Learning General Policies for Planning through GPT Models

Nicholas Rossetti, Massimiliano Tummolo, Alfonso Emilio Gerevini,
Luca Putelli, Ivan Serina, Mattia Chiari, Matteo Olivato

19/03/2025

LLMs still can't plan

- Models pre-trained on language datasets cannot plan using prompting techniques [Arora et al 2023, Valmeekam et al 2022, 2023]

- Via fine-tuning, Plansformer plans for various domains but with a small number of objects [Pallagani et al 2023a, 2023b]

- Does the problem lie on the **Transformer architecture** or on **the language pre-training dataset**?

Generative Pre-Trained Transformer (GPT)

- Once trained, given a sequence of initial words, GPT **predicts the next word** (e.g. names, verbs or adjectives)

- The predicted word **is added to the input** and GPT **repeats** the full process auto-regressively, obtaining the next word

- To be processed by GPT a sentence is divided into tokens (words) which are **embedded into a real number vector**

General Policies in Classical Planning

Given the current state and goal of the problem, a **general policy** is a function that provides the next action to apply:

$$\pi(s_i, Goal) = a \qquad a \in A(s_i)$$

A policy $\boldsymbol{\pi}$ solves a set of classical planning instances for the same domain $\boldsymbol{D}$ if each of these instances $\boldsymbol{I} = (\boldsymbol{O}, \boldsymbol{Init}, \boldsymbol{Goal})$ is **solved by the sequence of actions**:

$$\pi(s_0, Goal), \dots, \pi(s_n, Goal)$$

$$where\ s_0 = Init\ and\ Goal \subseteq s_{n+1}$$

General Policies in Classical Planning

Given the current state and goal of the problem, a **general policy** is a function that provides the next action to apply:

$$\pi(Init, G, P) = a \quad a \in A(s) \quad s = state\ produced\ by\ P$$

A policy $\boldsymbol{\pi}$ solves a set of classical planning instances for the same domain $\boldsymbol{D}$ if each of these instances $\boldsymbol{I} = (\boldsymbol{O}, \boldsymbol{Init}, \boldsymbol{Goal})$ is **solved by the sequence of actions**:

$$\pi(s_0, Goal), \ldots, \pi(s_n, Goal)$$

$$where\ s_0 = Init\ and\ Goal \subseteq s_{n+1}$$

Can LLMs plan, if trained?

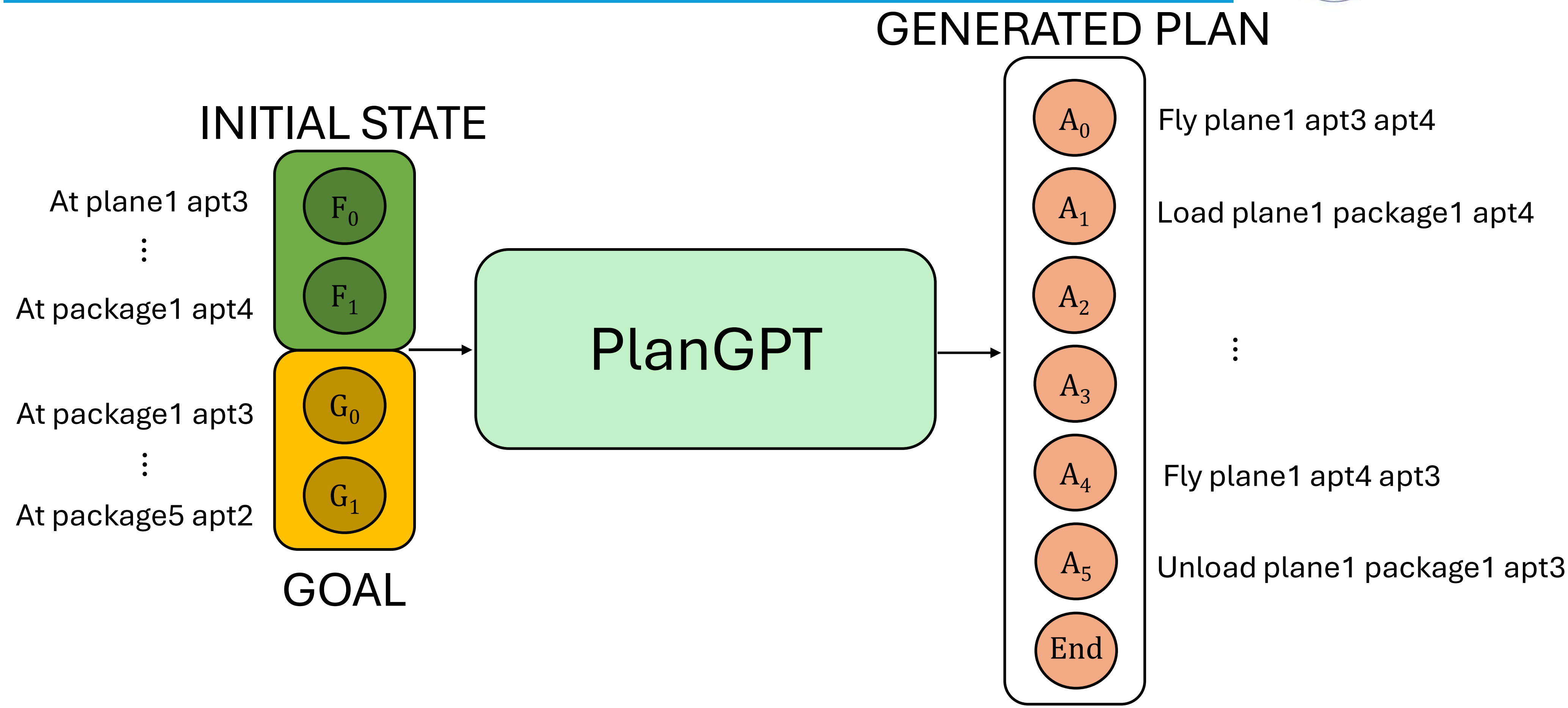
UNIVERSITÀ DEGLI STUDI DI BRESCIA

- We trained a GPT-2 model **from scratch** to obtain a **general policy**, using a dataset of **solved planning problems**

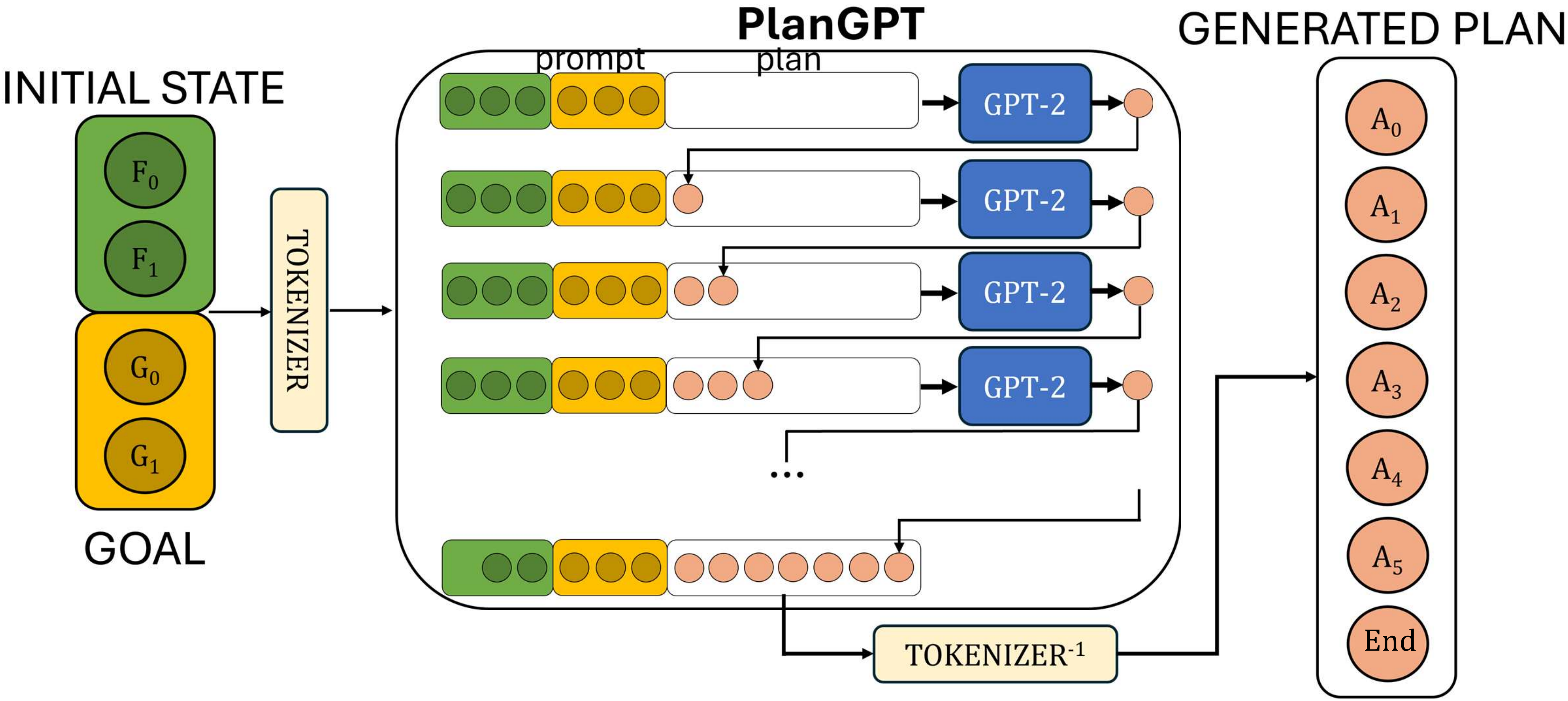
- We use the **initial state** and the **goal** of the problem as a prompt to GPT

- Given this prompt, we train GPT (**PlanGPT**) to **generate a valid plan**

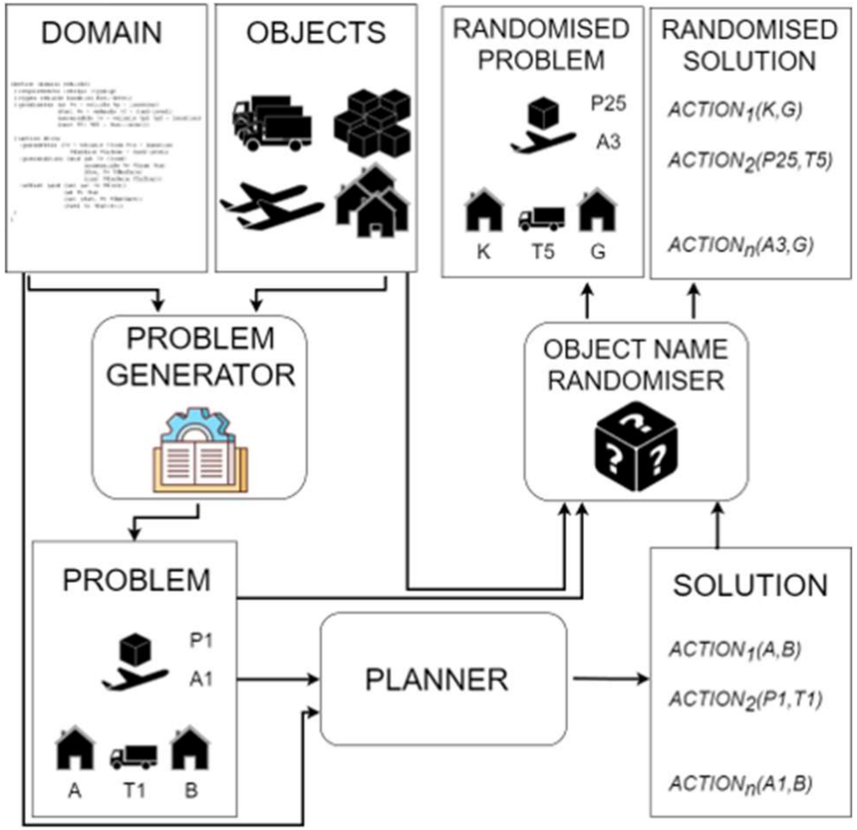
GPT for Planning (General Policies)



GPT for Planning (General Policies)



Dataset Generation



- We generated problems using a **problem generator**
- We solved these problems using **LPG** [Gerevini et al 2002]
- We **randomize the objects' names**

truck1 → **truck4**

city1 → **city7**

Training PlanGPT – the loss function

- The tokenizer splits each predicate and each action in their components: **predicate/action name and its objects**

at truck4 pos5 → **at - truck4 - pos5**

drive truck4 pos5 pos7 → **drive - truck4 - pos5 – pos7**

- We train PlanGPT by generating the next token and comparing it with the token in the solution plan using the **Cross Entropy loss**

$$L = -\sum_{t=1}^{|V|} \log(y'_t) * y_t$$

Cross Entropy Loss - Problem

- The use of the Cross Entropy Loss forces the learned model **to mimic the example label** (*a specific plan*)

- **Problem:** **Valid plans** different from the label plan are **considered incorrect** in the loss function!

Coverage Early Stopping

- **Solution:** We can mitigate this problem by *evaluating the model capability of generating valid plans during training*: let's include a **plan validator**!

- We evaluate our model based on the **coverage obtained on the validation set** instead of the validation loss using the **early stopping** technique.

Experimental Setup

- We generate **70k problems**, divided in 63k for training set (obtaining more than **200k plans**), **1k** for validation set and **6k** for test set (**Tset**).

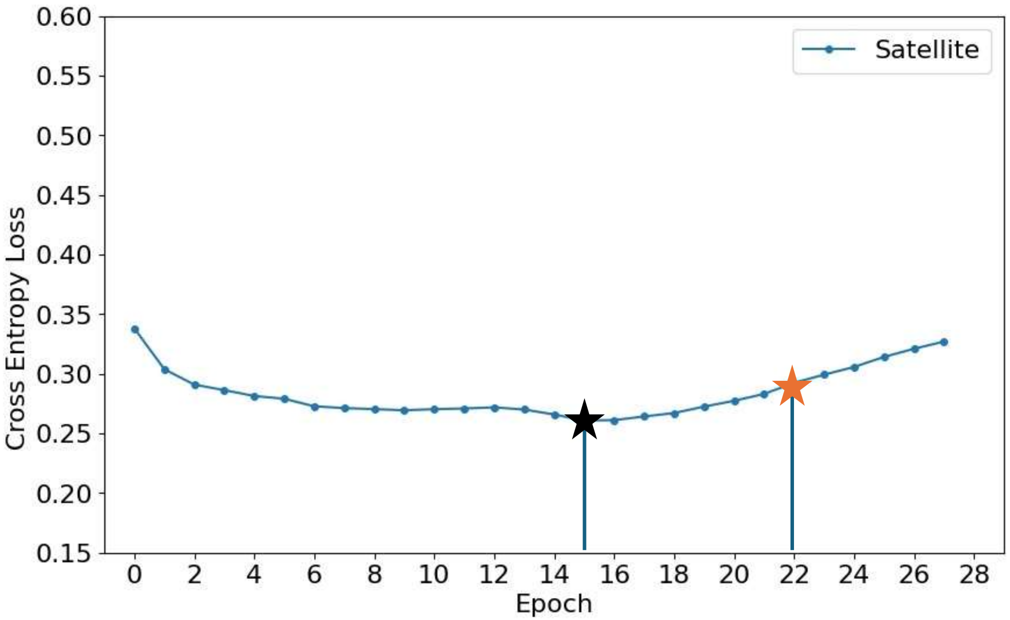
- We trained PlanGPT on **8 classical planning domains**: Blocksworld, Depots, Driverlog, Floortile, Logistics, Satellite, Visitall and Zenotravel

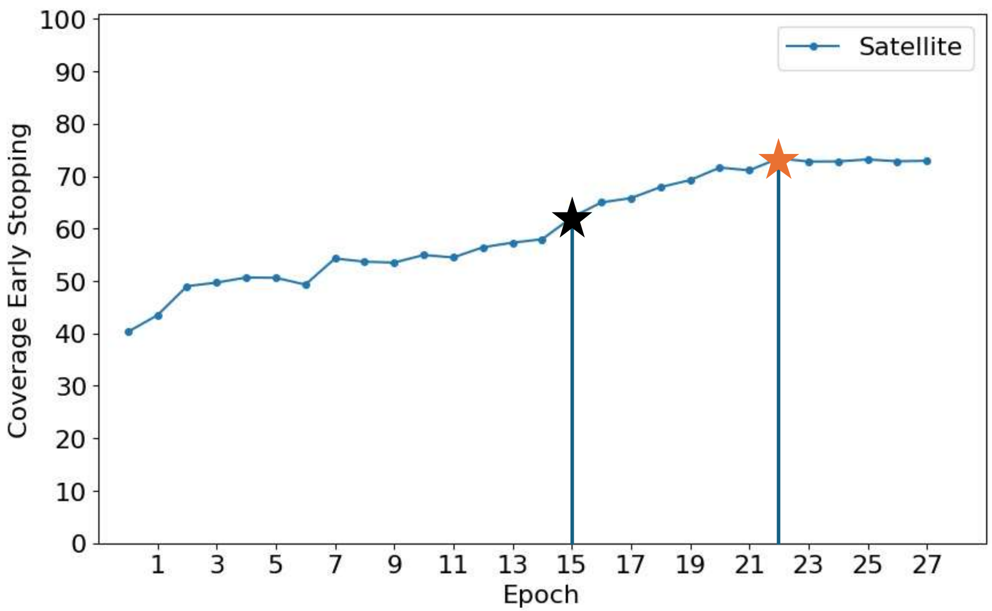
- We validate PlanGPT using classical planning metrics: **Coverage**, **IPC-Q** and **IPC-A**

Results – Coverage Early Stopping

Domain	Cross Entropy Loss Optimization	Coverage Early Stopping Optimization
Blocksworld	**100.0**	**100.0**
Depots	90.3	**94.5**
Driverlog	94.7	**96.5**
Floortile	98.2	**99.6**
Logistics	76.3	**77.3**
Satellite	81.3	**90.1**
Visitall	99.9	**100.0**
Zenotravel	**94.7**	**94.7**

Results – Coverage with Early Stopping





Comparison with Relational GNNs

- We compare with general policies obtained by Relational GNNs (Ståhlberg et al 2022a, 2022b):

Tset	Coverage		IPC-A		IPC-Q	
Domain	PlanGPT	R-GNN	PlanGPT	R-GNN	PlanGPT	R-GNN
Blocks	**100.0**	81.4	**1763.1**	1093.7	**1847.1**	1459.0
Logistics	**77.3**	21.6	**4752.2**	791.7	**5125.1**	772.1
Visitall	**100.0**	96.0	**5754.5**	3176.4	**6046.4**	6002.0

Conclusions

UNIVERSITÀ
DEGLI STUDI
DI BRESCIA

- We propose a method to learn a general policy based on GPT, called **PlanGPT**

- Our training procedure exploits an early stopping technique that we designed to increase the coverage

- For several domains, **PlanGPT** solves most of the problems in test sets, showing **competitive performance** w.r.t. SotA

- PlanGPT is limited to its **vocabulary** and **context window** and fails to generalize to problems **with more objects** compared to those in training

Future work

- We are exploring the use of PlanGPT has a heuristic: **providing a "good" plan seed to a plan-repair system (LPG)**

- We are investigating the integration of PlanGPT with **validator-driven RL techniques**, as in an RLHF setting

- We are investigating the integration of a **plan validator in the generation strategy** of PlanGPT

Thank you

Any questions?

Bibliography

- Arora, D.; and Kambhampati, S. 2023. Learning and Leveraging Verifiers to Improve Planning Capabilities of Pretrained Language Models.CoRR, abs/2305.17077
- Gerevini, A.; and Serina, I. 2002. LPG: A Planner Based on Local Search for Planning Graphs with Action Costs. In AIPS, 13–22. AAAI Press.
- Pallagani, V.; Muppasani, B.; Murugesan, K.; Rossi, F.; Srivastava, B.; Horesh, L.; Fabiano, F.; and Loreggia, A. 2023a. Understanding the Capabilities of Large Language Models for Automated Planning. CoRR, abs/2305.16151.
- Pallagani, V.; Muppasani, B.; Srivastava, B.; Rossi, F.; Horesh, L.; Murugesan, K.; Loreggia, A.; Fabiano, F.; Joseph, R.; and Kethepalli, Y. 2023b. Plansformer Tool: Demonstrating Generation of Symbolic Plans Using Transformers. In IJCAI, 7158–7162. IJCAI Org.
- Ståhlberg, S.; Bonet, B.; and Geffner, H. 2022a. Learning General Optimal Policies with Graph Neural Networks: Expressive Power, Transparency, and Limits. In ICAPS, 629–637. AAAI Press.
- Ståhlberg, S.; Bonet, B.; and Geffner, H. 2022b. Learning Generalized Policies without Supervision Using GNNs. In KR, 474–483. IJCAI Org.
- Valmeekam, K.; Hernandez, A. O.; Sreedharan, S.; and Kambhampati, S. 2022. Large Language Models Still Can't Plan (A Benchmark for LLMs on Planning and Reasoning about Change). CoRR, abs/2206.10498.
- Valmeekam, K.; Sreedharan, S.; Marquez, M.; Hernandez, A. O.; and Kambhampati, S. 2023. On the Planning Abilities of Large Language Models (A Critical Investigation with a Proposed Benchmark). CoRR, abs/2302.06706.