Semantically Layered Representation for Planning Problems and Its Usage for Heuristic Computation Using Cellular Simultaneous Recurrent Neural Networks

Michaela Urbanovská and Antonín Komenda
Department of Computer Science (DCS), Faculty of Electrical Engineering (FEE), Czech Technical University in Prague (CTU), Karlovo namesti 293/13 Prague, 120 00, Czech Republic

Keywords: Classical Planning, Cellular Simultaneous Recurrent Neural Networks, Semantically Layered Representation, Learning Heuristic Functions.

Abstract: Learning heuristic functions for classical planning algorithms has been a great challenge in the past years. The biggest bottleneck of this technique is the choice of an appropriate description of the planning problem suitable for machine learning. Various approaches were recently suggested in the literature, namely grid-based, image-like, and graph-based. In this work, we extend the latest grid-based representation with layered architecture capturing the semantics of the related planning problem. Such an approach can be used as a domain-independent model for further heuristic learning. This representation keeps the advantages of the grid-structured input and provides further semantics about the problem we can learn from. Together with the representation, we also propose a new network architecture based on the Cellular Simultaneous Recurrent Networks (CSRN) that is capable of learning from such data and can be used instead of a heuristic function in the state-space search algorithms. We show how to model different problem domains using the proposed representation as well as explain the new neural network architecture and compare its performance in the state-space search against existing classical planning heuristics and heuristics provided by the state-of-the-art.

1 INTRODUCTION

Classical planning in conjunction with machine learning can create a powerful general problem-solving system that can be applied to real-world problems as well as existing classical planning benchmarks. Using machine learning to infer a heuristic and possibly improve planning state-space search algorithm is a widely discussed topic that is being tackled from many sides. Despite many existing state-of-the-art approaches, there is not a single standard technique that could be used for heuristic computation, as problem representation often becomes a cornerstone. Standard modeling languages such as PDDL (Ghallab et al., 1998) has been used in off-shelf planners for years, but their logic-like structure is difficult to process by machine learning techniques such as neural networks. That is why many existing approaches use an alternative problem representation to make the data better structured for neural networks.

One of the first examples is (Groshev et al., 2018) where authors used a 2D grid representation of the Sokoban puzzle to compute a policy using a convolutional neural network (CNN). Following works such as (Urbanovská and Komenda, 2021) and (Urbanovská and Komenda, 2022) have used 2D grid representation for the Sokoban puzzle as well as other different domains that natively contain a grid structure. A similar approach is used with the Cellular Simultaneous Recurrent network (CSRN) in (Ilin et al., 2008) where authors use a neural network to compute a heuristic function for a maze traversal problem.

Another alternative is using the graph representation of the problem and processing it through graph neural networks (GNN). This approach has been used as a domain-independent heuristic in (Shen et al., 2020) as authors represented a relaxed version of the planning problem by a graph to compute the heuristic function. Another approach is (Ståhlberg et al., 2022) which uses graph representation created from PDDL predicates to learn a policy that leads the state-space search. A slightly different approach is proposed in (Toyer et al., 2020) where authors use the PDDL's structure to build the neural network for each planning domain.

The work (Urbanovská and Komenda, 022a) dis-

Urbanovská, M. and Komenda, A.
Semantically Layered Representation for Planning Problems and Its Usage for Heuristic Computation Using Cellular Simultaneous Recurrent Neural Networks.
DOI: 10.5220/0011691000003393
In *Proceedings of the 15th International Conference on Agents and Artificial Intelligence (ICAART 2023) - Volume 3*, pages 493-500
ISBN: 978-989-758-623-1; ISSN: 2184-433X

cusses different possible grid representations of planning problems suitable for machine learning. Many classical planning benchmarks contain an underlying grid structure and therefore problem representations proposed in (Urbanovská and Komenda, 022a) apply to them.

In this work, we propose a novel semantically layered problem representation based on principles introduced in (Urbanovská and Komenda, 022a) together with a modified version of the CSRN architecture suitable for processing such representation. We model several planning domains with underlying grids to show the advantages and limitations of this approach. Lastly, we compare the performance of the trained networks to the existing classical planning heuristics in terms of performance in a search algorithm.

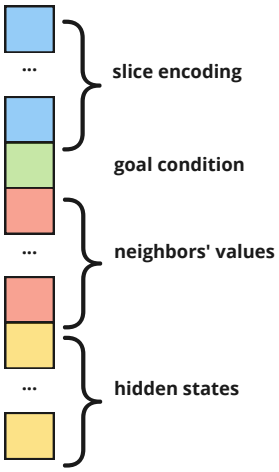

Figure 1: General structure of the vector representation created for every grid cell in the semantically layered representation.

2 SEMANTICALLY LAYERED PLANNING PROBLEM REPRESENTATION

Representation of the planning problems can often be a bottleneck for any machine learning algorithm. Many works focus on problems that are representable in 2D (Groshev et al., 2018), (Urbanovská and Komenda, 2021), (Chrestien et al., 2021), (Ilin et al., 2008), (Urbanovská and Komenda, 2022), however, those are often not domain-independent. This problem has been addressed in (Urbanovská and Komenda, 022a) which showed a possible extension to the 2D grid representation that could lead to a domain-independent representation for many existing planning benchmark domains. It also showed that the domains without an implicitly defined grid can be modified using expert knowledge to fit into the 3D grid representation and still be processable by existing grid-based approaches.

In this work, we build on the ideas proposed in (Urbanovská and Komenda, 022a) and show that they are beneficial for the learning of heuristic functions and that they allow for expressing more complex domains that cannot be formulated on a 2D grid.

We propose the **semantically layered grid representation** which is based on the various semantic elements of the planning problem (a subset of objects, predicates, facts, and/or actions), which can be expressed in STRIPS (Fikes and Nilsson, 1971) or PDDL (Ghallab et al., 1998). Therefore, this approach could be further extended in the future to contain an automatic conversion from PDDL straight to the 3D semantically accurate grid representation. At this time, we are using a handcrafted representation reflecting the problem's semantics.

The semantically layered representation has three dimensions. The first two copy the size of the grid on the input, and the third dimension contains semantically different layers, where each layer corresponds to a different element in the problem instance. The number of layers can be constructed in two ways. The first one is *layer per object type* which creates one semantic layer for one type of object. This way of generating the layers creates a smaller representation in terms of the number of layers, but it can also lead to problems with the encoding of certain problem domains. A good example is the Tetris domain, where we can have multiple blocks of the same type that would be impossible to tell apart in this representation.

The second one is the *layer per object instance* that creates one semantic layer for every instance of every type. One disadvantage of this representation is the high number of layers it can possibly generate. Therefore, it slows down the computation of the network as it enlarges the input data. One great advantage is the amount of information it can represent. Let us take the Tetris example again, where we can represent every Tetris block within its own layer. That provides us with complete information about the problem instance and its objects.

To differentiate between these two representations, we address the *layer per object type* input representation as *one-layer* representation and the *layer per object instance* input representation as *multi-layer* representation.

2.1 Example Problem Formulations

Let us start with the simplest domain originally used with CSRN, which is the maze domain. One instance of the maze problem contains walls and free cells and an agent which is supposed to get to a certain goal location in the maze. This problem can be represented by a 2D grid. We can also model it in the semantically layered representation. Since the maze has only one agent and one goal position, the *one-layer* and *multi-layer* representations are identical.

Figure 2 shows how to create such a representation. We create one layer for the free cells, one layer for the walls, and one layer for the agent. Each layer has binary values based on the objects' positions in the grid. Next, we create the vector representation for each grid cell in every layer as displayed in Figure 1. First, we have $n+1$ values, where $n =$ number of slices present in the representation and the last binary value represents a goal condition. Goal condition means that the object represented by the slice at that grid cell should appear in a goal state. For example, the maze would have that value equal to one at the agent slice on the grid cell that corresponds to the goal's location. Next, there are m values that are received from the neighboring grid cells every iteration, and lastly, we have h hidden states produced by the cell network in the last iteration. Examples of the vectors representations for the maze domains are shown in Figure 3.

Now, let us take the Sokoban puzzle as an example of a more complex domain. A Sokoban instance has walls, free tiles, boxes, an agent, and goal positions which have to have boxes on them to reach the goal. Different entities can overlap each other, therefore using a 2D grid by itself can be limiting for the neural network to express that a box is standing on a goal cell or that an agent is standing on a goal cell. This can be easily solved by the semantically layered representation. We create a layer for walls, spaces we can step on, the agent, boxes, and goal positions. This would be when using the *one-layer* representation. In the bottom row, we can see the difference when using the *multi-layer* representation, where we have a separate slice for every box and every goal position. Since boxes and goal positions are represented by a one grid cell and not multiple grid cells, we can tell them apart even in the *one-layer* representation. Also in Sokoban, the boxes and goals are homogeneous, meaning there is no difference in what box ends up on what goal position, which allows for the more concise *one-layer* representation.

3 SEMANTICALLY LAYERED CSRN

Since we proposed a new semantically layered representation of the grid-based planning problems, we have to create a new architecture that can process it. Therefore, we propose the semantically layered CSRN (slCSRN). This architecture is scale-free and holds the original CSRN's principles and extends them to the semantically layered representation with the same goals—to compute heuristic values usable in the state space search algorithms with grid-based planning problems.

Neighborhood Function. Originally, the CSRN architecture used the 4-neighborhood of every grid cell for sending and receiving information from other cells. By adding a third dimension to the input, we have to communicate across the layers as well. Since we want this neighborhood function to stay domain-independent, we replaced it with the 6-neighborhood function that communicates with all surrounding grid cells and also wraps in all 3 dimensions. The original 4-neighborhood was corresponding to the available actions in the maze domain. Therefore, we hypothesize that this can cause worse results as the neighborhood no longer correspond to actual planning actions.

Vector Representation of Grid Tiles. The cell networks process each grid cell's vector representation that is based on the information about the cell, values sent by neighboring cells, and hidden states produced by the cell network in the previous recurrent iteration. In (Ilin et al., 2008), the vector representation included information about the grid cell being a wall or a goal that was caused by the usage on one domain. In different domains, we need to express more than that, and extending to the 3D also calls for encoding the identity of the grid layer we are processing. Therefore, the first n values in the vector representation include one binary value for every layer type, therefore every layer's grid cells know their semantics. The general schema for the vector representation is shown in Figure 1.

Goal Condition. Another change to the vector representation of the grid cells is also the goal condition binary value that is placed after the layer encoding. The goal condition is equal to one in layers that have an effect on the goal at the position that is supposed to be present in the goal state. That means that we no longer just have a grid cell equal to a goal (position) like we had in the maze domain. Now we can express more complex goal conditions similar to how we express them in STRIPS, using a set of facts that have to be true in a goal state. For example, in the Tetris domain, we can now express in every block's

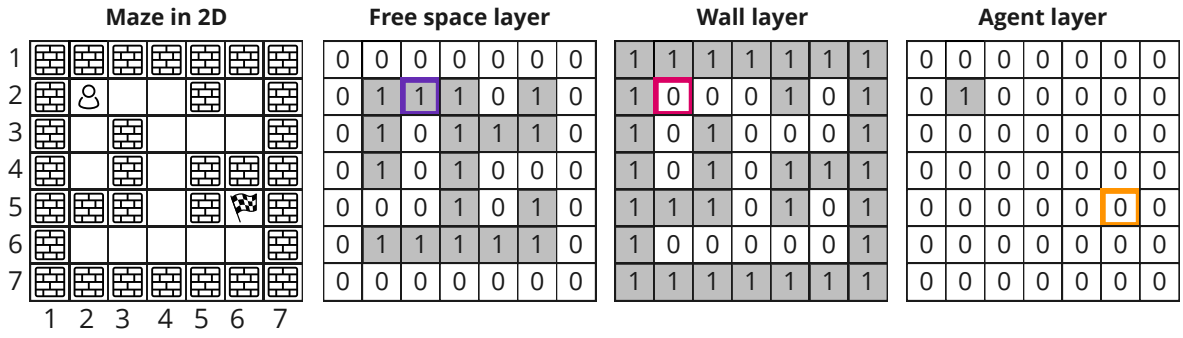

Figure 2: One-layer representation of an instance of the maze problem. Maze in 2D represents the maze on the input Next we see three semantic layers and their binary encoding on the input.

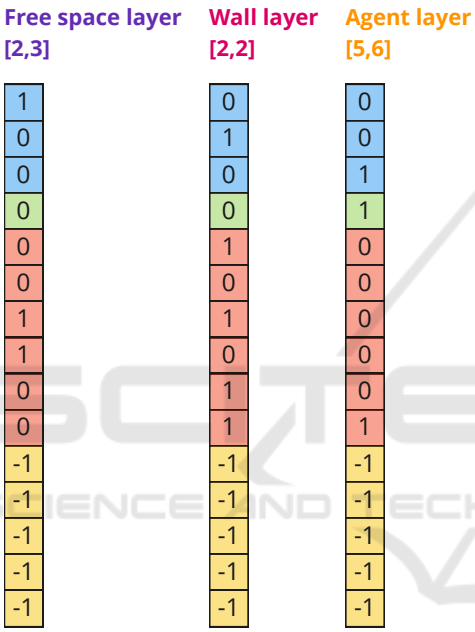

Figure 3: Vector representations of selected color-coded grid cells from the semantic layers of a maze instance shown in Figure 2 created based on the general vector representation template shown in Figure 1.

layer that its position is required in the bottom half of the grid to meet its goal condition.

Output Interpretation. Interpreting the output of the CSRN architecture is a challenge for every domain that is not fully representable in 2D. Since we are using 3D input representation, but we still need the heuristic represented on a 2D grid, we had to create a new computation of the output values that can be further used as a heuristic. The process of interpreting the heuristic is displayed in Figure 4.

At the beginning of the computation, we take the 2D input grid and create the semantically layered representation and binary masks that show the positions of each object in its respective semantical layer. We then process the semantically layered representation through our architecture and receive the output which has one layer corresponding to each input layer. By masking every output layer with the binary masks we created at the beginning, we have a masked input that only highlights values relevant to the position of the object in the current state that is on the input. We then flatten the masked output layers and obtain a 2D heuristic projection we can use in the search. This approach is inspired by the potential heuristic (Pommerening et al., 2015) from Classical Planning, where we sum the potentials of facts that are present in the state despite computing the potentials for all of them.

3.1 Unfolded-slCSRN

The new input representation provides us with more expressivity and information about the problem's semantics. However, it also requires more communication across the semantically layered grid to propagate any information. The capacity of a CSRN network where all cell networks share weights is relatively small, and we hypothesized that a very low number of parameters might be limiting to learning the heuristic function. Therefore, we created an alternative to the slCSRN with more trainable parameters, but the same scalability. We call it the unfolded-slCSRN. The structure of the network is the same as slCSRN, but every recurrent iteration has its own trainable set of weights that are shared among all cell networks.

3.2 Training

Training of the slCSNR and unfolded-slCSRN is done in the same manner as in (Urbanovská and Komenda, 2022). We use ADAM optimizer (Kingma and Ba, 2015) together with the monotonicity measuring loss function as described in (Urbanovská and Komenda, 2021).

As a side metric, we use the error function described in (Urbanovská and Komenda, 2022) that

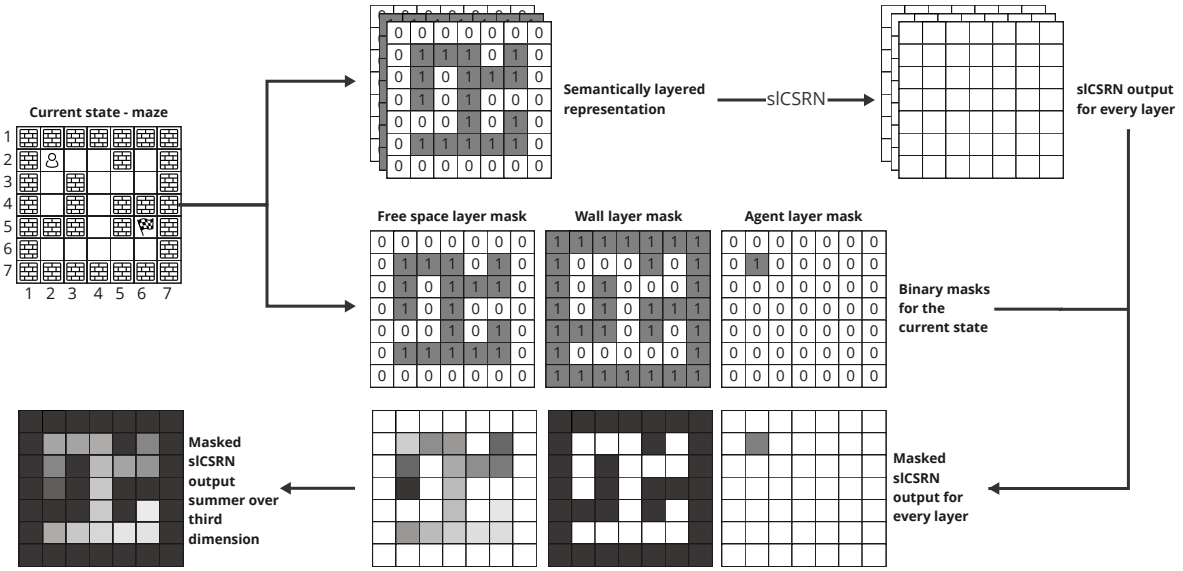

Figure 4: Example of the heuristic interpretation for an instance from the maze domain.

measures the number of erroneous decisions in the search that are not present in any plan that solves the problem.

To be able to compare the results with the existing state of the art, we train slCSRN and unfolded-slCSRN for the maze and the Sokoban puzzle domains. The training data set for the maze domain contains 10 samples of size 5×5. The base for the Sokoban puzzle training sets contains 100 fully evaluated 3*x*3 maps with one box and 28 3*x*3 maps with two boxes. Every batch is constructed so it contains samples from only one problem instance, which leaves us with 128 batches to train with. We trained the architectures on the whole training set as well as only 10 randomly selected batches to see how the selection of training data influences the performance of the trained networks.

The parameters of the architecture were selected from the following

- number of recurrent iterations = [10, 20, 30]
- number of hidden states = [5, 15, 30]

We also used both *one-layer* and *multi-layer* representations to see if they have any impact on the abilities of the trained networks.

4 EXPERIMENTS

The experimental evaluation is comparing the results with versions of CSRN from (Urbanovská and Komenda, 2022) and state-of-the-art classical planning heuristics. We take the selected trained networks and plug them into a search algorithm as a heuristic function. We measure three metrics in total to determine the performance of the trained networks—**average path length** (avg_pl), **average number of expanded states** (avg_ex), and **coverage** (cvg). Coverage is the most important of these as it shows the percentage of solved problems in the provided set.

4.1 Comparison of Trained Networks

We trained both slCSRN and unfolded-slCSRN with the *one-layer* and *multi-layer* representation to see how the performance changes and if there is a trade-off between the complexity of the input, training time, and final performance.

Of the two selected problem domains, Sokoban is the more complex one, and therefore we choose it as the main indicator of the performance of the individual architecture versions and input format combinations. Each slCSRN and unfolded-CSRN was trained on the full training data set (128 batches) and 10 randomly selected batches as described in Section 3.2.

We can see that training on the whole available training set produces consistently better errors on both train and validation sets. In general, the unfolded version of the slCSRN showed better results in both *one-layer* and *multi-layer* representations, so we assume that the higher number of trainable parameters positively influences the ability of the network to learn. The configuration with the best error values overall is the unfolded-slCSRN trained on the whole training set. To be precise, it is its parametrization with 10 recurrent iterations and 15 hidden states. Its conver-

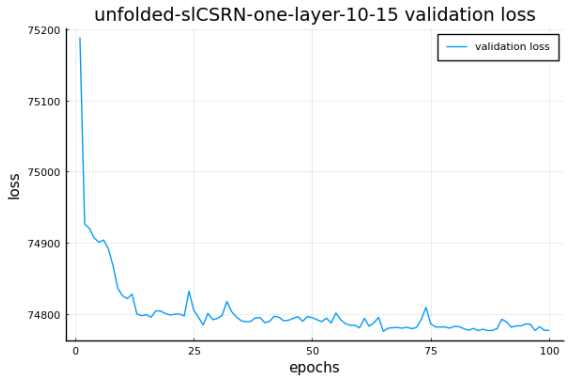

Figure 5: Convergence graph of the validation error for the unfolded-slCSRN architecture trained on Sokoban 128 batches with *one-slice* representation.

gence of the validation error is shown in Figure 5.

Based on these results, we selected the unfolded-slCSRN architecture as one of our heuristic functions for the following experiments. The unfolded-slCSRN trained on the maze domain achieved zero error on the configuration with 10 recurrent iterations and 30 hidden states so it is selected for the planning experiments as well.

4.2 Planning Experiments

We have selected one network for the maze domain and four networks for the Sokoban puzzle domain to run in the planning experiments and be used as a heuristic function during the search. Since we are trying to learn the monotonicity property of the heuristic function, we are using the Greedy Best First Search algorithm as the state space search algorithm, as it is guided solely by the heuristic values.

Although it is not our goal in this work to outperform the existing approaches, we seek to see how the performance changes. The unfolded-slCSRN has to deal with a much larger input and also contains more data processing when converting the 2D representation of problems to the *one-layer* semantically layered representation. All these actions may impact the performance in search as they slow down the network's evaluation.

The maze domain is evaluated on four data sets of sizes 8×8, 16×16, 32×32, and 64×64 where each one contains 50 unseen maze instances. The time limit for one instance is 10 minutes.

The Sokoban domain is evaluated on three data sets, 8×8 with two boxes, 10×10 from the Boxoban data set (Guez et al., 2018), 16×16 with two boxes where each set contains 50 unseen samples. The time limit for one instance is 10 minutes.

For both domains, we compare the results with the state-of-the-art planning heuristics LM-Cut (Pommerening and Helmert, 2013) and h^{FF} (Hoffmann, 2001) as well as with the blind heuristic and Euclidean distance as baselines. We also included the results from (Urbanovská and Komenda, 2022) that achieved the best performance so far, to see the influence of the semantically layered representation on the performance compared to the CSRN that uses the flat 2D representation.

4.3 Discussion

The results for the maze domain are presented in Table 1. We compare the results with both state-of-the-art heuristics and the best CSRN network from (Urbanovská and Komenda, 2022). The coverage on all four data sets is equal to one, which means that all problems were solved. That is by itself a success, as the larger input for the network and its extended structure do not negatively influence the evaluation time such that the search would be excessively slowed down and unable to find the solutions in time.

The smallest 8×8 data set is the only one that does not have an average path length equal to the optimal value. The average number of expanded states is larger than in the case of the CSRN. The maze domain does not benefit from the multidimensional representation due to its low complexity. Therefore, we were not expecting a great impact on the results when using the semantically layered representation. This hypothesis showed to be true, as the results of the unfolded-slCSRN are on par with the CSRN.

The results for the Sokoban domains are in Table 2. Since Sokoban is a PSPACE-complete problem (Culberson, 1997), the performance difference is expected to be a lot more prominent. We can see this on coverage, where the maximum amount of solved problems is equal to 96%. The best coverage achieved on the 10×10 data set is 10%.

A possible cause, in this case, could be the evaluation time of the new architectures. Especially the ones using the multi-layer representation which is more costly to evaluate as the input's size increases with the number of objects in the problem instance. That prolongs the runtime of the whole algorithm. This can be seen at the unfolded-slCSRN-multi-layer as it reaches the lowest coverage on the 10×10 data set and also at the slCSRN-multi-slice as it has the lowest coverage at the 8×8 data set.

The overall best results from the proposed architectures are the unfolded-slCSRN-one-layer results. This can be due to the increased information capacity that comes with the unfolded version and also with the relatively small input size. This suggests that the

Table 1: Planning experiments for the maze domain. All best results are in bold lettering.

	8x8			16x16			32x32			64x64		
	avg pl	avg ex	cvg	avg pl	avg ex	cvg	avg pl	avg ex	cvg	avg pl	avg ex	cvg
blind	11.64	27.42	**1**	**23.36**	96.82	**1**	**46.2**	267.28	**1**	**104.72**	1085.66	**1**
ED	**10.76**	14.58	**1**	**23.36**	48.14	**1**	**46.2**	129.7	**1**	**104.72**	561.44	**1**
h^{FF}	**10.76**	**10.76**	**1**	**23.36**	**23.36**	**1**	**46.2**	**46.2**	**1**	**104.72**	**104.72**	**1**
LM-cut	**10.76**	**10.76**	**1**	**23.36**	**23.36**	**1**	-	-	0	-	-	0
CSRN-ADAM-20-5	**10.76**	17.28	**1**	**23.36**	77.48	**1**	**46.2**	305.52	**1**	**104.72**	1292.78	**1**
unfolded-slCSRN-10-30	10.80	21.66	**1**	**23.36**	100.20	**1**	**46.2**	392.38	**1**	**104.72**	1391.12	**1**

Table 2: Planning experiments for the Sokoban domain. All best results are in bold lettering.

	8x8			10x10 - Boxoban			16x16		
	avg pl	avg ex	cvg	avg pl	avg ex	cvg	avg pl	avg ex	cvg
blind	111.24	3.5k	**1**	1.2k	66k	**1**	30k	52k	**0.54**
ED	**31.10**	**0.5k**	**1**	**45.64**	**8.2k**	**1**	**115.33**	**12.6k**	**0.54**
h^{FF}	-	-	0.04	-	-	0	-	-	0
LM-cut	-	-	0	-	-	0	-	-	0
CSRN-ADAM-10-15	51.44	7.3k	**1**	58.33	51.5k	0.36	224.93	263.4k	0.28
sl-CSRN-one-layer	34.52	1.1k	0.96	37.5	1.1k	0.08	-	-	0
sl-CSRN-multi-layer	35.0	1.3k	0.92	29.5	378.75	0.08	-	-	0
unfolded-slCSRN-one-layer	37.67	1.0k	0.96	37.8	1.7k	0.1	-	-	0
unfolded-slCSRN-multi-layer	34.04	1.2k	0.96	29.0	452.5	0.04	-	-	0

Sokoban puzzle might require the level of semantics provided by the *one-layer* representation and it does not benefit from a further extension of the information on the input by using the *multi-layer* representation.

Another reason could be the output interpretation described in Section 3. By getting rid of the third dimension of the output due to practical reasons, we might be losing a certain amount of information that might improve the heuristic estimate. In the future, we would like to focus on different ways of interpreting the heuristic that would avoid this issue and might provide even better information on how to guide the state space search.

Even though we did not outperform the state-of-the-art results of the CSRN architecture in the planning experiments, we have shown that the semantically layered representation can be used in the search, in the maze domain even without significantly slowing down the algorithm and compromising the results. In the Sokoban domain, we showed how the amount of training data influences the learning as the networks trained on the full available data set performed the best. The results encourage us to implement more domains into this representation and look for ones that are possibly going to benefit from the *multi-layer* representation and the extended information it provides.

5 CONCLUSIONS

We have proposed two novel representations for grid-based planning problems that can be used as a domain-independent representations. Both the *one-layer* and the *multi-layer* representation provide additional semantics to the planning problem, in opposition to the 2D grid representation used in the previous state of the art.

To process this new representation, we proposed two versions of the CSRN architecture. The slCSRN with similar principles and weights shared among the cell networks. And the unfolded-slCSRN provides a larger information capacity as it shares a different set of weights among the cell networks in every recurrent iteration.

We trained the slCSRN and unfolded-slCSRN with both *one-layer* and *multi-layer* representations and compared the performance of the trained networks to classical planning heuristics and the CSRN architecture.

The results for the maze domain were on par between the CSRN and unfolded-slCSRN. The main difference was seen in the Sokoban domain. Its performance is influenced by the complexity and runtime of the newly introduced unfolded-slCSRN network, as well as by the exponential state space the Sokoban puzzle has.

This representation may be a great step in the possible domain-independent heuristic computation for planning problems on grids. As proposed in (Urbanovská and Komenda, 022a), many planning benchmarks can be modeled on grids even without the necessary underlying structure. This representation allows us to use graph-based methods as well as image-based methods to analyze heuristic computation for planning problems using neural networks.

In the future, we would like to focus on modeling more problem domains and extending the results to a more domain-independent setting. We would also like to create a system that would be able to create the semantically layered representation solely from the PDDL as its structure copies the structure of the planning problem.

ACKNOWLEDGEMENTS

The work of Michaela Urbanovská was supported by the OP VVV funded project CZ.02.1.01/0.0/0.0/16019/0000765 "Research Center for Informatics" and by the Grant Agency of the Czech Technical University in Prague, grant No. SGS22/168/OHK3/3T/13. The work of Antonín Komenda was supported by the Czech Science Foundation (grant no. 22-30043S).

REFERENCES

Chrestien, L., Pevný, T., Komenda, A., and Edelkamp, S. (2021). Heuristic search planning with deep neural networks using imitation, attention and curriculum learning. *CoRR*, abs/2112.01918.

Culberson, J. (1997). Sokoban is pspace-complete.

Fikes, R. E. and Nilsson, N. J. (1971). Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208.

Ghallab, M., Knoblock, C., Wilkins, D., Barrett, A., Christianson, D., Friedman, M., Kwok, C., Golden, K., Penberthy, S., Smith, D., Sun, Y., and Weld, D. (1998). Pddl - the planning domain definition language.

Groshev, E., Tamar, A., Goldstein, M., Srivastava, S., and Abbeel, P. (2018). Learning generalized reactive policies using deep neural networks. In *2018 AAAI Spring Symposium Series*.

Guez, A., Mirza, M., Gregor, K., Kabra, R., Racaniere, S., Weber, T., Raposo, D., Santoro, A., Orseau, L., Eccles, T., Wayne, G., Silver, D., Lillicrap, T., and Valdes, V. (2018). An investigation of model-free planning: boxoban levels. https://github.com/deepmind/boxoban-levels/.

Hoffmann, J. (2001). Ff: The fast-forward planning system. *AI magazine*, 22(3):57–57.

Ilin, R., Kozma, R., and Werbos, P. J. (2008). Beyond feedforward models trained by backpropagation: A practical training tool for a more efficient universal approximator. *IEEE Transactions on Neural Networks*, 19(6):929–937.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Pommerening, F. and Helmert, M. (2013). Incremental lmcut. In *Twenty-Third International Conference on Automated Planning and Scheduling*.

Pommerening, F., Helmert, M., Röger, G., and Seipp, J. (2015). From non-negative to general operator cost partitioning. In Bonet, B. and Koenig, S., editors, *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, pages 3335–3341. AAAI Press.

Shen, W., Trevizan, F. W., and Thiébaux, S. (2020). Learning domain-independent planning heuristics with hypergraph networks. In Beck, J. C., Buffet, O., Hoffmann, J., Karpas, E., and Sohrabi, S., editors, *Proceedings of the Thirtieth International Conference on Automated Planning and Scheduling, Nancy, France, October 26-30, 2020*, pages 574–584. AAAI Press.

Ståhlberg, S., Bonet, B., and Geffner, H. (2022). Learning general optimal policies with graph neural networks: Expressive power, transparency, and limits. In Kumar, A., Thiébaux, S., Varakantham, P., and Yeoh, W., editors, *Proceedings of the Thirty-Second International Conference on Automated Planning and Scheduling, ICAPS 2022, Singapore (virtual), June 13-24, 2022*, pages 629–637. AAAI Press.

Toyer, S., Thiébaux, S., Trevizan, F. W., and Xie, L. (2020). Asnets: Deep learning for generalised planning. *J. Artif. Intell. Res.*, 68:1–68.

Urbanovská, M. and Komenda, A. (2021). Neural networks for model-free and scale-free automated planning. *Knowledge and Information Systems*, pages 1–36.

Urbanovská, M. and Komenda, A. (2022). Learning heuristic estimates for planning in grid domains by cellular simultaneous recurrent networks. In Rocha, A. P., Steels, L., and van den Herik, H. J., editors, *Proceedings of the 14th International Conference on Agents and Artificial Intelligence, ICAART 2022, Volume 2, Online Streaming, February 3-5, 2022*, pages 203–213. SCITEPRESS.

Urbanovská, M. and Komenda, A. (2022a). Grid representation in neural networks for automated planning. In Rocha, A. P., Steels, L., and van den Herik, H. J., editors, *Proceedings of the 14th International Conference on Agents and Artificial Intelligence, ICAART 2022, Volume 3, Online Streaming, February 3-5, 2022*, pages 871–880. SCITEPRESS.