A Appendix

A.1 List of abbreviations

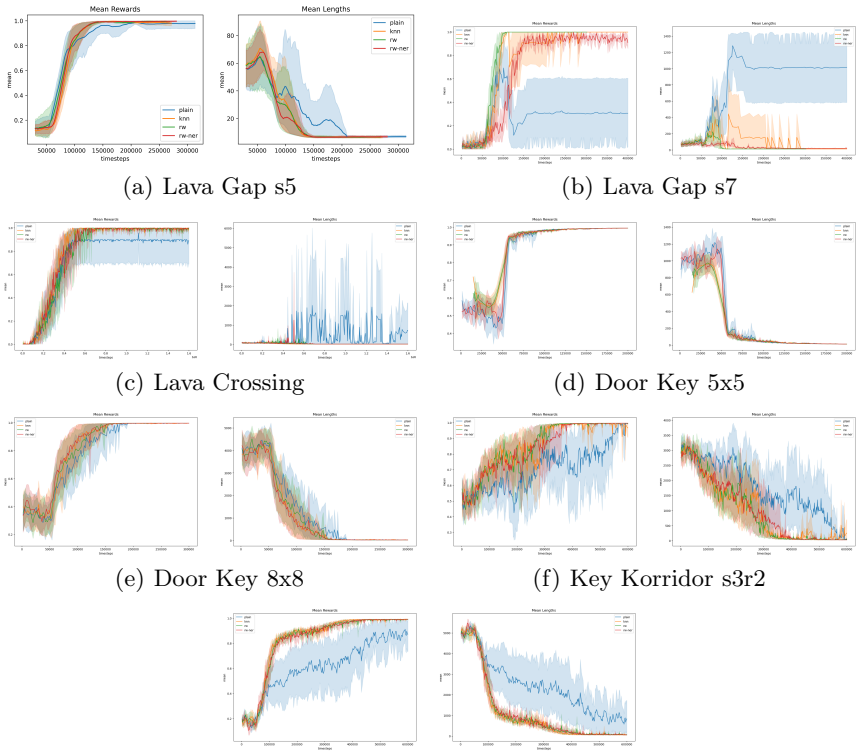
- **AI** Artificial Intelligence
- **AMDP** Abstract Markov Decision Process
- **API** Application Programming Interface
- **DQN** Deep Q-Network
- **ER** Entity and Relation (linking)
- **KG** Knowledge Graph
- **KGE** Knowledge Graph Embedding
- **KGEM** Knowledge Graph Embedding Model
- **KGRL** Knowledge Graphs (Injection in) Reinforcement Learning
- **KNN** K-Nearest Neighbors algorithm
- **LLM** Large Language Model
- **LM** Language Model
- **MDP** Markov Decision Process
- **ML** Machine Learning
- **NLU** Natural Language Understanding
- **NLP** Natural Language Processing
- **OWL** Web Ontology Language
- **RDF** Resource Description Framework
- **RL** Reinforcement Learning
- **RLHF** Reinforcement Learning from Human Feedback
- **RW** Random Walk
- **SEM** The Simple Event Model
- **W3C** WWW Consortium

A.2 Additional Learning curves

Training Runs with Random Action To test the generalization of the KGRL approach, we additionally evaluated the trained policy with introducing some noise. To this end, 10 episodes of the trained policy at each evaluation step are perturbed by a random action with small probability to verify that the policy has not been overfitted. The obtained results in Fig. 1 support the claim that KG injection into learning process improves sample efficiency, especially for complex environments.

A.3 Ablation studies

Dimensions of KGE We test the KGRL performance by varying the dimensions of the KGE from 0 to 100 and plotting mean rewards and mean episode lengths for each environment (see Fig. 3). The general trend shows, that there is a positive correlation between KGE dimensionality and sample efficiency, though values differ across the environments.



(a) Lava Gap s5

(b) Lava Gap s7

(c) Lava Crossing

(d) Door Key 5x5

(e) Door Key 8x8

(f) Key Korridor s3r2

(g) Obstructed Maze D1

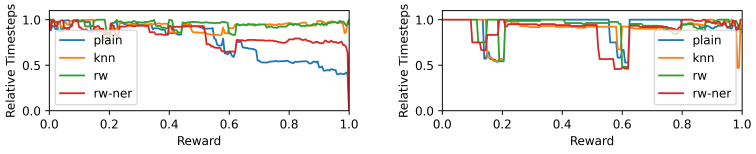
Fig. 1. Mean Reward and Episode length achieved by the trained approaches for different environments. Rewards and episode length are reported as average over ten seeds with one standard deviation confidence intervals. We evaluated the policies at 200 equally distance training steps running ten episodes at every evaluation with a small probability of disturbing the agent with a random action to test for overfitting.

A.4 Hyperparameter tuning

We perform hyperparameter tuning to obtain good training parameters for the baseline approach (DQN), in order to ensure the possibility of convergence for the Baseline. We do not perform hyperparameter tuning for the parameters of the proposed methods even though further performance increases would be possible. During training all methods use the same Hyperparameters.

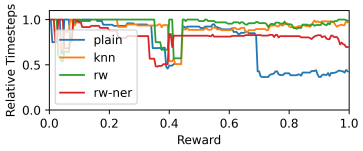
We search for the best parameter values detailed in Table 1, by performing 300 trials for all environments and optimizing over the rescaled area under the curve of the timesteps-reward graph. We use a Tree-structured Parzan Estimator for suggesting parameter values during the trials. All hyperparameter tuning has been implemented with Optuna[1]

[1] https://optuna.readthedocs.io/en/stable/index.html

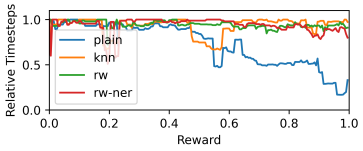


(a) All Environments

(b) Easy Environments

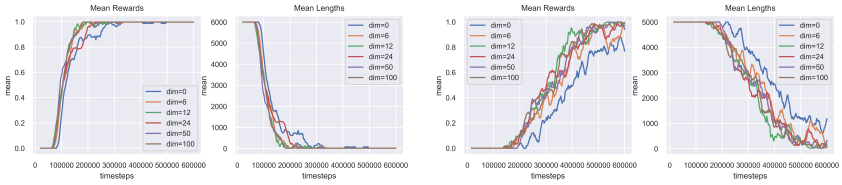


(c) Medium Environments



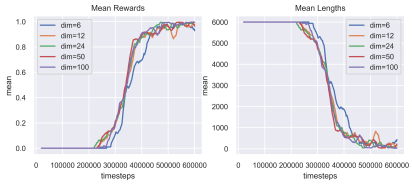
(d) Hard Environments

Fig. 2. Average of the relative number of training timesteps $(t/t^{best})^{-1}$ needed to achieve a reward, based on the least amount of training steps t^{best} to achieve that reward in the same environments (curves close to $y = 1$ are better). The average is computed over all environments, easy environments, medium environments and hard environments.



(a) Dimensions - DoorKey

(b) Dimensions - KeyKorridor



(c) Dimensions - ObstructedMaze

Fig. 3. Performance of different dimensions for the KGE model.

B Complexity of Minigrid Environments

Table 2 describes the distribution of environments across 3 complexity classes (easy, medium and hard) according to the complexity of the knowledge graph, describing this environment.

Parameter Name	Range	categorical/interval
gamma	[0.9, 0.95, 0.98, 0.99, 0.995, 0.999]	categorical
learning rate	[1e-6, 0.1]	interval
learning rate schedule	[constant, linear, quadratic, cubic, quartic]	categorical
batch size	[128, 256, 512, 1024]	categorical
buffer size	[1e5, 3e5, 1e6]	categorical
exploration final epsilon	[0, 0.5]	interval
exploration fraction	[0.3, 0.9]	interval
learning starts	[20000, 50000]	categorical
train frequency	[100 steps, 300 steps, 1000 steps, 2000 steps, 1 episode]	categorical
gradient steps	[1, 2, 4, 8, 16]	categorical
network architecture	[small, medium, large]	categorical

Table 1. Parameters optimized during Hyperparameter search.

Environment	N, nodes	N, edges	N, artifacts	Obstacles	Complexity
Lava Gap s5	84	140	0	yes	Easy
Lava Gap s7	156	288	0	yes	Medium
Door Key 5x5	86	132	1-2	no	Easy
Door Key 8x8	155	319	1-2	no	Medium
Lava Crossing	253	492	0	yes	Hard
Key Corridor s3r2	116	182	1-2	no	Hard
Obstructed Maze D1	775	1465	3+	yes	Hard

Table 2. Distribution of Minigrid Environments into Complexity Classes

B.1 Classes of the Minigrid Ontology

In Fig. 4 the class hierarchy of the Minigrid ontology is shown. Thesse classes allow to model the agent, action space, environment topology and artifacts.

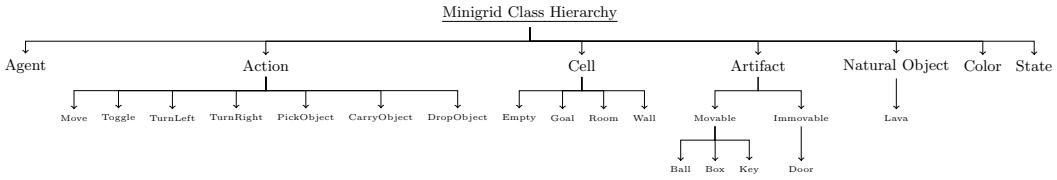


Fig. 4. The Minigrid class hierarchy in the proposed ontology

C Domain Adaptation Process

The steps required to adapt KGRL to a new domain are shown in Fig. 5. The description of the process is provided in the paper.

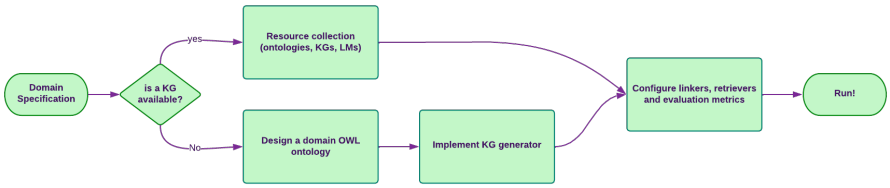


Fig. 5. Tailoring KGRL pipeline to a new domain: The image describes steps and data, that are required to adapt KGRL pipeline to a new domain. Resources (knowledge graphs, ontologies, language models, multimodal embeddings, etc.) reusability plays a major role in this process.