Father} \sqsubseteq \mathsf{Parent}\}$ and the rule $r = \mathsf{Father}(x) \wedge \mathsf{Parent}(x) \rightarrow \mathsf{Human}(x)$ where Human is a primitive concept, r is redundant since the atom $\mathsf{Parent}(x)$ is entailed by the atom $\mathsf{Father}(x)$ with respect to $\mathcal{K}$.

2.2 Metrics for Rules Evaluation

For determining the rules of interest for the goal in Def. 2.2, metrics for assessing the quality of a rule are necessary. In the following, the adopted metrics are summarized.

Given a rule $r = B_1 \wedge \ldots \wedge B_n \rightarrow H$, let us denote:

- $\Sigma_H(r)$ the set of distinct bindings of the variables occurring in the head of r, formally: $\Sigma_H(r) = \{binding\ V(H)\}$
- $E_H(r)$ the set of distinct bindings of the variables occurring in the head of r provided the body and the head of r are satisfied, formally: $E_H(r) = \{binding\ V(H) \mid \exists\ binding\ V(B_1 \wedge \cdots \wedge B_n) : B_1 \wedge \cdots \wedge B_n \wedge H\}$. Since rules are connected and closed, $V(H) \subseteq V(B_1 \wedge \cdots \wedge B_n)$
- $M_H(r)$ the set of distinct bindings of the variables occurring in the head of r also appearing as binding for the variables occurring in the body of r, formally: $M_H(r) = \{binding\ V(H) \mid \exists\ binding\ V(B_1 \wedge \cdots \wedge B_n) : B_1 \wedge \cdots \wedge B_n\}$
- $P_H(r)$ the set of distinct bindings of the variables occurring in the head of r provided that the body and the head of r are satisfied. Particularly, this applies when a role atom is in the head of the considered rule. Formally: $P_H(r) = \{binding\ V(H) \mid \exists\ binding\ V(B_1 \wedge \cdots \wedge B_n) \cup v_{rng}(H') :\ B_1 \wedge \cdots \wedge B_n \wedge H'\}$ where
  - H and H' are role atoms with the same the predicate symbol R;
  - $V(H) \subseteq V(B_1 \wedge \cdots \wedge B_n)$ since rules are connected and closed
  - $v_{dom}(H) = v_{dom}(H')$ and $v_{rng}(H) \neq v_{rng}(H')$;
  with v_{dom} and v_{rng} standing for the domain and range variables respectively of the predicate symbol R
  - $v_{rng}(H') \notin V(B_1 \wedge \cdots \wedge B_n)$;

Like in [4, 9], the classical definitions (as used in [1]) are modified to ensure monotonicity, as summarized in the following.

Definition 2.12 (Rule Support). Given a rule $r = B_1 \wedge \ldots \wedge B_n \rightarrow H$, its support is the number of distinct bindings of the variables in the head, provided the body and the head of r are satisfied jointly, formally:

$$\mathrm{supp}(r) = |E_H(r)|. \tag{1}$$

Example 2.13 (Computation of Rule Support). Given the rule $r = \mathsf{feed}(x, y) \rightarrow \mathsf{love}(x, y)$ and assuming the following bindings $\{\mathsf{feed}(Anna, Dog), \mathsf{feed}(Anna, Cat), \mathsf{feed}(Peter, Pig), \mathsf{love}(Anna, Dog), \mathsf{love}(George, Cat)\}$ exist then $\mathrm{supp}(r) = 1$, as there is just one binding for the rule head ($\mathsf{feed}(Anna, Dog)$) allowing the head $\mathsf{love}(Anna, Dog)$ and the body $\mathsf{feed}(Anna, Dog)$ to be jointly satisfied.

Definition 2.14 (Head Coverage for a Rule). Given the rule $r = B_1 \wedge \ldots \wedge B_n \rightarrow H$, its head coverage is the ration between the rule support and the distinct variable bindings from the head of the rule

$$\mathrm{headCoverage}(r) = |E_H(r)|/|\Sigma_H(r)|. \tag{2}$$

Example 2.15 (Computation of Head Coverage). Given the rule r in Ex. 2.13 and the corresponding bindings, $\mathrm{headCoverage}(r) = \frac{1}{2}$ since there are two bindings for the head of r: $\{\mathsf{love}(Anna, Dog), \mathsf{love}(George, Cat)\}$.

Definition 2.16 (Rule Confidence). Given a rule $r = B_1 \wedge \ldots \wedge B_n \rightarrow H$, its confidence is defined as the ratio of the number of the rule support and the number of bindings in the rule body:

$$\mathrm{conf}(r) = |E_H(r)|/|M_H(r)|. \tag{3}$$

Example 2.17 (Computation of Rule Confidence). Given the rule r in Ex. 2.13 and the corresponding bindings, $\mathrm{conf}(r) = \frac{1}{3}$, since there are three bindings, $\{\mathsf{feed}(Anna, Dog), \mathsf{feed}(Anna, Cat), \mathsf{feed}(Peter, Pig)\}$ for the body of r.

An issue with these definitions, and particularly Def. 2.16, is that an implicit closed-world assumption is made, since no distinction between *incorrect* predictions, i.e., bindings σ matching r such that $\mathcal{K} \models \neg H\sigma$, and *unknown* predictions, i.e., bindings σ matching r such that both $\mathcal{K} \models H\sigma$ and $\mathcal{K} \models \neg H\sigma$, is made. Reasoning on ontologies is grounded on the OWA and our goal is to maximize correct predictions, not just describing the available data. Hence, we also adopt the *PCA Confidence* [9] that is able to take into account the OWA.

Definition 2.18 (Rule PCA-Confidence). Given the rule $r = B_1 \wedge \ldots \wedge B_n \rightarrow H$, its PCA (Partial Completeness Assumption) confidence is defined as follows:

$$\mathrm{pcaconf}(r) = \begin{cases} |E_H(r)|/|M_H(r)|, & \text{if } H \text{ is a concept atom;} \\ |E_H(r)|/|P_H(r)|, & \text{if } H \text{ is a role atom.} \end{cases} \tag{4}$$

For the example described in Ex. 2.13, $\mathrm{pcaconf}(r) = \frac{1}{2}$.

Definition 2.19 (Rule Precision). Given the rule $r = B_1 \wedge \ldots \wedge B_n \rightarrow H$, its precision is the ratio of the number of correct predictions made by r and the total number of correct and incorrect predictions (predictions logically contradicting $\mathcal{K}$), leaving out the predictions with unknown truth value.

This metric expresses the ability of a rule to perform correct predictions, but it is not able to take into account the induced knowledge, that is the *unknown* predictions. For this reason, the metrics proposed in [8] are also considered (for the evaluation in Sect. 5):

- *match rate*: number of predicted assertions in agreement with facts in the complete ontology, out of all predictions;
- *commission error rate*: number of predicted assertions contradicting facts in the full ontology, out of all predictions;
- *induction rate*: number of predicted assertions that are not known (i.e., for which there is no information) in the complete ontology, out of all predictions.

3 EVOLUTIONARY DISCOVERY OF RELATIONAL ASSOCIATION RULES

Given a populated ontological KB, our goal is to discover frequent and accurate hidden patterns in the form of multi-relational ARs to be exploited for making predictions of new assertions in the KB. The discovered rules are compliant with the fixed language bias (see Sect. 2.1). Particularly, they are DL-Safe and expressed in SWRL (see Sect. 2). Hence, they can be integrated with the existing ontology, resulting in a KB with an enriched expressive power [10, 11].

EDMAR maintains a population of patterns (the individuals) and makes it evolve by iteratively applying a number of genetic operators. A pattern is the genotype of an individual and the corresponding rule is its phenotype, constructed using the first atom of the pattern as the head and the remaining atoms as the body. Since the goal is to discover rules capable of making (possibly a large number of) accurate predictions, the fitness of a pattern is the sum of the head coverage and of the PCA confidence of the corresponding rule, which measures its the overall quality. The EA we propose may be regarded as an improvement of a previous proposal by three of us [5] and at the same time as an alternative and complementary approach to level-wise generate-and-test algorithms for discovering relational ARs from RDF data-sets [9] and recent proposals that take into account terminological axioms and deductive reasoning capabilities [4].

3.1 Representation

The overall flow of the EA is shown by Algorithm 1. As in [4, 5, 9], a pattern is represented as a list of atoms, to be interpreted in conjunctive form. For each discovered frequent pattern, a multi-relational AR is constructed by considering the first atom in the list as the head of the rule and the remaining atoms as the rule body. The length of a rule is the number of atoms in the rule (including the head atom).

Example 3.1. Given the pattern $[\mathsf{sonOf}(x, y), \mathsf{fatherOf}(y, x), \mathsf{Male}(x)]$, the corresponding rule is $\mathsf{fatherOf}(y, x) \wedge \mathsf{Male}(x) \rightarrow \mathsf{sonOf}(x, y)$. The rule length is 3. Its meaning is: "if y is x's father and x is male, then x is a son of y".

The EA is steady-state: children are created by applying genetic operators on selected parents, and then the children are added back into the population to compete with individuals in the old population in order to allow transition into the new population at the next cycle. The selection operator chooses the best parents for reproduction.

Algorithm 1 Evolutionary algorithm for the discovery of multi-relational ARs from a populated ontological KB

Input: $\mathcal{K}$: ontological KB; θ_f: frequency threshold; n: the size of the population; p_{cross}: crossover probability; p_{mut}: mutation probability; τ: truncation proportion; θ_{fit}: fitness threshold;
Output: *pop*: set of frequent patterns discovered from $\mathcal{K}$
1: Creating a list A_f of frequent atoms in $\mathcal{K}$
2: Initialize a population *pop* of size n by using n times CREATENEWPATTERN() operator
3: Compute fitness values for all of the patterns in *pop*
4: Sort *pop* by decreasing fitness value
5: Initialize *the number of generation* (equals to 0)
6: **while** (*the number of generation* $<$ MAX_GENERATIONS) **do**
7: **for** ($i = 0, 1, \ldots, \lfloor \tau n \rfloor$) **do**
8: CROSSOVER(*pop*$[i]$, *pop*$[\lfloor \tau n \rfloor + i]$)
9: CROSSOVER(*pop*$[i]$, *pop*$[2\lfloor \tau n \rfloor + i]$)
10: Compute fitness value for all of offspring
11: **for each** offspring **do**
12: **with probability** p_{mut} **do** MUTATE(*offspring*)
13: Add all of offspring to *pop*
14: Sort *pop* by decreasing fitness value
15: Remove patterns located at the end of *pop* so that the size of *pop* is exactly n
16: Increase *the number of generation* by 1
17: Remove redundant and inconsistent rules from the final population *pop*
18: Remove rules where fitness value is less than θ_{fit} from the final population *pop*
19: **return** *pop*

Algorithm 2 The CREATENEWPATTERN() Operator.

Input: a global variable A_f: a list of frequent atoms;
Output: p: a new random pattern
1: *length* $\leftarrow random(2,$ MAX_RULE_LENGTH$)$
2: p is initialized to empty
3: **while** p.LENGTH() $<$ *length* **do**
4: pick an atom $a \in A_f$ at random
5: **if** (p is not empty) **then**
6: adjust the variables in a to ensure the language bias is respected
7: add a to the end of p
8: **return** p

The genetic operators of initialization, crossover, and mutation, are designed to enforce the respect of the language bias.

An important consequence of the fact that patterns are intended to be transformed into rules for evaluation is that the order of atoms counts only insofar as one atom is in the head position (and, therefore, the head of the rule). The relative position of atoms that are not in the head position is irrelevant.

At the end of the algorithm, discovered knowledge patterns in the final population satisfy three conditions: (1) each pattern is not redundant (as for Defn. 2.10); (2) each pattern, once considered jointly with the ontology, satisfies $\mathcal{K} \cup p \not\models \bot$, in which p is the pattern in the population; (3) the fitness value of each pattern is above a given threshold θ_{fit}.

3.2 Initialization

Before creating patterns in the population, the initialization operator requires a list A_f of frequent atoms, which is computed once and for all before launching the evolutionary process (line 1 of Alg. 1). A frequent atom is a pattern r consisting of a single atom of the form $C(x)$ or $R(x, y)$, such that $\text{supp}(r) \geq \theta_f$.

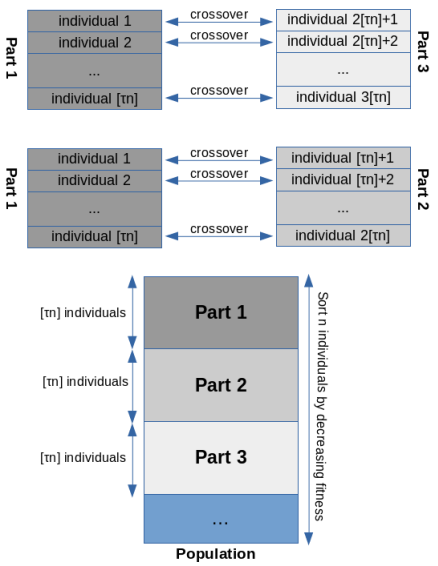
The initialization operator initializes the population with n patterns by using n times CREATENEWPATTERN() operator. A new pattern is seeded with a frequent pattern picked at random from the list A_f and a random target length uniformly distributed between 2 and MAX_RULE_LENGTH. The CREATENEWPATTERN() operator is detailed in Algorithm 2.

3.3 Selection

Before performing the selection operator, patterns in the population are sorted by decreasing fitness value and a given parameter τ is used to assist in the selection of individuals.

The selection operator is used before calling the crossover operator. The purpose of this operator is to select the best individuals to perform crossover. An illustrative diagram is depicted in Figure 1: $3\lfloor \tau n \rfloor$ individuals are selected for reproduction, each individual in Part 1 (the best individuals) is selected twice to mate with each individual in Part 2 and each individual in Part 3, respectively, in order from top to bottom.

Figure 1: Selection operator for crossover



3.4 Crossover

The crossover operator produces two offspring patterns from two parent patterns that are selected according to the selection operator (Fig. 1). The atoms of the offspring are randomly selected from the atoms of the parents. The operator, detailed in Algorithm 3, proceeds by creating a set L including all the atoms in the two input patterns and choosing a target length for the two offspring; then, atoms are picked from L at random and added to either pattern until the target length is attained, possibly changing their variables to ensure the language bias is respected.

Example 3.2. Given the two parent patterns

p_1: $[\mathsf{sonOf}(x, y), \mathsf{fatherOf}(y, x), \mathsf{Male}(x)]$,
p_2: $[\mathsf{fatherOf}(x, y), \mathsf{grandfatherOf}(x, z), \mathsf{sonOf}(z, y)]$,

according to Algorithm 3, we have:

$$L = \{\mathsf{sonOf}, \mathsf{fatherOf}, \mathsf{Male}, \mathsf{grandfatherOf}\}.$$

Suppose the (random) target length for the first child (O_1) is 4 and for the second child (O_2) is 3. The offspring after performing crossover may be the following patterns:

Algorithm 3 The Crossover Operator CROSSOVER(p_1, p_2).

Input: p_1, p_2: the two patterns to be crossed over;
Output: O_1, O_2: two patterns that are a recombination of the input patterns.
1: Creating a set L containing all of the atoms in both parents
   $A^1 \leftarrow$ a set contains atoms of p_1
   $A^2 \leftarrow$ a set contains atoms of p_2
   $L \leftarrow A^1 \cup A^2$
2: Randomly choose a target length for each offspring in the range of 2 to MAX_RULE_LENGTH (a given parameter)
   $length\_O_1 \leftarrow random(2, \text{MAX\_RULE\_LENGTH})$
   $length\_O_2 \leftarrow random(2, \text{MAX\_RULE\_LENGTH})$
3: O_1 and O_2 are initialized to empty
4: **for** $(i = 1, 2)$ **do**
5: **while** O_i.LENGTH() $<$ $length\_O_i$ **do**
6: Pick an atom $a \in L$ at random
7: **if** (O_i is not empty) **then**
8: adjust the variables in a to ensure the language bias is respected
9: add a to the end of O_i
10: **return** O_1, O_2

Algorithm 4 The Mutation Operator MUTATE(p).

Input: p: the pattern to be mutated;
Output: p': the mutated pattern.
1: **if** p.GETFITNESSVALUE() $> \theta_{\text{mut}}$ **then**
2: **if** p.LENGTH() $<$ MAX_RULE_LENGTH **then**
3: $p' \leftarrow$ SPECIALIZATION(p)
4: **else**
5: $p' \leftarrow$ CREATEBODYPATTERN(p)
6: **else**
7: **if** p.LENGTH() > 2 **then**
8: $p' \leftarrow$ GENERALIZATION(p)
9: **else**
10: $p' \leftarrow$ CREATEBODYPATTERN(p)
11: **return** p'

O_1: $[\mathsf{grandfatherOf}(x, y), \mathsf{fatherOf}(x, z), \mathsf{fatherOf}(z, y), \mathsf{Male}(y)]$,
O_2: $[\mathsf{fatherOf}(x, y), \mathsf{grandfatherOf}(x, z), \mathsf{fatherOf}(y, z)]$.

3.5 Mutation

The mutation operator perturbs a pattern from the offspring of crossover with a given probability p_{mut}. Our mutation operator, detailed in Algorithm 4, uses two operators based on the idea of specialization and generalization in ILP: it applies the specialization operator, if the fitness of the pattern is above a given threshold θ_{mut}, or the generalization operator, if its fitness is below the threshold θ_{mut}, to the pattern undergoing it. In case a pattern is too long to undergo specialization or too short to undergo generalization, mutation will apply function CREATEBODYPATTERN(p), which creates a completely new body by picking atoms from the list A_f of frequent atoms, while keeping the same head as the the parent pattern and respecting the language bias.

Example 3.3. Assume $p = [\mathsf{siblingOf}(x, y), \mathsf{stayWith}(y, x)]$, with $\text{fitness}(p) < \theta_{\text{mut}}$ undergoes mutation; then, $p' =$ CREATEBODYPATTERN(p), for instance, and $p = [\mathsf{siblingOf}(x, y), \mathsf{brotherOf}(z, x), \mathsf{Female}(x), \mathsf{sisterOf}(y, z)]$.

The specialization operator, detailed in Algorithm 5, appends a new atom to a pattern, while preserving the language bias.

Example 3.4 (Specialization). Given $p = [\mathsf{GrandFather}(x), \mathsf{Father}(x)]$, $p' =$ SPECIALIZATION(p) might yield, for instance, $p' = [\mathsf{GrandFather}(x), \mathsf{Father}(x), \mathsf{Husband}(x)]$.

The generalization operator, detailed in Algorithm 6, removes a random number of atoms located at the end of the body of p. After

Algorithm 5 The Specialization Operator SPECIALIZATION(p).

Input: p: the pattern to be specialized; a global variable A_f: a list of frequent atoms;
Output: p': the specialized pattern.
1: pick an atom $a \in A_f$ at random
2: Adjust the variables in a to ensure the language bias is respected (adjust according to p)
3: $p' \leftarrow$Add a to the end of p
4: **return** p'

Algorithm 6 The Generalization Operator GENERALIZATION(p).

Input: p: the pattern to be generalized;
Output: p': the generalized pattern.
1: Randomly generate a number n which represents the number of atoms will be removed.
$n \leftarrow random(1, p.body.\text{LENGTH}() - 1)$
2: **for** $(i = 1, \ldots, n)$ **do**
3: Remove the last atom from p
4: $p' \leftarrow$ Adjust the variables of atoms in p (if necessary) to ensure the language bias is respected.
5: **return** p'

removing atoms, the length of the body must remain at least one atom and preserve the language bias.

Example 3.5 (Generalization). Given a pattern $p = [\mathsf{sonOf}(x, y), \mathsf{fatherOf}(y, x), \mathsf{Male}(x), \mathsf{motherOf}(z, x), \mathsf{spouseOf}(z, y)]$, $p' = \text{GENERALIZATION}(p)$ might yield, assuming the operator randomly chooses to remove three atoms, $p' = [\mathsf{sonOf}(x, y), \mathsf{fatherOf}(y, x)]$.

3.6 Fitness

In order to assess the fitness value, the head coverage metrics (Def. 2.14) is primarily considered, as also argued in the related literature [5]. Indeed, this metric captures the generality of a pattern, since one expects good-quality patterns to cover a large share of the known true facts. In addition, we argue that a measure of rule confidence should be combined with the notion of coverage in order to reward those patterns that, besides covering a large number of true facts, also cover as few false facts as possible. Since DLs adopt the OWA, we must use the PCA-Confidence metric (Def. 2.18) to measure the confidence of a pattern. This metric is an indication of how often the pattern has been found to be true. High confidence value means that the pattern has a low error rate and *vice versa*. Hence, the selected fitness function is

$$\text{fitness}(r) = \text{headCoverage}(r) + \text{pcaconf}(r).$$

A high fitness indicates that a pattern is meaningful (general and accurate). The two terms of the fitness function might be viewed as two conflicting objectives; therefore, they could be weighted differently or a two-objective EA could be used to find non-dominated rules. We leave the exploration of both ideas for future work.

3.7 Consistency Check

Inconsistent rules, i.e., rules that are unsatisfiable when considered jointly with the ontology, are of no use for KB enrichment and have thus to be discarded.[6] Notice that this case should never occur if the ontological KB is consistent and noise-free. Nevertheless, since EDMAR can also be applied to noisy ontologies, it may happen that an unsatisfiable rule/pattern (when considered jointly with the ontology) is extracted, particularly if low frequency and fitness thresholds (see Sect. 2.2 for details about the metrics) are considered.

Since checking rules for consistency may be computationally very expensive, we have decided not to check patterns for consistency during evolution. Instead, we defer this check and we apply it to the final population (see line 17 of Alg. 1). The satisfiability check is performed by calling an off-the-shelf OWL reasoner. Our current implementation makes use of Pellet [17].

[6] As remarked in [12], the satisfiability check is useful only if disjointness axioms occur in the ontology. This check can be omitted (thus saving computational cost) if no disjointness axioms occur.

4 RELATED WORK

The exploitation of data mining methods for discovering hidden knowledge patterns is not new in the SW context. First proposals have been formalized in [12, 13], where solutions for discovering frequent patterns in the form of, respectively, DATALOG clauses and conjunctive queries from hybrid sources of knowledge (i.e., a rule set and an ontology) have been presented. These methods are grounded on a notion of *key*, standing for the basic entity/attribute to be used for counting elements for building the frequent patterns. Unlike these methods, our solution focuses on an ontological KB and does not require any notion of *key* and as such it is able to discover any kind of frequent hidden knowledge patterns in the ontology. A method for learning ARs from RDF datasets, with the goal of inducing a schema ontology has been proposed in [20], while a method for inducing new assertional knowledge from RDF datasets has been presented in [9]. Unlike our approach, these two methods do not take any background/ontological knowledge into account and do not exploit any reasoning capabilities. Furthermore, our solution allows to discover rules that can be directly added to the ontology, since our rules are represented in SWRL, which is not the case for the existing methods.

As regards exploiting EAs in combinations with ILP, several proposals started to appear in the literature at the beginning of the new millennium. An EA has been exploited as a wrapper around a population of ILP algorithms in [16]; alternatively, a hybrid approach combining an EA and ILP operators has been proposed in [6, 7]. A similar idea is also followed by [15, 18, 19], in which an EA is used to evolve and recombine clauses generated by a stochastic bottom-up local search heuristic. Finally, [5] built on those ideas and combined them with recent work on relational AR discovery from populated KBs in the SW [4, 9] to propose an EA for the discovery of relational AR. The rationale for using EAs as a meta-heuristic for ILP is to mitigate the combinatorial explosion generated by the inductive learning of rich representations, such as those used in DLs [4], while maintaining the quality of the results.

Our approach adopts several solutions that allow it to outperform the evolutionary method proposed in [5], as shown in Tab. 3 and 4. Specifically, in [5], the mutation operator is only applied to the worst individuals in the population, in the hope that the mutants be better than their parents. However, while it is easier to improve a poor individual by chance, such mutant can hardly compete with the best individuals. Instead, we apply mutation to all individuals independently of their fitness. This allows the EA to explore the search space around promising solutions as well.

Another difference is that our approach strictly considers closed rules (see Def. 2.8), unlike RARD and the method proposed in [5] where open rules are allowed. This restriction allows us to reduce

Table 1: Key facts about the ontological KBs used.

Ontology	# Concepts	# Roles	# Indiv.	# Declared Assertions	# Decl.+Derived Assertions
Financial	59	16	1000	3359	3814
BioPAX	40	33	323	904	1671
NTMerged	47	27	695	4161	6863

the search space and the number of returned rules while remaining competitive in terms of generated predictions (see Tab. 4).

Another reason for the better performance of our method is our use of a measure of rule confidence in the fitness function (cf. Section 3.6). Indeed, [5] uses just the head coverage of a rule as its fitness ($\text{fitness}(r) = \text{headCoverage}(r)$). Such choice only focuses on the maximization of the number of the known correct predictions without taking into account the confidence of the rule; in particular, it completely overlooks the incorrect predictions of the rule. Thanks to the PCA confidence, we can obtain rules with a low error rate. Furthermore, it is also suitable to operate under the OWA.

5 EXPERIMENTS AND RESULTS

We applied our method to the same populated ontological KBs used in [5]: Financial,[7] describing the banking domain; Biological Pathways Exchange (BioPAX)[8] Level 2 Ontology, describing biological pathway data; and New Testament Names Ontology (NTN),[9] describing named entities (people, places, and other classes) in the New Testament, as well as their attributes and relationships. Details on these ontologies are reported in Tab. 1.

The first goal of our experiments consisted in assessing the ability of the discovered rules to predict new assertional knowledge for a considered ontological KB. For that purpose, different samples of each ontology have been built for learning multi-relational ARs (as presented in Sect. 3) while the full ontology versions have been used as a testbed. Specifically, for each ontology, three samples have been built by randomly removing, respectively, 20%, 30%, and 40% of the concept assertions, according to a stratified sampling procedure.

We ran the EA presented in Sect. 3 by repeating, for each run, the sampling procedure. We performed 30 runs for each stratified sample of each ontology using the following parameter setting:

$$\begin{array}{rclrcl} n & = & 5,000; & p_{\text{mut}} & = & 5\%; \\ \text{MAX\_GENERATIONS} & = & 200; & \theta_{\text{mut}} & = & 0.2; \\ \text{MAX\_RULE\_LENGTH} & = & 10; & \tau & = & \frac{1}{5} \\ & & & \theta_f & = & 1. \end{array}$$

The charts in Figure 2 demonstrate the unfolding of the evolutionary process over 30 distinct runs. The top chart shows the growth over generations of the number of patterns having a fitness greater than θ_{fit}. The bottom chart shows the growth over generations of the average fitness of the entire population. We can observe that high-quality patterns are gradually discovered over generations.

As in [9], we applied the discovered rules to the full ontology versions and collected all predictions, that is the head atoms of the instantiated rules. Given the collected predictions, those already contained in the reduced ontology versions were discarded while the

[7] http://www.cs.put.poznan.pl/alawrynowicz/financial.owl.
[8] http://www.biopax.org/release/biopax-level2.owl.
[9] http://www.semanticbible.com/ntn/ntn-view.html.

Figure 2: The growth of population over generations

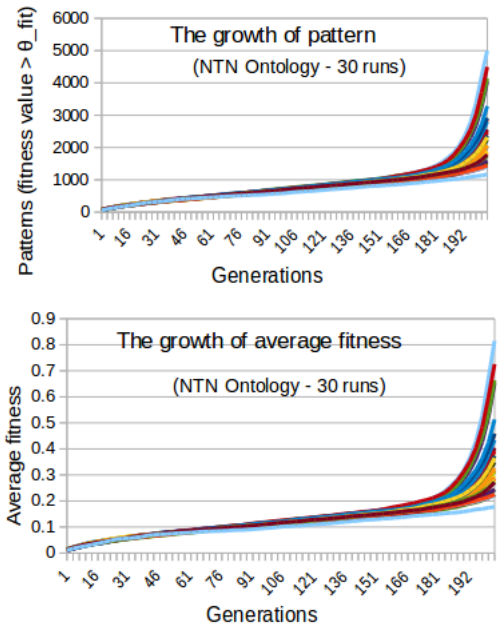


Table 2: Avg ($\pm$ st.dev.) performance on each ontology.

Ontology	Sample	Match Rate	Comm. Rate	Ind. Rate	Precision	Total # Predictions
Financial	20%	0.871 $\pm$ 0.036	0 0	0.129 $\pm$ 0.036	1.0	44,962 $\pm$ 41,949
	30%	0.855 $\pm$ 0.047	0 0	0.145 $\pm$ 0.047	1.0	39,401 $\pm$ 44,645
	40%	0.864 $\pm$ 0.039	0 0	0.136 $\pm$ 0.039	1.0	31,226 $\pm$ 33,952
BioPAX	20%	0.571 $\pm$ 0.028	0 0	0.429 $\pm$ 0.028	1.0	86,920 $\pm$ 11,691
	30%	0.59 $\pm$ 0.025	0 0	0.41 $\pm$ 0.025	1.0	79,543 $\pm$ 11,850
	40%	0.584 $\pm$ 0.031	0 0	0.416 $\pm$ 0.031	1.0	97,559 $\pm$ 13,049
NTNMerged	20%	0.632 $\pm$ 0.059	0 0	0.368 $\pm$ 0.059	1.0	3,439,660 $\pm$ 554,720
	30%	0.6 $\pm$ 0.055	0 0	0.4 $\pm$ 0.055	1.0	2,353,420 $\pm$ 477,735
	40%	0.711 $\pm$ 0.075	0 0	0.289 $\pm$ 0.075	1.0	2,899,464 $\pm$ 563,711

remaining predicted facts were considered. A prediction is assessed as *correct* if it is contained/entailed by the full ontology version and as *incorrect* if it is inconsistent with the full ontology version. Results (see Tab. 2) have been averaged over 30 different runs with the above parameter setting and have been measured in terms of *precision* (see Def. 2.19), *match rate*, *commission error rate*, and *induction rate* (see Sect. 2.2).

These results fully confirm the capability of EDMAR to discover accurate rules (precision = 1 on all samples of all ontologies considered) and, which is even more relevant, to come up with rules that induce previously unknown facts (induction rate > 0), with a very large absolute number of predictions by the standards of alternative state-of-the art approaches.

We also compared the performance of EDMAR to state-of-the-art methods which are closest to it in purpose, namely the evolutionary

Table 3: Comparison of the number of discovered rules.

Ontology	Samp.	# The total number of rules discovered			
		EDMAR	[5]	RARD	AMIE
Financial	20%	27 ± 3	94 ± 34	**177**	2
	30%	26 ± 3	86 ± 32	**181**	2
	40%	24 ± 4	78 ± 50	**180**	2
Biopax	20%	132 ± 10	144 ± 47	**298**	8
	30%	118 ± 12	188 ± 26	**283**	8
	40%	137 ± 12	159 ± 38	**272**	0
NTNMerged	20%	**1,834** ± 782	1,046 ± 593	243	1,129
	30%	**1,235** ± 495	946 ± 218	225	1,022
	40%	**1,810** ± 733	897 ± 473	239	1,063

Table 4: Comparison of the number of extracted predictions.

Ontology	Samp.	Top m	# Predictions			
			EDMAR	[5]	RARD	AMIE
Financial	20%	2	**42,575 *** ± 38,239	14,442 ± 17,280	29	208
	30%	2	**36,799** ± 41,667	29,890 ± 29,576	57	197
	40%	2	**30,263** ± 33,825	18,958 ± 21,954	85	184
Biopax	20%	8	**41,024 *** ± 7,567	2,045 ± 740	25	2
	30%	8	**39,283 *** ± 6,485	1,653 ± 779	34	2
	40%	8	**43,698 *** ± 6,524	1,704 ± 1,437	50	0
NTNMerged	20%	10	**933,248 *** ± 110,786	98,470 ± 25,261	620	420
	30%	10	**724,020 *** ± 162,851	11,940 ± 41,960	623	281
	40%	10	**828,317 *** ± 250,804	103,100 ± 38,903	625	332

method proposed by [5] and the two level-wise generate-and-test algorithms which are the multi-relational association rule discovery (RARD) method by d'Amato *et al.* [4] and AMIE [9]. Table 3 reports the average number of rules discovered by each system given each KB sample. We can observe a clear tendency for EDMAR to perform better, as compared to its competitors, the more the ontology to which it is applied is complex.

Additional comparative results are reported in Table 4. Here, given the top m rules, with m equal to 10 or equal to the number of rules discovered by AMIE, when fewer than 10 rules were discovered, the number of correct predictions generated by the top m rules discovered by each system is compared. The results reported in Table 4 corroborate the claim that EDMAR can substantially outperform the existing systems, not only in terms of rules discovered, but also (and more importantly) in terms of their predictive power. A star in the EDMAR column means that Welch's t-test on the comparison of EDMAR vs. [5] rejects the null hypothesis with a confidence level of at least 95%.

6 CONCLUSIONS

We presented an evolutionary method for discovering multi-relational ARs, coded in SWRL, from ontological KBs, to be used primarily for enriching assertional knowledge. EDMAR has been experimentally evaluated through its application to publicly available ontologies and compared to the three most relevant state-of-the-art algorithms having the same goal.

Future work will focus on two main aspects: (i) scalability, by considering experimenting our method on datasets from the Linked Data Cloud; (ii) upgrading the algorithm for running on parallel systems such as the MapReduce programming model and the Hadoop framework, in order to be able to perform big data analytics.

REFERENCES

[1] R. Agrawal, T. Imielinski, and A. N. Swami. 1993. Mining Association Rules between Sets of Items in Large Databases. In *Proc. of the Int. Conf. on Management of Data*. ACM Press, 207–216.

[2] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider (Eds.). 2003. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge Univ. Press.

[3] T. Berners-Lee, J. Hendler, and O. Lassila. 2001. The Semantic Web. *Scientific American* (2001).

[4] C. d'Amato, S. Staab, A. Tettamanzi, D. M. Tran, and F. Gandon. 2016. Ontology Enrichment by Discovering Multi-Relational Association Rules from Ontological Knowledge Bases. In *Proc. of SAC 2016*. ACM.

[5] C. d'Amato, A. Tettamanzi, and D. M. Tran. 2016. Evolutionary Discovery of Multi-relational Association Rules from Ontological Knowledge Bases. In *EKAW*. 113–128.

[6] F. Divina. 2010. *Genetic Relational Search for Inductive Concept Learning: A Memetic Algorithm for ILP*. LAP LAMBERT Academic Publishing.

[7] F. Divina and E. Marchiori. 2002. Evolutionary Concept Learning. In *GECCO 2002*, W. B. Langdon *et al.* (Ed.). Morgan Kaufmann, 343–350.

[8] N. Fanizzi, C. d'Amato, and F. Esposito. 2008. Learning with Kernels in Description Logics. In *ILP 2008*, F. Železný and N. Lavrac (Eds.). Springer, 210–225.

[9] L. Galárraga, C. Teflioudi, K. Hose, and F. Suchanek. 2013. AMIE: Association Rule Mining under Incomplete Evidence in Ontological Knowledge Bases. In *WWW '13*. ACM, 413–422.

[10] I. Horrocks and P. F. Patel-Schneider. 2004. A proposal for an OWL rules language. In *Proc. of the Int. Conf. on World Wide Web*. ACM, 723–731.

[11] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. 2004. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. (2004). http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/

[12] J. Józefowska, A. Lawrynowicz, and T. Lukaszewski. 2010. The role of semantics in mining frequent patterns from knowledge bases in description logics with rules. *Theory and Practice of Logic Programming* 10, 3 (2010), 251–289.

[13] F. A. Lisi. 2011. AL-QuIn: An Onto-Relational Learning System for Semantic Web Mining. *Int. J. of Semantic Web and Information Systems.* (2011).

[14] B. Motik, U. Sattler, and R. Studer. 2005. Query Answering for OWL-DL with rules. *Web Semantics* 3, 1 (2005), 41–60.

[15] S. Muggleton and A. Tamaddoni-Nezhad. 2008. QG/GA: a stochastic search for Progol. *Machine Learning* 70, 2-3 (2008), 121–133.

[16] P. Reiser and P. Riddle. 2001. Scaling Up Inductive Logic Programming: An Evolutionary Wrapper Approach. *Applied Intelligence* 15, 3 (2001), 181–197.

[17] E. Sirin, B. Parsia, B. Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. 2007. Pellet: A practical OWL-DL reasoner. *Web Semantics* 5, 2 (June 2007), 51–53.

[18] A. Tamaddoni-Nezhad and S. Muggleton. 2000. Searching the Subsumption Lattice by a Genetic Algorithm. In *ILP 2000*, J. Cussens and A. Frisch (Eds.). Springer, 243–252.

[19] A. Tamaddoni-Nezhad and S. Muggleton. 2002. A Genetic Algorithms Approach to ILP. In *ILP 2002*, S. Matwin and C. Sammut (Eds.). Springer, 285–300.

[20] J. Völker and M. Niepert. 2011. Statistical Schema Induction. In *ESWC'11 Proc. (LNCS)*, Vol. 6643. Springer, 124–138.