A Simulated Annealing Meta-heuristic for Concept Learning in Description Logics

Patrick Westphal[1,2][0000-0002-3855-4485], Sahar Vahdati[3,2][0000-0002-7171-169X], and Jens Lehmann[1,4][0000-0001-9108-4278]

[1] Fraunhofer Institute for Intelligent Analysis and Information Systems, Dresden, Germany
[2] Institute for Applied Informatics, Leipzig, Germany
[3] University of Bonn, Bonn, Germany

Abstract. Ontologies – providing an explicit schema for underlying data – often serve as background knowledge for machine learning approaches. Similar to ILP methods, concept learning utilizes such ontologies to learn concept expressions from examples in a supervised manner. This learning process is usually cast as a search process through the space of ontologically valid concept expressions, guided by heuristics. Such heuristics usually try to balance *explorative* and *exploitative* behaviors of the learning algorithms. While exploration ensures a good coverage of the search space, exploitation focuses on those parts of the search space likely to contain accurate concept expressions. However, at their extreme ends, both paradigms are impractical: A totally random explorative approach will only find good solutions by chance, whereas a greedy but myopic, exploitative attempt might easily get trapped in local optima. To combine the advantages of both paradigms, different meta-heuristics have been proposed. In this paper, we examine the Simulated Annealing meta-heuristic and how it can be used to balance the exploration-exploitation trade-off in concept learning. In different experimental settings, we analyse how and where existing concept learning algorithms can benefit from the Simulated Annealing meta-heuristic.

Keywords: Inductive Logic Programming (ILP) · Description Logic (DL) · Concept Learning (CL) · Meta-heuristics.

1 Introduction

The availability of vast amounts of structured data with explicit semantics offers great opportunities for analytics in downstream AI tasks. One such data source is the Linked Open Data cloud[4] which is a distributed and interlinked collection of knowledge bases expressed by means of ontologies covering many different domains. Ontologies provide means for a logic-based description of a domain of interest and allow to express *terminological* knowledge such as hierarchies of relations and *concepts* i.e. categories to classify individuals with

[4] `https://lod-cloud.net/`

common relevant features [1]. Further, knowledge bases also provide *assertional* knowledge, i.e. statements about individuals of the domain and how they relate to each other. An established formalism to express such background knowledge are Description Logics (DL) [1] which allow to formulate statements about the domain of interest in the form of axioms. One way to gain insights from such relational data is to inductively learn from positive and negative examples. Starting with a set of example individuals for a target concept, a concept description is learned s.t. it accurately covers positive examples while not covering negative examples [12]. Such concept descriptions can then serve as human and machine readable binary classifiers to classify unseen individuals w.r.t. the target concept. The idea of *concept learning* (CL) is often cast as a systematic search in the space of possible concept descriptions which is mainly aiming to maximize the example classification accuracy.

A common approach for exploring the search space is to follow a refinement-based approach which is guided by a heuristic [12]. Such approaches often take advantage of iteratively improving intermediate concept descriptions based on a given quality measure e.g. accuracy. This procedure can be seen as *Hill Climbing* where the whole refinement process is understood as a sequence of incremental changes to a candidate concept expression [16]. As many Hill Climbing methods, a concept learning algorithm might be prone to get stuck in a local optimum, which renders the algorithm *myopic*. In CL, this occurs when an intermediate candidate concept description scores too low with respect to the employed quality measure. Such concept descriptions are then ignored in further refinement iterations, but may be essential steps on the refinement path towards the optimal solution which is then never found by the algorithm. Such myopic behavior is usually caused by applying a heuristic which is greedily guiding the search process to achieve sensible solutions, while being incapable of leaving local optima or plateaus. To remedy this, different solutions have been explored in the field of mathematical optimization, however, only very few of them have been applied to CL. As one promising attempt to tackle myopia in CL, we extend existing algorithms with the established Simulated Annealing meta-heuristic. As our main contribution of this article, we propose a formalism to integrate Simulated Annealing into concept learning. Furthermore, we provide an evaluation of the resulting algorithm showing that we are able to tackle the myopia problem and achieve competitive results on synthetic and real world learning problems.

2 Related Work

Inductive Learning Systems. Different research fields have addressed *inductive* inference of general principles or patterns from specific facts considering background knowledge. Most prominently *Inductive Logic Programming* (ILP) contributes several systems implementing different strategies [14]. In CL the goals of ILP are adapted and extended using a different family of knowledge representation languages, namely Description Logics. Here, a common task is to learn concept descriptions from individuals serving as positive and negative ex-

amples. Such concept descriptions should cover the positive examples while not covering the negative ones. Several concept learning systems have been developed in this regard namely YingYang [9], DL-Foil [5,8] and the DL-Learner [2].

YingYang is one of the early systems with the main strategy of applying the *counterfactual* method, i.e. finding concepts that cover negative examples and conjunctively add the negations of such concepts to the overall solution to rule out the wrongly covered negative examples.

DL-Foil is a 'Foil-like' algorithm that applies a sequential covering approach to concept learning. It partly reuses other CL techniques but extends the binary setting of evaluating the covered positive and negative examples to a three-valued setting, also considering examples of uncertain membership when evaluating a concept description.

The DL-Learner framework is a collection of learning algorithms and strategies for CL. The most prominent algorithms are the *OWL Class Expression Learner* (OCEL) [11] and the *Class Expression Learner for Ontology Engineering* (CELOE) [10]. Inspired by OCEL and CELOE, the authors of [18] proposed the ParCEL algorithm which was subsequently implemented in the DL-Learner framework. ParCEL tries to compute multiple partial concept descriptions in parallel which are eventually combined to form the target concept. This idea is further extended, in a way similar to the counterfactuals idea, by the same authors which lead to the SPaCEL algorithm [19].

Extended Meta-heuristics Approaches. Each of the aforementioned systems employs refinement strategies to systematically explore the space of ontologically valid concept descriptions. Similar to the findings made for refinement operators in the ILP domain [17], certain properties of upward and downward refinement operators in Description Logics were investigated [6,4,12]. Starting with an initial concept description, new concepts are derived by applying refinement rules through a downward (upward) refinement operator. Usually, the refined concepts are evaluated based on a *heuristic* which guides the search process by picking promising concepts for the next refinement iteration.

This basic paradigm lends itself for comparison with the *Hill Climbing* optimization techniques. In [3] the authors relate refinement-based ILP methods to Hill Climbing and propose to apply certain meta-heuristics, namely *Beam Search*, *Look-Ahead* methods, and the introduction of *Determinate Literals*, to overcome the Hill Climbing search-inherent myopia problem. Similarly, in [17] the authors introduce *Simulated Annealing* for non-myopic ILP. However, this approach cannot be applied to concept learning without modifications. A more general investigation about the introduction of randomized restarts into the search process was performed in [20]. Whereas the core idea of randomly picking a clause from an active set of considered clauses is similar to the random choice from the built search tree in CL as proposed here, many details differ.

Also motivated by the Hill Climbing analogy, in [16,15] DL-Focl is proposed, which extends DL-Foil by meta-heuristic strategies such as *Repeated Hill Climbing*, *Look-Ahead* mechanisms, and *Tabu Search*. In this paper we extend the research on meta-heuristics in CL learning by considering the Simulated

Table 1: Terms and Notations in Description Logics (summarized from [1]).

Notation	Description
N_C, N_R, N_I	Set of all concept names, role names, and individual names respectively
A, B, C, D	Concepts (or "classes") denoting sets of individuals where A, B are atomic concepts
$\top, \bot$	Concept denoting the complete domain Δ, and the empty concept, respectively
r, s	Roles (or "object properties")
$\sqcap, \sqcup, \neg$	Concept/role constructors allowing to define the concept/role *intersection*, *union*, and *negation*, respectively
$\exists r.C, \forall r.C$	Existential and universal restriction on the role r, respectively
$C \sqsubseteq D$	Inclusion axiom, meaning that C is a sub-concept/subclass of D
a, b	Individuals of a considered domain Δ
$C(a), r(a, b)$	Class assertion stating that individual a is an instance of concept C, and role assertion stating that individuals a and b are related via role R
$\mathcal{K} \models \alpha$	Entailment of axiom α from knowledge base $\mathcal{K}$

Annealing technique to systematically adjust explorative and exploitative traits of a learning algorithm.

3 Approach

Balancing exploration and exploitation when searching solutions is one of the key aspects in designing CL algorithms. Heuristics guiding an algorithm's behavior in this respect can follow established patterns published in the meta-heuristics literature [3,17,16], or combine different, more specific strategies as in [11,10]. Especially for refinement-based CL approaches, many meta-heuristics from combinatorial optimization theory are applicable. We concentrate on the Simulated Annealing meta-heuristic and its application to concept learning using the OCEL [11] and CELOE [10] algorithms. We further introduce the notion of *Adaptive Simulated Annealing* for concept learning and provide the details about the respective Simluated Annealing extensions for OCEL and CELOE.

3.1 Notation and Preliminaries

To express knowledge of a certain domain of discourse Δ we distinguish *individuals* being elements of the domain, and concepts, or "classes", representing sets of individuals. The relations between individuals are expressed by *roles*. An overview of the basic notations is given in Table 1. We also refer to candidate solutions, or hypotheses, as well as their refinements (in an abstract or more concrete sense). For notational clarity, a single candidate solution is denoted by the Greek letter σ, whereas sets of candidate solutions are represented by the uppercase Greek letter Σ (both optionally with indexes).

3.2 Concept Learning in Description Logics

We focus on the problem of inductively learning a (usually complex) concept describing a set of observed individuals from N_I. Given the set $N^+ = \{a_1^+, a_2^+, \ldots\} \subseteq N_I$ of *positive examples*, and the set $N^- = \{a_1^-, a_2^-, \ldots\} \subseteq N_I$ of *negative examples* a target concept C_t shall be learned, such that $\mathcal{K} \models C_t(a_i^+)$ for all $a_i^+ \in N^+$, and

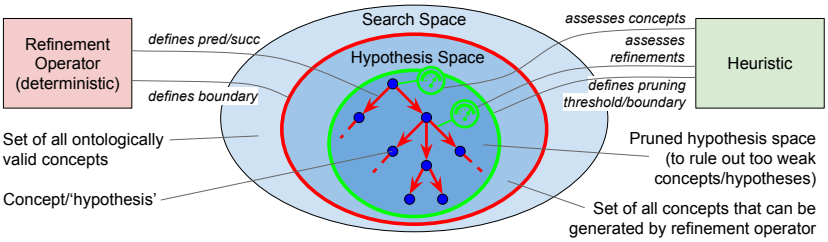

Fig. 1: Terms and Spaces in Concept Learning.

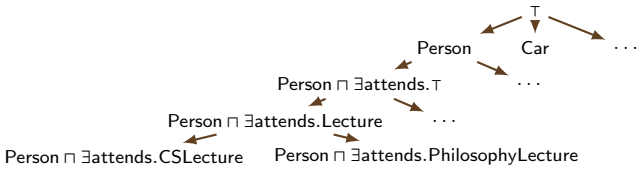

Fig. 2: Search tree of an example refinement

$\mathcal{K} \not\models C_t(a_i^-)$ for all $a_i^- \in N^-$, i.e. positive examples are entailed by the complex concept (not the negative examples). *Refinement-based* CL approaches achieve this by traversing the space of all ontologically valid concept descriptions, i.e. the *search space*. This is done in a systematic manner, e.g. by applying concept constructors and introducing existential/universal restrictions [12]. In this context, such generated concept descriptions are often called *hypotheses*. Given a *quasi-ordered* space $(\Sigma, \preceq)$, a downward (upward) refinement operator ρ is a mapping from Σ to 2^Σ such that for any $\sigma \in \Sigma$ we have that $\sigma' \in \rho(\sigma)$ implies $\sigma' \preceq \sigma$ ($\sigma \preceq \sigma'$) [10]. Depending on the properties of the refinement operator this spans a certain *hypothesis space*. Since we are extending the OCEL and CELOE algorithms we are assuming a downward refinement operator ρ which is complete and proper, as reported in [12]. Moreover, as in [12], we restrict the concept length of the hypotheses generated in each refinement step to achieve finiteness. Further, we consider downward refinement operators over the quasi-ordered space $(\mathcal{L}, \sqsubseteq)$, i.e. over concept descriptions $\mathcal{L}$ and the subsumption relation $\sqsubseteq$ s.t. for each refinement $C' \in \rho(C)$ of an input concept C it holds that $C' \sqsubseteq C$. ρ, in general, is capable of generating concepts in the DL language $\mathcal{ALCHOQ}$ with concrete roles. An example refinement process is shown in Figure 2. The set of nodes in this refinement graph can be seen as a sub-space of the hypothesis space which is dynamically extended per iteration. Moreover, the refinement operator ρ, in combination with the learning algorithm (OCEL or CELOE), eliminate redundancies up to weak equality [12,10]. Hence, the refinement graph forms a tree known as *search tree* in the literature [12,10].

This example also illustrates the problem of myopia. Imagine we have a set of individuals representing CS students as positive examples, and another set of individuals representing philosophy students as negative examples. The background knowledge base contains information about the students' courses and the course taxonomy (with the axioms CSLecture $\sqsubseteq$ Lecture and PhilosophyLecture $\sqsubseteq$

Algorithm 1: Basic concept learning algorithm

```
   Result: Best hypothesis σ_best
 1 SearchTree ← initial empty search tree
 2 σ_best ← initial candidate expression
 3 SearchTree.add(σ_best)
 4 repeat
 5 |   # Find hypothesis in the whole search tree with highest score according to heuristc χ
 6 |   for σ_tmp in SearchTree do
 7 |   |   if χ(σ_tmp) > χ(σ_best) then
 8 |   |   |   σ_best ← σ_tmp
 9 |   Σ_{ρ,σ_best} ← ρ(σ_best)
10 |   for σ_new in Σ_{ρ,σ_best} do
11 |   |   if σ_new is not too weak then
12 |   |   |   SearchTree.add(σ_new)
13 until stop criterion is satisfied;
```

Lecture). Now, if we want to learn a concept description for CS students, a valid hypothesis telling CS students apart from philosophy students would be Person $\sqcap$ $\exists$attends.CSLecture. However, during the refinement process, the hypotheses that are generated in all but the last iteration cover positive and negative examples to the same extent. Thus, a greedy and myopic learning algorithm could prune the refinement chain in earlier iterations, e.g. after refining to Person $\sqcap$ $\exists$attends.$\top$ as it does not bring any quality improvements in terms of the concept description's classification accuracy.

A *heuristic*, being the component in a CL setting that takes care of the assessment of generated hypotheses, assigns numeric quality scores. Thus, it makes two hypotheses comparable, and imposes an order on the elements of the hypothesis space. Moreover, a heuristic may mark certain refinements as 'too weak' for further consideration, thus 'cutting' refinement edges in the search tree which will then never be followed. This quality-based pruning further restricts the considered part of the hypothesis space as sketched in Figure 1. There are several actual quality metrics proposed for the binary [10] and three-valued [16] CL settings. Further, a heuristic might as well consider structural properties of a concept, e.g. penalizing overly complex hypotheses. In essence, the heuristic of a CL algorithm defines the exploitation strategy. However, it may also contain explorative traits, e.g. when penalizing hypotheses that were refined already too often, thus forcing the algorithm to explore other regions of the hypothesis space. In the CL model we introduced here, the heuristic is the core component to balance the exploration vs. exploitation trade-off. It will be denoted as a function $\chi : \Sigma \mapsto \mathbb{R}$ mapping hypotheses to real numbers in the following.

Having the notions of a refinement and a quality score in place, the analogies to the Hill Climbing strategy can be illustrated. In essence, Hill Climbing can be explained as iteratively assessing new candidate solutions retrieved by refining a given candidate and adopt new candidates if the quality score improved [13] (cf. Algorithm 2). To compare this with concept learning, a basic algorithm description abstracted from OCEL and CELOE is provided in Algorithm 1. One difference between refinement-based CL and Hill Climbing is that the refinement operator in concept learning, ρ, is defined to return a set of candidates,

whereas the refinement in Hill Climbing, ρ_{hc}, returns just one refined hypothesis per invocation. Another difference is that a Hill Climbing algorithm would just keep track of the current best candidate hypothesis, whereas the CL algorithm presented here keeps track of the whole pruned sub space of the hypothesis space, explored so far. It is stored in the form of a search tree making use of the predecessor/successor relations imposed by the refinement operator. While this comes with a certain memory overhead, the search tree approach is more flexible, especially when the assessment scores of hypotheses may be dynamic. Also, since this allows to get a good overview of how well a certain region of the hypothesis space is explored (e.g. in terms of the number of the child nodes, tree depth etc.) this is of particular use for a balanced exploration strategy. In this regard, even though we kept the variable name $\sigma_{\mathbf{best}}$ to ease the comparison with Algorithm 2, the hypothesis refined in a certain iteration does not have to be the *best*, e.g. in terms of its classification quality, but should rather be understood as the best choice in terms of the "exploration-exploitation" strategy of the applied heuristic. Accordingly, neither OCEL nor CELOE strictly follow a greedy (and thus myopic) Hill Climbing approach but apply such balancing strategies. In their default settings both algorithms have built-in measures to escape local optima. However, the presented concept learning model can still be used to implement a pure greedy Hill Climbing-based approach by defining the heuristic accordingly. To do so, in Section 4 we configure the OCEL and CELOE algorithm in the most greedy way to show the potentials of a more balanced heuristic in terms of exploration. In the following, we will introduce the established Simulated Annealing meta-heuristic which dynamically adjusts the explorative and exploitative traits of a learning algorithm.

3.3 Simulated Annealing

When applying a Hill Climbing strategy, a refinement will only be applied if it improves the quality of the hypothesis w.r.t. a heuristc χ. Then every move will go 'uphill' which makes Hill Climbing a purely exploitative approach leading to myopia in case of local optima. The Simulated Annealing meta-heuristic dynamically adds explorative facets to the Hill Climbing procedure in a controlled way. This means, that with some probability, a move might go 'downhill', irrespective of the check whether the quality of a refined hypothesis $\rho_{\text{hc}}(\sigma)$ improved over the initial hypothesis σ. The probability of allowing such downhill moves is given by $p = e^{\frac{\chi(\rho_{\text{hc}}(\sigma))-\chi(\sigma)}{t}}$. This probability depends on the magnitude of the quality degradation of $\rho_{\text{hc}}(\sigma)$ over σ, where greater degradations make it less likely to take a downhill move. On the other hand, p depends on the parameter t which is usually interpreted as the *temperature* of the Simulated Annealing process. Higher temperatures increase the probability of taking downhill steps (which favors exploration), whereas lower temperatures decrease the probability p making the process more Hill Climbing-like. Simulated Annealing starts with a high temperature and cools down during the execution, turning the initially fully explorative strategy into a fully exploitative one. A sketch of the whole procedure is given in Algorithm 3. Besides cooling down one 'degree' at each

Algorithm 2: Hill Climbing

```
Result: The best solution σ
σ ← initial candidate solution
repeat
    σ_new ← ρ_hc(σ) ;
    if χ(σ_new) > χ(σ) then
        σ ← σ_new
until σ is the ideal solution or we
  run out of time;
```

Algorithm 3: Simulated Annealing

```
Result: The best solution σ_best
t ← high initial temperature
σ ← initial candidate solution
σ_best ← σ
repeat
    σ_new ← ρ_hc(σ));
    p ← e^((χ(σ_new)−χ(σ))/t)
    if χ(σ_new) > χ(σ) or rand(0,1)< p then
        σ ← σ_new
    t ← t − 1
    if χ(σ) > χ(σ_best) then
        σ_best ← σ
until σ_best is the ideal solution, we run out
  of time, or t = 0;
```

iteration (as shown in Algorithm 3), other cooling schedules are possible, too.

If we relax the stop criterion of t being equal to 0, it might be the case that, after being cooled down, the algorithm gets stuck in a local optimum. To remedy this, the Simulated Annealing approach can be extended to 'heat up' again, whenever there are no improvements. We will call this *Adaptive Simulated Annealing* and will investigate its performance impact in Section 4. For this adaptive attempt further parameters need to be adjusted. First, analogous to the cooling schedule, there should be a strategy how fast to heat up again. Besides this, one needs to declare when to heat up, i.e. after how many iterations without improvement the cooling process will be reverted to a more explorative strategy.

3.4 Simulated Annealing in Concept Learning

The application of Simulated Annealing to concept learning follows the same goal of balancing exploration and exploitation during the learning process in a systematic way. The core intuition is to be more explorative when the hypotheses are of lower quality and gradually switch to a more greedy strategy when the quality of the learned concepts increases. Other than the aforementioned Hill Climbing approach, the refinement operator we consider in our CL setting returns a *set* of hypotheses, not just a single refinement. Therefore, introducing explorative behavior goes beyond probabilistically accepting or rejecting a single refinement. Instead, we refer to the search tree to introduce probabilistic, explorative moves. To pick a node from the search tree for applying the refinement operator, we usually choose the best one w.r.t. the algorithm's search heuristic (line 6-8 in Algorithm 1). In the proposed Simulated Annealing approach to concept learning, this is replaced by randomly choosing an arbitrary node with the probability p. The definition of p, however, also needs to be adjusted. Since we no longer have one hypothesis and just one refined version of it, we cannot refer to the quality improvement induced by the refinement step, anymore. To follow the intuition for introducing Simulated Annealing into the CL process, we base the value of p on the quality of the currently best hypothesis $\chi(\sigma_{\text{best}})$. Accordingly, p is adjusted to $p = e^{\frac{-\chi(\sigma_{\text{best}})}{t}}$. In its basic form, the proposed approach will decrease t after each iteration (i.e. the repeat loop in line 4-13 of Algorithm 1).

The value of t is not considered as a stop criterion. To introduce an adaptive behavior for each iteration, we keep track whether the quality score of the best hypothesis, i.e. $\chi(\sigma_{\text{best}})$, improved. After a certain number of iterations without improvements, the temperature t is increased again, pushing the algorithm back to a more explorative behavior.

4 Empirical Evaluation

For empirical evaluations, we implemented our Simulated Annealing approach as part of the DL-Learner framework. In particular, we extended the OCEL and CELOE algorithms to investigate the potential of this meta-heuristic in CL. A new heuristic was designed realising the Simulated Annealing strategy as introduced in Subsection 3.4. OCEL and CELOE can be run in a *pure* or *adaptive* Simulated Annealing setting. Both versions are evaluated in our experiments. The source code of the implemented algorithms is freely available in the *feature/extended-metaheuristics* branch of the DL-Learner GitHub project[5]. In the following, we explain the evaluation setting and discuss the results.

4.1 Evaluation Setup

Two types of evaluation setups have been considered: 1) a setting for evaluating an algorithm's capabilities of tackling the myopia problem, 2) a setting for providing insights into the performance of the CL approaches on real world learning problems. For the first part, we developed a dataset generator which can create knowledge bases in OWL of arbitrary size (in terms of the defined classes, object properties, datatype properties and individuals). For each dataset, a target concept description $C_{\text{target}} \equiv \exists r_1.(\cdots \exists r_n.C_+)\cdots)$ is generated, where all r_i (with $1 \leq i \leq n$) and $C_+ \in N_C$ are chosen randomly. The parameter n defines the *(nesting) depth* of the nested existential restriction. Further, the generator declares a defined number of positive and negative examples, as well as additional individuals being neither part of the positive nor part of the negative examples. The dataset generator takes care of creating axioms such that all positive examples will be instances of the target concept. Moreover, the atomic class filler concept C_+ has a sibling class C_-, such that all negative examples are instances of $\exists r_1.(\cdots \exists r_n.C_-)\cdots)$. Accordingly, all hypotheses on the refinement chain up the second last step will cover positive and negative examples to the same extent. We generated three learning scenarios for each depth $n \in \{2, 3, 4\}$. The datasets are available for download on the dataset generator project page on GitHub[6]. All datasets have 50 classes, 20 object properties, 10 data properties, as well as 50 positive and 50 negative example individuals. Moreover, the total number of individuals is 300 per each dataset for depth 2, 400 for each dataset of depth 3,

[5] `https://github.com/SmartDataAnalytics/DL-Learner/tree/feature/extended-metaheuristics`

[6] `https://github.com/patrickwestphal/learning_scenario_generators/releases/tag/v0.1.0`

and 500 for each dataset of depth 4. The number of axioms for depth 2 datasets is 989, 985, and 986, respectively. For depth 3, the generated datasets have an axiom count of 1297, 1348, and 1371, respectively, and for depth 4 it is 1738, 1705, and 1694. We refer to this part of the evaluation as *synthetic datasets*. In the second part of our evaluation we used the datasets provided by the SML-Bench benchmarking system[7]. The properties of the respective datasets and learning problems are discussed in [21].

In the evaluation, we compare OCEL and CELOE with different configurations. As a baseline for both of the algorithms we configured OCEL and CELOE to be as greedy (and thus, myopic) as possible. Please note that the applied settings still will not make OCEL and CELOE purely greedy algorithms but still leave some measures against myopic behavior in place which cannot be disabled. We will refer to these baseline settings as OCEL (greedy) and CELOE (greedy).

We compare these two base line systems with their respective versions having the Simulated Annealing extensions in place. The start temperature for t is set to 2000 for each system. We configure them to be evaluated in two different settings: One configuration using the 'pure' Simulated Annealing method (OCEL SA, CELOE SA), and one version applying Adaptive Simulated Annealing (OCEL ASA, CELOE ASA). For OCEL ASA and CELOE ASA we set the *reHeatThreshold* to 2, which means that the temperature will be increased after two iterations without improvement.

For comparison against the algorithms in their default settings we also evaluated OCEL (default) and CELOE (default). For all OCEL versions we set the required *noisePercentage* parameter to 35, which makes OCEL accept a candidate expression even if 35% of the examples were misclassified. This parameter was set to a higher value to make OCEL accept and report hypotheses even if they are quite weak, instead of just returning an error message saying that no concept could be learned that complies with the (stricter) noise setting. We could not compare with other concept learning systems, e.g. those presented in [8,7,15], since, to the best of our knowledge, the respective implementations are tied to pre-defined evaluation scenarios and it would require a considerable refactoring effort to make them run standalone. All experiments were executed on a machine with 2 Intel Xeon 'Broadwell' CPUs with 8 cores running at 2.1GHz with 128GB of RAM using the SML-Bench benchmark executor.

4.2 Results

Synthetic Datasets The experiments performed on the synthetic datasets allow us to investigate the influence of the allotted execution time and the complexity of the target concept to learn. Whereas it is expected that a greater execution time will lead to better results, we are also interested in the learning behavior of the different algorithms having only a restricted time budget. We

[7] `https://github.com/SmartDataAnalytics/SML-Bench/tree/updates/learningtasks`

Table 2: Evaluation results on synthetic datasets with a **nesting depth of 2**. Reported are the average accuracy and its standard deviation (top), as well as average F_1-score and its standard deviation (bottom) of the 10-fold cross validation across all three datasets.

Runtime	OCEL (default)	OCEL (greedy)	OCEL SA	OCEL ASA	CELOE (default)	CELOE (greedy)	CELOE SA	CELOE ASA
5s	0.76 ±0.17	0.76 ±0.17	0.71 ±0.17	0.71 ±0.17	1.00 ±0.00	0.50 ±0.00	0.50 ±0.00	0.50 ±0.00
10s	0.76 ±0.17	0.76 ±0.17	0.71 ±0.17	0.71 ±0.17	1.00 ±0.00	0.50 ±0.00	0.67 ±0.24	0.67 ±0.24
30s	0.76 ±0.17	0.76 ±0.17	0.73 ±0.17	0.73 ±0.18	1.00 ±0.00	0.50 ±0.00	0.68 ±0.24	0.68 ±0.24
60s	0.76 ±0.17	0.73 ±0.16	0.73 ±0.18	0.70 ±0.16	1.00 ±0.00	0.50 ±0.00	1.00 ±0.00	1.00 ±0.00
Runtime	OCEL (default)	OCEL (greedy)	OCEL SA	OCEL ASA	CELOE (default)	CELOE (greedy)	CELOE SA	CELOE ASA
5s	0.61 ±0.32	0.61 ±0.32	0.53 ±0.32	0.53 ±0.32	1.00 ±0.00	0.67 ±0.00	0.67 ±0.00	0.67 ±0.00
10s	0.61 ±0.32	0.61 ±0.32	0.52 ±0.32	0.52 ±0.32	1.00 ±0.00	0.67 ±0.00	0.78 ±0.16	0.78 ±0.16
30s	0.61 ±0.32	0.61 ±0.32	0.56 ±0.32	0.56 ±0.33	1.00 ±0.00	0.67 ±0.00	0.79 ±0.16	0.79 ±0.16
60s	0.61 ±0.32	0.56 ±0.33	0.55 ±0.34	0.52 ±0.31	1.00 ±0.00	0.67 ±0.00	1.00 ±0.00	1.00 ±0.00

compared all OCEL and CELOE variants on the synthetic datasets with different nesting depths n. We chose nesting depths of 2, 3, and 4. With higher nesting depths the performance of all the evaluated learning algorithms dropped and rendered a comparison rather meaningless. We performed our evaluation in a 10-fold cross validation setting on each of three different random learning scenarios we generated per nesting depth and report the overall average accuracy and F_1-score of the best solutions found together with their standard deviations. We repeated the evaluation run with different allotted execution times ranging from $5s$ to $60s$. The results are discussed for each nesting depth.

Nesting depth = 2 (Table 2). With a nesting depth of 2 the target concept description to learn has the shape of a doubly nested existential restriction $\exists r_1.(\exists r_2.C_+)$. This expression is still simple enough to favor CELOE's learning strategy of finding simple and human-readable concept descriptions. It can be seen that the explorative trait of the Simulated Annealing variants seemingly come as an overhead here, that does not pay off unless a long enough execution time is granted. The OCEL variants perform worse than CELOE (default) and establish a middle ground with accuracies around 75%. It can be seen that especially OCEL (default) and OCEL (greedy) behave very similar in terms of the susceptibility to execution time constraints. This also suggests that, other than CELOE (greedy), the OCEL (greedy) still has enough built-in explorative characteristics that allow it to overcome myopia irrespective of the settings we applied. This also holds for the following experiments which renders OCEL (greedy) rather unsuitable as a myopic base line for OCEL SA and OCEL ASA. However, we still kept it for reference.

Nesting depth = 3 (Table 3). With the more complex structure of the target concept description $\exists r_1.(\exists r_2.(\exists r_3.C_+))$ the CELOE variants fall behind in terms of accuracy, as expected due to their bias for simpler concepts. The numbers still show a considerable advantage of applying the Simulated Annealing meta-heuristic in terms of accuracy and give the overall best F_1-scores for this experiment which shows the potential of adding further explorative traits to

Table 3: Evaluation results on synthetic datasets with a **nesting depth of 3**. Reported are the average accuracy and its standard deviation (top), as well as average F_1-score and its standard deviation (bottom) of the 10-fold cross validation across all three datasets.

Runtime	OCEL (default)	OCEL (greedy)	OCEL SA	OCEL ASA	CELOE (default)	CELOE (greedy)	CELOE SA	CELOE ASA
5s	0.74 ±0.22	0.51 ±0.16	0.79 ±0.27	0.76 ±0.27	0.52 ±0.06	0.50 ±0.04	0.68 ±0.23	0.68 ±0.23
10s	0.74 ±0.22	0.72 ±0.24	0.80 ±0.27	0.76 ±0.27	0.52 ±0.06	0.50 ±0.04	0.68 ±0.23	0.68 ±0.23
30s	0.74 ±0.22	0.74 ±0.22	0.78 ±0.22	0.77 ±0.23	0.52 ±0.06	0.50 ±0.04	0.68 ±0.23	0.68 ±0.23
60s	0.74 ±0.22	0.73 ±0.22	0.78 ±0.22	0.76 ±0.23	0.52 ±0.06	0.50 ±0.04	0.69 ±0.23	0.69 ±0.23

Runtime	OCEL (default)	OCEL (greedy)	OCEL SA	OCEL ASA	CELOE (default)	CELOE (greedy)	CELOE SA	CELOE ASA
5s	0.63 ±0.31	0.36 ±0.17	0.73 ±0.32	0.71 ±0.32	0.67 ±0.03	0.66 ±0.03	0.78 ±0.16	0.78 ±0.16
10s	0.63 ±0.31	0.62 ±0.32	0.74 ±0.32	0.71 ±0.32	0.67 ±0.03	0.66 ±0.03	0.78 ±0.16	0.78 ±0.16
30s	0.63 ±0.31	0.63 ±0.31	0.70 ±0.31	0.73 ±0.28	0.67 ±0.04	0.66 ±0.03	0.78 ±0.16	0.78 ±0.16
60s	0.63 ±0.31	0.61 ±0.31	0.69 ±0.31	0.70 ±0.30	0.67 ±0.04	0.66 ±0.03	0.78 ±0.16	0.78 ±0.16

CELOE's heuristic. However, it can be seen, that the adaptive setting seemingly hardly influences the outcomes of CELOE ASA in comparison with CELOE SA.

The OCEL variants in general perform better in terms of accuracy and provide a similar F_1-score compared to CELOE (default) and CELOE (greedy). The execution time constraints do not influence OCEL (default) but drastically do so in case of the other OCEL variants. Especially in case of very short execution times OCEL (greedy), OCEL SA, and OCEL ASA fail to find solutions for a considerable number of folds across all datasets. Accordingly, the numbers provided by SML-Bench might be misleading, as it only gives the averages of the folds where a solution was found. In Table 3 we marked these cases in brown. For the Simulated Annealing variants this shows that a sufficiently long runtime is needed to benefit from their explorative nature. With increased execution times, however, this 'exploration penalty' pays off and OCEL SA and OCEL ASA outperform the other OCEL variants both in terms of accuracy and F_1-score. Furthermore in case of OCEL we can see an influence of the adaptive setting of OCEL ASA which gives an improved F_1-score in our experiment.

Overall, this scenario seems to mark the sweet spot as the target concept description to learn is complex enough to gain advantage from additional explorative characteristics during the learning phase. Whereas runtime restrictions do not matter for the CELOE variants, OCEL's Simulated Annealing adaptions seem to need a certain amount of time to guarantee that an acceptable solution (w.r.t. to the configured noise setting mentioned above) can be found.

Nesting depth = 4 (Table 4). With a nesting depth of 4, i.e. the target concept structure $\exists r_1.(\exists r_2.(\exists r_3.(\exists r_4.C_+)))$, the performance of most of the examined algorithm decreased. The experiment seems to mark a tipping point especially for the OCEL variants as none of them managed to find acceptable solutions (given their configured noise settings) for all the folds across the datasets. Accordingly, the results should be interpreted with caution as they might relate to different subsets of the training and test folds. For CELOE in particular the accuracy and F_1-score of the Simulated Annealing variants dropped compared to the nesting

Table 4: Evaluation results on synthetic datasets with a **nesting depth of 4**. Reported are the average accuracy and its standard deviation (top), as well as the average F_1-score and its standard deviation (bottom) of the 10-fold cross validation across all three datasets.

Runtime	OCEL (default)	OCEL (greedy)	OCEL SA	OCEL ASA	CELOE (default)	CELOE (greedy)	CELOE SA	CELOE ASA
5s	0.61 ±0.18	0.60 ±0.18	0.59 ±0.19	0.62 ±0.13	0.54 ±0.11	0.49 ±0.03	0.53 ±0.10	0.53 ±0.10
10s	0.65 ±0.14	0.65 ±0.10	0.60 ±0.13	0.60 ±0.13	0.54 ±0.11	0.49 ±0.03	0.53 ±0.10	0.53 ±0.10
30s	0.61 ±0.18	0.62 ±0.18	0.62 ±0.13	0.60 ±0.13	0.53 ±0.11	0.49 ±0.03	0.53 ±0.11	0.53 ±0.11
60s	0.61 ±0.18	0.60 ±0.18	0.59 ±0.19	0.62 ±0.13	0.53 ±0.11	0.49 ±0.03	0.53 ±0.11	0.53 ±0.11

Runtime	OCEL (default)	OCEL (greedy)	OCEL SA	OCEL ASA	CELOE (default)	CELOE (greedy)	CELOE SA	CELOE ASA
5s	0.50 ±0.28	0.48 ±0.28	0.53 ±0.26	0.55 ±0.19	0.68 ±0.08	0.66 ±0.02	0.67 ±0.08	0.67 ±0.08
10s	0.57 ±0.20	0.54 ±0.18	0.58 ±0.16	0.58 ±0.16	0.68 ±0.08	0.66 ±0.02	0.67 ±0.08	0.67 ±0.08
30s	0.50 ±0.28	0.53 ±0.26	0.55 ±0.19	0.57 ±0.16	0.67 ±0.08	0.66 ±0.02	0.67 ±0.08	0.67 ±0.08
60s	0.50 ±0.28	0.48 ±0.28	0.53 ±0.26	0.55 ±0.19	0.67 ±0.08	0.66 ±0.02	0.67 ±0.08	0.67 ±0.08

depth of 3. This suggests that explorative behavior does not bring any advantages in a complex setting like this. Judging from the results of OCEL and CELOE in their default settings, this might be due to their exploitative strategies which are unfit for such a scenario, but which are essential for a balanced learning approach taking advantage of both exploration *and* exploitation. Accordingly, this suggests that a tailored heuristic would be needed to manage such scenarios, which is beyond the scope of this work.

Overall, however, in this experimentation setting we could show that the existing learning algorithms OCEL and CELOE can benefit from the Simulated Annealing meta-heuristic, especially in more complex cases which require a non-myopic search approach. In the following subsection we will examine the performance of the Simulated Annealing-based approaches on the real-world learning tasks provided by the SML-Bench framework.

SML-Bench The experiments performed with the SML-Bench dataset library will give an overview of how well the proposed Simulated Annealing extensions work on real world problems. We chose all the datasets evaluated in [21] and granted a maximum execution time of 3 minutes. (Longer execution times gave no substantial improvements in terms of the results.) Overall it can be seen that the Simulated Annealing variants are competitive w.r.t. their base algorithms. In some cases, namely *carcinogenesis/1*, *mutagenesis/42* and *premierleague/1*, the time spent on exploration seems to cause that OCEL SA and OCEL ASA could not find acceptable solutions for all folds that comply with their noise settings. However, in case of *carcinogenesis/1* OCEL in its standard settings even failed on all but one fold which suggests that this not a problem inherent to the Simulated Annealing approach. We marked these cases in brown in Table 5. The only dataset where the Simulated Annealing variants did not provide any benefits is *mammographic/1*. Here, the base algorithms outperform the extensions in terms of accuracy, as well as F_1-score. In all other cases we can see improvements, either for OCEL or CELOE. Except for the *premierleague/1* and *pyrimidine/1* datasets OCEL SA and OCEL ASA outperform OCEL (default). It seems, though, that

Table 5: Average accuracy and its standard deviation (top), and average F_1-score and its standard deviation (bottom) of 10-fold cross validation run on SML-Bench datasets

Learning problem	OCEL (default)	OCEL (greedy)	OCEL SA	OCEL ASA	CELOE (default)	CELOE (greedy)	CELOE SA	CELOE ASA
carc./1	0.23 ±0.00	0.66 ±0.18	0.66 ±0.18	0.64 ±0.24	0.55 ±0.02	0.54 ±0.01	0.55 ±0.02	0.55 ±0.02
hepat./1	0.68 ±0.08	0.68 ±0.08	0.71 ±0.07	0.71 ±0.07	0.49 ±0.06	0.41 ±0.01	0.47 ±0.05	0.47 ±0.05
lymph./1	0.73 ±0.12	0.73 ±0.12	0.73 ±0.12	0.73 ±0.12	0.70 ±0.15	0.77 ±0.11	0.76 ±0.15	0.76 ±0.15
mam./1	0.82 ±0.05	0.82 ±0.05	0.77 ±0.08	0.77 ±0.08	0.49 ±0.02	0.46 ±0.00	0.46 ±0.00	0.46 ±0.00
mut./42	0.49 ±0.34	0.55 ±0.36	0.71 ±0.10	0.75 ±0.04	0.94 ±0.13	0.30 ±0.07	0.90 ±0.20	0.90 ±0.20
nctrer/1	0.80 ±0.09	0.80 ±0.09	0.82 ±0.11	0.82 ±0.11	0.59 ±0.03	0.59 ±0.01	0.61 ±0.04	0.61 ±0.04
prem./1	0.85 ±0.10	*no results*	0.83 ±0.15	0.83 ±0.15	0.98 ±0.05	0.49 ±0.02	0.64 ±0.25	0.64 ±0.25
pyrim./1	0.85 ±0.24	0.85 ±0.24	0.75 ±0.26	0.75 ±0.26	0.83 ±0.17	0.50 ±0.00	0.88 ±0.13	0.88 ±0.13

Learning problem	OCEL (default)	OCEL (greedy)	OCEL SA	OCEL ASA	CELOE (default)	CELOE (greedy)	CELOE SA	CELOE ASA
carc./1	0.15 ±0.00	0.65 ±0.20	0.65 ±0.20	0.62 ±0.27	0.71 ±0.01	0.70 ±0.01	0.71 ±0.01	0.71 ±0.01
hepat./1	0.53 ±0.14	0.53 ±0.13	0.58 ±0.26	0.58 ±0.26	0.61 ±0.03	0.58 ±0.01	0.61 ±0.02	0.61 ±0.02
lymph./1	0.76 ±0.10	0.76 ±0.10	0.76 ±0.10	0.76 ±0.10	0.78 ±0.10	0.82 ±0.09	0.81 ±0.11	0.81 ±0.11
mam./1	0.78 ±0.08	0.78 ±0.08	0.73 ±0.12	0.73 ±0.12	0.64 ±0.01	0.63 ±0.00	0.63 ±0.00	0.63 ±0.00
mut./42	0.25 ±0.42	0.32 ±0.43	0.10 ±0.25	0.17 ±0.31	0.93 ±0.14	0.46 ±0.08	0.90 ±0.16	0.90 ±0.16
nctrer/1	0.84 ±0.06	0.84 ±0.06	0.87 ±0.07	0.87 ±0.07	0.74 ±0.01	0.74 ±0.01	0.75 ±0.02	0.75 ±0.02
prem./1	0.81 ±0.13	*no results*	0.83 ±0.16	0.83 ±0.16	0.98 ±0.05	0.66 ±0.02	0.76 ±0.17	0.76 ±0.17
pyrim./1	0.84 ±0.22	0.84 ±0.22	0.67 ±0.38	0.67 ±0.38	0.84 ±0.15	0.67 ±0.00	0.89 ±0.13	0.89 ±0.13

the Adaptive Simulated Annealing approach provides very similar results to the 'pure' variant. This would suggest, that switching back to a more explorative strategy does not bring great advantages over keeping the exploitative strategy in OCEL. This picture is even more clear in case of CELOE SA and CELOE ASA which always provided the same numbers throughout all experiments (incl. those on the synthetic datasets). Nonetheless, CELOE's Simulated Annealing variants could prove better or equal to CELOE (default) in most of the cases. Overall, even though the Simulated Annealing variants proved to be better suited to learn the complex target expressions in the synthetic experiments this does not mean that they usually will produce more complex hypotheses. The results of the SML-Bench experiments rather show that while the learned expressions do differ, there are no such tendencies towards more complex class expressions.

5 Conclusions

In this paper we proposed an extension for two established concept learning algorithms from the DL-Learner framework based on the Simulated Annealing meta-heuristic. The design of the algorithm extensions is motivated by the goal to overcome the myopia problem by means of a systematic strategy for balancing the exploration and exploitation traits of a concept learning algorithm. On dedicated synthetic datasets we showed that our approach does indeed outperform OCEL and CELOE in more greedy, exploitation-oriented configurations, as well as in their default settings. We could further prove the competitiveness of our Simulated Annealing approach on real world concept learning problems privided by the SML-Bench benchmarking framework.

In future work, we will investigate settings for improving the Adaptive Simulated Annealing, especially for the CELOE algorithm where the 're-heat strategy' only had a minor impact. Further, we will examine other options to decide whether to focus more on explorative or exploitative search. Besides this, gained insights will also help to investigate related meta-heuristics not evaluated on the concept learning setting, yet.

References

1. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook. Cambridge University Press (2007)
2. Bühmann, L., Lehmann, J., Westphal, P.: DL-Learner – a framework for inductive learning on the Semantic Web. Web Semantics **39**, 15–24 (2016)
3. Castillo, L.P., Wrobel, S.: A comparative study on methods for reducing myopia of hill-climbing search in multirelational learning. In: ICML '04. ACM (2004)
4. Esposito, F., Fanizzi, N., Iannone, L., Palmisano, I., Semeraro, G.: Knowledge-intensive induction of terminologies from metadata. In: ISWC 2004 (2004)
5. Fanizzi, N., d'Amato, C., Esposito, F.: DL-FOIL – concept learning in description logics. In: ILP 2008. pp. 107–121 (2008)
6. Fanizzi, N., Ferilli, S., Iannone, L., Palmisano, I., Semeraro, G.: Downward refinement in the $\mathcal{ALN}$ description logic. In: HIS'04. pp. 68–73. IEEE (2005)
7. Fanizzi, N., Rizzo, G., d'Amato, C.: Boosting DL concept learners. In: The Semantic Web – 16th International Conference. pp. 68–83. Springer (2019)
8. Fanizzi, N., Rizzo, G., d'Amato, C., Esposito, F.: DLFoil: Class expression learning revisited. In: EKAW 2018. pp. 98–113
9. Iannone, L., Palmisano, I., Fanizzi, N.: An algorithm based on counterfactuals for concept learning in the Semantic Web. Applied Intelligence **26**(2), 139–159 (2007)
10. Lehmann, J., Auer, S., Bühmann, L., Tramp, S.: Class expression learning for ontology engineering. Journal of Web Semantics **9**(1), 71–81 (2011)
11. Lehmann, J., Hitzler, P.: A refinement operator based learning algorithm for the ALC description logic. In: ILP 2008. pp. 147–160
12. Lehmann, J., Hitzler, P.: Concept learning in description logics using refinement operators. Machine Learning Journal **78**(1-2), 203–250 (2010)
13. Luke, S.: Essentials of Metaheuristics. Lulu, 2nd edn. (2013)
14. Muggleton, S., Watanabe, H. (eds.): Latest Advances in Inductive Logic Programming. World Scientific (2014)
15. Rizzo, G., Fanizzi, N., d'Amato, C.: Class expression induction as concept space exploration: From DL-Foil to DL-Focl. FGCS **180**, 256–272 (2020)
16. Rizzo, G., Fanizzi, N., d'Amato, C., Esposito, F.: A framework for tackling myopia in concept learning on the web of data. In: EKAW 2018. pp. 338–354
17. Serrurier, M., Prade, H.: Improving inductive logic programming by using simulated annealing. Information Sciences **178**(6), 1423–1441 (2008)
18. Tran, A.C., Dietrich, J., Guesgen, H.W., Marsland, S.: An approach to parallel class expression learning. In: RuleML 2012. pp. 302–316. Springer (2012)
19. Tran, A.C., Dietrich, J., Guesgen, H.W., Marsland, S.: Parallel symmetric class expression learning. Journal of Machine Learning Research **18**(1), 2145–2178 (2017)
20. Železný, F., Srinivasan, A., Page, C.D.: Randomised restarted search in ILP. Machine Learning **64**(1-3), 183–208 (2006)
21. Westphal, P., Bühmann, L., Bin, S., Jabeen, H., Lehmann, J.: SML-Bench – a benchmarking framework for structured machine learning. SWJ **10**(2) (2019)