An Agent-Based Learning Approach for Finding and Exploiting Heuristics in Unknown Environments

Daan Apeldoorn, Gabriele Kern-Isberner
Technische Universität Dortmund
daan.apeldoorn@tu-dortmund.de, gabriele.kern-isberner@cs.tu-dortmund.de

Abstract

This paper presents an agent-based hybrid symbolic/sub-symbolic learning approach as a basic model for the human ability of quickly recognizing and exploiting heuristic rules in an initially unknown environment. Instead of mechanically iterating a task until the optimal policy was found (as usually done by common Reinforcement Learning techniques), the agent finds heuristics expressed by symbolic knowledge bases which allow for an intelligent exploitation of knowledge gained from few experiences. For this purpose, a measure is proposed which allows an agent to decide on its own at which point during the learning process, its decisions should rely on the recognized heuristics rather than on a learned state-action pair representation. The approach will be evaluated both in the context of several grid world scenarios and a game from the GVGAI framework. It will be shown that our approach outperforms standard Q-Learning in the selected scenarios and arguments will be provided that also other agent-based machine learning techniques could profit from the proposed approach.

1 Introduction

Common machine learning techniques used in the context of learning agents (e. g., Reinforcement Learning based approaches like *Q-Learning* (Watkins 1989), (Sutton and Barto 1998)) are usually based on a trial-and-error principle: In unknown environments, agents based on such approaches start with purely random actions and successively improve their behavior as the learning process progresses. Having no *a priori* knowledge about the environment, such agents have to explore large amounts of the state-action space, even if the underlying structure of the environment follows simple rules.

Humans, in contrast, are known for successfully applying rough heuristics learned from very few examples. This, of course, can lead to wrong decisions (see, e. g., (Dörner 1992) for several examples), however, in the common cases many problems can be quickly solved by applying such heuristics derived from few examples. One reason for that is, that many problem domains (e. g., games) are geared towards the *commonsense human way of thinking*, and these kinds of problems can therefore be quickly solved by exploiting such heuristics.

In this paper, we propose a hybrid symbolic/sub-symbolic agent model which is able to autonomously find and exploit useful heuristics in a previously unknown environment. During a Reinforcement Learning process, the agent creates a symbolic knowledge base from learned experiences which comprises rule-based knowledge on multiple levels of abstraction as a model for heuristics. We do not incorporate any *a priori* knowledge (as done e. g. in (Shapiro, Langley, and Shachter 2001)—although this would be possible in our model), instead the heuristics are inferred by the agent *itself* during the learning process. Furthermore, the agent is able to decide itself at which point during the learning process the decisions should be relied on the found heuristics rather than on the weighted state-action pairs of the Reinforcement Learning process. More concretely, the following contributions are made:

- A hybrid symbolic/sub-symbolic agent model which incorporates both symbolic rule-based knowledge and weighted state-action pair representations learned by a sub-symbolic learning approach.
- An approach for estimating the *subjective structural complexity* of a previously unknown environment from an agent perspective during a learning process (based on a "commonsense" measure for *strategic depth* (Apeldoorn and Volz 2017)).

Compared to our previous works on *knowledge base extraction* presented in (Apeldoorn and Kern-Isberner 2016) and (Apeldoorn and Kern-Isberner 2017), the novelty of this paper lies in the incorporation of our extraction approach into a learning agent model which is able to decide *on its own* when to exploit the extracted knowledge as a heuristic in an unknown environment. This is realized by incorporating the strategic depth measure introduced in (Apeldoorn and Volz 2017) as a decision criterion.

As a side-product, the resulting agent model is still able to explain the learned knowledge and the made decisions to some extent at any point during the learning process, as described in (Apeldoorn and Kern-Isberner 2017).

In Section 2 related similar approaches will be discussed. A brief introduction to the fundamental preliminary works will be given in Section 3. The agent model will be introduced in Section 4. In Section 5 the model will be evaluated in the context of several simple grid world scenarios and later in different levels of a game from the *General Video Game Playing Artificial Intelligence* (GVGAI)

framework, which is usually used for the GVGAI competition where agents have to compete in playing previously unknown video games (Perez-Liebana et al. 2016).

2 Related Work

Several attempts have been made so far to closer relate symbolic knowledge representation and sub-symbolic machine learning approaches in the context of agents. The basic ideas can be roughly assigned to one of the following groups (representatives for every group will be discussed subsequently):

1. Extraction techniques to gain different knowledge representations (e. g., propositional rules) learned from data through machine learning techniques
2. Methods to integrate *a priori* knowledge into machine learning approaches to support the learning process (e. g., by shaping the search space with *a priori* knowledge and later refining this knowledge with learning techniques)
3. Cognitive architectures that combine 1. and 2.

Representatives of the first group are, e. g., (Sun 2002), where extraction techniques have been proposed to gain simple rules or plans from reinforcement learning. In (Junges and Klügl 2013), decision trees are created from learned weighted state-action representations with the primary goal of supporting agent developers in the implementation of adequate agent behavior. These works focus on the *extraction* of knowledge in different forms. In contrast, in this paper, we propose to (re)incorporate the extracted knowledge back into the learning process as a model for finding and exploiting rough heuristics in a previously unknown environment.

As representatives for the second group, e. g., (Singh et al. 2011) and (Shapiro, Langley, and Shachter 2001) can be considered: In these approaches, machine learning techniques are combined with mechanisms to incorporate *a priori* knowledge to accelerate the learning process or to later refine and adapt the *a priori* knowledge to a dynamically changing environment (in (Singh et al. 2011) this is realized in the context of a BDI agent model where initial beliefs which can be refined during a learning process). In contrast, our approach works (nearly) without any *a priori* knowledge about the environment.

The third group is represented, e. g., by models such as CLARION (Sun, Peterson, and Merrill 1999), a cognitive architecture which incorporates both sub-symbolic learning and symbolic knowledge during learning and decision making. In this model, a sub-symbolic representation is combined with a rule network on top, and both the sub-symbolic and the rule level are used in the agent's decision making process. In our approach, however, we assume a reasonable point during a learning process, where a transition from a sub-symbolic (implicit) knowledge to a symbolic (explicit) knowledge representation should take place with the primary objective to quickly find useful *heuristics* in a previously unknown environment. Another representative related to the third group, the SPHINX approach, is described in (Leopold, Kern-Isberner, and Peters 2008), where reinforcement learning is combined with *ranking functions* and *belief revision* to support the learning process. It was shown on a object recognition task that the considered learning agent can benefit from the combined approach (with and without involving additional background knowledge). However, in contrast to our approach, it was not investigated, in which phase during the learning process the agent should rely its behavior more on the explicit, symbolic part of the knowledge (i. e., the ranking function) rather than on the implicit representation of the weights learned through the reinforcement learning part. Furthermore, the symbolic part is not primarily dedicated to represent a model for heuristics.

3 Preliminaries

This section introduces the two preliminary works needed for our approach: First, the concept of *exception-tolerant Hierarchical Knowledge Bases* (HKBs) (Apeldoorn and Kern-Isberner 2016, 2017) will be briefly introduced and how they can be retrieved (Sections 3.1 and 3.2).[1] HKBs will be used later as a model for heuristics. After that, a "commonsense" strategy measure (recently introduced in (Apeldoorn and Volz 2017)), is described which will be used later in our agent model to let an agent *subjectively* estimate whether or not heuristics should be exploited (Section 3.3).

3.1 Definition of HKBs

To introduce HKBs and how they can be learned, we closely follow (Apeldoorn and Kern-Isberner 2016, 2017). There, an extraction approach is proposed which is able to extract an HKB from a weighted state-action pair representations learned by a common reinforcement learning approach like Q-Learning (Watkins 1989).[2]

This section only provides the main definitions needed to understand the basic idea of HKBs. For more details on HKBs, the reader should refer to the original literature.

Basics of the Agent Model We consider an agent which is learning a policy by acting autonomously in a previously unknown environment. The agent is equipped with n sensors through which it can perceive its current state in the environment. The agent is able to perform actions from a predefined action space and can furthermore perceive, whether or not the performed actions were good, in form of (numeric) rewards. The perceived rewards are then used to learn a weighted state-action pair representation where the weights determine which action has to be performed, given a perceived state (usually the one with the highest weight).

More formally, in such a representation, a state s is an element of a multi-dimensional state space $\mathbb{S} = \mathbb{S}_1 \times ... \times \mathbb{S}_n$ where n is the number of the agent's sensors (through which the agent is able to perceive its state in the environment)

[1] Note that in (Apeldoorn and Kern-Isberner 2016, 2017), *exception-tolerant Hierarchical Knowledge Bases* were originally called *Hierarchical Knowledge Bases*—but to avoid confusion with the *Hierarchical Knowledge Bases* by Borgida et al. (Borgida and Etherington 1989) and in order to further outline the primary purpose of our approach, we call them *exception-tolerant* here.

[2] Note that the used learning approach is not of importance as long as it results in a weighted state-action pair representation, where the weights determine which action has to be chosen given a perceived state (usually the one with the highest weight).

and every $\mathbb{S}_i$ is a set of possible sensor values of the corresponding sensor. Furthermore, the agent selects actions from a predefined action set $\mathbb{A}$ and the learned weights are stored in a multi-dimensional matrix $\hat{Q} = (q_{s_1,...,s_n,a})$ with $s_i \in \mathbb{S}_i$ and $a \in \mathbb{A}$. The weights can be learned by different machine learning approaches, provided that the learning approach converges such that given a state, the highest weight determines the best action to be selected (i. e., $a^{\max}_{s_1,...,s_n} = \arg\max_{a' \in \mathbb{A}} q_{s_1,...,s_n,a'}$).

Exception-Tolerant Hierarchical Knowledge Bases An HKB consists of rules which are organized on different levels of abstraction. In contrast to *Exception Lists* (Michael 2011), an HKB can handle multiple rules per level and the rules also comprise weights. To be able to define these rules, two different kinds of states and two different kinds of rules will be distinguished (Apeldoorn and Kern-Isberner 2017):

Definition 1 (Complete States/Partial States) *A complete state is a conjunction $s := s_1 \wedge ... \wedge s_n$ of all values s_i currently perceived by an agent's sensors, where n is the number of sensors (and every perceived sensor value $s_i \in \mathbb{S}_i$ of the corresponding sensor value set $\mathbb{S}_i$ is assumed to be a fact in the agent's current state). A partial state is a conjunction $s := \bigwedge_{s' \in S} s'$ of a subset $S \subset \{s_1, ..., s_n\}$ of the sensor values of a complete state.*

Definition 2 (Complete Rules/Generalized Rules) *Complete rules and generalized rules are of the form $p_\rho \Rightarrow a_\rho\ [w_\rho]$, where p_ρ is either a complete state (in case of an complete rule) or a partial state (in case of a generalized rule), the conclusion $a_\rho \in \mathbb{A}$ is an action of an agent's action space $\mathbb{A}$ and $w_\rho \in [0, 1]$ is the rule's weight.*

Thus, *complete rules* map *complete states* to actions and *generalized rules* map *partial states* to actions. An HKB can now be defined as follows:

Definition 3 (Exception-Tolerant Hierarchical Knowledge Base) *An exception-tolerant Hierarchical Knowledge Base (HKB) is an ordered set $\mathcal{KB} := \{R_1, ..., R_{n+1}\}$ of $n + 1$ rule sets, where n is the number of sensors (i. e., the number of state space dimensions). Every set $R_{i<n+1}$ contains generalized rules and the set R_{n+1} contains complete rules, such that every premise $p_\rho = \bigwedge_{s \in S_\rho} s$ of a rule $\rho \in R_i$ is of length $|S_\rho| = i - 1$.*

According to Definition 3, the set R_1 contains the most general rules (with empty premises) and the set R_{n+1} contains the most specific (i. e., complete) rules.

For the relations of rules, the term of *needed exception* will be used, according to the following definition (cf. (Apeldoorn and Kern-Isberner 2017)):

Definition 4 (Needed Exception) *A rule $\rho \in R_{j>1}$ is an exception to a rule $\tau \in R_{j-1}$ with premise $p_\tau = \bigwedge_{s \in S_\tau} s$, action a_τ as conclusion and weight w_τ, if $S_\tau \subset S_\rho$ and $a_\rho \neq a_\tau$. The exception is needed, if there exists no other rule $\upsilon \in R_{j-1}$ with premise $p_\upsilon = \bigwedge_{s \in S_\upsilon} s$ and action a_υ as conclusion where $S_\upsilon \subset S_\rho$, $a_\upsilon = a_\rho$ and $w_\upsilon > w_\tau$.*

3.2 Learning an HKB

An HKB can be extracted from a weighted state-action pair representation $\hat{Q}$ (that is learned, e. g., through a Reinforcement Learning technique) using the following approach (Apeldoorn and Kern-Isberner 2016):[3]

The approach takes a weighted state-action pair representation $\hat{Q}$ as input and returns an HKB $\mathcal{KB}^{\hat{Q}}$ which reflects the knowledge contained in $\hat{Q}$ by performing the following steps:

1. *Initial creation of rule sets:* In the first step, the multiple abstraction levels $R_1, ..., R_{n+1}$ of the knowledge base are initially filled with rules. We only consider state-action pairs here that contribute to the best policy found during the learning process so far.[4]
2. *Removal of worse rules:* In all sets R_j, a rule $\rho \in R_j$ is removed, if there exists another rule $\sigma \in R_j$ with the same partial state as premise having a higher weight (i. e., in every set R_j only the best rules for a given partial state are kept).
3. *Removal of worse more specific rules:* In all sets $R_{j>1}$, a rule $\rho \in R_j$ with premise $p_\rho = \bigwedge_{s \in S_\rho} s$, conclusion a_ρ and weight w_ρ is removed, if there exists a more general rule $\sigma \in R_{j'<j}$ with premise $p_\sigma = \bigwedge_{s \in S_\sigma} s$ where $S_\sigma \subset S_\rho = \{s_1, ..., s_{j-1}\}$ and weight $w_\sigma \geq w_\rho$.
4. *Removal of too specific rules:* In all sets R_j, a rule $\rho \in R_{j>1}$ with premise $p_\rho = \bigwedge_{s \in S_\rho}$ and conclusion a_ρ is removed, if there exists a more general rule $\sigma \in R_{j'<j}$ with the same action $a_\sigma = a_\rho$ as conclusion and with premise $p_\sigma = \bigwedge_{s \in S_\sigma} s$ where $S_\sigma \subset S_\rho = \{s_1, ..., s_{j-1}\}$ and if ρ is not a *needed exception* to a rule $\tau \in R_{j-1}$.
5. *Optional filter step:* Optionally, filters may be applied to filter out further rules which are, e. g., helpful to explain the knowledge contained in $\hat{Q}$ through the optimal found policy so far, but which are not needed for reasoning later.

After performing these steps on $\hat{Q}$, the knowledge base $\mathcal{KB}^{\hat{Q}}$ comprises all sets $R_j \neq \emptyset$ with the extracted rules representing the implicit knowledge contained in the learned weights of $\hat{Q}$ in a compact way.

Reasoning on HKBs A reasoning algorithm on HKBs was proposed in (Apeldoorn and Kern-Isberner 2016): Given perceived sensor values $s_1, ..., s_n$, this algorithm searches an HKB upwards, starting from the bottom-most level R_{n+1}, for the first rule whose premise is fulfilled. This rule is then returned as concluding action (see (Apeldoorn

[3] Note that a faster version of the algorithm improving its first step is proposed in (Apeldoorn and Kern-Isberner 2017) (cf. footnote 4).

[4] Furthermore, since the number of possible premises of the rules can grow drastically with the size of the given problem instance, an efficient algorithm has to be used here. An attempt for efficiently preselecting the rules using adapted ideas from the APRIORI algorithm (Agrawal et al. 1996) has been made in (Apeldoorn and Kern-Isberner 2017) to increase the overall performance of the extraction approach.

and Kern-Isberner 2016) for details). By this, the algorithm selects the most specific rule that fits to the perceived sensor values and falls back to the next more unspecific rule as a heuristic, in case no more specific rule with a fitting premise could be found.

Due to the possibility of falling back to more general rules in case no matching rule could be found for the given sensor values, the reasoning mechanism on an HKB allows for drawing meaningful conclusions over states that have not been seen before by an agent. The idea that a more general rule serves as a rough heuristic for unknown states (for which no better knowledge is available) will later contribute to accelerate the learning process.

3.3 Strategic Depth

Based on an HKB, in (Apeldoorn and Volz 2017), a strategy measure was introduced which seems to adequately reflect the *commonsense strategic depth* of games. The measure was evaluated in the context of a survey. This measure will be used later in our agent model, to let the agent judge *subjectively* when to apply heuristics. According to (Apeldoorn and Volz 2017), the measure is defined as follows:

Definition 4 (Strategic Depth Measure d_s) *Let $\mathcal{KB} = \{R_1, ..., R_{n+1}\}$ be an HKB and $\mathfrak{S} = \{\mathbb{S}_1, ..., \mathbb{S}_n\}$ be the set of all sensor value sets of an agent, then the strategic depth is defined as a function*

$$d_s(\mathcal{KB}, \mathfrak{S}) := \sum_{i=1}^{n+1} \binom{n}{i-1} b^{i-1} \frac{|R_i|}{\sum\limits_{\substack{\mathbf{S} \subseteq \mathfrak{S} \\ |\mathbf{S}|=i-1}} \prod\limits_{\mathbb{S} \in \mathbf{S}} |\mathbb{S}|} , \quad (1)$$

where $n+1 = |\mathcal{KB}| = |\mathfrak{S}| + 1$ is the number of levels in $\mathcal{KB}$, b is a weighting constant, and $R_i \in \mathcal{KB}$ is the i-th level of $\mathcal{KB}$.

As rules on a level $R_{j>1}$ can be considered *exceptions* to the rules on level R_{j-1}, the intuition behind d_s is that it measures the weighted relative number of rules (hence, exceptions) represented by an HKB.

In (Apeldoorn and Volz 2017), also a modified version of the measure is proposed, where d_s is additionally divided by the sum of weights to normalize it to values of range $[0, 1]$. For our purpose, this version of the measure is extremely useful, since it allows our agent model to estimate the strategic depth of a previously unknown environment relatively to the *maximum* strategic depth that can be expected, given the sets of possible sensor values $\mathbb{S}_1, ..., \mathbb{S}_n$. Thus, in the following, the normalized version of the measure (according to (Apeldoorn and Volz 2017)) will be used:

Definition 5 (Normalized Strategic Depth $\bar{d}_s$) *Let $\mathcal{KB}$, $\mathfrak{S}$, n, b and d_s be as in Definition 5. Then the normalized strategic depth is defined as a function*

$$\bar{d}_s(\mathcal{KB}, \mathfrak{S}) := \frac{d_s(\mathcal{KB}, \mathfrak{S})}{\sum_{i=1}^{n+1} \binom{n}{i-1} b^{i-1}} . \quad (2)$$

In the following, $\bar{d}_s$ will be used in our agent model to allow the agent to estimate *subjectively* when it could be useful to exploit heuristics during a learning process in an *a priori* unknown environment: If $\bar{d}_s$ falls below a certain threshold th then this indicates that heuristics could possibly be applied successfully.

The intuition behind this is that the (normalized) strategic depth based on an HKB which was extracted from a weighted state-action pair representation learned by an agent, will successively decrease as the agent's learning process progresses: Since the agent's knowledge about the problem increases, the problem appears successively less complex to the agent and the subjectively measures normalized strategy depth converges to the *de facto* strategic depth of the task to be learned (i. e., the strategic depth according to the HKB of the optimal policy to solve the task). We will demonstrate this in the following on several examples.

Examples for Subjective Normalized Strategic Depth We consider four different grid worlds of different structural complexities where an agent has to learn to navigate from a starting point A to a target point B (see Figure 2). We use standard *Q-Learning* (Watkins 1989) as a learning approach with a learning rate of $\alpha = 0.1$, a random action probability of $\epsilon = 0.1$ and a discount factor $\gamma = 0.9$. We run every scenario for 250 runs and we measure the normalized strategic depth subjectively perceived by the agent (averaged over 30 repetitions) in dependency of the number of runs already performed. Two phenomena can be observed in Figure 2:

- The subjective strategic depth converges to the *de facto* strategic depth of the respective scenario as the agent behavior converges to the optimal policy.
- The subjective strategic depth decreases in general as the learning process progresses, since the agent's knowledge about the scenario increases, and therefore the scenarios appears to be simpler to the agent. (In case of the fourth scenario, after the first runs, the agent seems to underestimate the depth of the scenario, since the scenario locally has straight path structures that form together a more complex path from a global point of view.)

4 Agent Model

This section introduces our model of a learning agent which is able to determine and exploit heuristics based on HKBs during a learning process in a previously unknown environment. In our agent model, a sub-symbolic learning approach can be used to learn weighted state-action pairs (where the maximum weight determines the preferred action, given a state). Figure 1 illustrates the main ideas of our agent model which mainly consists of two parts:

- an *initialization part* which is executed every time before a new learning episode starts, and
- a common *agent cycle* which is executed every time step.

In the former, the agent decides, whether it relies its decisions on the weighted state-action pairs (represented by the $\hat{Q}$ matrix) or on the heuristics (represented by the current

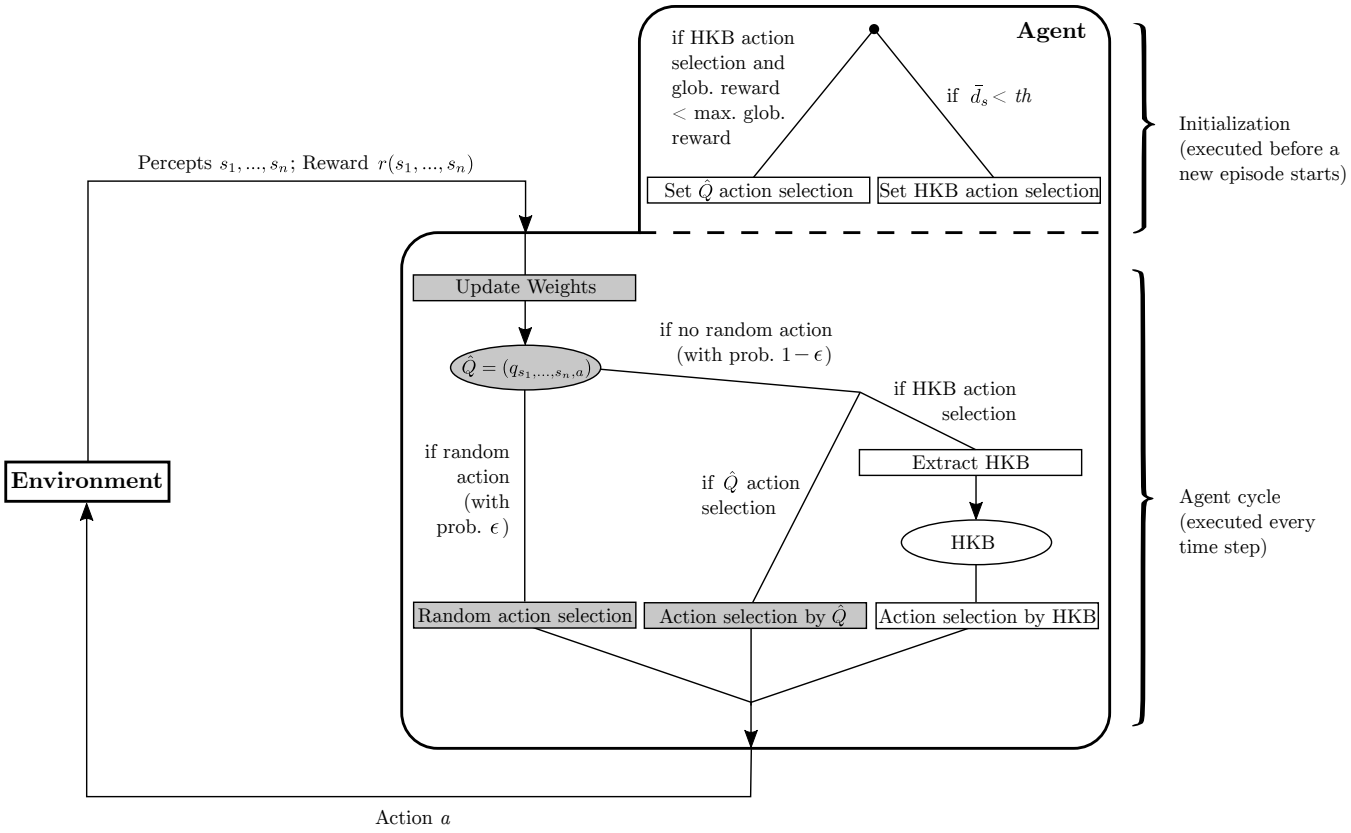

Figure 1: Learning Agent Model with the Ability of Finding and Exploiting Heuristics

extracted HKB) in the upcoming learning episode. This decision is made depending on the $\bar{d}_s$ value calculated from the current extracted HKB.

In the latter, the agent either chooses a random action (for exploration purpose) or decides to choose its action according to the $\hat{Q}$ matrix or the extracted HKB, respectively, depending on the decision made during the initialization. Note that in Figure 1, components belonging to the sub-symbolic learning approach are indicated by a gray coloring, whereas those parts belonging to our heuristics extension are colored white. The gray parts could be replaced by a different learning approach.

5 Evaluation

This Section first evaluates the approach in the four grid world scenarios from Figure 2 in Section 3.3. Later, the approach will be evaluated additionally in a more realistic example: a game from the GVGAI framework (Perez-Liebana et al. 2016).

5.1 Evaluation in the Context of Grid World Scenarios

We run the four grid world Scenarios from Figure 2 in Section 3.3 for 50 runs each and measure the percentage of how many times the optimal policy was found by the agent over 200 repetitions of the experiment. As learning approach we use again Q-Learning with the same parameters used for measuring the subjective strategic depth at the end of Section 3.3. As threshold parameter to determine the exploitation of found heuristics by means of the subjective strategic depth $\bar{d}_s$, we choose a threshold of $th = 0.2$. According to Figure 2, this means that—in average—the agent would try to exploit the found heuristics after

- ≈ 25 runs in case of the first scenario,
- ≈ 75 runs in case of the second scenario and
- ≈ 100 runs in case of the third scenario.

For the second and the third scenario, this may sound confusing since we perform a total number of 50 runs only. However, the reader should be aware that Figure 2 shows the *average* development of $\bar{d}_s$ measure and thus there are enough cases where the agent individually decides to exploit a found heuristic that leads to an optimal policy.

As for the fourth scenario, the agent never considers to exploit any heuristics, since the subjective strategic depth does not decrease below $th = 0.2$. This can be interpreted in a way that the scenario comprises too many exceptions, such that an exploitation of heuristics does not make sense from the agent's point of view (this point of view is in fact what the parameter th controls, depending on whether the desired agent should be more "heuristics-affine" or more "conservative"). Table 1 contains the comparison of our agent model

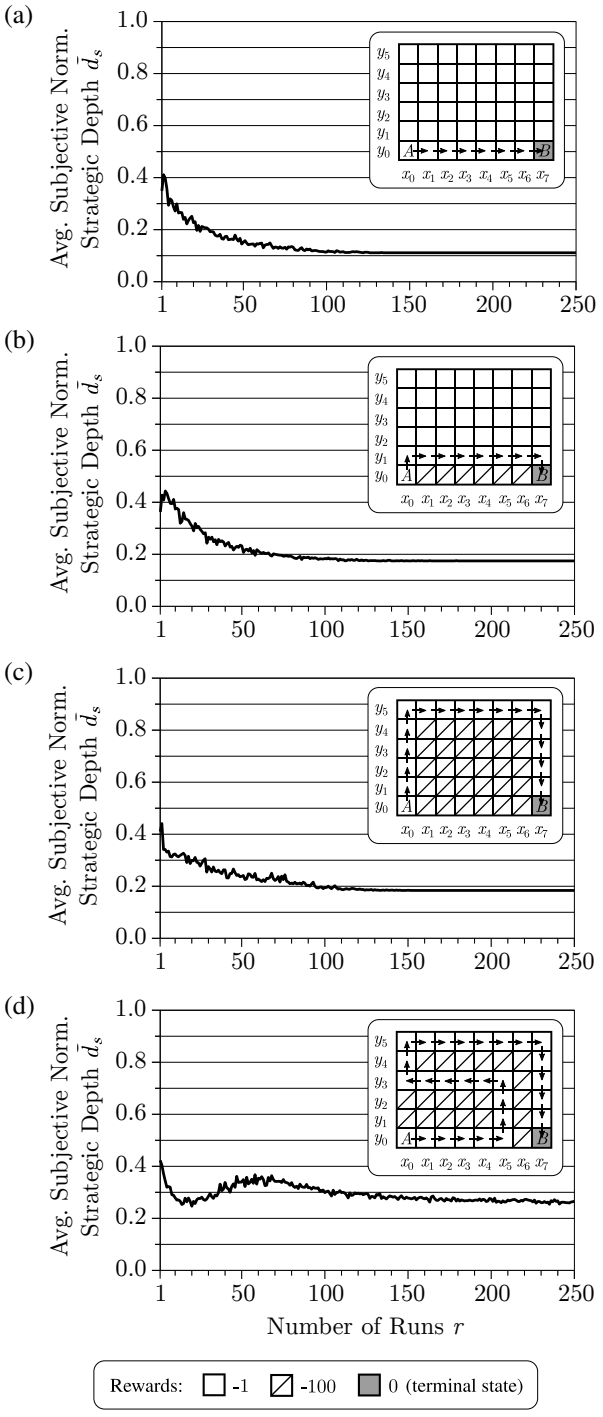

Figure 2: Examples of Subjective Normalized Strategic Depth for the Considered Grid Worlds Scenarios of Increasing Structural Complexity with the Optimal Policies Indicated by Arrows—*Scenarios 1 (a) and 2 (b) are taken from (Apeldoorn and Kern-Isberner 2016), Scenario 3 (c) is from (Apeldoorn and Kern-Isberner 2017) and Scenario 4 (d) from (Apeldoorn and Volz 2017).*

	Standard Q-Learning	**With Heuristics** ($th = 0.2$)	**Heuristics Only**
Scenario 1	3.5%	66.5%	43.0%
Scenario 2	4.5%	20.0%	10.0%
Scenario 3	0.0%	10.5%	0.03%
Scenario 4	0.0%	0.0%	0.0%

Table 1: Standard Q-Learning vs Heuristics Approach in the Context of the Grid Worlds Scenarios—*The table shows the percentage of* 100 *repetitions in which the optimal policy was found by the agent during the first* 50 *runs.*

against a standard Q-Learning agent with the same parameters for the learning part. In addition, for reasons of comparison, Table 1 also provides results of a *heuristics-only* version of the agent model[5], where the agent was enforced to rely its decisions always on the extracted heuristics (i. e., in the initialization phase in Figure 1 the action selection mode is *always* set to *HKB action selection* and the weights contained in the $\hat{Q}$ matrix are *never* considered directly for action selection in the agent cycle).

As can be seen in Table 1, standard Q-Learning rarely manages to solve the grid worlds during the first 50 runs, whereas our heuristics approach clearly outperforms these results. As for the *heuristics-only* version, the results are slightly worse: This seems to be the case, since the agent starts too early to rely its decisions on the heuristics (which are nearly random in the beginning of the learning process). This may prevent the agent from further exploring the environment to an adequate extent and may result in local optimal policies.

One could argue that standard Q-Learning is a rather old approach and there are nowadays more elaborate learning approaches available. However, if a better learning approach will be chosen in our setup (*better* in the sense that it will converge faster to the optimal policy), the extracted HKBs and hence the measure $\bar{d}_s$ will be more accurate as well and therefore, better heuristics could be exploited earlier during the learning process. This means that in our approach, both the heuristics and the measure $\bar{d}_s$ (to decide when to exploit them) will improve with the quality of the learning approach used for the agent.

5.2 Evaluation in the Context of a GVGAI Game

We evaluate the agent model now in a more realistic example by consider the game *Camel Race* from the GVGAI framework (Perez-Liebana et al. 2016). In this game, the agent has to control a camel which must be moved to the opposite site of the screen—faster than any other camel in the game (additionally avoiding obstacles in the higher levels). Figure 3 shows the first and the third level of the game (with obstacles).[6]

[5]Thanks to an anonymous reviewer for proposing this idea.

[6]The game was slightly modified for our purpose by reducing the number of camels, in order to reduce the dimensionality of the state-action space without changing the basic game mechanics.

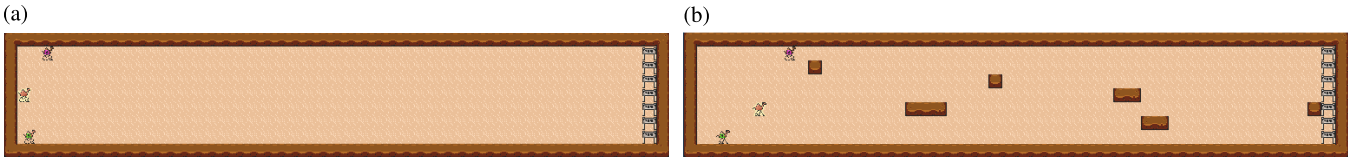

Figure 3: First Level (a) and Third Level (b) of the GVGAI Game *Camel Race*

	Standard Q-Learning	**With Heuristics** ($th = 0.2$)	**Heuristics Only**
1st Level	0.0%	63.3%	66.0%
3rd Level	0.0%	53.3%	60.0%

Table 2: Standard Q-Learning vs Heuristics Approach in the Context of the GVGAI Game *Camel Race—The table shows the percentage of* 30 *repetitions in which the agent won the game in the first* 100 *runs.*

The agent's state space is described by the sensor value sets $\mathbb{S}_x^{C_1} \times \mathbb{S}_x^{C_2} \times \mathbb{S}_y^{C_2} \times \mathbb{S}_x^{C_3}$ containing the coordinates of the camels in the game (where $C_1, ..., C_3$ correspond to the three camels, C_2 is the camel in the middle controlled by the agent and C_1, C_3 only move in x direction) and $\mathbb{A} = \{\text{up}, \text{down}, \text{left}, \text{right}, \text{none}\}$ are the five actions that can be performed by the agent. As reward, the agent perceives the *current distance to the fastest opponent camel* in x direction.

Even if the game is rather simple, it has an interesting aspect: Due to the dynamics of the environment caused by the movement of the two opponent camels, our agent perceives new and previously unseen states in nearly every game tick of the early learning phase. Thus, the agent has to explore large parts of the state-action space to improve its behavior, although the game could be easily won by applying a rough heuristic like *always move right* ($\top \Rightarrow \text{right}\ [1.0]$) for the first level. This renders the game an eligible test environment for our agent model.

We perform 100 runs for each level and measure the percentage of wins by the agent over 30 repetitions. Again, standard Q-Learning with and without the heuristics approach, as well as the *heuristics-only* approach are used, given the same standard parameters as provided in Section 5.2. Table 2 shows the results for the first and the third level of the game: With standard Q-Learning, the agent is not able to win the game within 100 runs, whereas using the heuristics approach, the agent wins both levels in over 50% of the cases. Surprisingly, the *heuristics-only* approach performs even better here. This seems to be the case since—although the game has some dynamics and can thus be considered more complex than the grid world scenarios—the considered levels allow for rougher heuristics than the grid worlds: In contrast to the grid worlds, the goal to be reached is not located in a single corner, instead, the game can be won by simply reaching the rightmost side of the screen (which can already be achieved by a very rough heuristic).

6 Conclusion and Future Work

In this paper, we described a model of a learning agent which is able to find and to exploit heuristics without *a priori* knowledge in unknown environments. We showed on several examples (four grid world scenarios and one game from the GVGAI framework) that our approach drastically accelerates the learning process to find optimal policies.

However, the approach is not perfect yet and leaves room for future work: (1) The runtime to extract an HKB should be further improved (a faster algorithm has been proposed in (Apeldoorn and Kern-Isberner 2017) already; cf. footnote 4) and (2) a quality criterion could be incorporated into the measure that helps to decide whether heuristics should be exploited, depending on previous experiences with applying these heuristics. The latter could possibly be useful to help preventing an agent from repeatedly trying out bad heuristics.

Furthermore, it could be interesting to extend the model with revision techniques for the found heuristics and to incorporate mechanisms to store and reuse heuristics.

References

Agrawal, R.; Mannila, H.; Srikant, R.; Toivonen, H.; and Verkamo, A. 1996. *Fast Discovery of Association Rules*. Advances in Knowledge discovery and Data Mining. Cambridge, MA, USA: MIT Press. 307–328.

Apeldoorn, D., and Kern-Isberner, G. 2016. When should learning agents switch to explicit knowledge? In Benzmüller, C.; Sutcliffe, G.; and Rojas, R., eds., *GCAI 2016. 2nd Global Conference on Artificial Intelligence*, volume 41 of *EPiC Series in Computing*, 174–186. EasyChair Publications.

Apeldoorn, D., and Kern-Isberner, G. 2017. Towards an understanding of what is learned: Extracting multi-abstraction-level knowledge from learning agents. In Rus, V., and Markov, Z., eds., *Proceedings of the Thirtieth International Florida Artificial Intelligence Research Society Conference*, 764–767. Palo Alto, California: AAAI Press.

Apeldoorn, D., and Volz, V. 2017. Measuring strategic depth in games using hierarchical knowledge bases. In *2017 IEEE Conference on Computational Intelligence and Games (CIG)*, 9–16. IEEE. *To be published.*

Borgida, A., and Etherington, D. W. 1989. Hierarchical knowledge bases and efficient disjunctive reasoning. In Brachman, R. J.; Levesque, H. J.; and Reiter, R., eds., *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, 33–43. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Dörner, D. 1992. *Die Logik des Mißlingens – Strategisches Denken in komplexen Situationen*. Reinbek bei Hamburg: Rowohlt Taschenbuch Verlag.

Junges, R., and Klügl, F. 2013. Learning tools for agent-based modeling and simulation. *KI – Künstliche Intelligenz* 27(3):273–280.

Leopold, T.; Kern-Isberner, G.; and Peters, G. 2008. *Belief Revision with Reinforcement Learning for Interactive Object Recognition*. ECAI 2008 – 18th European Conference on Artificial Intelligence Proceedings. Amsterdam: IOS Press. 65–69.

Michael, L. 2011. Causal Learnability. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011)*, 1014–1020. Palo Alto, California: AAAI Press.

Perez-Liebana, D.; Samothrakis, S.; Togelius, J.; Schaul, T.; and Lucas, S. M. 2016. General video game ai: Competition, challenges and opportunities. In Schuurmans, D., and Wellman, M., eds., *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence and the Twenty-Eighth Innovative Applications of Artificial Intelligence Conference*, 4335–4337. Palo Alto, California: AAAI Press.

Shapiro, D.; Langley, P.; and Shachter, R. 2001. *Using Background Knowledge to Speed Reinforcement Learning in Physical Agents*. AGENTS '01 Proceedings of the fifth international conference on Autonomous agents. New York: ACM. 254–261.

Singh, D.; Sardina, S.; Padgham, L.; and James, G. 2011. Integrating learning into a bdi agent for environments with changing dynamics. In Walsh, T., ed., *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, 2525–2530. Menlo Park, California: AAAI Press/International Joint Conferences on Artificial Intelligence.

Sun, R.; Peterson, T.; and Merrill, E. 1999. A hybrid architecture for situated learning of reactive sequential decision making. *Applied Intelligence* 11(1):109–127.

Sun, R. 2002. *Knowledge Extraction from Reinforcement Learning*. New Learning Paradigms in Soft Computing. Berlin Heidelberg: Springer. 170–180.

Sutton, R., and Barto, A. 1998. *Reinforcement Learning: An Introduction*. Camebridge, Massachusetts; London, England: The MIT Press.

Watkins, C. 1989. *Learning from Delayed Rewards*. England: University of Cambridge.