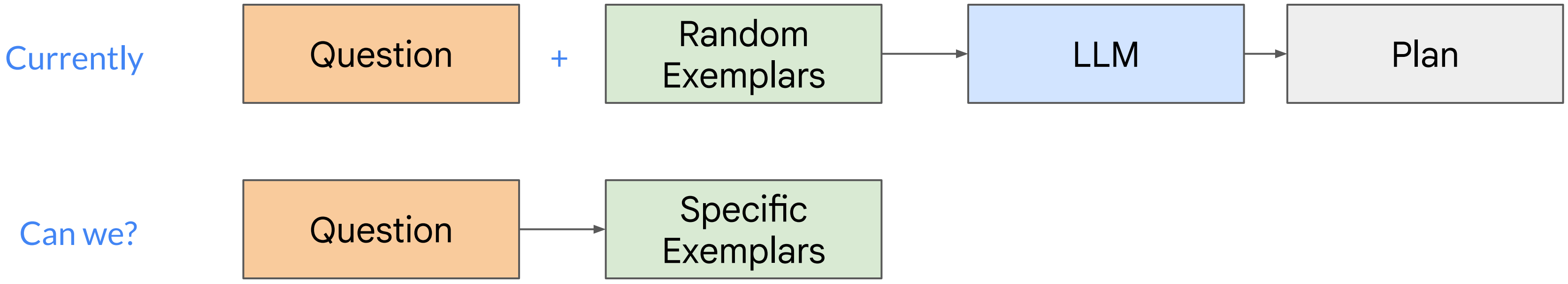
Google DeepMind

Improving Large Language Model Planning with Action Sequence Similarity

Xinran Zhao, Hanie Sedghi, Bernd Bohnet, Dale Schuurmans, Azade Nova

Student Researcher: Xinran
Mentors: Azade and Hanie

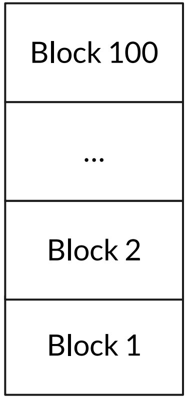
Planning with in-Context Learning (ICL)



- Previous work (Bohnet et al., 2024) shows good model performance on planning tasks with ICL
- However, for each question, the best exemplars may be different with different similarity among problems
- What signals help models select specific exemplars for each question?

Intuition

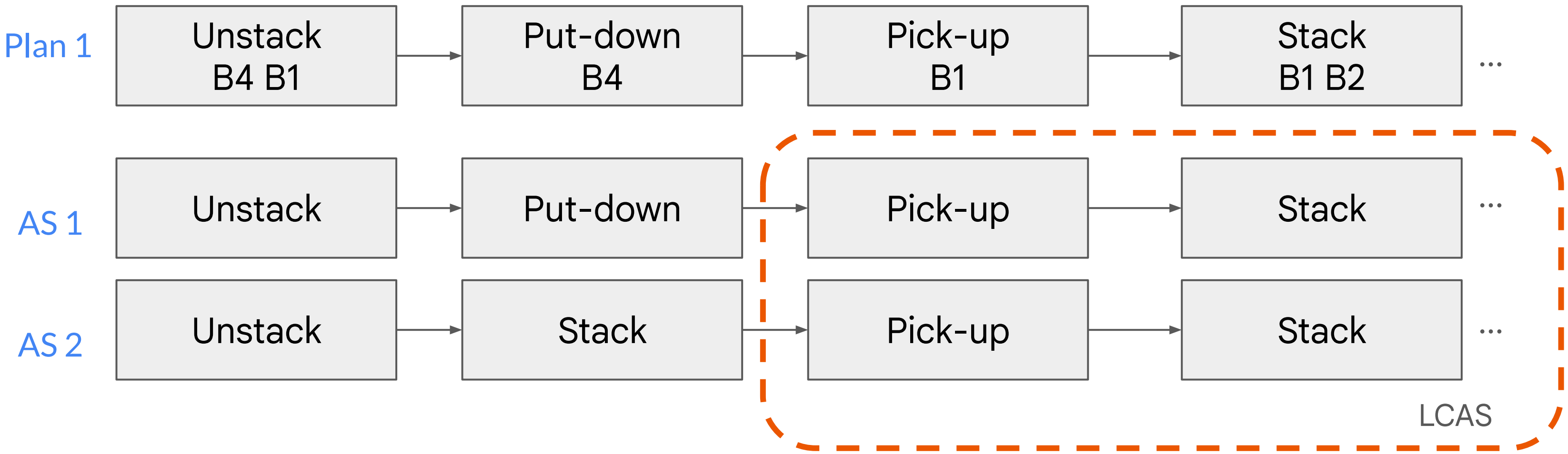
- Literature in NLP discuss the exemplar selection mainly as a retrieval problem from the similarity of task descriptions
- However, for planning, intuitively, plans are the key (100-block pile example)
- We first analyze the performance of ICL with exemplars selected by task description vs. plan similarity
- How to capture plan similarity?



How to capture plan similarity?

We propose: Action Sequence Similarity

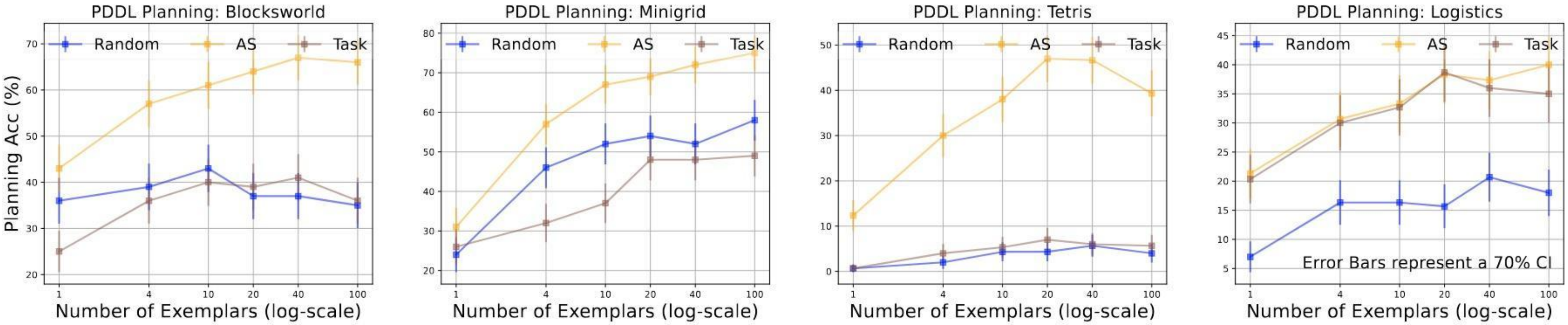
- Unlike many NLP tasks, a plan is a sequence of ordered and dependent actions
- We represent plans with action sequences (AS) and capture the similarity with the normalized longest common action sequence (LCAS)



Action Sequence Similarity

Analytical Performance on PlanBench

- With AS (Oracle plan), we can then analyze which signal helps LLM planning with ICL
- We show that AS is a robust signal with referential Oracle plans, compared to task description (token overlap of task descriptions)
- How to empirically leverage AS?



Action Sequence Similarity

Empirical application

We use the model generated plans to replace Oracle plans

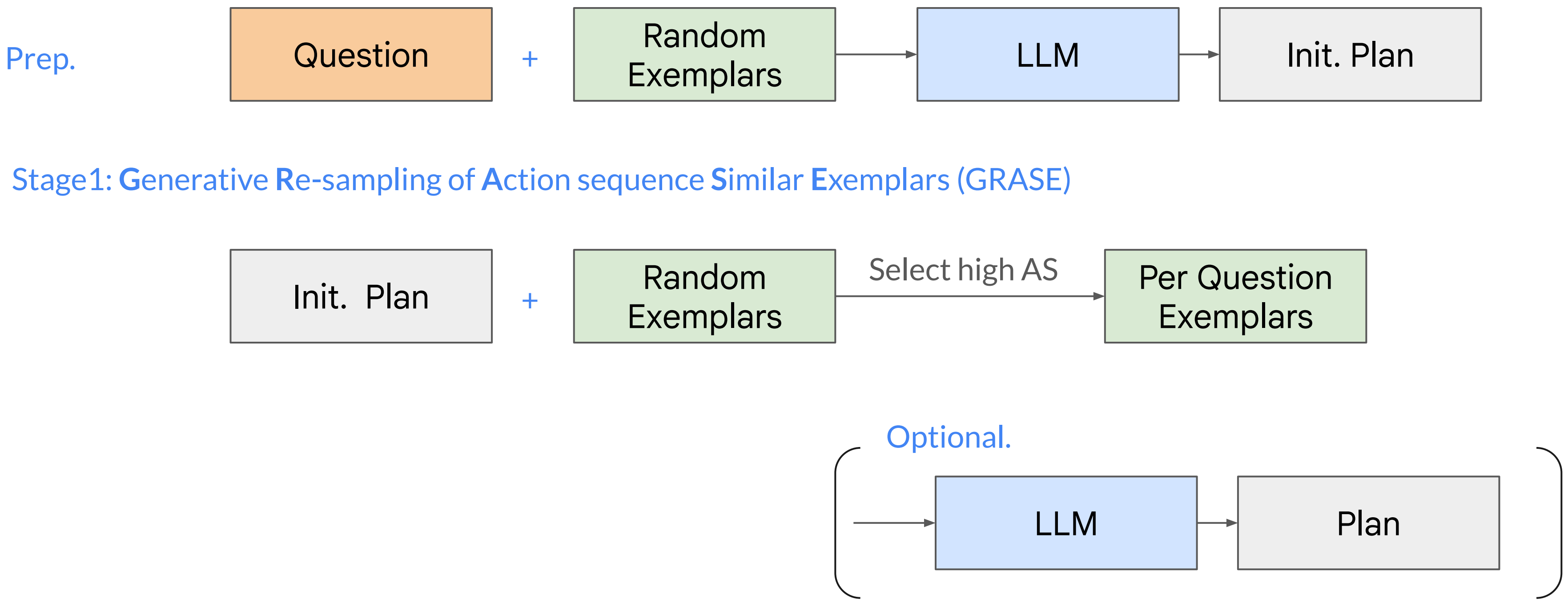
- They are not necessarily correct
- They can indicate some model preferred directions (e.g., which pile to start)

AS captures the similarity of planning problems between

- test example vs. exemplar candidates: rank the exemplars
- Groups of exemplar candidates: improve diversity and reduce redundancy

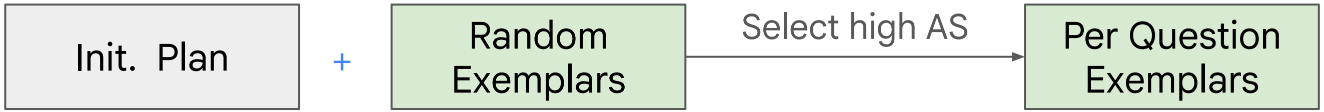
We propose GRASE-DC to empirically apply AS in planning with ICL

Our pipeline: GRASE-DC

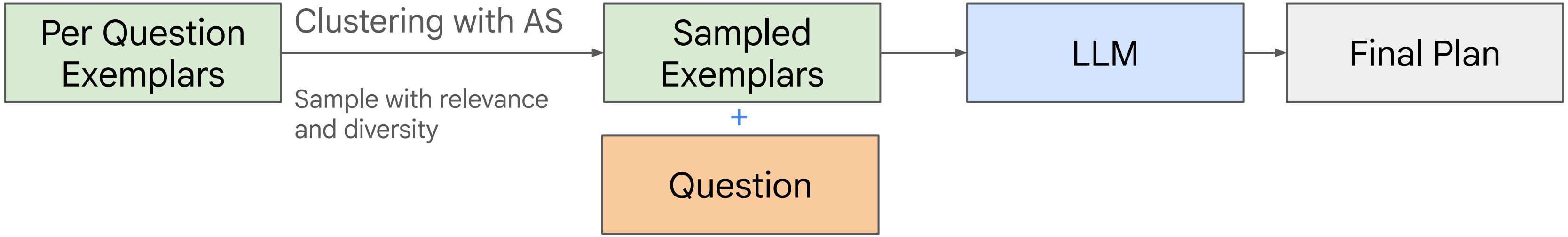


Our pipeline: GRASE-DC

Stage1: **G**enerative **R**e-sampling of **A**ction sequence **S**imilar **E**xemplars (GRASE)



Stage2: **D**ynamic **C**lustering (DC)



Our pipeline: GRASE-DC

GRASE-DC maintains the original ICL pipeline

- All operations are done on the exemplar set
- This feature makes it possible to collaborate with other methods

DC helps decide when to stop adding in-context exemplars for each question

GRASE-DC can be applied iteratively (GRASE-DC*)

- When we use the final plan to replace the initial plan

GRASE-DC can collaborate with validators (+VAL)

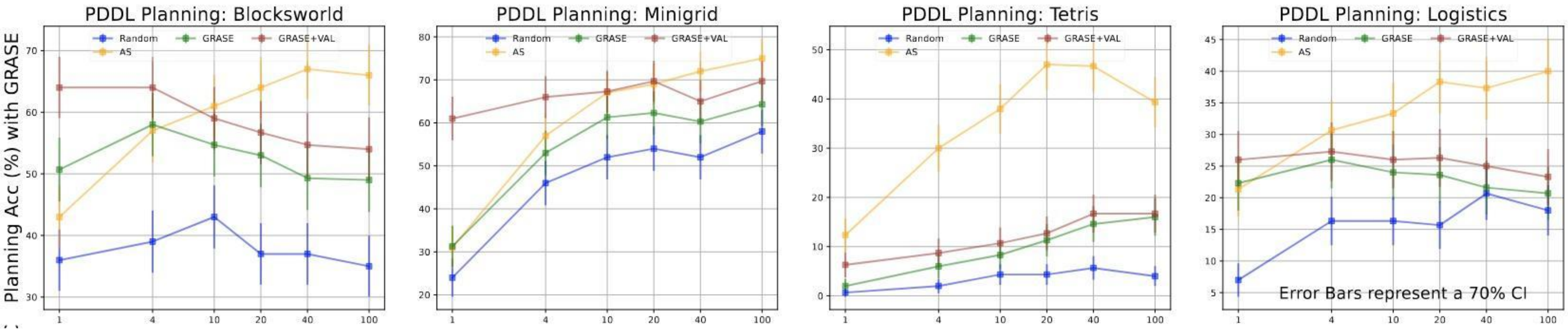
- After preparation, we only process the cases where the model did wrong

We then test the performance of GRASE-DC with Gemini 1.5 Pro

GRASE-DC performance on PDDL

GRASE on [PlanBench]

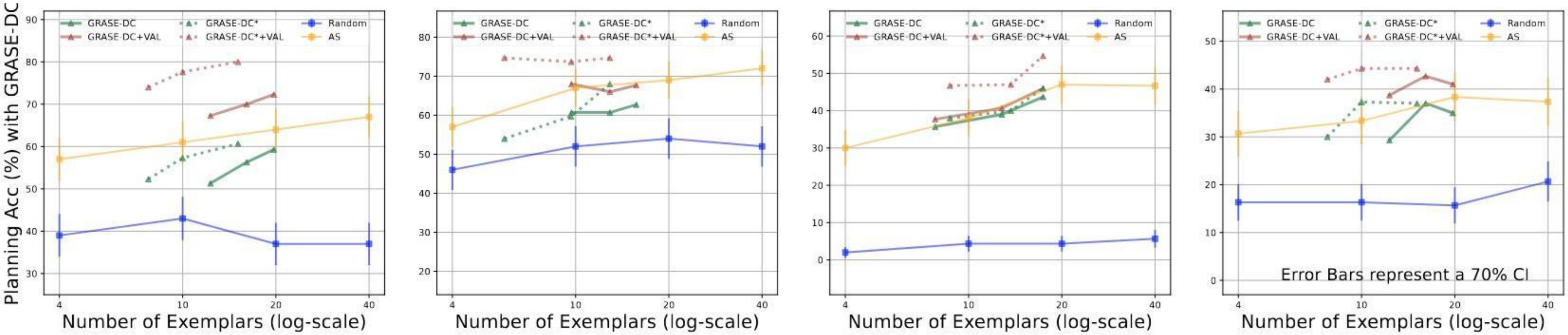
- GRASE already shows improved performance with per-task exemplar candidates
- VAL further improve the performance, at low-shot cases, they outperforms the analytical AS
- Blocksworld is 3-7 to 3-7



GRASE-DC performance on PDDL

GRASE on PlanBench

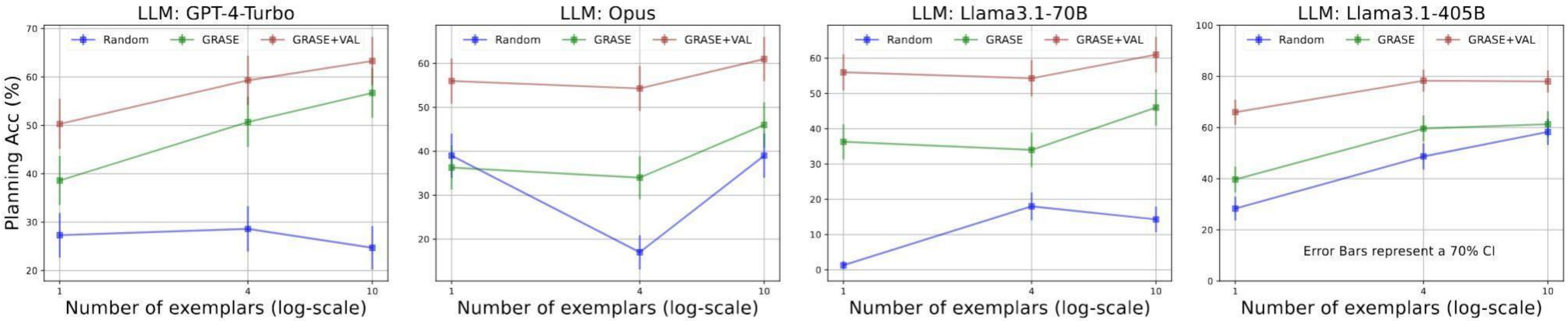
- GRASE-DC offers further performance improvement with less exemplars (auto-decide)
- The performance is close to using Oracle test plans
- Iterations (GRASE-DC*) and VAL allow further performance gain



GRASE-DC performance on PDDL

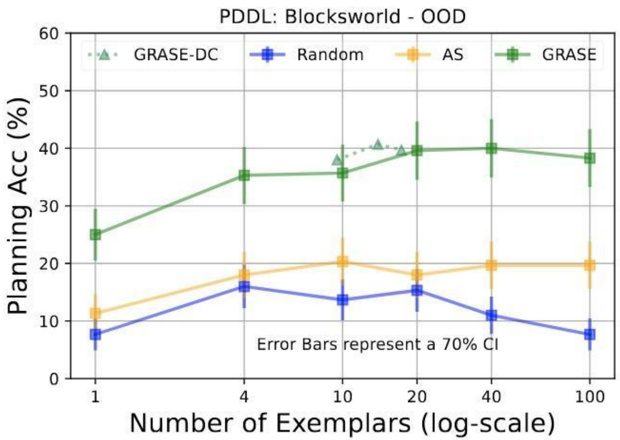
Other models on PlanBench

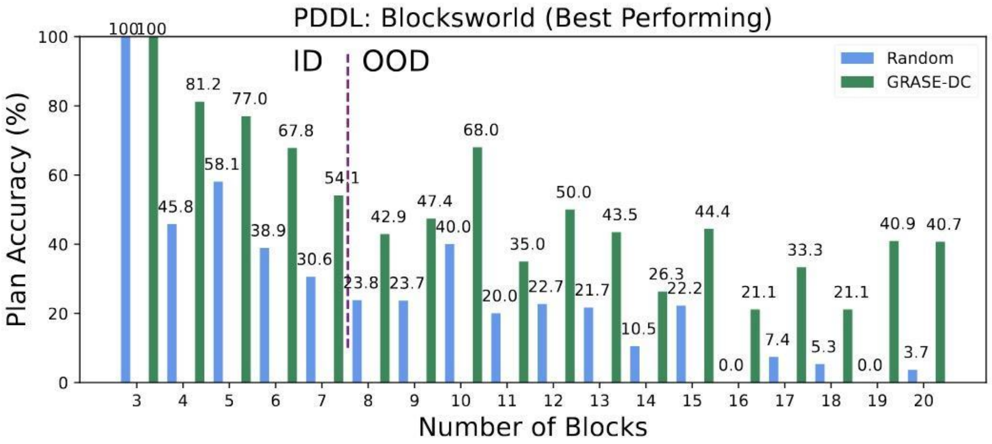
- GRASE shows consistent performance with other LLMs
- With GRASE, Llama 3.1-70B can outperform GPT-4-Turbo with random



GRASE-DC performance on PDDL (easy-to-hard)

- We use exemplar candidates with 3-7 blocks per problem to solve test cases with 8-20 blocks (out-of-distribution, OOD)
- We show that GRASE-DC strongly outperform random on OOD data (SFT with Gemini 1.0 S in this setting: 34.2)

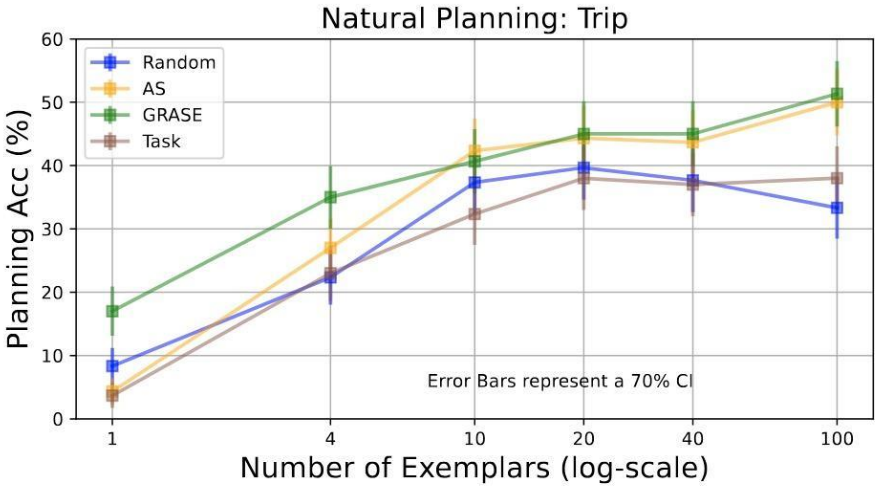




GRASE-DC performance on Natural Plan

Trip planning in Natural Plan

- We conduct experiments on Trip Planning, where the actions are flights across cities
- Similarly, we observe good performance, especially when with a large number of cities



Our pipeline: GRASE-DC

Efficiency

GRASE-DC requires **one** additional prompt to LLMs

- Compared to search-based algorithm, it is less costly.
- Using ICL allows easy application and shows the LLM internal capability

Is there any other ways do empirically leverage AS?

Further analysis on Efficiency

We propose two other methods to leverage AS

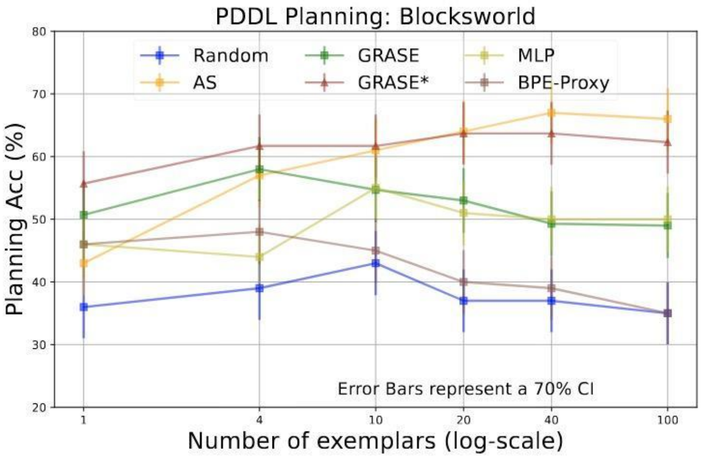
- **MLP**: using embeddings of exemplar (task+plan) and test task description to train an MLP regressor to approximate AS (similar to Few Shot Retrieval)

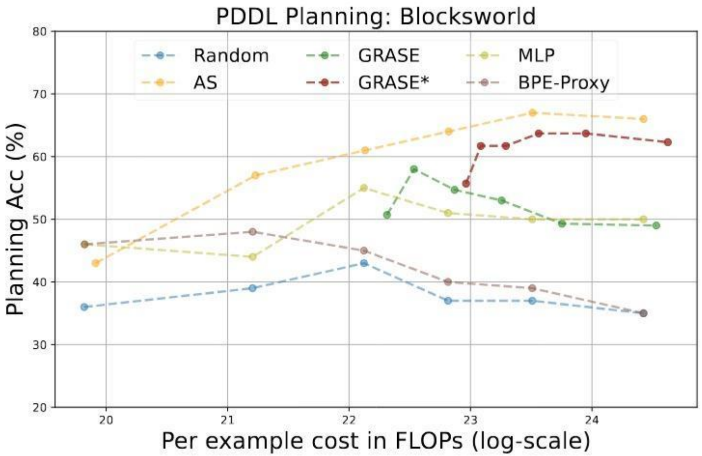
- Byte Pair Encoding as the Proxy (**BPE**): MLP compares the test example with all candidates. To save that cost, we capture frequence action sequences with BPE (over actions) as the proxy between them

- We use **Gecko** to acquire embeddings

Further analysis on Efficiency

- MLP and BPE-Proxy can achieve good performance, but can not be iterated
- From the FLOPs (selection and generation) view, alternatives are good, e.g., MLP achieves 95% performance with 66% FLOPs of GRASE. With BPE-Proxy, it is 83% performance with 27% FLOPs





Future work…

- [Planning] Collaborate GRASE-DC with other orthogonal methods: e.g., model-generated critiques or improved detailed instructions
- [Other tasks] Extend GRASE-DC to other tasks with potential action sequences (or general trajectories), e.g., coding, math, and agent

Thank you for Listening! Q&A time

- Thank you for listening!
- Welcome all Questions to xinranz3@andrew.cmu.edu
- Xinran would also like to thank his mentors, senior researchers, fellow student researchers, and anonymous reviewers for their great help during the wonderful summer

End-of-Presentation line

@xinranzhao, @hsedghi, @azade

——---