

Goal Recognition with Deep Learning and Embedded Representation of State Traces

Mattia Chiari¹, Alfonso Emilio Gerevini¹, Luca Putelli¹ and Ivan Serina¹

¹Department of Information Engineering, Università degli Studi di Brescia, Via Branze 38, Brescia, Italy

Abstract

The identification of the goal that an agent is going to achieve is an important task with several applications in robotics and security. Despite several approaches on Goal Recognition (GR) relied on automated planning techniques, recently this task has been addressed by GRNet, which exploits deep learning techniques and has reached a new state-of-the-art that solves GR instances more accurately and more quickly. The information required by GRNet is a trace of actions, indicating the names of the observed actions. However, we intend to study this approach in the case of having as input a state trace instead of an action trace. In this situation, two problems arise immediately: how to encode a state in a form that can be processed by a neural network? Is it possible to analyse a sequence of states with the same techniques used for the actions? In this work, we propose a modification of GRNet in order to make it effective also for observations made by traces of states. In particular, we add an autoencoder which has the capability of deriving a numerical representation of a state. We then perform an experimental analysis over two well known benchmark domains.

1. Introduction

Goal Recognition is defined as the task of recognising the goal that an agent is trying to achieve from observations about the agent's behaviour in the environment [2, 3]. Such observations are usually made by a trace of actions executed by the agent for achieving its goal, or a trace of world states progressively generated by the agent's actions. Goal recognition has been studied in AI for many years, and it is an important research field with many applications including human-computer interactions [4], computer games [5], network security [6], financial applications [7], and others.

In the literature, several systems to solve goal recognition problems have been proposed [8]. While most of these approaches are based on classical planning algorithms [9, 10, 11], more recently an entirely different line of work has been introduced. In fact, in more recent studies [7, 12] this problem has been tackled with deep learning algorithms into which a neural network is trained (using a dataset made by observations of the agent and their relative goal) to solve goal recognition problems structured as a classification task. In particular, the deep


IPS-RCRA-SPIRIT 2023: Italian Workshop on Planning and Scheduling, RCRA Workshop on Experimental evaluation of algorithms for solving problems with combinatorial explosion, and SPIRIT Workshop on Strategies, Prediction, Interaction, and Reasoning in Italy. November 7-9th, 2023, Rome, Italy [1]

✉ mattia.chiari@unibs.it (M. Chiari); alfonso.gerevini@unibs.it (A. E. Gerevini); luca.putelli@unibs.it (L. Putelli); ivan.serina@unibs.it (I. Serina)

ORCID 0000-0003-1041-6627 (M. Chiari)



© 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

learning architecture GRNet [13] has reached a new state-of-the-art on goal recognition in several planning benchmarks, improving both accuracy and runtime. Given a planning domain specified by a set of propositions and a set of actions names, each one denoting an agent’s action whose execution can be observed, GRNet is a model based on Recurrent Neural Networks which processes traces of observed actions to compute how likely it is that each domain proposition is part of the agent’s goal. A fundamental aspect of GRNet is that it is trained only once for a given domain, i.e., the same trained network can be used to solve a large set of goal recognition instances in the domain. Moreover, even better results are achieved combining GRNet with LGR [14], which is based on reasoning.

In this work, we extend GRNet allowing it to process also traces of observed states. In order to do that, we introduce a completely new autoencoder for computing a vectorial representation of states. The autoencoder is a feed-forward neural network which is trained to “copy” the input (i.e. a state) to the output. In particular, we exploit an undercomplete autoencoder, into which the hidden layers are smaller than the input. With this kind of architecture, the neural network has the goal to compress the information contained in the input in a smaller, meaningful vector that it is used to reconstruct the output [15]. In our context, these vectors are used for representing states. Therefore, a trace of observed states can be seen as a sequence of vectors that can be processed by a Recurrent Neural Network such as the one used by GRNet.

We perform an experimental analysis on two well known benchmark domains, DEPOTS and ZENOTRAVEL, which confirms the effectiveness of our approach and the applicability of deep learning techniques for goal recognition also with traces of states.

2. Preliminaries and Related Work

Goal recognition (GR) can be defined as the task of identifying the intention (goal) of an agent from observations about the agent’s behaviour in an environment. These observations can be represented as an ordered sequence of actions or states generated by those actions. The agent’s goal can be expressed either as a set of propositions or a probability distribution over alternative sets of propositions (each even forming a distinct candidate goal).

In the “goal recognition over a domain theory” approach [2, 16], an underlying model of the behavior of the agent and of the environment is available. This model represents the agent/environment states, called S , and the set of actions A that the agent can take; typically this is specified by a planning language such as PDDL [17]. Given the set of all possible propositions F , also called *fluents* or *facts*, each possible state of the agent and of the environment $S_i \in S$ is formalised as subsets of F (i.e. $S_i \subseteq F$). Each domain action in A is modelled by a set of preconditions and a set of effects, both over F .

An instance of a GR problem $T = \langle \Pi, I, O, \mathcal{G} \rangle$ is specified by:

1. a given domain $\Pi = \langle F, A \rangle$, which specifies the set F of possible fluents and the set of available actions A ;
2. an initial state of the agent and the environment $I \in S$
3. a sequence of observations $O = \langle obs_1, \dots, obs_n \rangle$, with $n \geq 1$. In this work, each $obs_i \in S$ is an state reached by the agent;
4. a set of possible goals $\mathcal{G} = \{G_1, \dots, G_m\}$, with $m \geq 1$, where each $G_i \subseteq F$.

We define the full sequence of actions $a_i \in A$ performed by the agent to achieve the goal as π . The observation trace σ is a subsequence of states generated by the actions in π . These states might be non-consecutive but they have the same order as in σ . Solving a GR instance consists in identifying $G^* \in \mathcal{G}$ that corresponds to the (unknown) goal of the agent.

There are two typical approaches for GR: the *model-based goal recognition* (MBGR), in which GR is defined as a reasoning task addressable by automated planning techniques [8, 18], and *model-free goal recognition* (MFGR) [3, 7, 13], in which GR is formulated as a classification task addressed through machine learning. MFGR requires minimal information about the domain actions and states (each observation is specified by just a label) and it can operate without the specification of an initial state, which can be completely unknown. Moreover, since running a learned classification model is usually fast, an MFGR system is expected to run much faster than an MBGR system based on planning algorithms. On the other hand, MFGR needs a data set of solved GR instances from which to learn a classification model for the new GR instances of the domain.

Concerning GR systems using neural networks, some works use them for specific applications, such as game playing [5]. As for goal recognition from traces of states, Amado et al. [19] used a pre-trained encoder and a LSTM network for representing and analysing a sequence of observed images representing states. In our approach, our states are instead encoded in PDDL. Amado et al. [20] trained a LSTM-based system to identify missing observations about states in order to derive a more complete sequence of states by which a MBGR system can obtain better performance. Instead, our approach solves directly the goal recognition without any planners involved.

Although the approaches in [7, 12] use similar techniques, with Recurrent Neural Networks trained for goal recognition, one major difference between our work and theirs is that they train a specific machine learning model *for each goal recognition instance*. Instead, in our approach, we train a general-purpose neural network that can be used to solve a large number of different goal recognition instances, without the need of designing or training a new model. Moreover, these approaches work only with traces of actions while we focus on states.

3. Processing State Traces for Goal Recognition

Our approach to goal recognition is depicted in Figure 1. It consists of three main components: the Embedding Component, the Sequential Component and the Instance Component.

The main idea of the first component is to calculate a meaningful representation of a state expressed in PDDL. This component, called *Embedding Component* is shown on the left of Figure 1 and it is made by an undercomplete autoencoder [15] which has to build a shorter representation of each state trying to capture the information necessary to copy its input in the output. Thanks to this component, we can represent a trace of observed states as a sequence of vectors. The analysis of this sequence is made by the *Sequential Component* (middle part of Figure 1), which produces as output a score (between 0 and 1) for each proposition in the domain proposition set F . The third component, called *Instance Component*, can be seen on the right of Figure 1 and it takes as input the proposition ranks generated by the environment component for a GR instance, and uses them to select a goal from the candidate goal set \mathcal{G} .

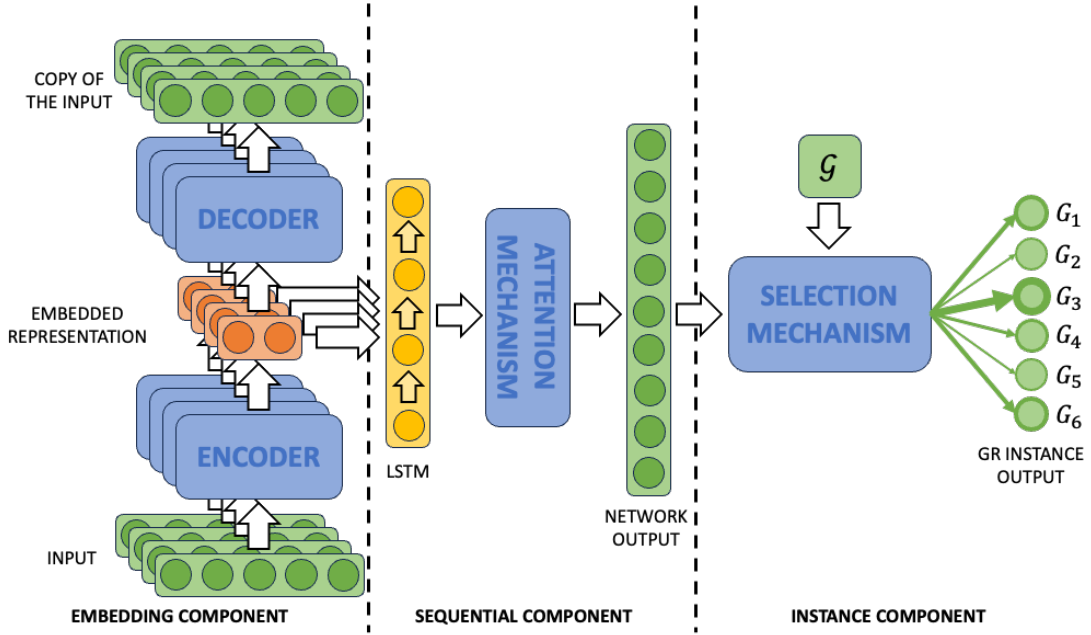


Figure 1: Neural network architecture for performing goal recognition from observed state traces. On the left, the Embedding Component calculates a vectorial representation of each state in the trace through an undercomplete autoencoder. In the middle, the Sequential Component (made by a LSTM layer and an Attention Mechanism) analyse the trace. On the right the Instance Component processes the results of the Sequential Component for selecting the most probable goal among a set of hypotheses.

Please note that the first two components are trained only once for each domain, therefore they can be used for every GR instance over F . The third component does not require any sort of training.

3.1. Embedding Component

The input of the Embedding Component is a sequence of states. Initially, each state s is represented as a binary vector o_s with length $|F|$, into which each component represents a different proposition $f \in F$, where F is lexically ordered. This vector is built as follows:

$$o_s = \begin{cases} 0 & f \notin s \\ 1 & f \in s \end{cases} \quad (1)$$

i.e. the vector assumes value 1 in the positions related to a proposition belonging to s , 0 otherwise.

Each state in the observed trace is then passed to the undercomplete autoencoder, as we show on the left of Figure 1. The autoencoder is composed of two main parts: the Encoder, which reduces the binary vector into a smaller representation (the Embedded representation) made by real numbers, and the Decoder, which processes the embedded representation in order to provide in output a copy of the input.

Given that the Decoder has to reconstruct the binary vector provided as input starting from the Embedded Representation, the main idea behind this architecture is that the Encoder should capture the most important aspects of the inputs and compress them into the Embedded Representation. Therefore, if the Decoder has very good results in the task of reproducing the input, we can assume that the Embedded Representation contains some meaningful information about the state and therefore the representation can be used for goal recognition.

Both the Encoder and the Decoder are made by two feed-forward neural network layers with ReLu activation function. The autoencoder is trained using binary cross-entropy as loss function.

3.2. Sequential Component

The Sequential Component is depicted in the middle of Figure 1. After representing each state as an informative vector of real numbers through the embedding component, the overall observed trace can be seen as a sequence of vectors and it can be processed through a Long Short-Term Memory network (LSTM), which is a kind of neural network specifically designed for processing sequential data like digital signals or free text [21]. In our case, the dataset is made by sequences of observed states.

A LSTM layer is composed of cells, which process each element of the input sequence (each observed state) considering also the previous inputs (states in the sequence). As in GRNet [13], the output of each cell is processed by an Attention Mechanism [22]; in particular, we implement the variant proposed by [23], which computes the weights representing the contribution of each element of the sequence, and generates a unique representation (also called the *context vector*) of the entire plan trace. The context vector is then passed to a feed-forward layer, which has N output neurons with *sigmoid* activation function. N is the number of the domain fluents (propositions) that can appear in any goal of \mathcal{G} for any GR instance in the domain; for our experiments N was set to the size of the domain fluent set F , i.e., $N = |F|$. The neurons should have value 1 if its corresponding fact is part of the agent’s goal. In other words, we have trained our neural network to perform a multi-label classification task, into which each domain fluent can be considered as a different binary class. As loss function, we used binary cross-entropy.

3.3. Instance Component

After the training and optimisation phases of the previous two components, the resulting model can be used for solving many different goal recognition instances.

This is done by the Instance Component (right part of Figure 1), which simply performs an evaluation of the candidate goals in \mathcal{G} of the GR instance, using the output of the environment component fed by the observations of the GR instance. As in the original GRNet, to choose the most probable goal in \mathcal{G} (solving the multi-class classification task corresponding to the GR instance), we use a simple score function that indicates how likely it is that G is the correct goal, according to the output provided by the neural network. This score is defined as:

$$Score(G) = \sum_{f \in G} \bar{o}_f$$

where \bar{o}_f is the network output for fact f of the current GR instance. For each candidate goal $G \in \mathcal{G}$, we consider only the output neurons that have associated facts in G . By summing only these predicted values, we derive an overall score for G being the correct goal. The element with the highest score is the most probable goal in \mathcal{G} .

3.4. Training and Configurations

In order to provide a meaningful representation and to solve goal recognition instances, the Embedding and the Sequential components have to be trained and optimised.

A core part of the optimisation procedure is the hyperparameter tuning. In our work, the number and the dimensions of the feed-forward layers in the Embedding Component, the dimension of the LSTM layer and all the other hyperparameters of the network were selected using the Bayesian-optimisation approach provided by the Optuna framework [24].

However, considering both the Embedding and the Sequential components of our architecture, we have designed two different training and optimisation configurations:

- The **Independent Training** configuration (IT), into which the Embedding Component is trained separately from the Sequential Component. The main idea behind this configuration is that the autoencoder should obtain a meaningful representation of the states by itself and this representation could be exploited in several different applications (such as goal recognition, in our case) without the need to change it. Therefore, in this case the Sequential Component is trained after and independently from the Embedding Component.
- The **Combined Training** configuration (CT), into which after a preliminary training of the Embedding Component (in the same way we described for IT), the training of the Sequential Component has an impact on the Embedding Component too. More in detail, the weights of the latter are specifically fine-tuned for goal recognition. Therefore, the embedded representation produced by the autoencoder becomes more application-oriented and less general.

All the implementation details regarding these configurations are reported in Section 4.1.

4. Experimental Analysis

4.1. Benchmark Suite and Data Sets

We consider two well-known benchmark domains: `DEPOTS` and `ZENOTRAVEL` [25, 26]. Of course GRNet can be trained and tested also using other domains.

Training sets In order to create the (solved) GR instances for the training and test sets in the considered domains, we used automated planning techniques. Concerning the training set, for each domain, we randomly generated a large collection of (solvable) plan generation problems of different size. We considered the same ranges of the numbers of involved objects as in the experiments of Pereira et al. [10]. For each of these problems, we computed up to four

Domain	$ S $	$ F $	$ S_i $	$ G_i $	$ \mathcal{G} $
DEPOTS	1384373	150	340	[2,8]	[7,10]
ZENOTRAVEL	5632466	66	111	[5,9]	[6,11]

Table 1

Number of considered states (S) and facts (F), length of the states ($S_i \in S$), size of the goals ($G_i \in \mathcal{G}$) and length of the goalsets (\mathcal{G}) in the considered GR instances for each considered domain. Interval $[x, y]$ indicates a range of integer values from x to y .

(sub-optimal) plans solving them. As planner we used LPG [27, 28, 29], which allows to specify the number of requested different solutions for the planning problem it solves.

To generate the training set for the Embedding Component, we collected all the different states from the generated plans. This trainset consists of tuples $\langle S_i, S_i \rangle$. Please note that, as a common practice in all autoencoders, in this dataset the input state is the same as the output state in order for the network to first be able to create a hidden representation and then to reconstruct the input. The number of states used to build this dataset is reported in Table 1 (column $|S|$).

To generate the training set for the Sequential Component, we derived the observation sequences from the generated plans by randomly selecting states (preserving their relative order). The selected states are between 30% and 70% of the plan states. The generated training set consists of pairs (O, G^*) where O is a sequence of observed states obtained by sampling a state sequence σ , and G^* is the hidden goal corresponding to the goal of the planning problem reached by σ . For each considered domain, we created a training set with 55000.

For all the experiments, we used 80% of each dataset as actual training set, 10% as validation set and the last 10% as test set.

Test set For evaluating the architecture, we generated a test sets formed by GR instances *not seen at training time*. Such test instances were generated as for the train instances, except that the observation sequences were derived from plans computed by LAMA [30], while for the training instances we used plans computed by LPG; this change is to make the testing more robust. This test set is a generalisation and extension of the test set used in [10] for the same domains that we consider. In particular, for DEPOTS and ZENOTRAVEL, the test set contains 7000 instances.

For each plan generated for being sampled, we removed the last five states as we considered them too informative and we randomly derived three different state traces formed by 30%, 50% and 70% of the plan states, respectively. This gives three groups of test instances, for each considered domain, allowing to evaluate the performance of the presented architectures also in terms of different amounts of available observations.

Table 1 gives information about the size of the GR instances in our test and training sets for each domain, in terms of number of considered states ($|S|$), facts ($|F|$), the maximum number of facts for a given state ($|S_i|$), min/max size of a goal ($|G_i|$) in a goal set \mathcal{G} , and min/max size of a goal set ($|\mathcal{G}|$).

MODEL	DEPOTS			ZENOTRAVEL		
	30%	50%	70%	30%	50%	70%
GRNet	60.9	75.8	86.3	77.0	89.7	96.0
IT	86.8	93.6	96.2	85.4	89.4	91.5
CT	90.3	95.0	97.7	95.2	97.7	98.7

Table 2

Goal recognition accuracy (% of GR instances correctly predicted) by GRNet, and by our architecture in IT and CT configurations on our test set.

Evaluation measures We use the GR *accuracy* for a set of test instances as the main evaluation criteria, which is defined as the percentage of instances whose goals are correctly identified (predicted) over the total number of instances in the test set. If for an instance the evaluated system provides k different goals with the same highest score, then, in the overall count of the solved instances, this instance has value $1/k$ if the true goal is one of these k goals, 0 otherwise.

4.2. Experimental Results

We experimentally evaluate IT and CT configurations, and we use the state-of-the-art system GRNet as a benchmark. In order to have a fair comparison with GRNet, which takes as input actions instead of states, we provided the actions that generate the states used to evaluate our model. For the Embedding Component of both IT and CT, we set the embedded representation dimension to 70.

Table 2 summarizes the performance results for GRNet, IT and CT in terms of accuracy on the test set. As we can see all the tested models perform generally well and they improve their performances with the increase of the percentage of the observed states. In particular, we can see that CT is the model with the highest performance, achieving more than 90 of accuracy in all test configurations. The fact that it performs better than the IT configuration proves that the embedded representation obtained from the autoencoder is not optimal for obtaining the best performance in goal recognition tasks. On the other hand, the good performance of IT proves that the representation provided by the autoencoder of the state is still quite informative.

We can notice that, with 30% of the states, both IT and CT perform significantly better than GRNet; in our opinion this is due to the higher information content of a state with respect to an action which makes goal recognition with few observation easier. In fact we can see that while in ZENOTRAVEL with 30% of the states IT reaches 85.4 of accuracy against the 77.0 of GRNet, with 70% of the states GRNet outperforms IT obtaining 96.0 of accuracy against 91.5 of the latter.

5. Conclusions

We have proposed an extension of GRNet [13], the state-of-the-art technique for goal recognition, for dealing with traces made by observed states. Our systems, through the autoencoder which

composes the Embedding Component, learns a meaningful vectorial representation of a state expressed in PDDL, which is later exploited by the Sequential Component. This part of the architecture is made by a LSTM layer and an Attention Mechanism and it analyses the sequence of states for predicting the goal of the agent. As in GRNet, the learning process is done once for each considered domain, allowing to solve (through the Instance Component) many GR instances.

An experimental analysis shows that our model performs generally well, for the ZENOTRAVEL and DEPOTS benchmark domains, in terms of accuracy. In fact, our model obtains a higher accuracy with respect to the original version of GRNet, which works with traces made by observed actions.

As future work, we intend to investigate the use of other deep learning architectures such as Transformer-based models [31]. Moreover, we aim to study different applications of autoencoders and neural networks in the planning context, such as predicting trajectory constraints [32] or the overall cost of solving a planning problem [33].

References

- [1] R. De Benedictis, M. Castiglioni, D. Ferraioli, V. Malvone, M. Maratea, E. Scala, L. Serafini, I. Serina, E. Tosello, A. Umbrico, M. Vallati, Preface to the Italian Workshop on Planning and Scheduling, RCRA Workshop on Experimental evaluation of algorithms for solving problems with combinatorial explosion, and SPIRIT Workshop on Strategies, Prediction, Interaction, and Reasoning in Italy (IPS-RCRA-SPIRIT 2023), in: Proceedings of the Italian Workshop on Planning and Scheduling, RCRA Workshop on Experimental evaluation of algorithms for solving problems with combinatorial explosion, and SPIRIT Workshop on Strategies, Prediction, Interaction, and Reasoning in Italy (IPS-RCRA-SPIRIT 2023) co-located with 22th International Conference of the Italian Association for Artificial Intelligence (AI* IA 2023), 2023.
- [2] F. A. Van-Horenbeke, A. Peer, Activity, plan, and goal recognition: A review, *Frontiers Robotics AI* 8 (2021).
- [3] H. Geffner, Model-free, Model-based, and General Intelligence, in: Proceedings of IJCAI 2018, 2018.
- [4] L. Batrinca, N. Mana, B. Lepri, N. Sebe, F. Pianesi, Multimodal Personality Recognition in Collaborative Goal-Oriented Tasks, *IEEE Transactions on Multimedia* 18 (2016). doi:10.1109/TMM.2016.2522763.
- [5] W. Min, B. W. Mott, J. P. Rowe, B. Liu, J. C. Lester, Player Goal Recognition in Open-World Digital Games with Long Short-Term Memory Networks, in: Proceedings of IJCAI 2016, IJCAI/AAAI Press, 2016.
- [6] R. Mirsky, Y. Shalom, A. Majadly, K. Gal, R. Puzis, A. Felner, New Goal Recognition Algorithms Using Attack Graphs, in: CSCML 2019, Proceedings, volume 11527, Springer, 2019.
- [7] D. Borrajo, S. Gopalakrishnan, V. K. Potluru, Goal recognition via model-based and model-free techniques, Proceedings of FinPlan 2020 (2020).

- [8] F. Meneguzzi, R. F. Pereira, A Survey on Goal Recognition as Planning, in: Proceedings of IJCAI 2021, 2021.
- [9] M. Ramírez, H. Geffner, Plan Recognition as Planning, in: Proceedings of IJCAI 2009, 2009.
- [10] R. F. Pereira, N. Oren, F. Meneguzzi, Landmark-based approaches for goal recognition as planning, *Artif. Intell.* 279 (2020).
- [11] S. Sohrabi, A. V. Riabov, O. Udrea, Plan Recognition as Planning Revisited, in: S. Kambhampati (Ed.), Proceedings of IJCAI 2016, IJCAI/AAAI Press, 2016.
- [12] M. Maynard, T. Duhamel, F. Kabanza, Cost-Based Goal Recognition Meets Deep Learning, Proceedings of PAIR 2019 (2019).
- [13] M. Chiari, A. E. Gerevini, F. Percassi, L. Putelli, I. Serina, M. Olivato, Goal recognition as a deep learning task: the GRNet approach, volume 33, 2023, p. 560 – 568. doi:10.1609/iccaps.v33i1.27237.
- [14] R. Pereira, N. Oren, F. Meneguzzi, Landmark-based heuristics for goal recognition, in: Proceedings of AAAI 2017, volume 31, 2017.
- [15] G. E. Hinton, R. R. Salakhutdinov, Reducing the dimensionality of data with neural networks, *science* 313 (2006) 504–507.
- [16] M. Ramírez, H. Geffner, Probabilistic Plan Recognition Using Off-the-Shelf Classical Planners, in: Proceedings of AAAI 2010, AAAI Press, 2010.
- [17] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, D. Wilkins, PDDL-the planning domain definition language, Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control (1998).
- [18] M. Ghallab, D. S. Nau, P. Traverso, Automated Planning and Acting, Cambridge University Press, 2016.
- [19] L. Amado, J. P. Aires, R. F. Pereira, M. C. Magnaguagno, R. Granada, F. Meneguzzi, LSTM-Based Goal Recognition in Latent Space, CoRR abs/1808.05249 (2018).
- [20] L. Amado, G. P. Licks, M. Marcon, R. F. Pereira, F. Meneguzzi, Using Self-Attention LSTMs to Enhance Observations in Goal Recognition, in: Proceedings of IJCNN 2020, IEEE, 2020.
- [21] S. Hochreiter, J. Schmidhuber, Long Short-term Memory, *Neural computation* 9 (1997).
- [22] D. Bahdanau, K. Cho, Y. Bengio, Neural Machine Translation by Jointly Learning to Align and Translate, in: ICLR 2015, Conference Track Proceedings, 2015.
- [23] Z. Yang, D. Yang, C. Dyer, X. He, A. J. Smola, E. H. Hovy, Hierarchical Attention Networks for Document Classification, in: K. Knight, A. Nenkova, O. Rambow (Eds.), Proceedings of NAACL HLT 2016, 2016.
- [24] T. Akiba, S. Sano, T. Yanase, T. Ohta, M. Koyama, Optuna: A next-generation hyperparameter optimization framework, in: Proceedings of the 25th ACM SIGKDD, 2019, pp. 2623–2631.
- [25] D. V. McDermott, The 1998 AI Planning Systems Competition, *AI Mag.* 21 (2000) 35–55.
- [26] D. Long, M. Fox, The 3rd International Planning Competition: Results and Analysis, *J. Artif. Intell. Res.* 20 (2003).
- [27] A. Gerevini, A. Saetti, I. Serina, Planning Through Stochastic Local Search and Temporal Action Graphs in LPG, *J. Artif. Intell. Res.* 20 (2003) 239–290.
- [28] A. Gerevini, I. Serina, Planning as propositional CSP: from walksat to local search techniques for action graphs, *Constraints An Int. J.* 8 (2003) 389–413.
- [29] A. Gerevini, A. Saetti, I. Serina, Temporal planning with problems requiring concurrency

through action graphs and local search, in: Proceedings of the 20th International Conference on Automated Planning and Scheduling, ICAPS 2010, Toronto, Ontario, Canada, AAAI, 2010, pp. 226–229.

- [30] S. Richter, M. Westphal, The LAMA planner: Guiding cost-based anytime planning with landmarks, *J. Artif. Intell. Res.* 39 (2010) 127–177.
- [31] L. Serina, M. Chiari, A. E. Gerevini, L. Putelli, I. Serina, A preliminary study on BERT applied to automated planning, in: IPS/RiCeRcA/SPIRIT@AI*IA, volume 3345 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2022.
- [32] L. Bonassi, E. Scala, A. E. Gerevini, Planning with PDDL3 qualitative constraints for cost-optimal solutions through compilation (short paper), in: IPS/RiCeRcA/SPIRIT@AI*IA, volume 3345 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2022.
- [33] F. Percassi, A. E. Gerevini, E. Scala, I. Serina, M. Vallati, Generating and exploiting cost predictions in heuristic state-space planning, in: Proceedings of the International Conference on Automated Planning and Scheduling, volume 30, 2020, pp. 569–573.