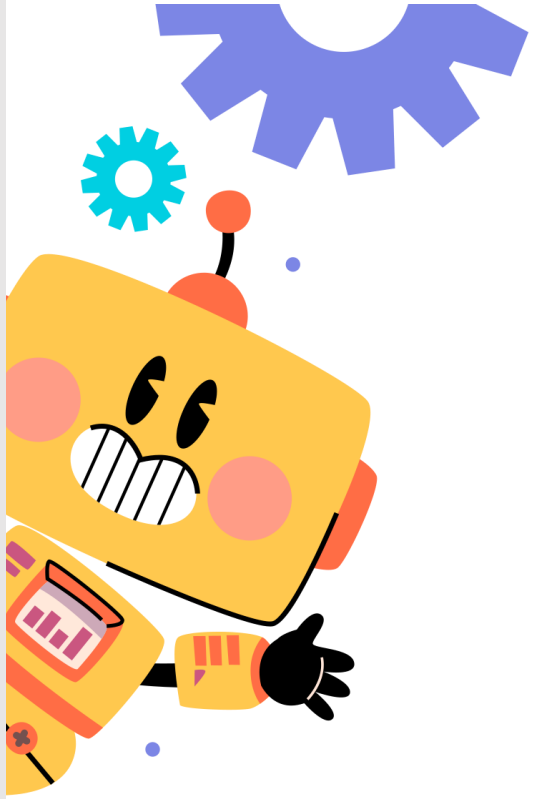
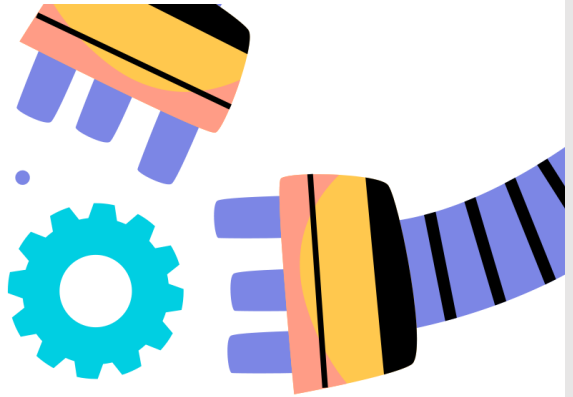




Search. Plan. Learn.
Ищи. Планируй. Учись.

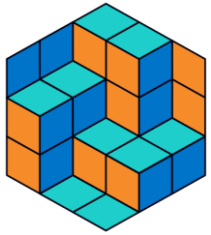
К.С. Яковлев

24.09.2025
СПбГУ
Лаборатория Маркова

О докладчике

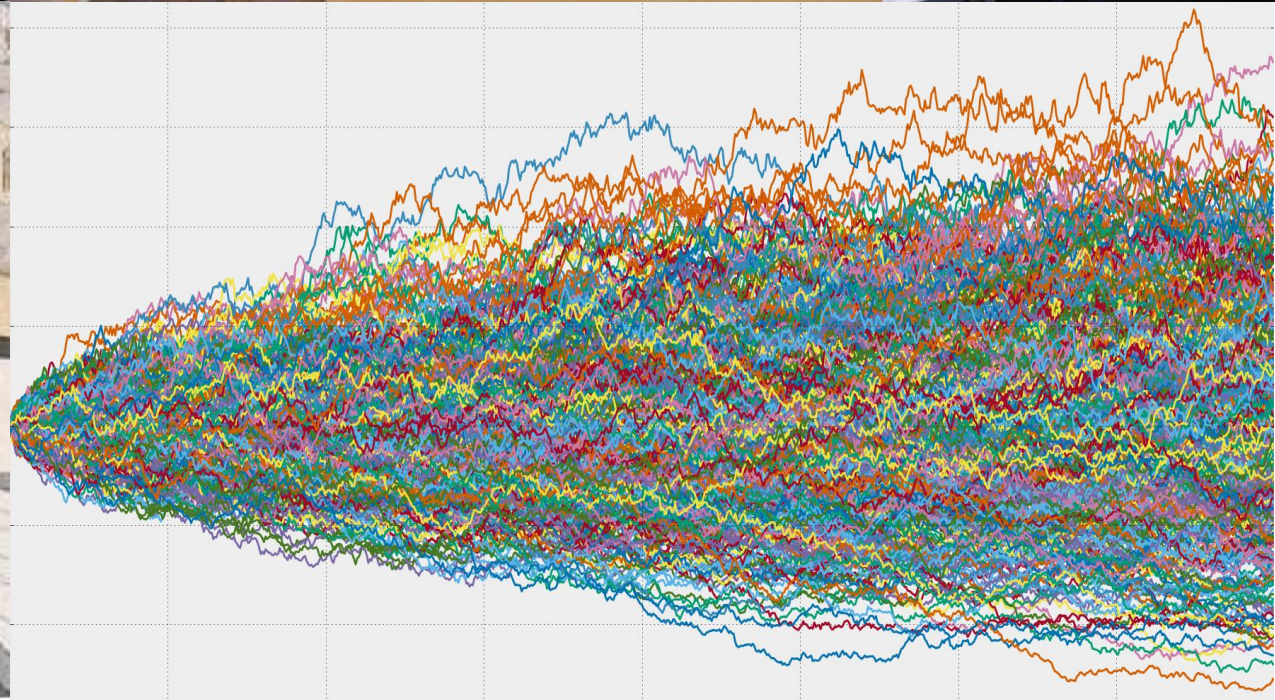
Константин Сергеевич Яковлев, к.ф.-м.н.

- Ведущий научный сотрудник, **ФИЦ ИУ РАН** (www.frccsc.ru)
- Ведущий научный сотрудник, **AIRI** (www.airi.net)
- Заведующий лабораторией Маркова, **СПбГУ** (no site yet)
- Доцент, **ФПМИ МФТИ** (www.mipt.ru)
- Доцент, **ФКН ВШЭ** (www.hse.ru)

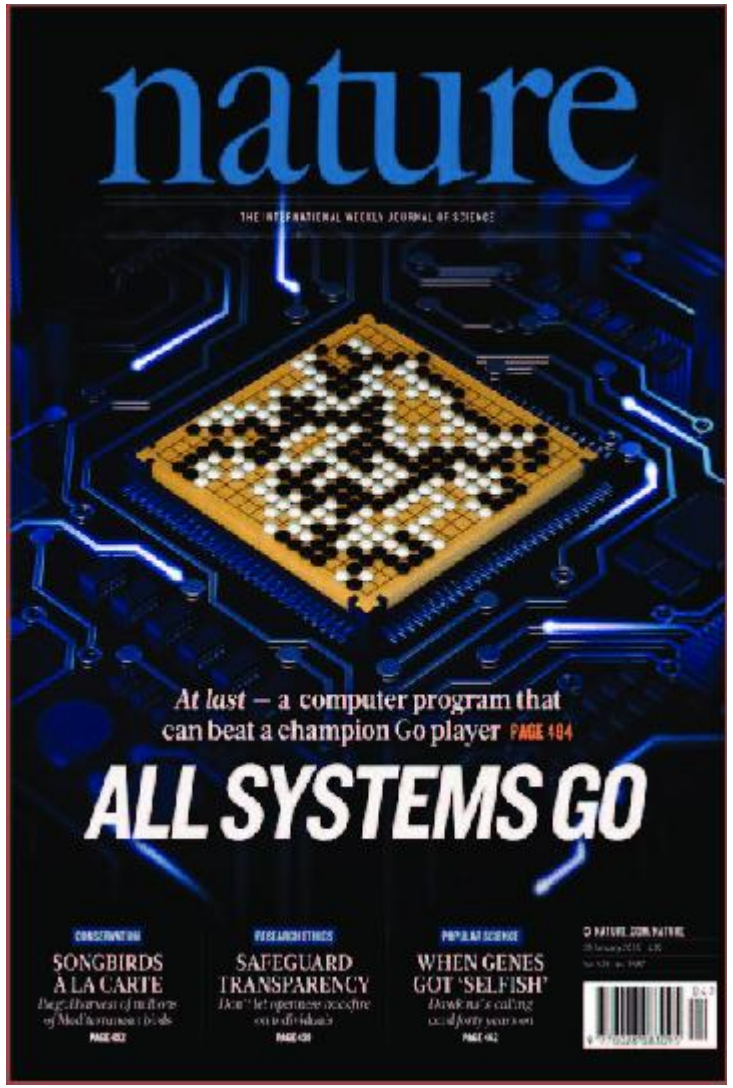


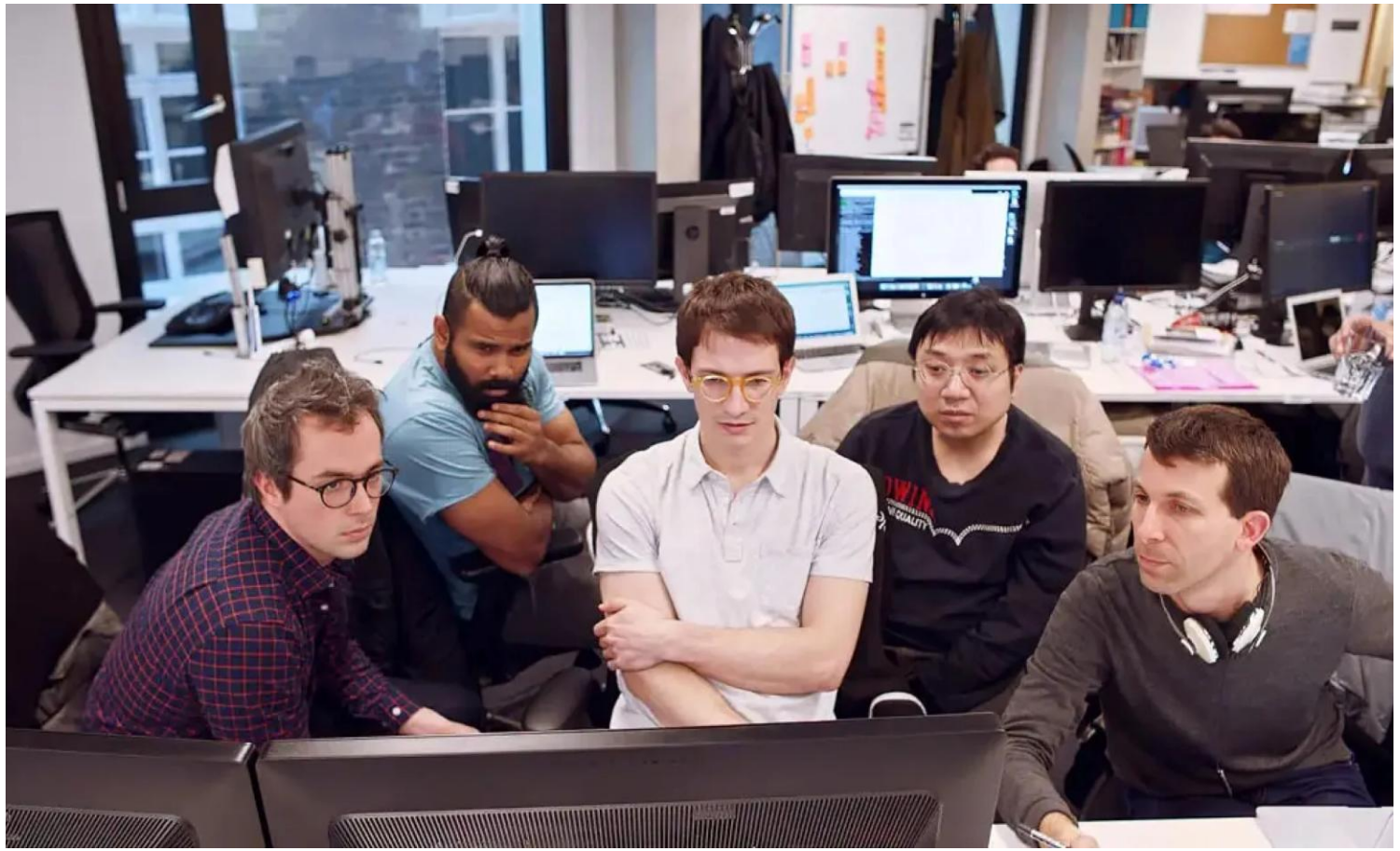
Научные интересы: искусственный интеллект, интеллектуальные системы управления, интеллектуальные динамические системы, робототехника, автоматическое планирование, эвристический поиск, многоагентные системы, когнитивные агенты

Простыми словами: **www.kyakovlev.me** -> Media
Посмотреть видео: **www.kyakovlev.me** -> Research

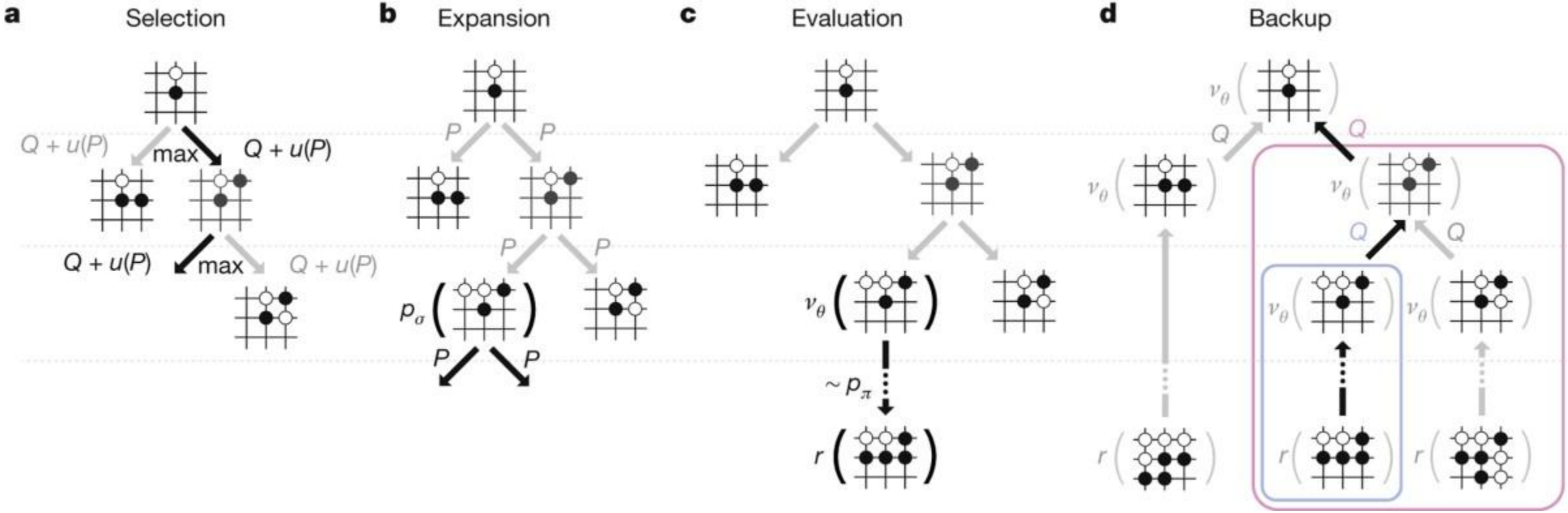


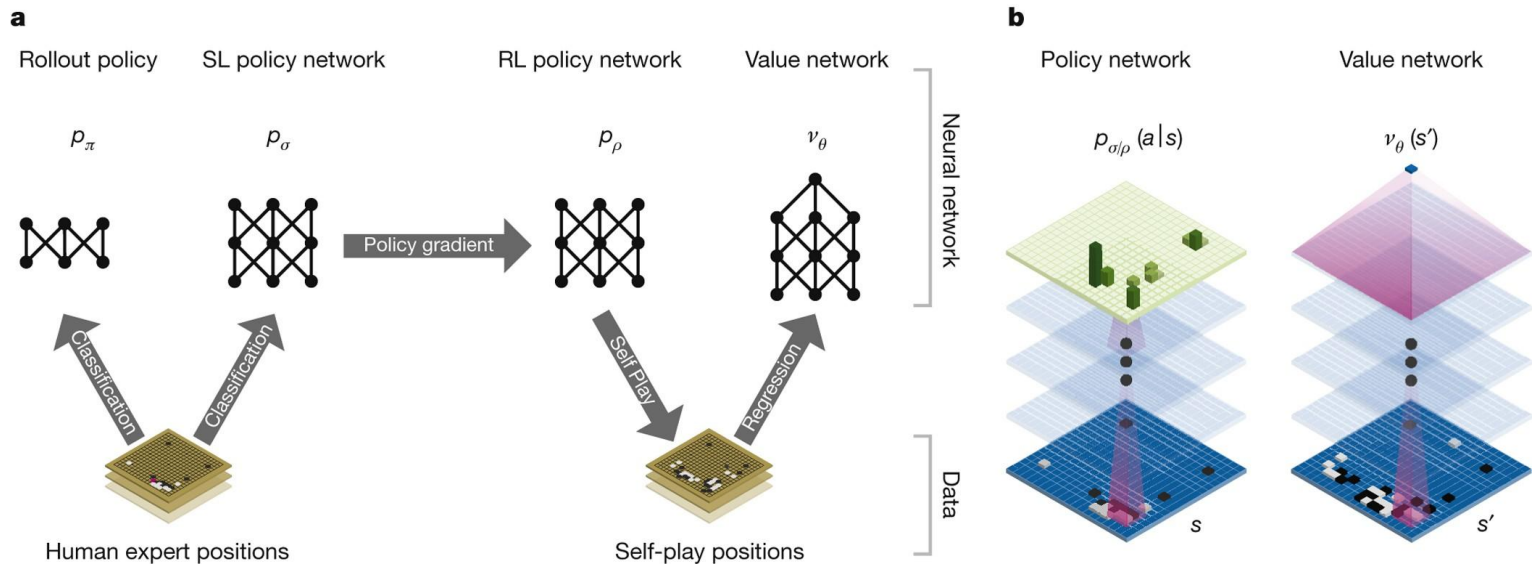
AlphaGo





https://deepmind.google/research/projects/alphago/


a
Selection
Q + u(P)
max
Q + u(P)
Q + u(P)
max
Q + u(P)
b
Expansion
P
P
P
P
p_σ
P
P
c
Evaluation
v_θ
$\sim p_\pi$
r
d
Backup
v_θ
Q
Q
v_θ
v_θ
Q
Q
v_θ
v_θ
r
r
r


a
Rollout policy
SL policy network
RL policy network
Value network
p_π
p_σ
p_ρ
v_θ
Policy gradient
Classification
Classification
Self Play
Regression
Neural network
Data
Human expert positions
Self-play positions
b
Policy network
Value network
$p_{\sigma/\rho}(a|s)$
$v_\theta(s')$
s
s'

AI = SPL (ИИ=ИПУ)

I see one move, but always the best move

José Raúl Capablanca

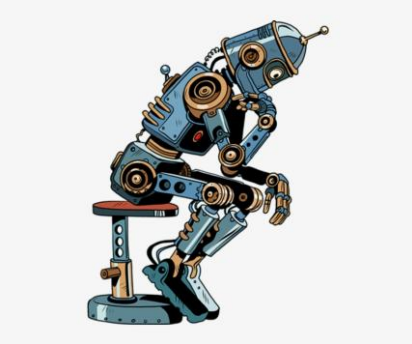
Планирование ~ умение рассуждать об эффектах действий и о том, как этим эффекты способствуют достижению целей

Поиск ~ умение перебирать (в голове) различные варианты решения и выбирать лучший

Обучение ~ умение улучшать стратегии поиска за счет использования прецедентов (настоящих или синтетических)

Planning vs. Sequential Decision Making

Текущее состояние → **<u>Последовательность</u> действий (план)** → **Желаемое состояние**

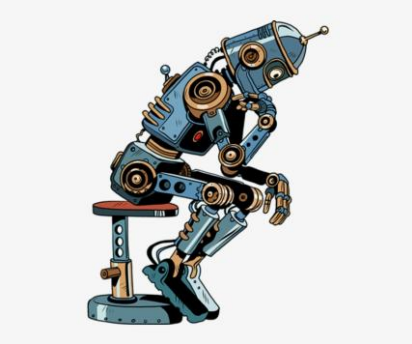


Робот в позиции (3, 7)

Робот в позиции (12, 2)

Planning vs. Sequential Decision Making

Текущее состояние → **<u>Последовательность</u> действий (план)** → **Желаемое состояние**



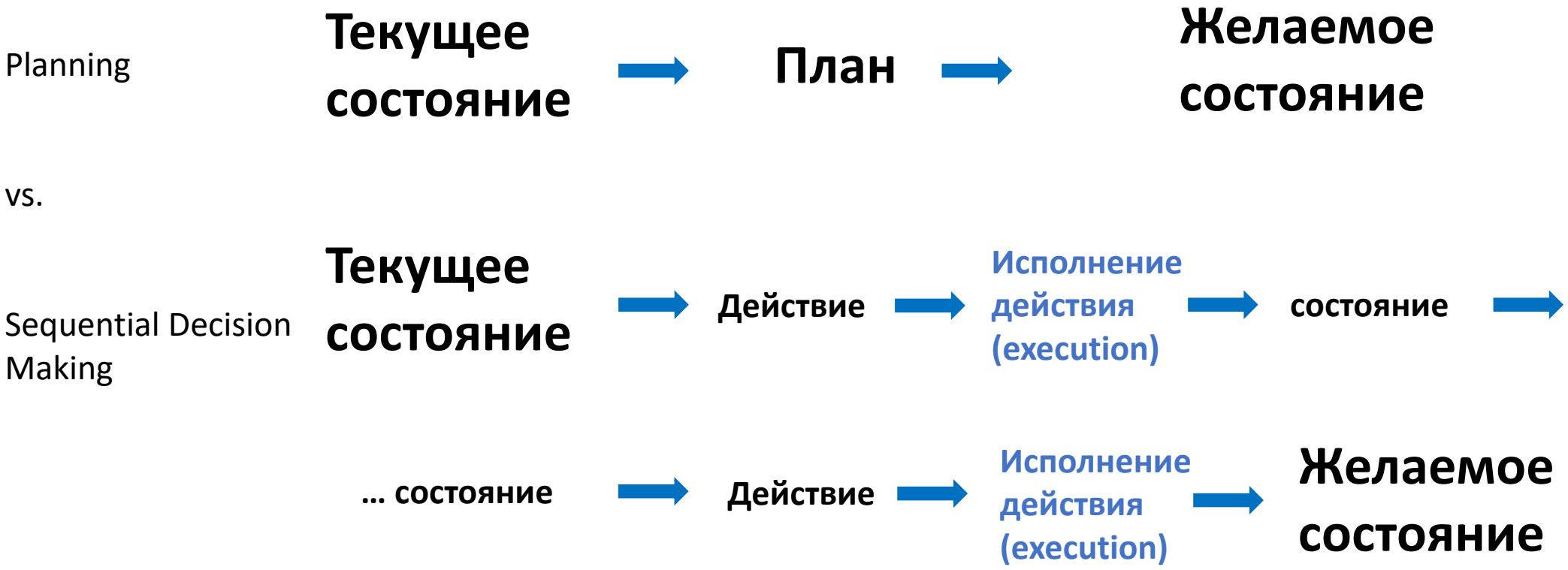
- **План**
- **НЕ факт достижения целевого состояния**

Это просто описание целевого состояния

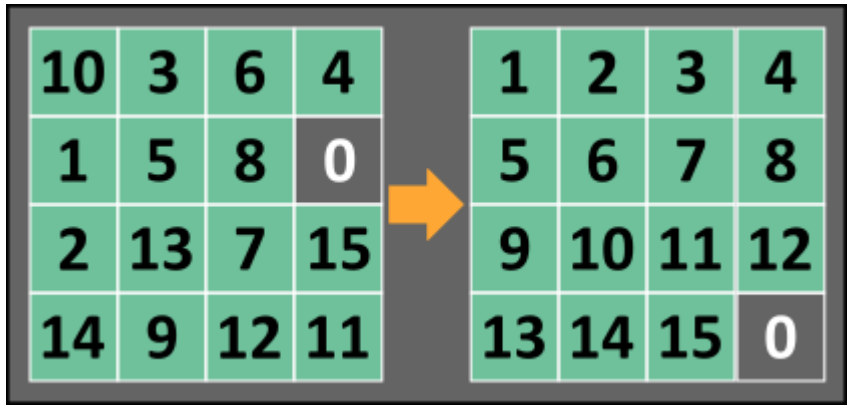
Робот в позиции (3, 7)

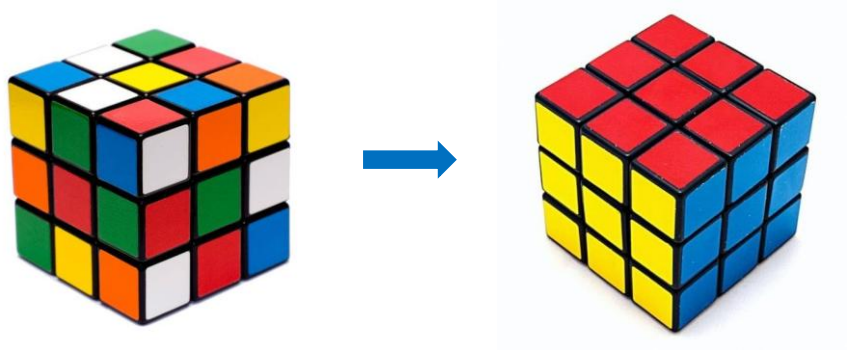
Робот в позиции (12, 2)

Planning vs. Sequential Decision Making



Поиск в пространствах со сложной комбинаторной структурой





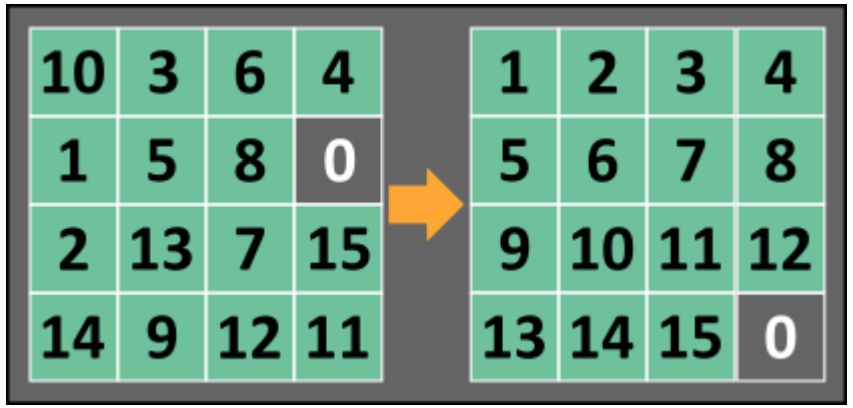
Число состояний
16! = 20 922 789 888 000

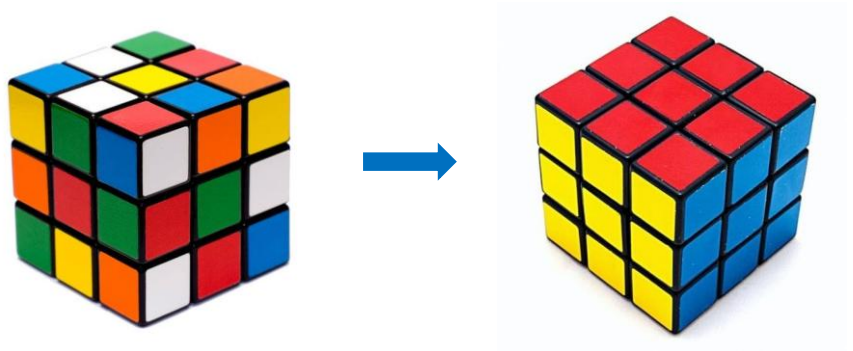
Валидных состояний
$16!/2 \sim 10^{12}$

Если 1 состояние ~ 1 byte, то

10^{12} byte ~ 10^{9} Kb ~ 10^{6} Mb ~
~ 10^{3} Gb ~ **1 Tb**

Поиск в пространствах со сложной комбинаторной структурой





Хорошая новость

Есть эффективнее алгоритмы сборки пятнашек и кубика Рубика
(субоптимальные решения)

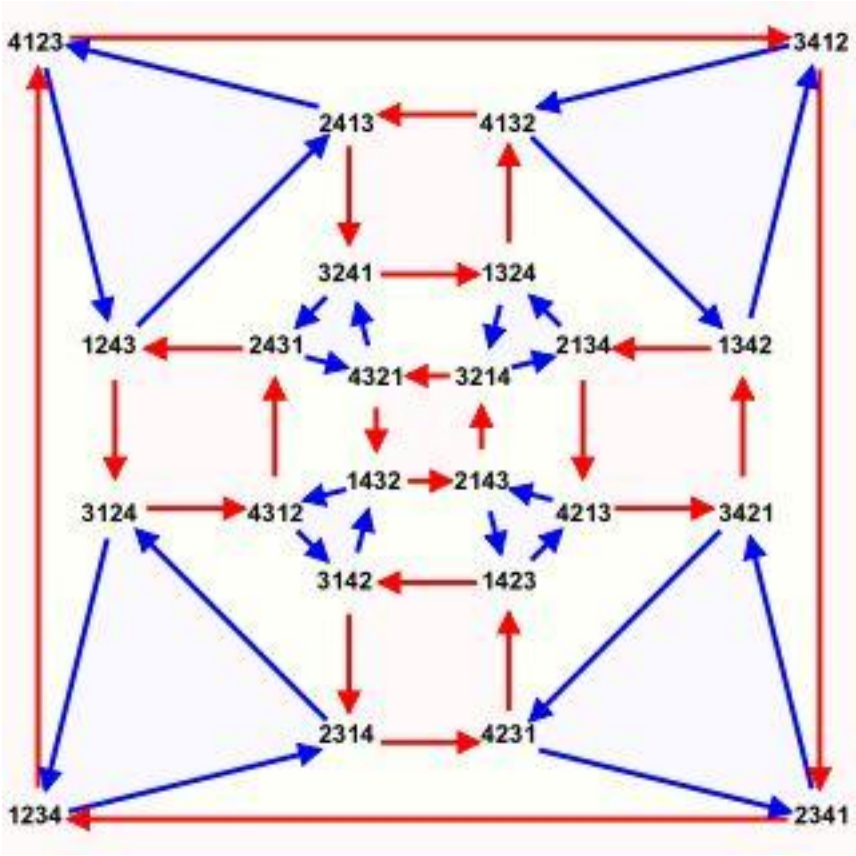
Плохая новость

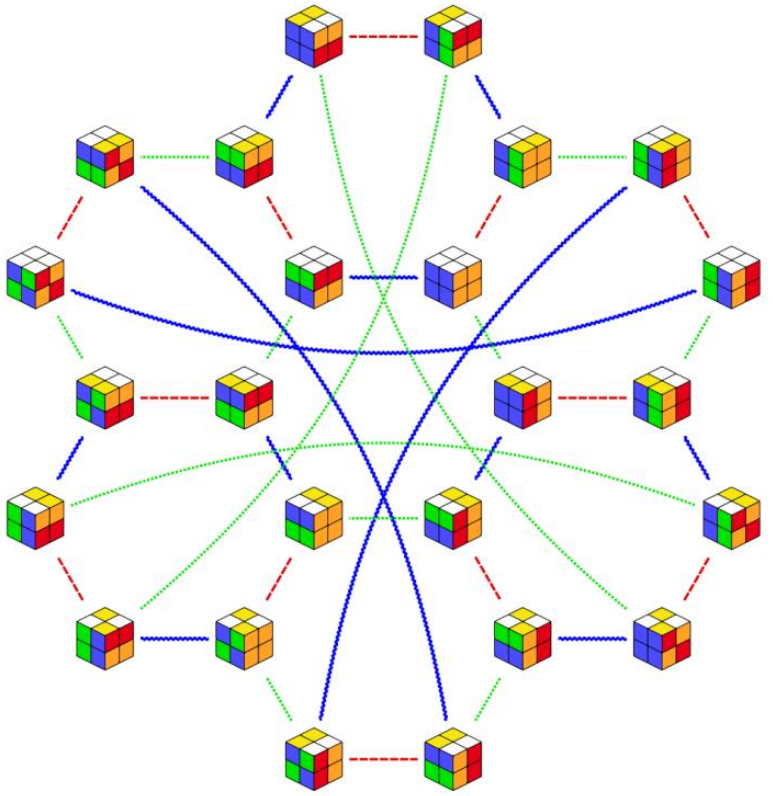
Это не универсальные алгоритмы.
Меняем кубик 333 на 4444 – алгоритм не работаем

Хотим

Универсальный алгоритм

Графы Кэли (Cayley Graphs)





Группа + Генераторы = Граф Кэли

Как искать путь до единичного элемента?

Универсальный алгоритм. SPL.

Поиск по графу/дереву (домено-независимый)

+

Машинное обучение для фокусировки поиска

DeepCubeA (2019)

ARTICLES
https://doi.org/10.1038/s42256-019-0070-z
nature machine intelligence

Solving the Rubik's cube with deep reinforcement learning and search

Forest Agostinelli[1,3], Stephen McAleer[2,3], Alexander Shmakov[1,3] and Pierre Baldi[1,2*]

The Rubik's cube is a prototypical combinatorial puzzle that has a large state space with a single goal state. The goal state is unlikely to be accessed using sequences of randomly generated moves, posing unique challenges for machine learning. We solve the Rubik's cube with DeepCubeA, a deep reinforcement learning approach that learns how to solve increasingly difficult states in reverse from the goal state without any specific domain knowledge. DeepCubeA solves 100% of all test configurations, finding a shortest path to the goal state 60.3% of the time. DeepCubeA generalizes to other combinatorial puzzles and is able to solve the 15 puzzle, 24 puzzle, 35 puzzle, 48 puzzle, Lights Out and Sokoban, finding a shortest path in the majority of verifiable cases.

https://deepcube.igb.uci.edu/

Идея:

Нейросеть аппроксимирует значение эвристической функции *h(s)*

h(s) – число ходов до цели из состояния s

A Machine Learning Approach That Beats Large Rubik's Cubes (2025)

A Machine Learning Approach That Beats Large Rubik's Cubes

The CayleyPy Project

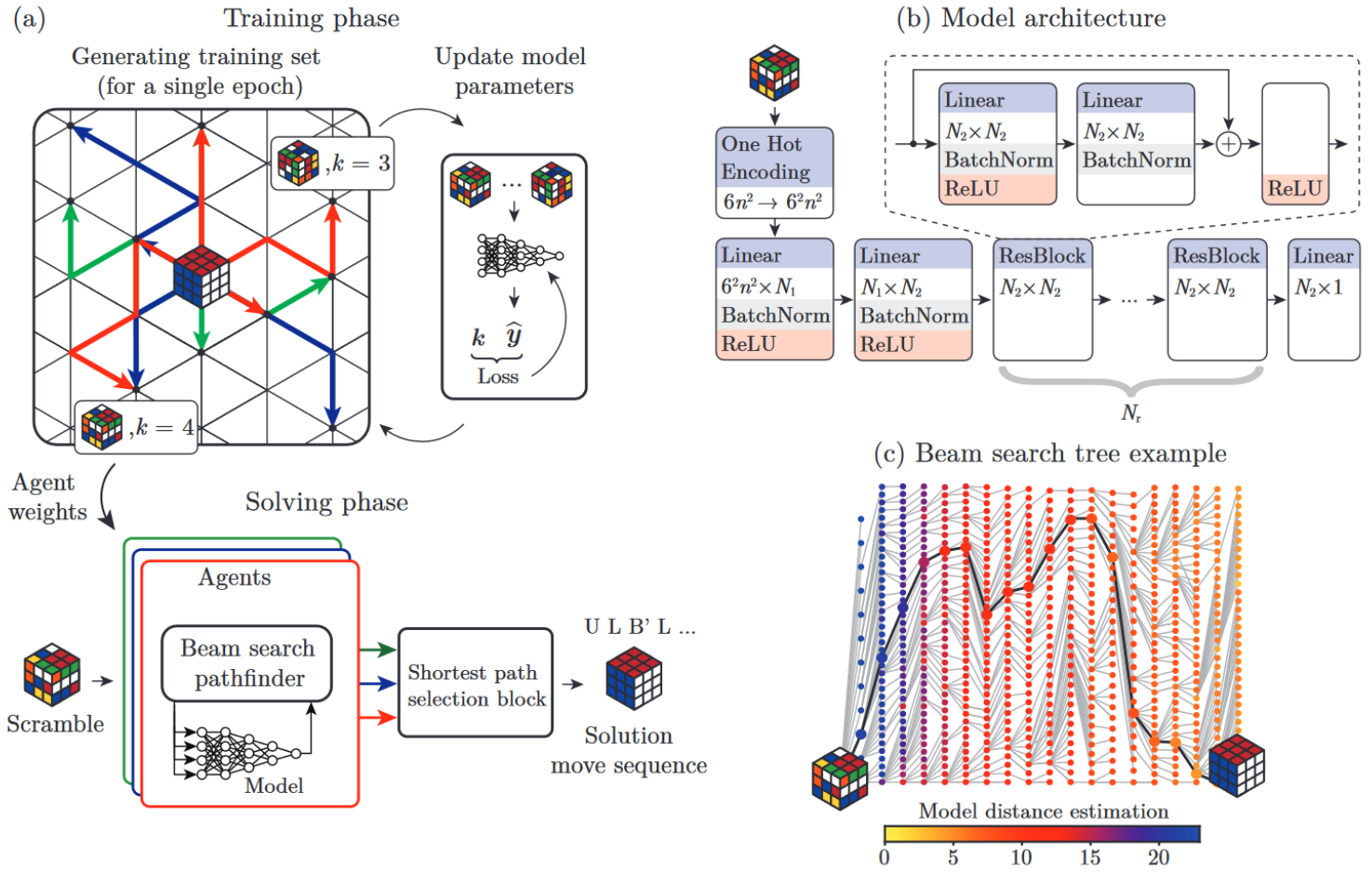
A. Chervov[1,2,3†], K. Khoruzhii[4†], N. Bukhal[5], J. Naghiyev[1], V. Zamkovoy[5], I. Koltsov[5], L. Cheldieva[5], A. Sychev[5], A. Lenin[5], M. Obozov[6], E. Urvanov[5], A. Romanov[5*]

[1] *Institut Curie, Université PSL, Paris, F-75005, France;*
[2] *INSERM U900, Paris, F-75005, France;*
[3] *CBIO, Mines ParisTech, Université PSL, Paris, F-75005, France;*
[4] *Technical University of Munich, Garching, 85748, Germany;*
[5] *Institute of Artificial Intelligence, RTU MIREA, Moscow, 119454, Russia;*
[6] *Yandex Research, Moscow, 119021, Russia*

[*] Corresponding author: romanov@mirea.ru
[†] These authors contributed equally to this work.

https://arxiv.org/pdf/2502.13266

A Machine Learning Approach That Beats Large Rubik's Cubes



https://arxiv.org/pdf/2502.13266

Интересные задачи, идеи, направления работ

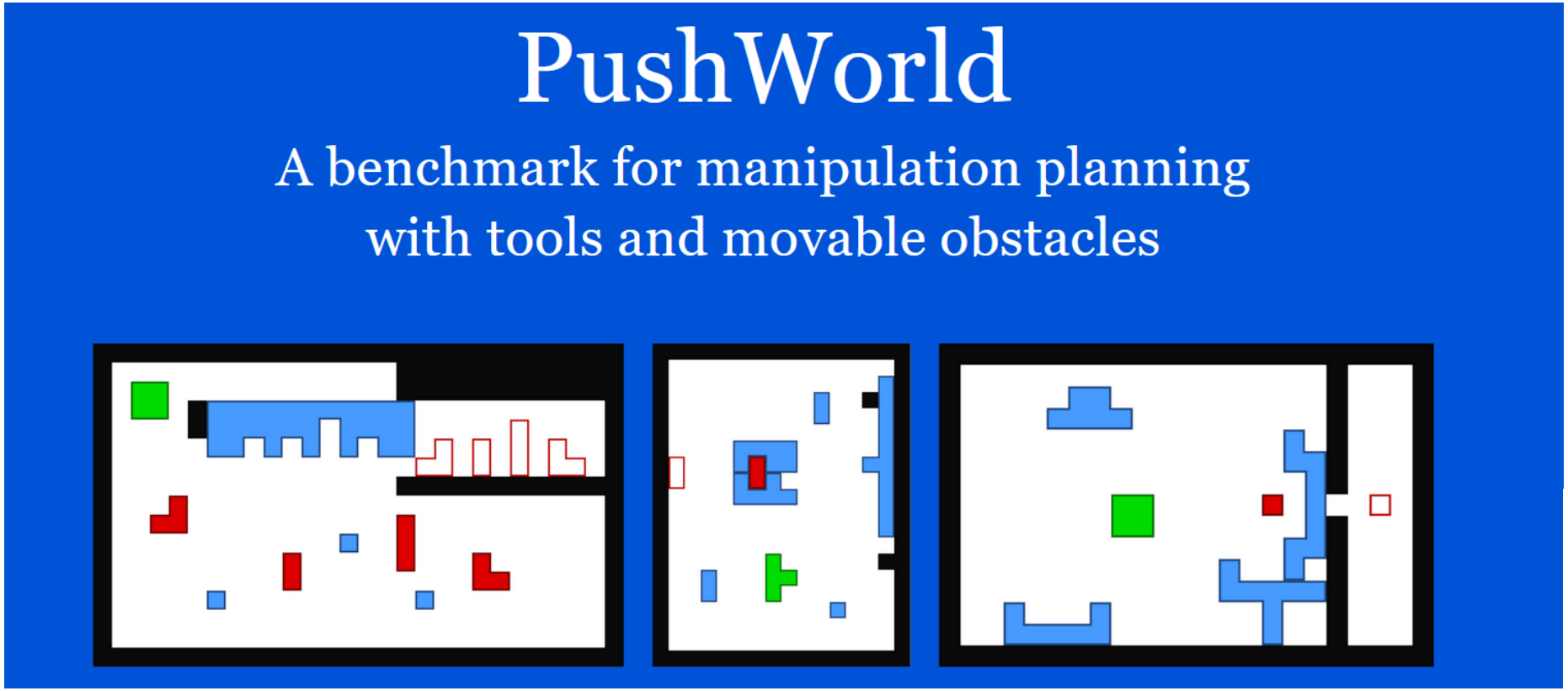
Перейти от «простого» кубика 3x3 к более сложному, например, 4x4

- Улучшение ML части (что учим, как учим, откуда берём данные)
- Улучшение алгоритмов поиска (beamsearch >> A*)

Соревнование на Kaggle
https://www.kaggle.com/competitions/cayley-py-professor-tetraminx-solve-optimally/overview
Другая головоломка, но смысл тот же

Ultimate Goal
Решатель для произвольных групп/генераторов

PushWorld



https://deepmind-pushworld.github.io/play/

Pushworld

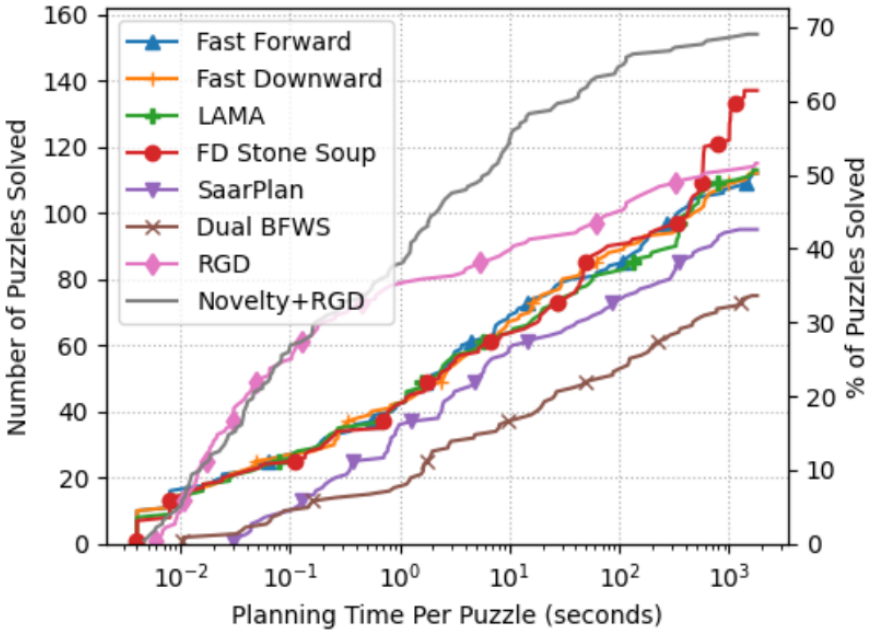


Figure 5: Number of solved puzzles vs. planning time. Planning time is measured independently for each puzzle.

	PPO		DQN	
Level 0 Puzzle Set	Train	Test	Train	Test
Base	88.5%	40.4%	24.9%	19.5%
Larger Puzzle Sizes	93.2%	71.5%	61.2%	61.8%
More Walls	77.9%	23.3%	18.5%	13.1%
More Obstacles	64.7%	21.4%	17.6%	13.1%
More Shapes	74.5%	24.2%	23.9%	17.1%
Multiple Goals	5.3%	1.1%	0.9%	0.2%
All	21.2%	5.7%	11.4%	10.1%

Table 2: The percentage of training and testing puzzles solved by PPO and DQN.

Классические алгоритмы поиска/планирования – работают, но долго

RL «из коробки» - работает быстро, но плохо

Pushworld

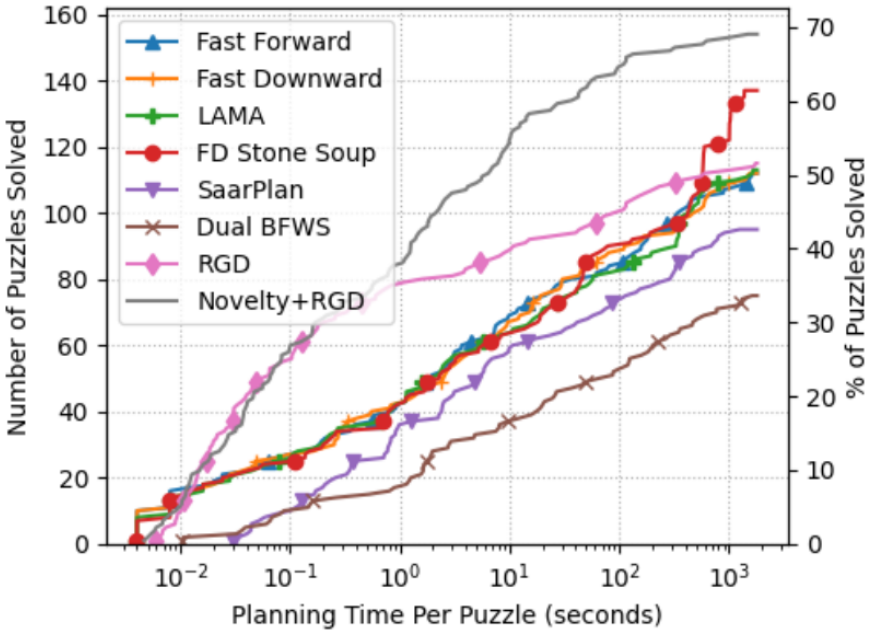


Figure 5: Number of solved puzzles vs. planning time. Planning time is measured independently for each puzzle.

	PPO		DQN	
Level 0 Puzzle Set	Train	Test	Train	Test
Base	88.5%	40.4%	24.9%	19.5%
Larger Puzzle Sizes	93.2%	71.5%	61.2%	61.8%
More Walls	77.9%	23.3%	18.5%	13.1%
More Obstacles	64.7%	21.4%	17.6%	13.1%
More Shapes	74.5%	24.2%	23.9%	17.1%
Multiple Goals	5.3%	1.1%	0.9%	0.2%
All	21.2%	5.7%	11.4%	10.1%

Table 2: The percentage of training and testing puzzles solved by PPO and DQN.

Классические алгоритмы поиска/планирования – работают, но долго

RL «из коробки» - работает быстро, но плохо

Sounds like a project

Головоломки с картиночным входом

Planning from Pixels in Environments with Combinatorially Hard Search Spaces

Marco Bagatella
Max Planck Institute for Intelligent Systems
Tübingen, Germany
mbagatella@tue.mpg.de

Mirek Olšák
Computer Science Department
University Innsbruck, Austria
mirek@olsak.net

Michal Rolínek
Max Planck Institute for Intelligent Systems
Tübingen, Germany
michal.rolinek@tue.mpg.de

Georg Martius
Max Planck Institute for Intelligent Systems
Tübingen, Germany
georg.martius@tue.mpg.de

https://papers.nips.cc/paper_files/paper/2021/file/cf708fc1decf0337aded484f8f4519ae-Paper.pdf

Головоломки с картиночным входом

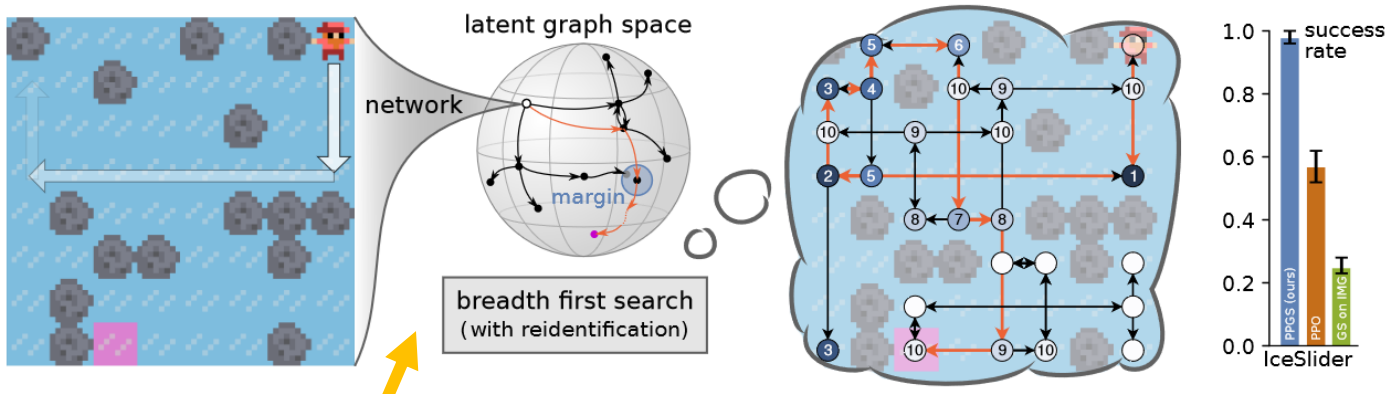
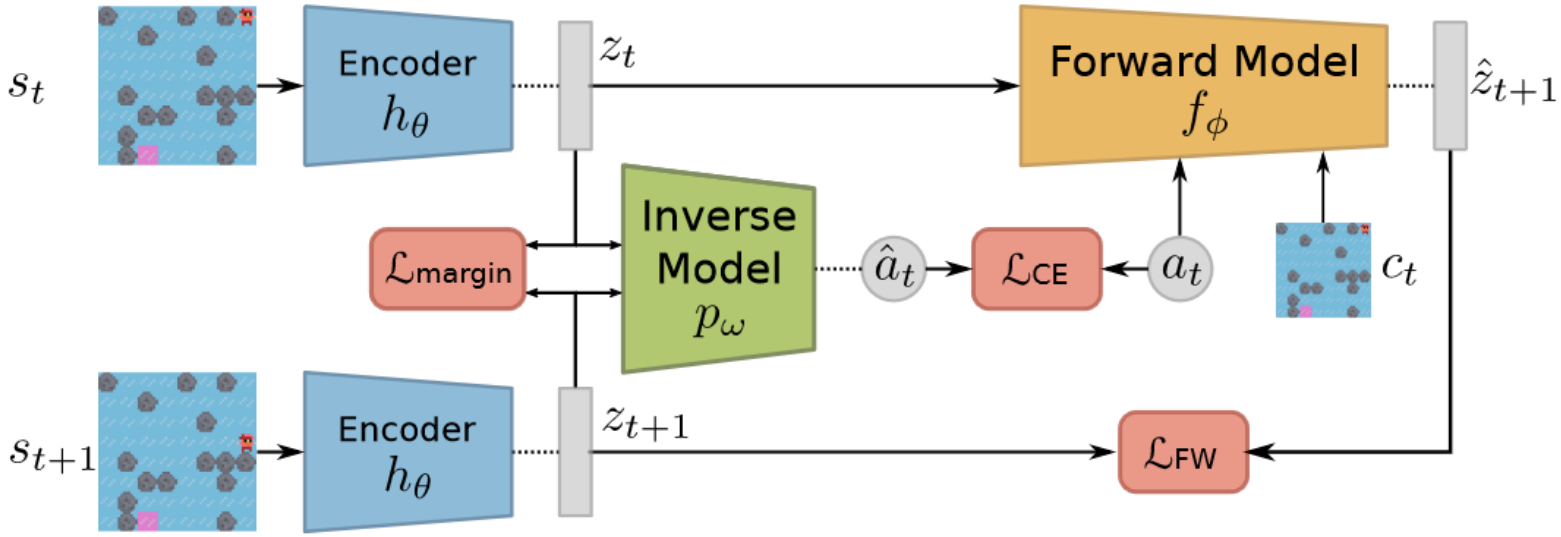


Figure 1: Planning from Pixels with Graph Search. Our method leverages learned latent dynamics to efficiently build and search a graph representation of the environment. Resulting policies show unrivaled performance across a distribution of hard combinatorial tasks.

Заменить BFS на более эффективный алгоритм поиска

Улучшить ML часть (архитектура сетей, loss'ы и пр.)



Нейросимвольная интеграция

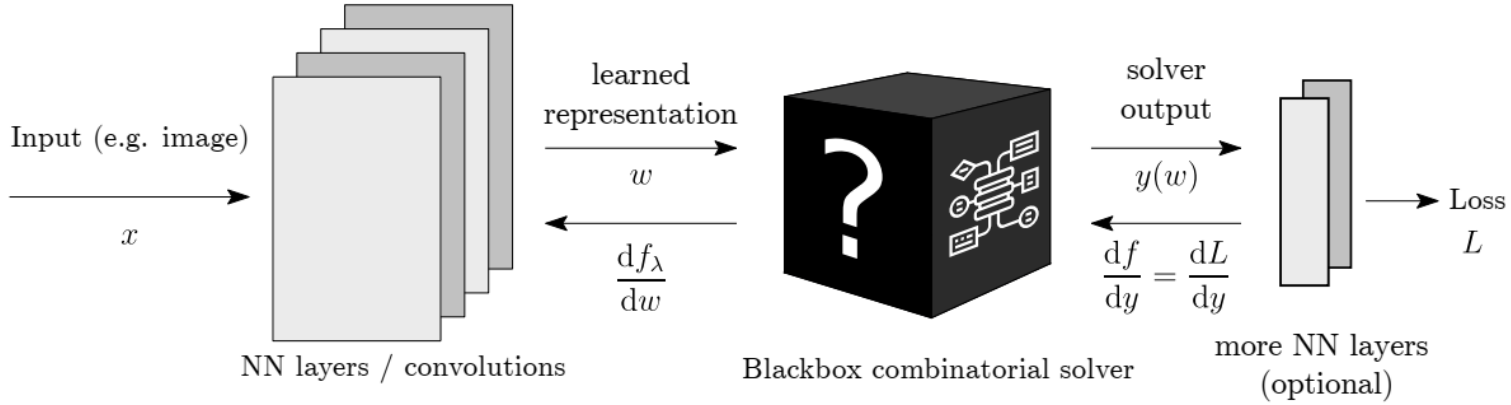


Figure 1: Architecture design enabled by Theorem 1. Blackbox combinatorial solver embedded into a neural network.

Differentiation Of Blackbox Combinatorial Solvers // ICLR 2020

https://iclr.cc/virtual_2020/poster_BkevoJSYPB.html

https://openreview.net/pdf?id=BkevoJSYPB

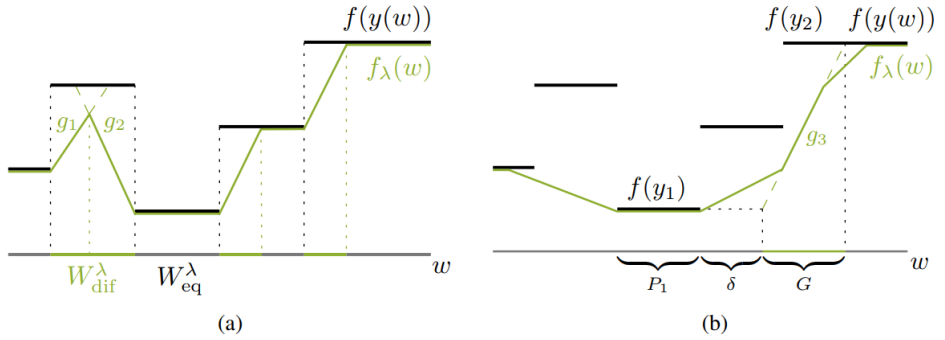
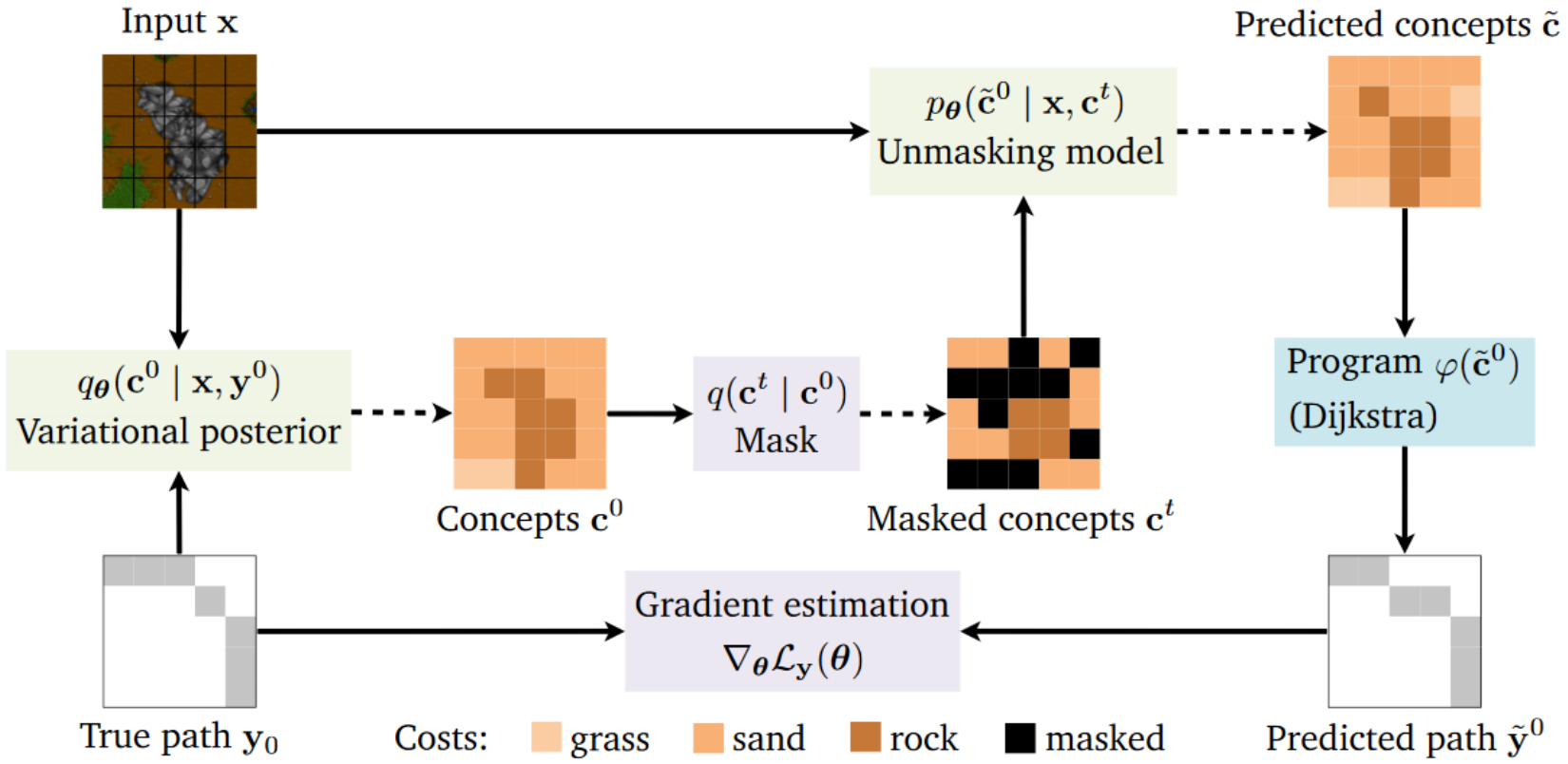


Figure 2: Continuous interpolation of a piecewise constant function. (a) f_λ for a small value of λ; the set W^λ_{eq} is still substantial and only two interpolators g_1 and g_2 are incomplete. Also, all interpolators are 0-interpolators. (b) f_λ for a high value of λ; most interpolators are incomplete and we also encounter a δ-interpolator g_3 (between y_1 and y_2) which attains the value $f(y_1)$ δ-away from the set P_1. Despite losing some local structure for high λ, the gradient of f_λ is still informative.

Нейросимвольная интеграция



Neurosymbolic Diffusion Models // NeurIPS 2025

https://arxiv.org/pdf/2505.13138

Трансформером по гриду

TransPath: Learning Heuristics For Grid-Based Pathfinding via Transformers

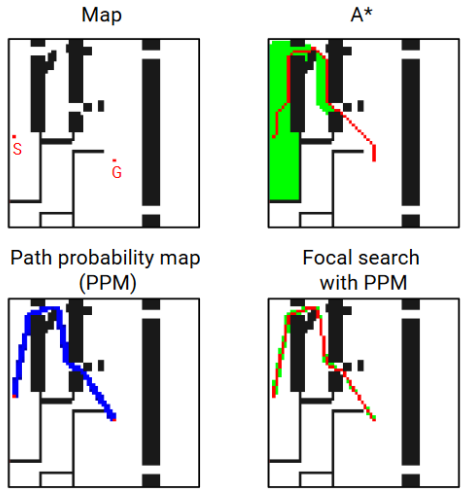
Daniil Kirilenko, [1] **Anton Andreychuk,** [2] **Aleksandr Panov,** [1, 2] **Konstantin Yakovlev** [1, 2]

[1] Federal Research Center for Computer Science and Control of Russian Academy of Sciences, Moscow, Russia
[2] AIRI, Moscow, Russia
anedanman@gmail.com, andreychuk@airi.net, panov@airi.net, yakovlev@isa.ru

Abstract

Heuristic search algorithms, e.g. A*, are the commonly used tools for pathfinding on grids, i.e. graphs of regular structure that are widely employed to represent environments in robotics, video games etc. Instance-independent heuristics for grid graphs, e.g. Manhattan distance, do not take the obstacles into account and, thus, the search led by such heuristics performs poorly in the obstacle-rich environments. To this end, we suggest learning the instance-dependent heuristic proxies that are supposed to notably increase the efficiency of the search. The first heuristic proxy we suggest to learn is the correction factor, i.e. the ratio between the instance independent cost-to-go estimate and the perfect one (computed offline at the training phase). Unlike learning the absolute values of the cost-to-go heuristic function, which was known before, when learning the correction factor the knowledge of the instance-independent heuristic is utilized. The second heuristic proxy is the path probability, which indicates how likely the grid cell is lying on the shortest path. This heuristic can be utilized in the Focal Search framework as the secondary heuristic, allowing us to preserve the guarantees on the bounded sub-optimality of the solution. We learn both suggested heuristics in a supervised fashion with the state-of-the-art neural networks containing attention blocks (transformers). We conduct a thorough empirical evaluation on a comprehensive dataset of planning tasks, showing that the suggested techniques *i*) reduce the computational effort of the A* up to a factor of 4x while producing the solutions, which costs exceed the costs of the optimal solutions by less than 0.3% on average; *ii*) outperform the competitors, which include the conventional techniques from the heuristic search, i.e. weighted A*, as well as the state-of-the-art learnable planners.

The project web-page is: https://airi-institute.github.io/TransPath/.



Простыми словами на Хабре

https://airi-institute.github.io/TransPath/

Понятная задача:
искать путь на сетчатом графе (grid)

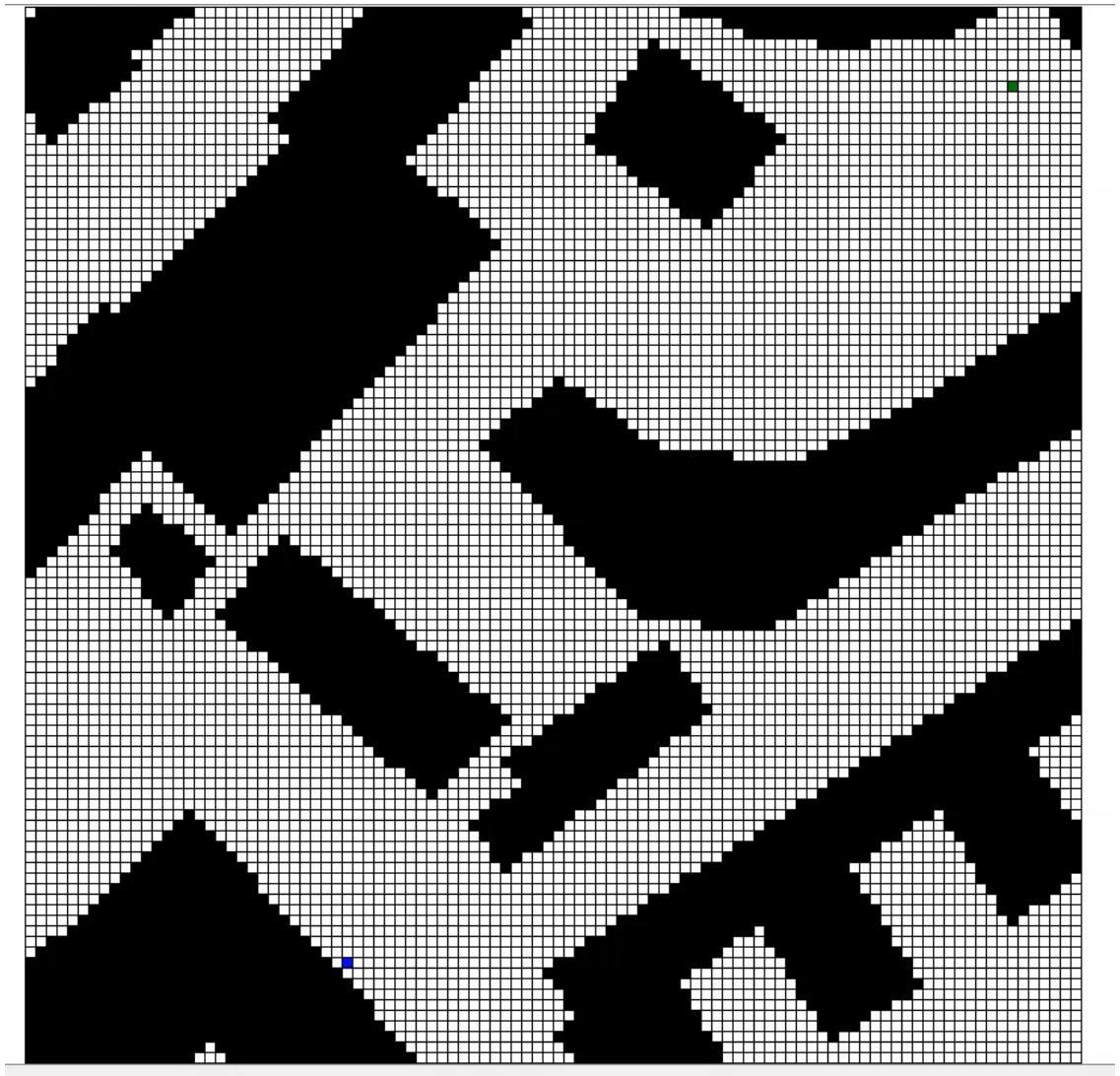
Простая идея:
нейросеть + A*

Key ingredient:
студент «с головой и руками»

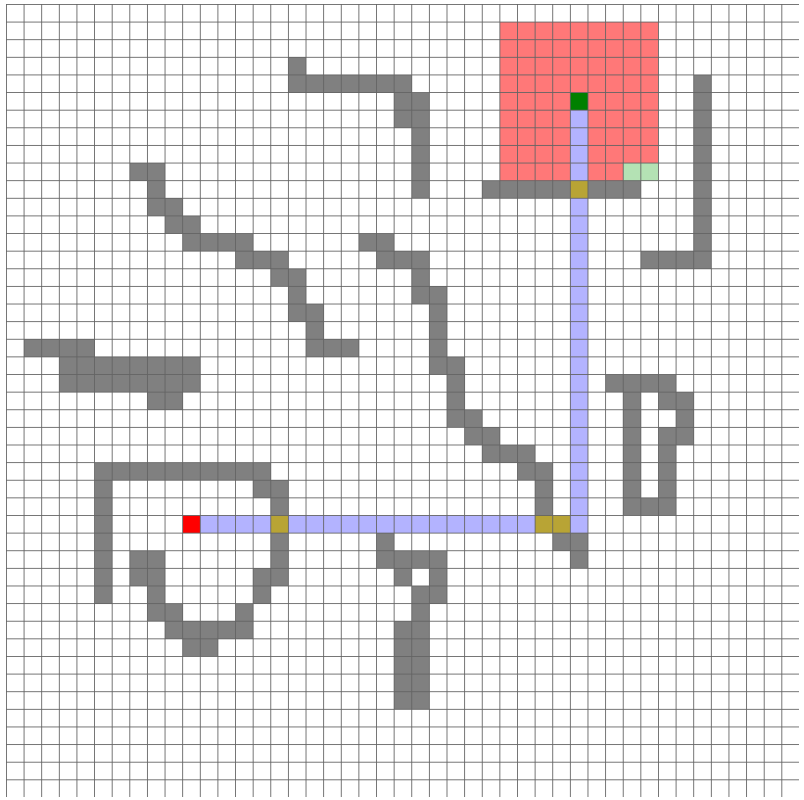
PROFIT
Статья на AAAI’23 (одной из ведущих конференции по ИИ в мире)

Работ сделана < 4 месяца

Планирование траектории

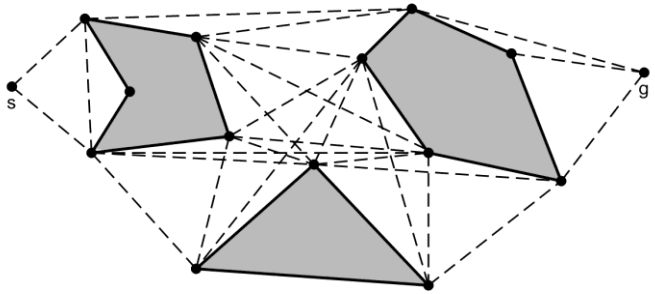


LIAN



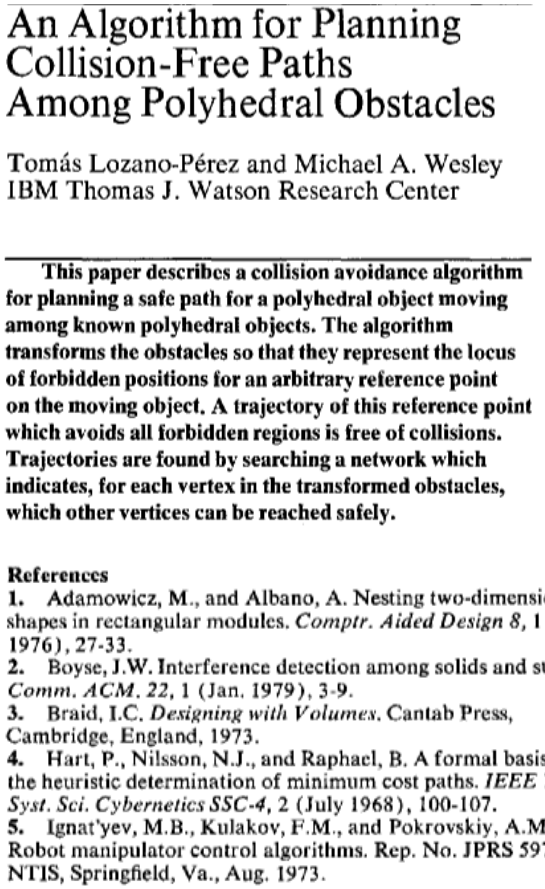
D*Lite

Планирование траектории



Nodes – vertices of the polygons bounding the obstacles in *C_free*

Edges – visible connections in C_free (determined via line-of-sight function)



An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles

Tomás Lozano-Pérez and Michael A. Wesley
IBM Thomas J. Watson Research Center

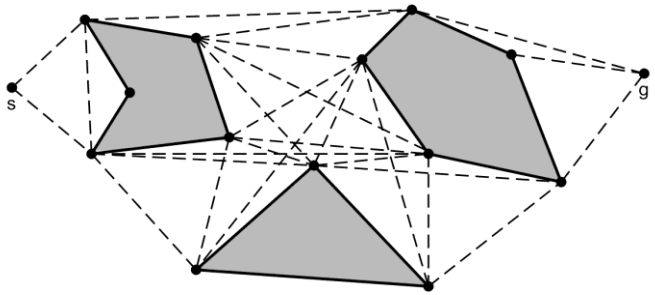
This paper describes a collision avoidance algorithm for planning a safe path for a polyhedral object moving among known polyhedral objects. The algorithm transforms the obstacles so that they represent the locus of forbidden positions for an arbitrary reference point on the moving object. A trajectory of this reference point which avoids all forbidden regions is free of collisions. Trajectories are found by searching a network which indicates, for each vertex in the transformed obstacles, which other vertices can be reached safely.

References

1. Adamowicz, M., and Albano, A. Nesting two-dimensional shapes in rectangular modules. *Comptr. Aided Design 8*, 1 (Jan. 1976), 27-33.
2. Boyse, J.W. Interference detection among solids and surfaces. *Comm. ACM. 22*, 1 (Jan. 1979), 3-9.
3. Braid, I.C. *Designing with Volumes*. Cantab Press, Cambridge, England, 1973.
4. Hart, P., Nilsson, N.J., and Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybernetics SSC-4*, 2 (July 1968), 100-107.
5. Ignat'yev, M.B., Kulakov, F.M., and Pokrovskiy, A.M. Robot manipulator control algorithms. Rep. No. JPRS 59717, NTIS, Springfield, Va., Aug. 1973.

Lozano-Pérez, T., & Wesley, M. A. (1979). An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, *22*(10), 560-570.

Планирование траектории



Nodes – vertices of the polygons bounding the obstacles in *C_free*

Edges – visible connections in C_free (determined via line-of-sight function)

An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles

Tomás Lozano-Pérez and Michael A. Wesley
IBM Thomas J. Watson Research Center

This paper describes a collision avoidance algorithm for planning a safe path for a polyhedral object moving among known polyhedral objects. The algorithm transforms the obstacles so that they represent the locus of forbidden positions for an arbitrary reference point on the moving object. A trajectory of this reference point which avoids all forbidden regions is free of collisions. Trajectories are found by searching a network which indicates, for each vertex in the transformed obstacles, which other vertices can be reached safely.

References

1. Adamowicz, M., and Albano, A. Nesting two-dimensional shapes in rectangular modules. *Comptr. Aided Design 8*, 1 (Jan. 1976), 27-33.

2. Boyse, J.W. Interference detection among solids and surfaces. *Comm. ACM. 22*, 1 (Jan. 1979), 3-9.

3. Braid, I.C. *Designing with Volumes*. Cantab Press, Cambridge, England, 1973.

4. Hart, P., Nilsson, N.J., and Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybernetics SSC-4*, 2 (July 1968), 100-107.

5. Ignat'yev, M.B., Kulakov, F.M., and Pokrovskiy, A.M. Robot manipulator control algorithms. Rep. No. JPRS 59717, NTIS, Springfield, Va., Aug. 1973.

Lozano-Pérez, T., & Wesley, M. A. (1979). An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, *22*(10), 560-570.

Background

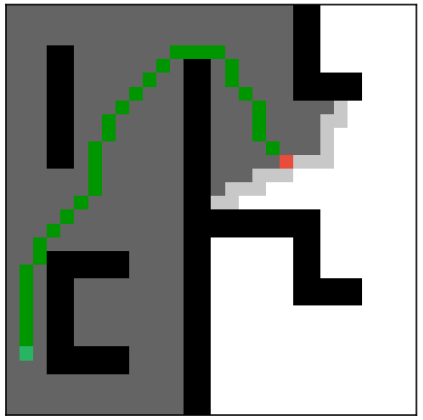
A* - **heuristic search** algorithm, that iteratively expands nodes based on their **f**-values

$$f(s) = g(s) + h(s)$$

g(s) - current cost of the path from **start** to **s** (computed by the algorithm)

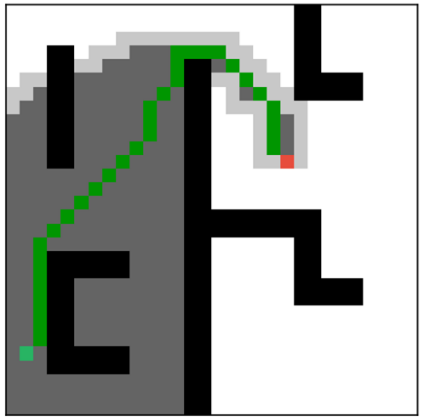
h(s) - heuristic estimate of the cost from **s** to **goal** (provided as an input)

h(s) = 0



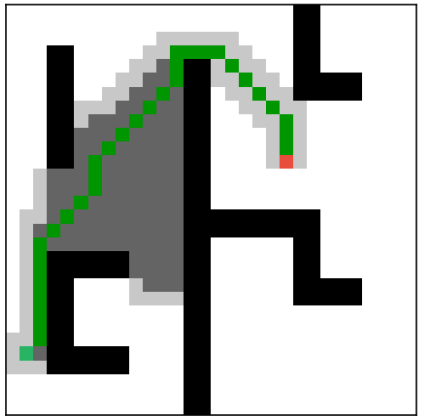
Dijkstra

h(s) = Octile Distance



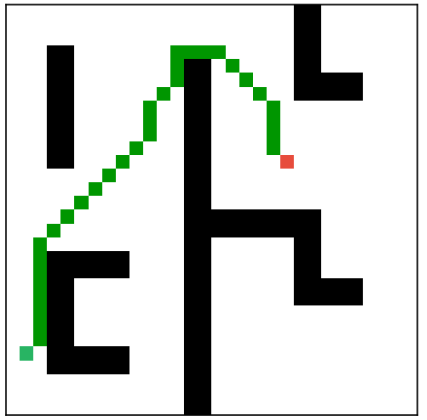
Regular A*

h(s) = 2*Octile Distance



Weighted A*

h(s) = h*(s)



A* with perfect heuristic

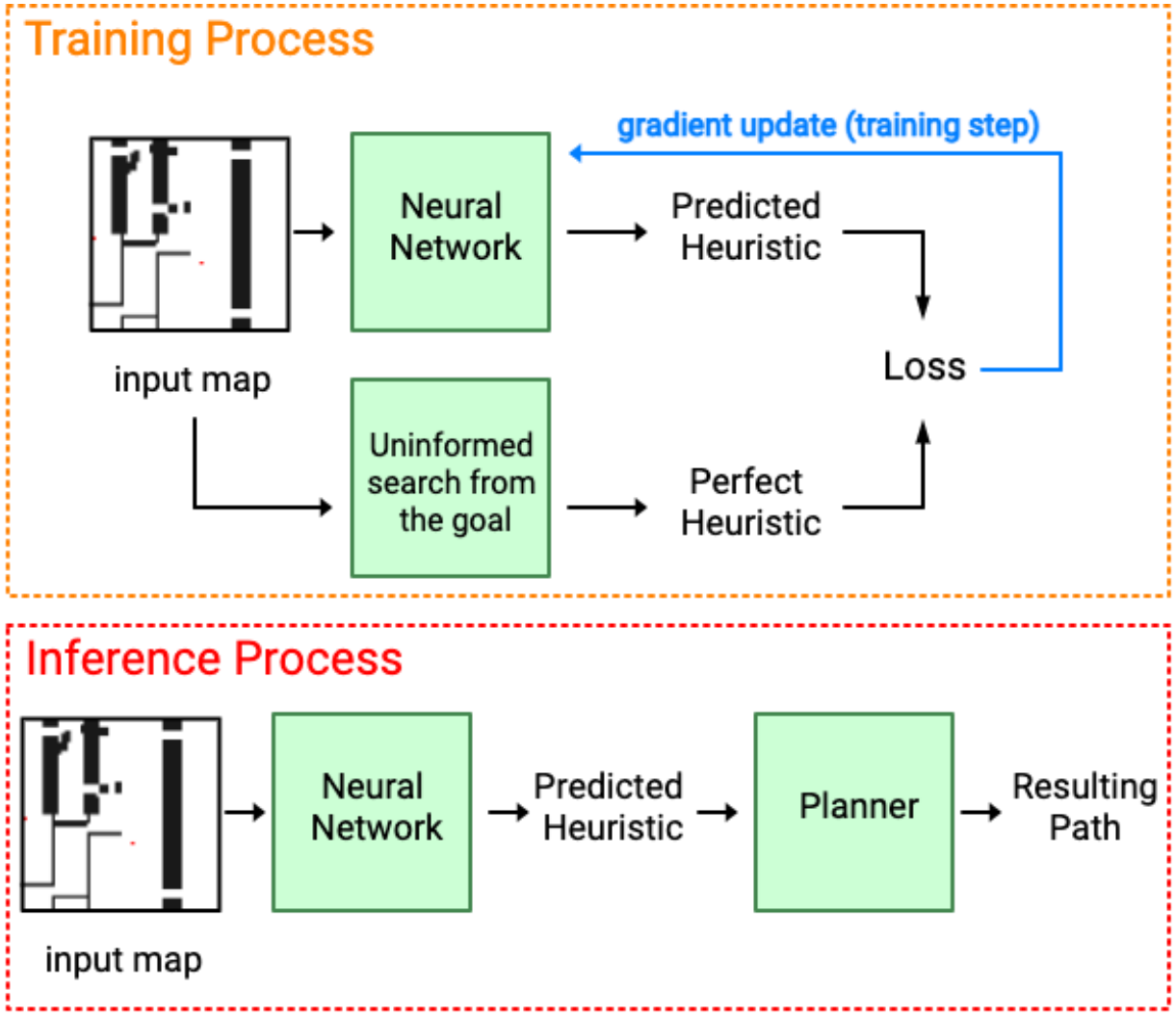
Problem and Solution Idea

- Heuristic search performance strongly depends on a *given* heuristic function
- Typically, heuristic functions are **not instance-dependent**, which results in **excessive expansions**
  - obstacles are not taken into account

Problem

Solution

- **informed heuristics** that are instance- dependent and take obstacles into account
- Use SOTA **machine learning** to construct such heuristics

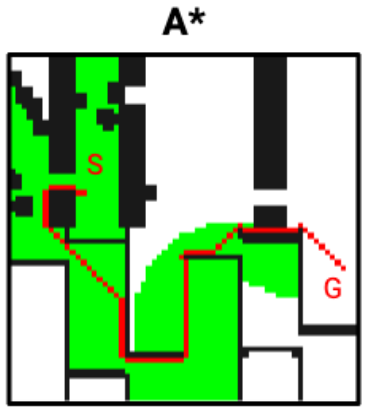


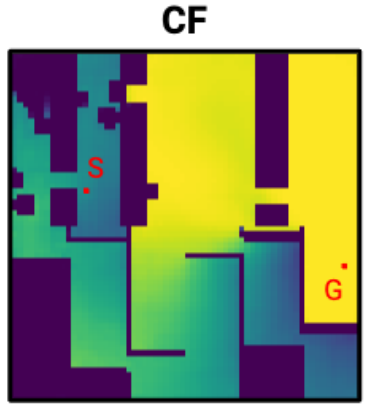
What Do We Learn 1: Correction Factor

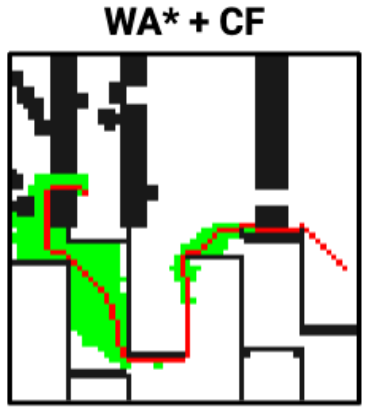
*Correction factor (*CF*)*: the ratio between the Octile distance (instance-independent) and the perfect heuristic (instance dependent)

$$cf(n) = h(n)/h^*(n)$$

- CF is naturally bounded by [0, 1] interval (good for neural networks)
- CF emcopasses the information on two heuristic functions
- For training phase: uninformed search from the goal node is utilized to get the ground-truth values
- After training: use CF as an additional weight within Weighted A* (Pohl et al., 1970)







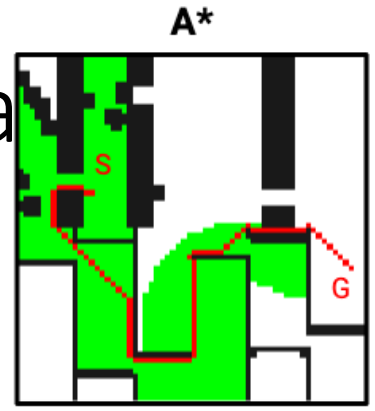
Pohl, I. 1970. Heuristic search viewed as path finding in a graph. Artificial intelligence, 1(3-4): 193–204.

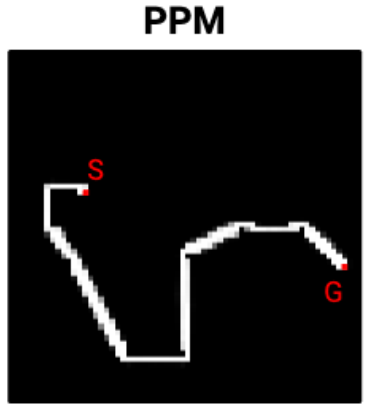
What Do We Learn 2: Path Probability Ma

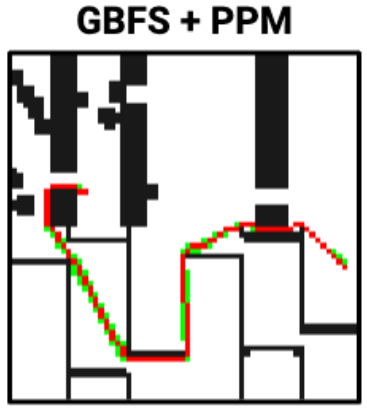
- *Path probability map (*PPM*)*: How likely the node is lying on the (shortest) path from start to goal

$$cost(\pi(s,g))/(cost(\pi(s,n)+cost(\pi(n,g))$$

- For training phase Theta* (Nash et al., 2007) is used to get ground truth PPM values
- After training: Use PPM as the secondary heuristic in Focal Search (Pearl and Kim, 1982)







Nash, A.; Daniel, K.; Koenig, S.; and Felner, A. 2007. Theta*: Any-Angle Path Planning on Grids. In Proceedings of The 22nd AAAI Conference on Artificial Intelligence (AAAI 2007), 1177–1183.

Pearl, J.; and Kim, J. H. 1982. Studies in semi-admissible heuristics. IEEE transactions on pattern analysis and machine intelligence, (4): 392–399.

Method: Planners

Generic search algorithm to exploit different types of heuristics:

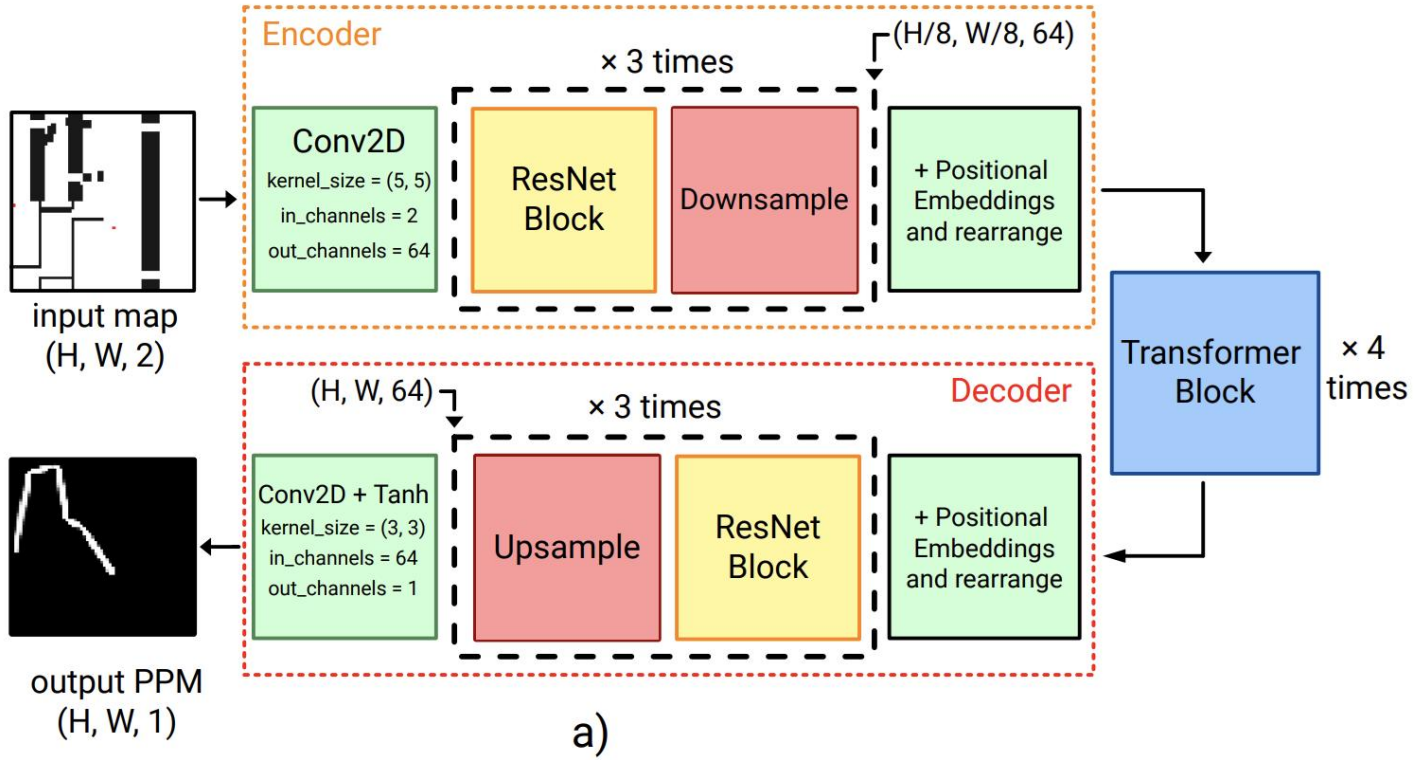
- A* executes black parts only
- WA* - black and **red**
- Focal Search (FS) – black and **purple**
- FS turns into the Greedy Best First Search (GBFS), if *w* is big enough
- WA* + CF – black and **blue**
- Octile distance is used as the basic heuristic function

Input: Grid Gr, *start* node, *goal* node, heuristic function h, sub-optimality factor w, h_{FOCAL} – secondary heuristic for Focal Search
Output: path π

```
1  g(start) := 0; ∀n ≠ start g(n) := ∞
2  OPEN := {start}; CLOSED := ∅
3  while OPEN ≠ ∅ do
4      n := GetBestNode(OPEN, FOCAL, h_FOCAL)
5      remove n from OPEN and FOCAL
6      insert n into CLOSED
7      if f_min has changed then
8          update FOCAL
9      if n is goal then
10         return ReconstructPath(n)
11     for each n' in GetSuccessors(Gr,n) do
12         if g(n') > g(n) + cost(n, n') then
13             g(n') := g(n) + cost(n, n')
14             f(n') := g(n')+w·h(n')/w(n')
15             update or insert n' in OPEN
16             if f(n') ≤ w · f_min then
17                 update or insert n' in FOCAL
18 return path not found
```

Method: Neural Network

- Main components: convolutional encoder, decoder and transformer blocks
- The network includes ~1m trainable parameters
- Netw $\mathrm{Loss} = \mathrm{MSE}(h_{gt}, h_{pred})$ irect supervision

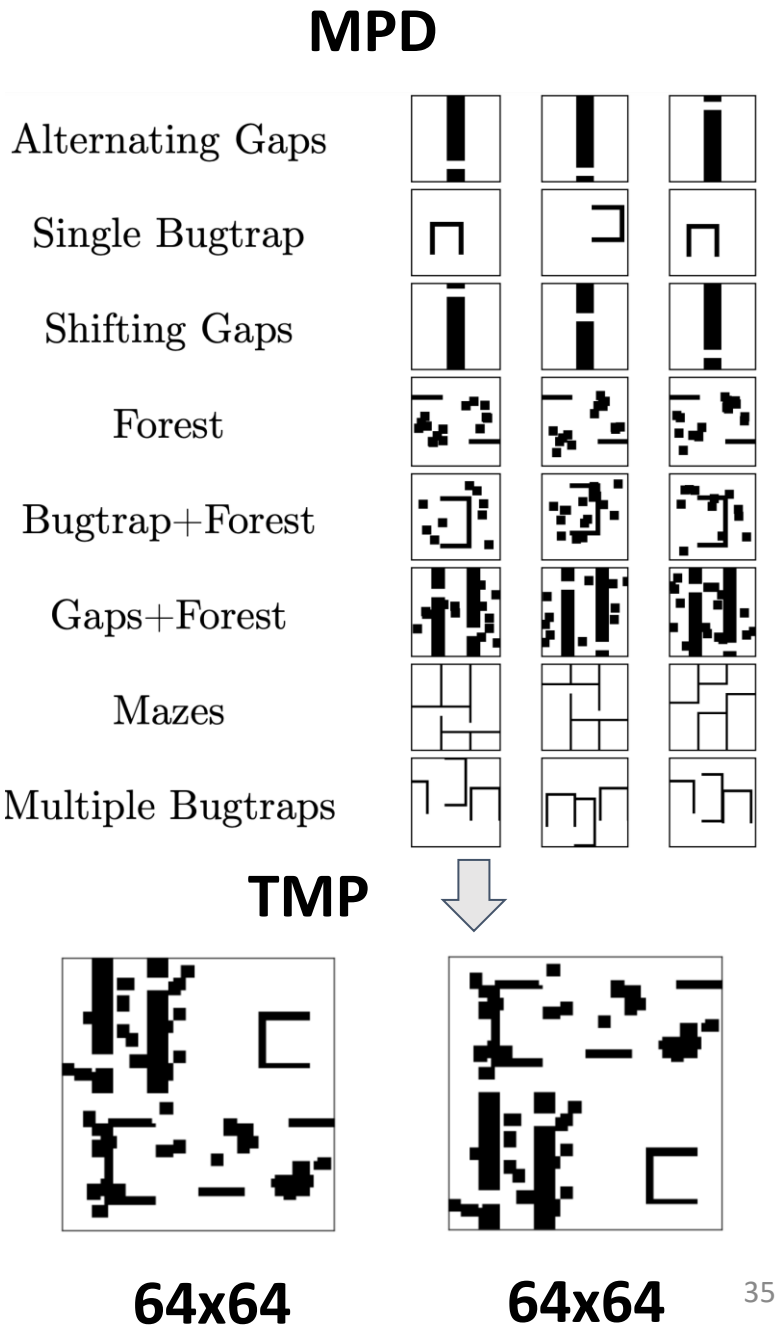


Experiments: Dataset

- TMP (Yonetani et al. 2021) maps based on the obstacles from the Motion Planning Dataset (MPD) (Bhardwaj et al. 2017).
  - Map size 64 x 64
- TMP data was augmented to 64k samples via mirroring and rotating
- Goals and starts were chosen randomly
  - 1 goal
  - 10 starts

= 10 instances per map

- Resulting dataset includes 640k samples
  - Train 512k
  - Validation 64k
  - Test 64k

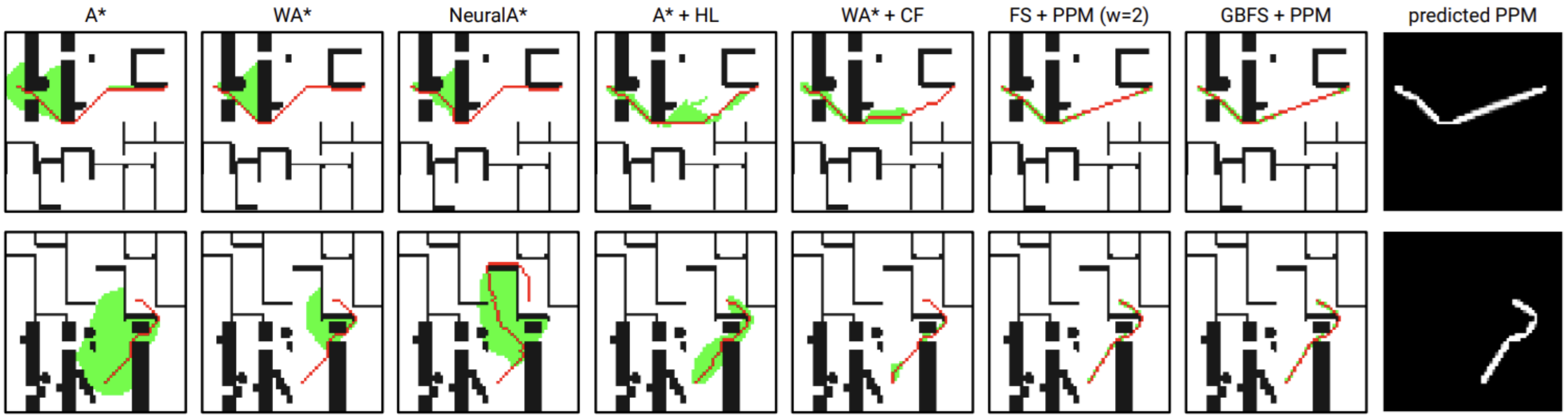


Bhardwaj, M., Choudhury, S., and Scherer, S. Learning heuristic search via imitation. In Proceedings of the Conference on Robot Learning (CoRL), 2017.

Experiments: Results

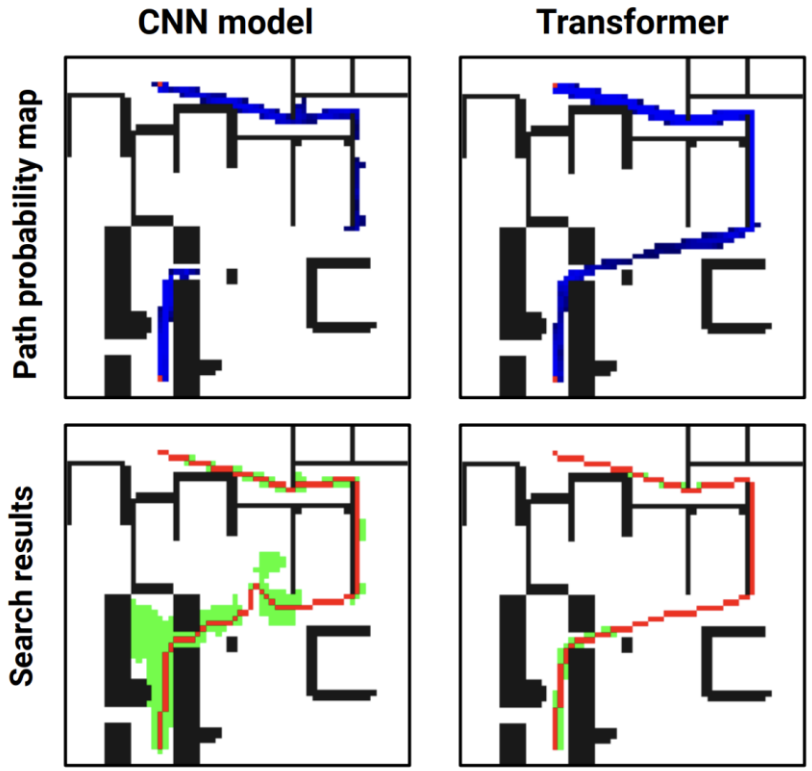
- Expansion Ratio best: **GBFS + PPM**
- Cost Ratio best: **FS + PPM**
- Optimal Found Ratio best: **WA* + CF**

	Optimal Found Ratio (%) ↑	Cost Ratio (%) ↓	Expansions Ratio (%) ↓
A*	100	100	100
WA*	40.66	103.52 ±4.85	44.43 ±25.92
Neural A*	29.82	104.90 ±6.56	52.30 ±30.47
A*+HL	79.11	100.27 ±0.62	80.50 ±74.40
WA*+CF	**85.40**	100.25 ±1.13	36.98 ±21.18
FS+PPM	82.97	**100.24 ±0.74**	26.36 ±21.08
GBFS+PPM	83.02	100.25 ±0.90	**23.60 ±18.34**



Experiments: Transformer Blocks

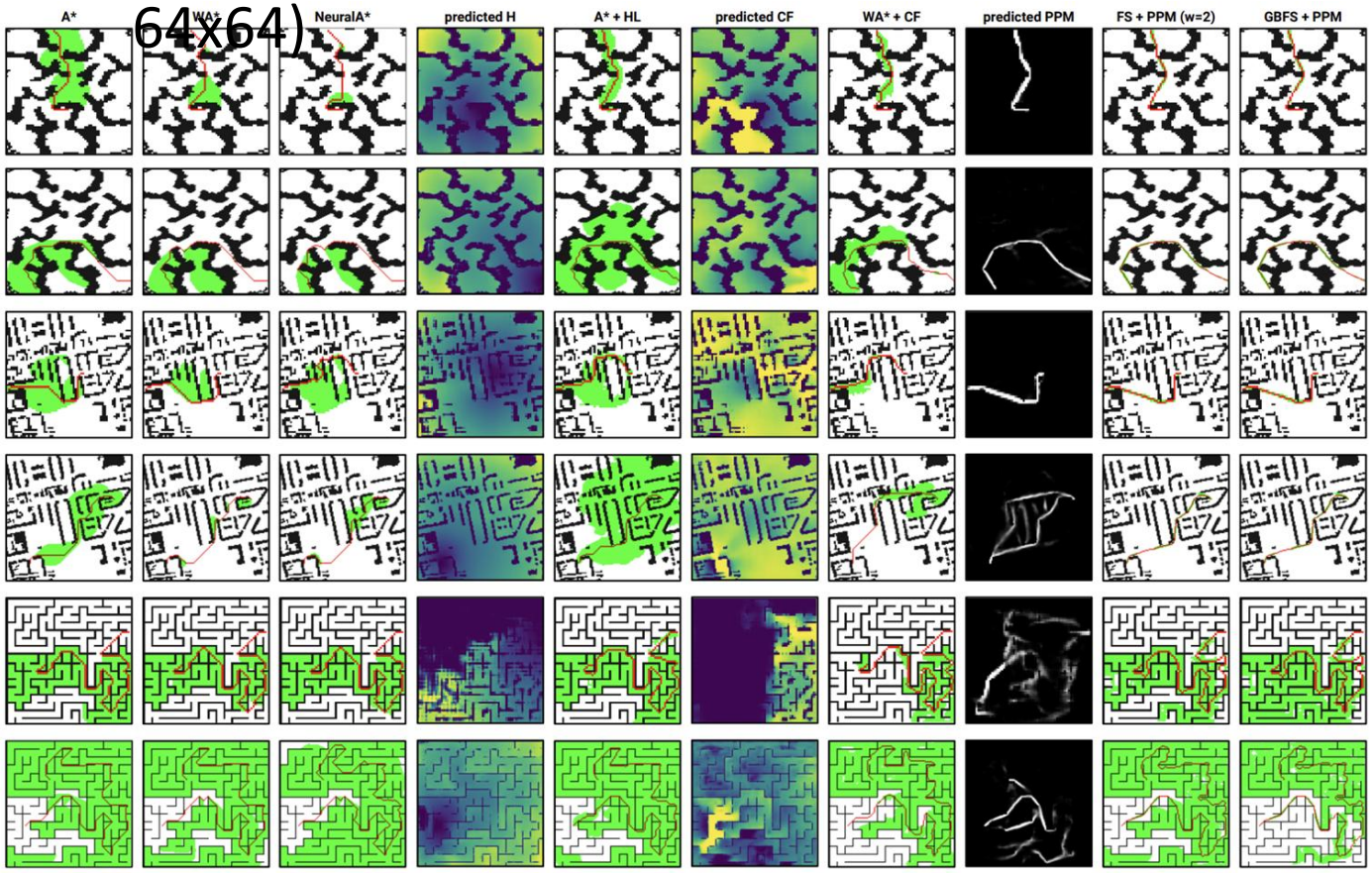
- Replacing Transformer blocks with ResNet blocks reduces performance
- CNN model predict fragmented paths, which lead to excessive expansions and costly solutions



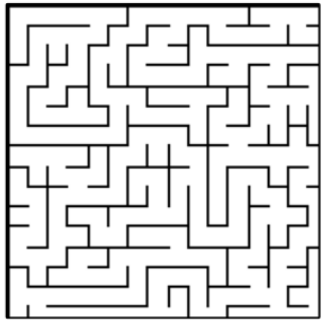
FS+PPM ($w = 4$)	w/ Trans	w/o Trans
Optimal Found Ratio (%)	85.22	61.74
Average Cost Ratio (%)	100.31 ± 1.58	101.12 ± 2.19
Average Expansions Ratio (%)	16.06 ± 11.57	19.65 ± 17.03
MSE $\times 10^{-3}$	**3.2**	5.3

Experiments: Out of Distribution Performance

MovingAI benchmark
Different map structure and size (128x128 and 64x64)




Berlin_1


Maze512-32-0


BigGameHunters

	Optimal Found Ratio (%) ↑	Cost Ratio (%) ↓	Expansions Ratio (%) ↓
A*	100	100	100
WA*	8.13	104.31 ±4.76	57.52 ±30.72
Neural A*	3.24	107.10 ±6.77	63.08 ±34.63
A*+HL	**29.02**	**101.90** ±2.72	148.94 ±136.95
WA*+CF	10.61	106.10 ±5.59	63.64 ±36.31
FS+PPM	18.66	105.62 ±5.61	55.06 ±39.57
GBFS+PPM	18.59	106.12 ±6.54	**54.33** ±**47.24**

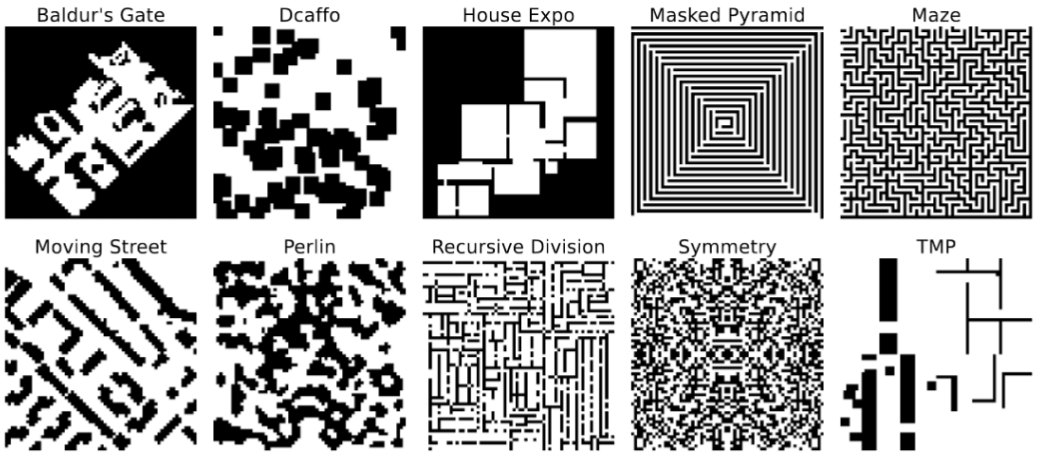
Follow-up. МКН выходит на сцену

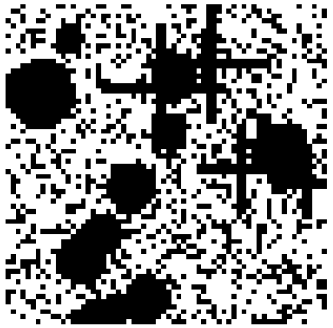
Идея:

Универсальный TransPath
работает одинаково хорошо
на *любой* карте

Решение:

- Модифицированная (слегка) архитектура
- «Абстрактная», но «богатая» обучающая выборка





	Optimal Found Ratio (%) ↑	Length Ratio (%) ↓	Expansions Ratio (%) ↓
A*	100.00	100.0	100.0
UPath(woFig)	55.24	105.1±16.2	**45.3±31.9**
UPath	**72.63**	**101.1±4.1**	47.4±27.7
WA*, w=2	32.35	103.7±4.9	54.6±30.1
WA*, w=3	24.13	105.2±6.5	49.8±30.1
WA*, w=5	14.38	107.9±8.9	47.3±29.8
TransPath	32.34	125.9±49.7	111.4±134.3

Table 1: Performance comparison on 64x64 UPF.

Результаты

1. Всех побили
2. Подали статью на AAAI’26
   - Прошли во 2ю фазу отбора, ждём рецензии

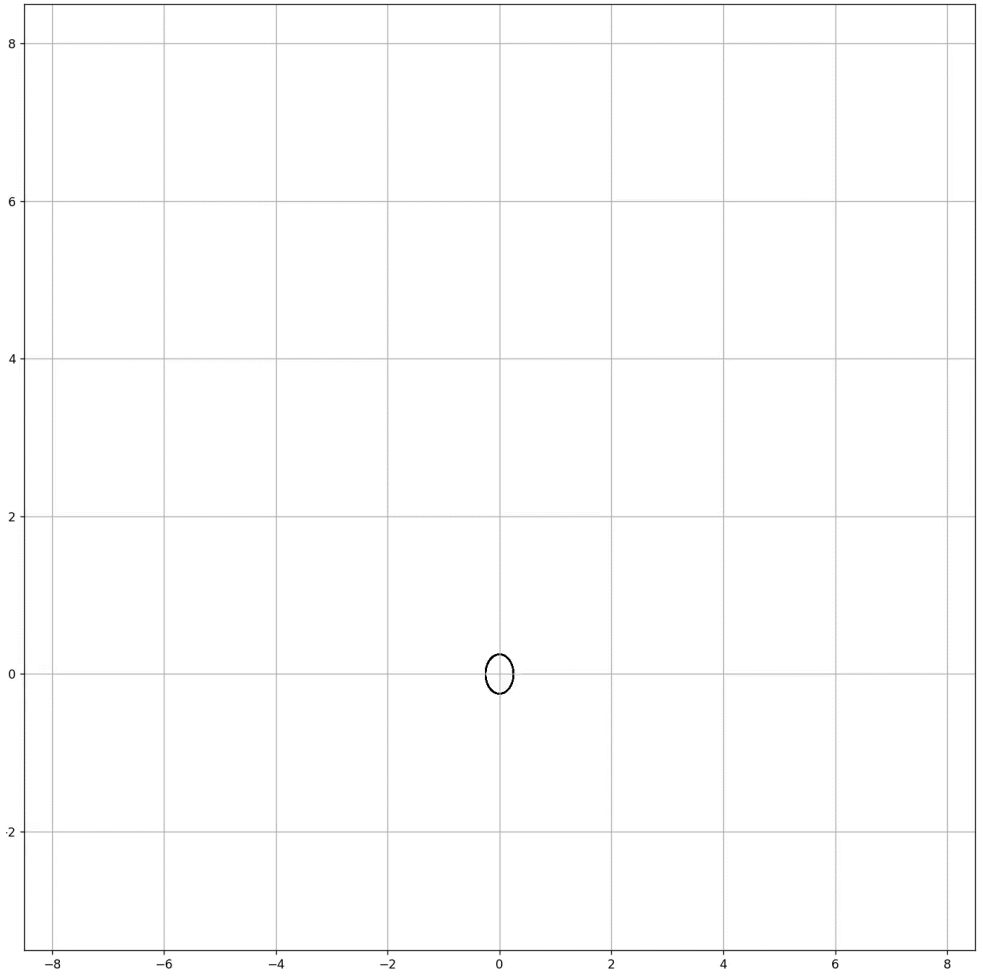
Выч.ресурсы

Размер сетки 1-2M
1 x A100 40Gb

Follow-up. МКН выходит на сцену

Идеи для проектов

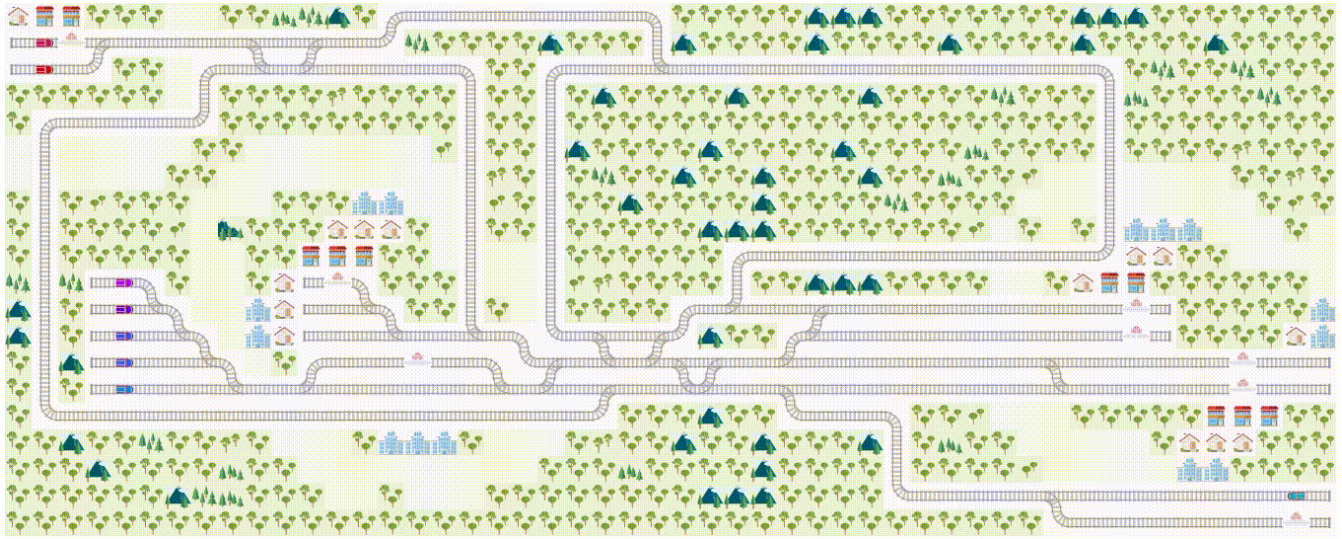
- Более сложная постановка задачи
  - 3D
  - Небинарный грид (каждая клеточка имеют свою собственную стоимость)
  - Примитивы движения
- Дистиляция сетки до того состояния, что она работает в real-time на Nvidia Jetson



Multi-agent Pathfinding (MAPF)

О задаче

- *n* агентов
- общая среда
- *n* стартов, *n* финишей
- Нужно довести всех агентов до финиша



Централизованный и децентрализованный вариант

Centralized

- single planner
- full observability
- information sharing

Idea

- [Central planner] Plans a set of collision-free paths
- Each agent follows its **path**

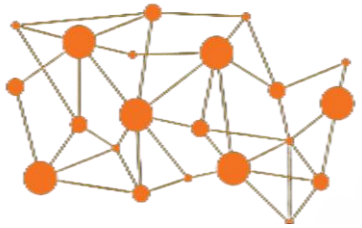
Pros

- Strong theoretical guarantees
- High-quality (optimal) solutions

Contras

- Computationally intensive
- The scalability is limited

Decentralized



- each agent decides
- partial observability
- limited information sharing

Idea

- [Each agent] Observes local surrounding
- Each agent decides the **next action**

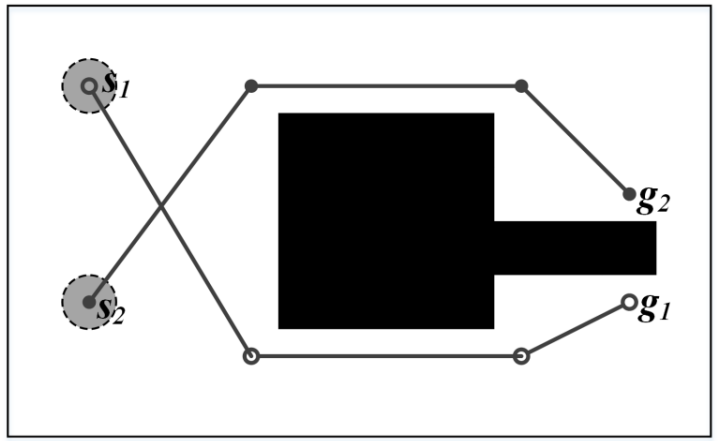
Pros

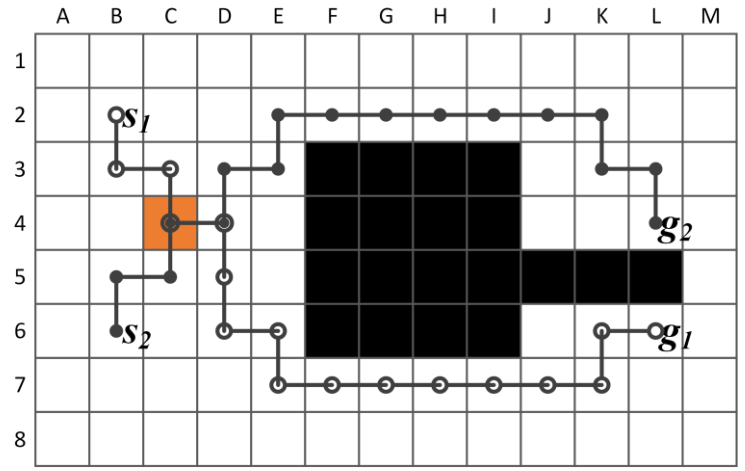
- Fast
- Highly scalable

Contras

- No (limited) theoretical guarantees
- Sub-optimal solutions

Централизованный MAPF. Классические допущения.





Время - дискретно

T = {0, 1, 2, 3, 4, …}

Раб.пространство - граф

G = (V, E)

V — локации

E — перемещения между ними

Действия

- ожидание (в текущей вершине)
- перемещение (в смежную вершину)

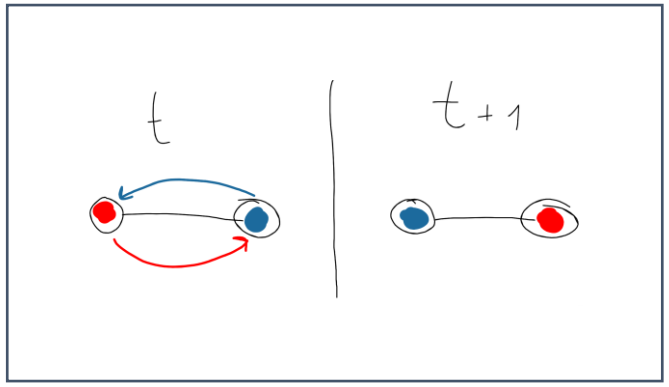
План

Последовательность действий

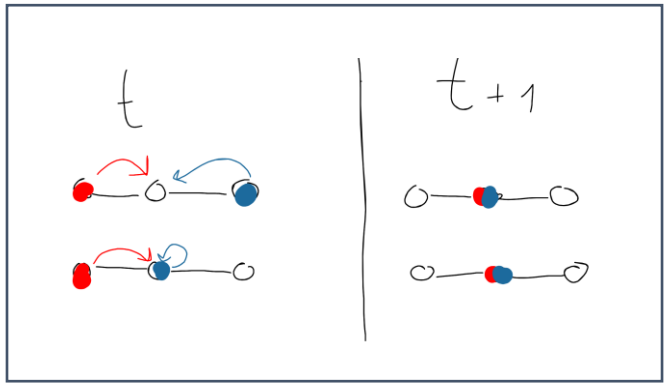
Стоимость/вес плана – число действий в нём

Классическая задача (MAPF)

- Вход: Граф, n начальных вершин, n конечных вершин
- Выход: n **совместно неконфликтных** планов
  - минизируем либо суммарную стоимость либо makespan



Edge (swap) conflict



Vertex conflict

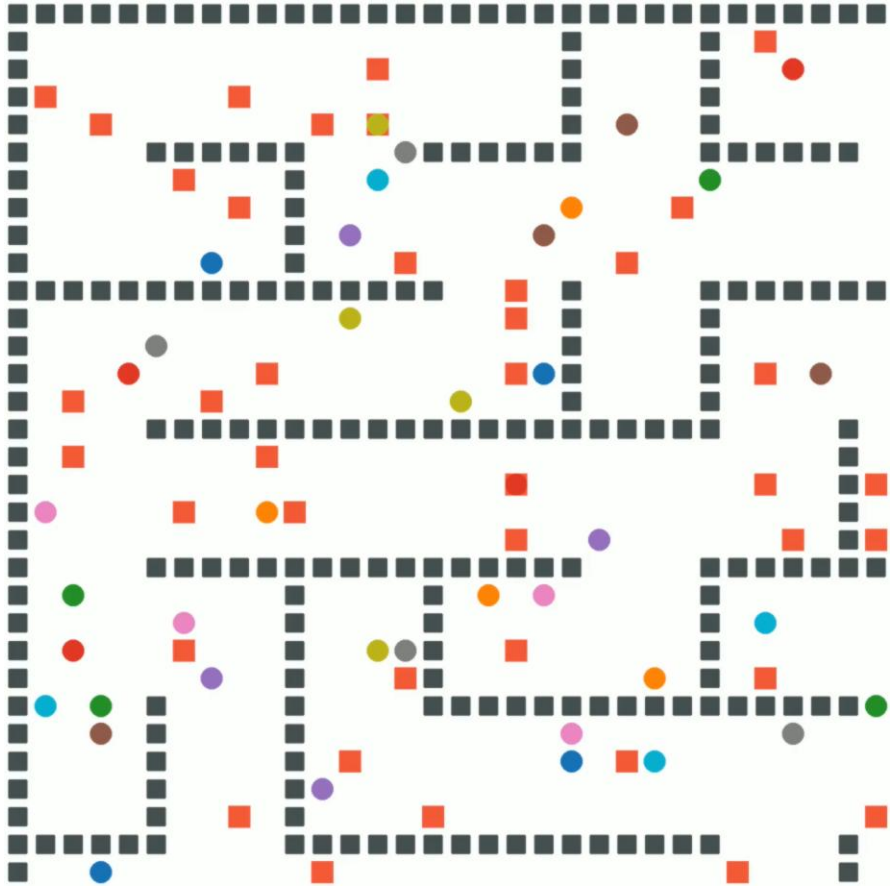
Objective (↓):

Sum-of-Costs $\sum_{i=1}^{n} cost(\pi_i)$

Makespan $\max_{i=1..n}(cost(\pi_i))$

Анонимный вариант (AMAPF)

- Вход: Граф, n начальных вершин, n конечных вершин, **назначения «старт-финиш» не фиксированы**
- Выход: n совместно неконфликтных планов
  - минизируем либо суммарную стоимость либо makespan



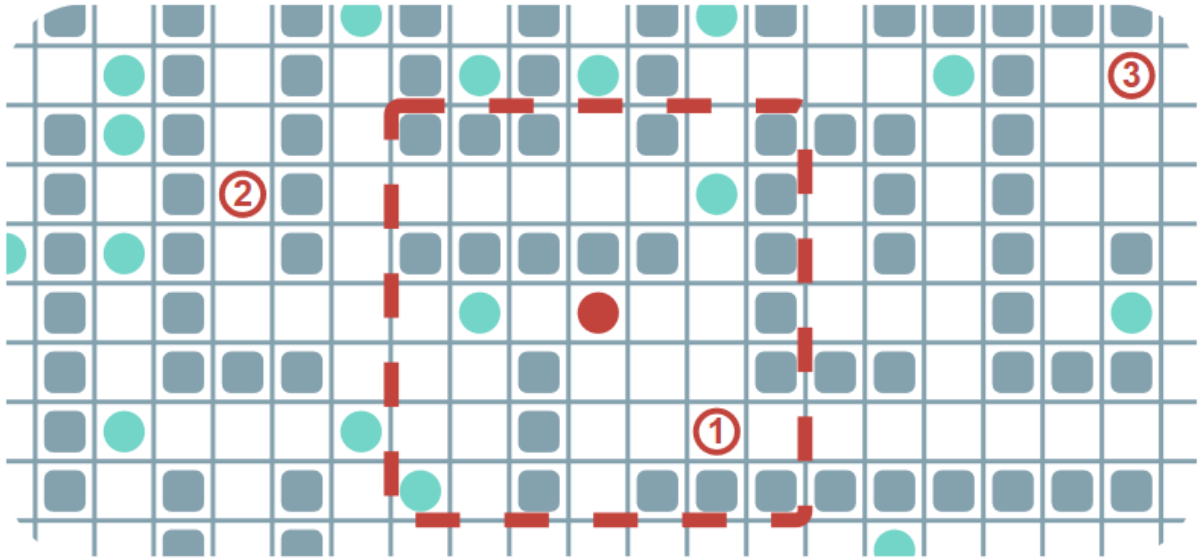
Objective (↓):

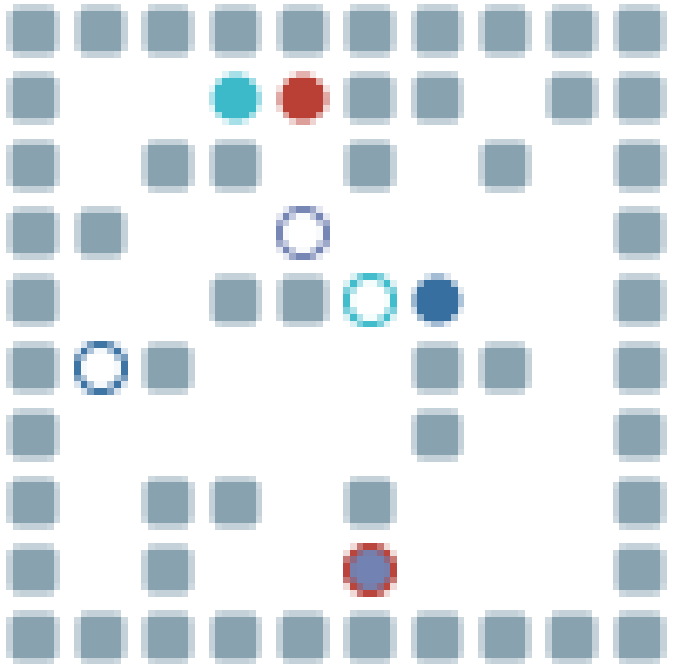
Sum-of-Costs $\sum_{i=1}^{n} cost(\pi_i)$

Makespan $\max_{i=1..n}(cost(\pi_i))$

Lifelong MAPF (LMAPF)

- Вход: Граф, n начальных вершин, n конечных вершин, **цели постоянно переназначаются**
- Выход: n совместно неконфликтных планов
  - На каждом шаге времени





Сложность задачи MAPF

?

Сложность задачи MAPF

Undirected graphs

- Decision variant **P**
  Kornhauser et al, 1984

- Optimization variant **NP-hard**
  Yu et al, 2016

Directed graphs

- Decision variant **NP-hard**
  Nebel, 2020

- Optimization variant **NP-hard**

Daniel Martin Kornhauser et al. Coordinating pebble motion on graphs, the diameter of permutation groups, and applications. In Proceedings of the 25th Annual Symposium on Foundations of Computer Science (FOCS 1984), pages 241–250, 1984.

Jingjin Yu et al. Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics. IEEE Transactions on Robotics, 32(5):1163–1177, 2016.

Bernhard Nebel. On the computational complexity of multi-agent pathfinding on directed graphs. In Proceedings of the 30th International Conference on Automated Planning and Scheduling (ICAPS 2020), pages 212–216, 2020

Сложность задачи MAPF

Undirected graphs

- Decision variant **P**
  Kornhauser et al, 1984
- Optimization variant **NP-hard**
  Yu et al, 2016

Directed graphs

- Decision variant **NP-hard**
  Nebel, 2020
- Optimization variant **NP-hard**

Направления исследований/задачи

- Параметризованная сложность
- Особые классы графов

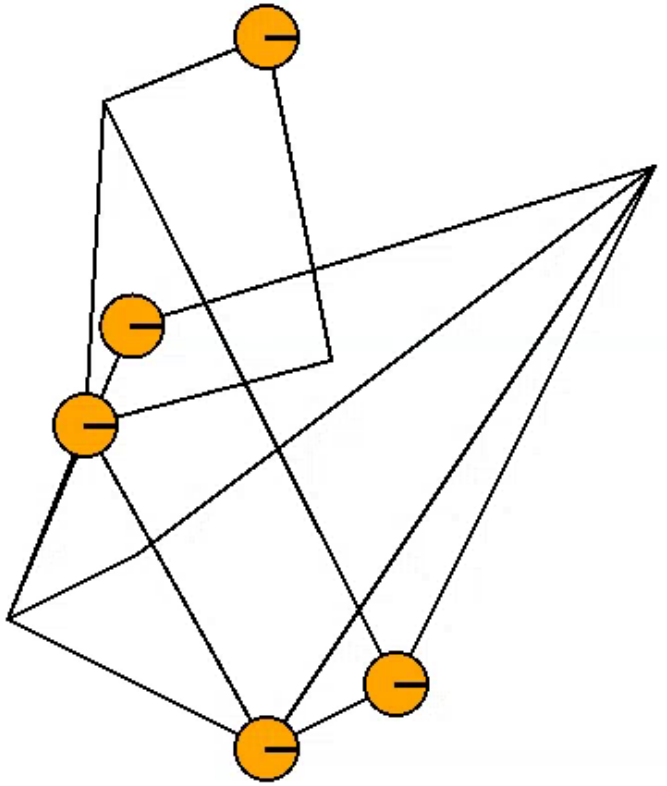
MAPF with Large Agents (MAPF-LA)

MAPF Undirected graphs

- Decision variant **P**
  Kornhauser et al, 1984

MAPF-LA Undirected graphs

- Decision variant **????**



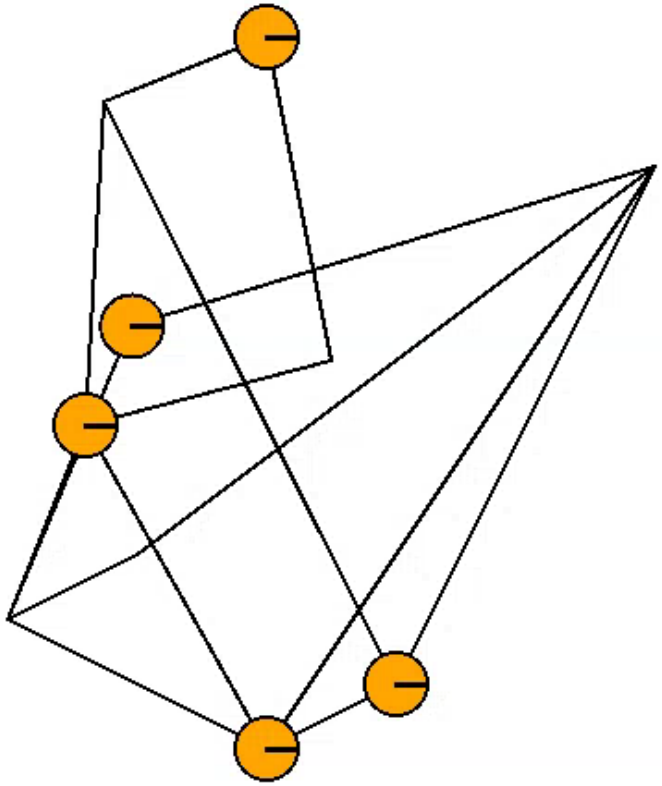
MAPF with Large Agents (MAPF-LA)

MAPF Undirected graphs

- Decision variant **P**
  Kornhauser et al, 1984

MAPF-LA Undirected graphs

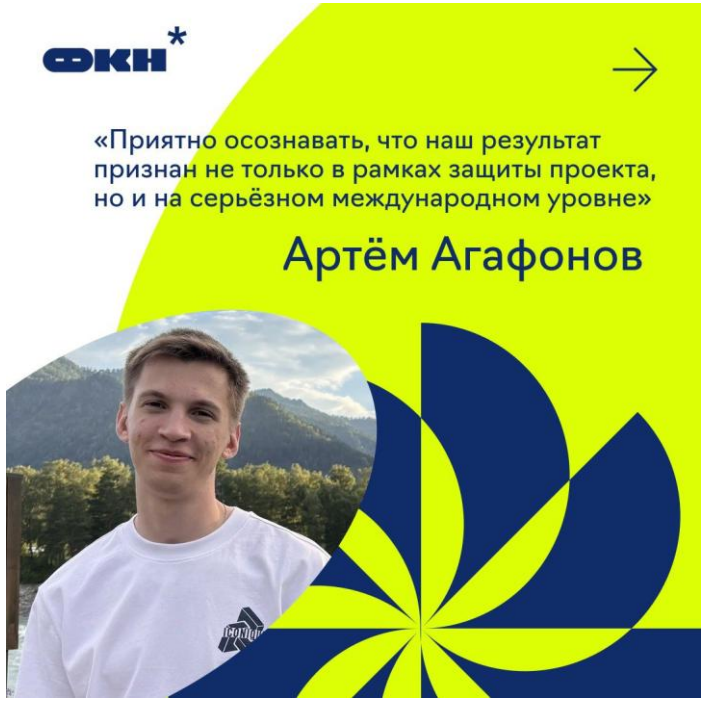
- Decision variant **NP-Hard**
  Agafonov & Yakovlev, 2025



MAPF-LA

MAPF-LA Undirected graphs

- Decision variant **NP-Hard**
  Agafonov & Yakovlev, 2025



Студент 2го курса

«Сначала я пытался придумать требуемый алгоритм»

Для этого были написаны программы для генерации теста задачи, для его решения полным перебором и для визуализации движения агентов в нём.
Я выдвигал разные гипотезы и проверял их с помощью этих программ. Но каждый раз я сталкивался с тем, что на каком-то тесте программа не работала.

Те проблемы, с которыми сталкивалось моё решение, навели меня на мысль, что решить задачу за полиномиальное время невозможно.

Мне показалось, что эта гипотеза объясняла, почему другие тоже не могли решить данную задачу. Поэтому я решил попробовать доказать это.

Ноябрь 24 – начало работы
Февраль 25 – первые оформленные результаты
Апрель 25 – подача статьи
Июль 25 – статья принята

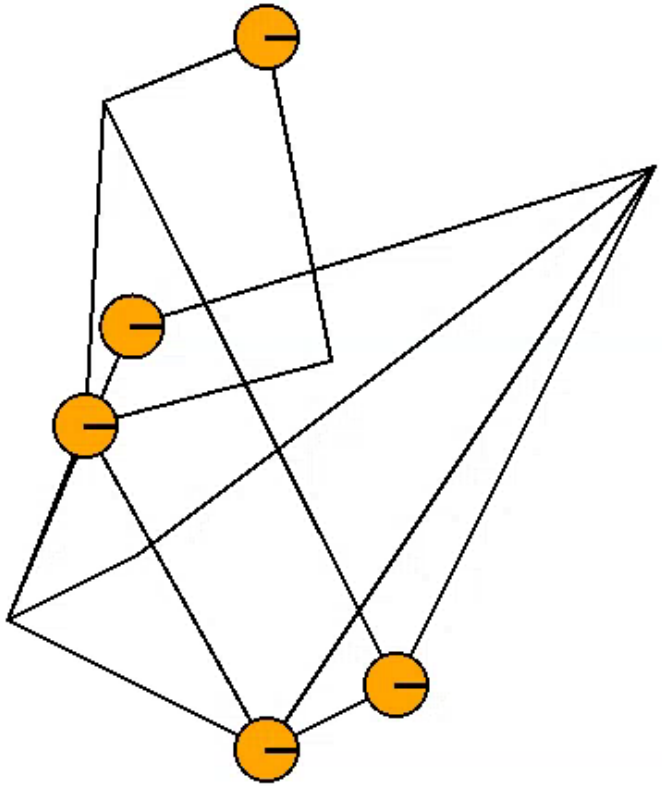
ECAI 2025 (рейтинг A)

На его месте мог бы быть ты! =)

MAPF-LA – надо копать дальше

MAPF-LA Undirected graphs

- Decision variant **NP-Hard**
  Agafonov & Yakovlev, 2025



Идея

Посмотреть другие классы графов (планарные, гриды)

Текущее доказательство опирается на близость вершин друг другу (и на наложение ребер)

Если (вдруг) получится сконструировать P алгоритм для (какого-то подкласса) MAPF-LA – **это статья! =)**

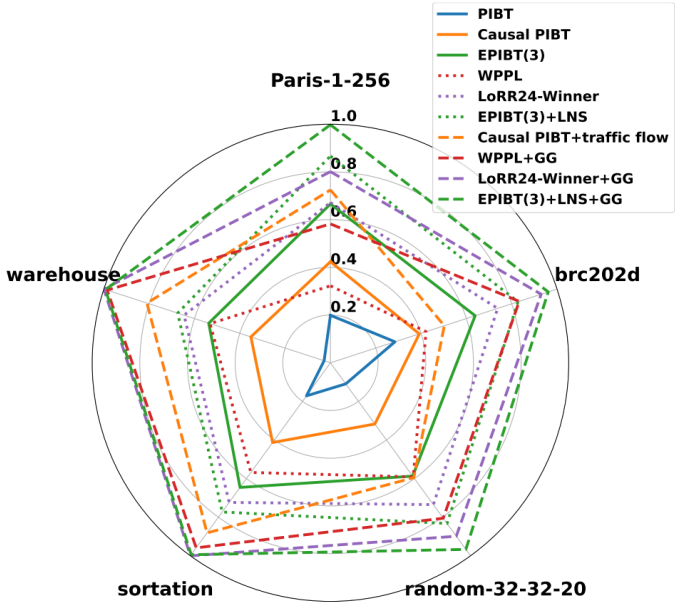
Централизованный MAPF

Много других интересных/нерешенных задач

- Которые решаются чисто алгоритмически
  - (если вы не любите ML, но любите алгоритмы – это для вас)

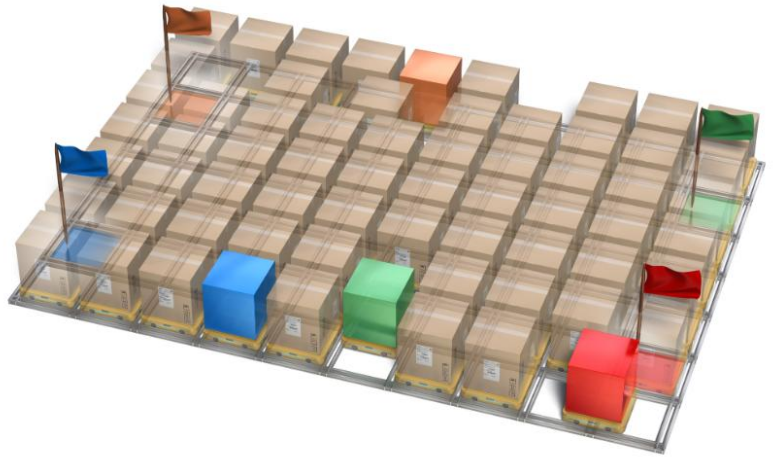
MAPF with Turns

SOTA решать (PIBT) «перестает» работать



MAPF HD

https://arxiv.org/abs/2509.06374



Не изучал внимательно, но уверен «мы можем сделать лучше»

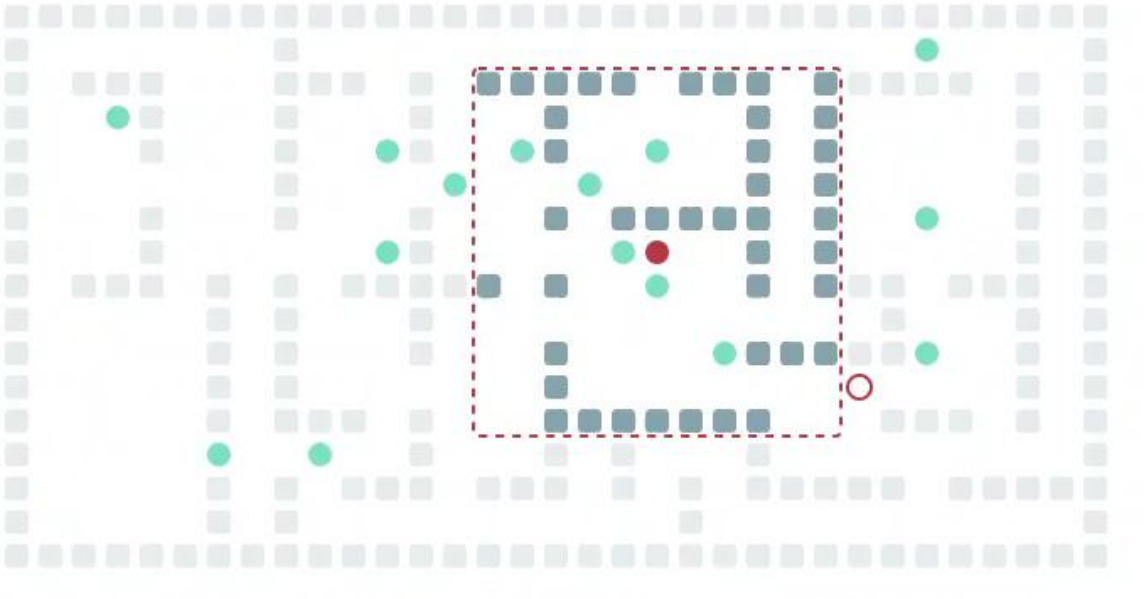
Децентрализованный MAPF/LMAPF/AMAPF

[decentralized] **No central planner**

[decentralized] **Each agent at each time step decides its next action**

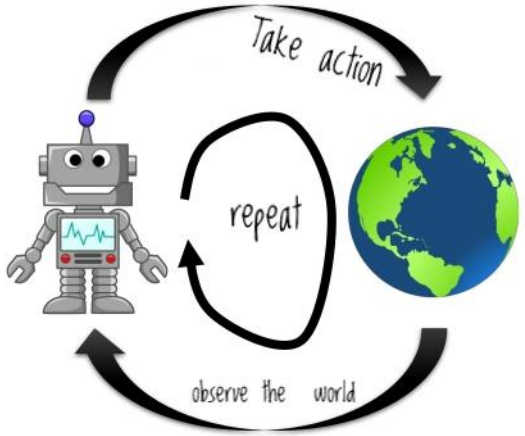
- limited observability
- limited communication

[lifelong] **After a goal is reached a new one is assigned**



Sequential decision making

- Observe
- Decide
- Act
- Repeat



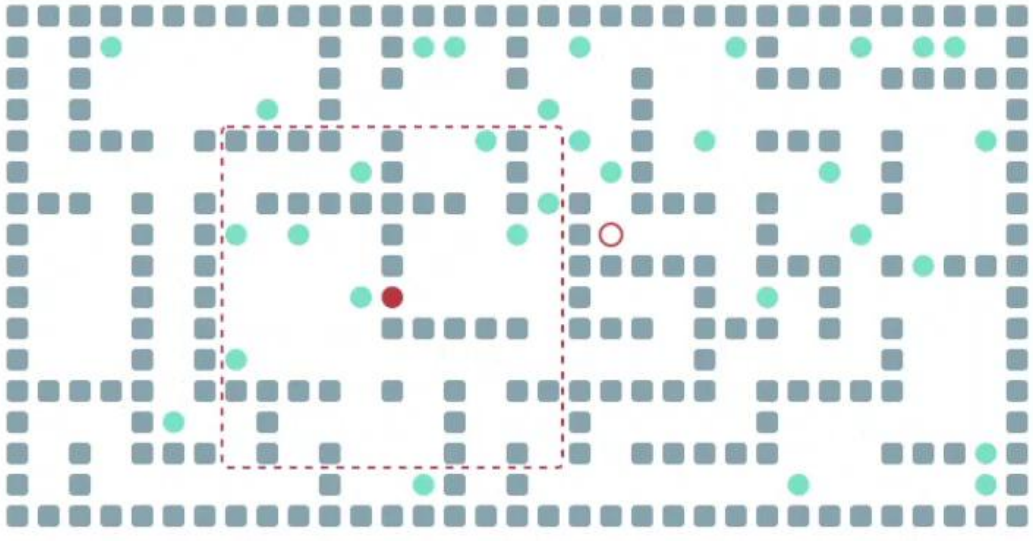
Почему бы просто каждому агенту не планировать A*?

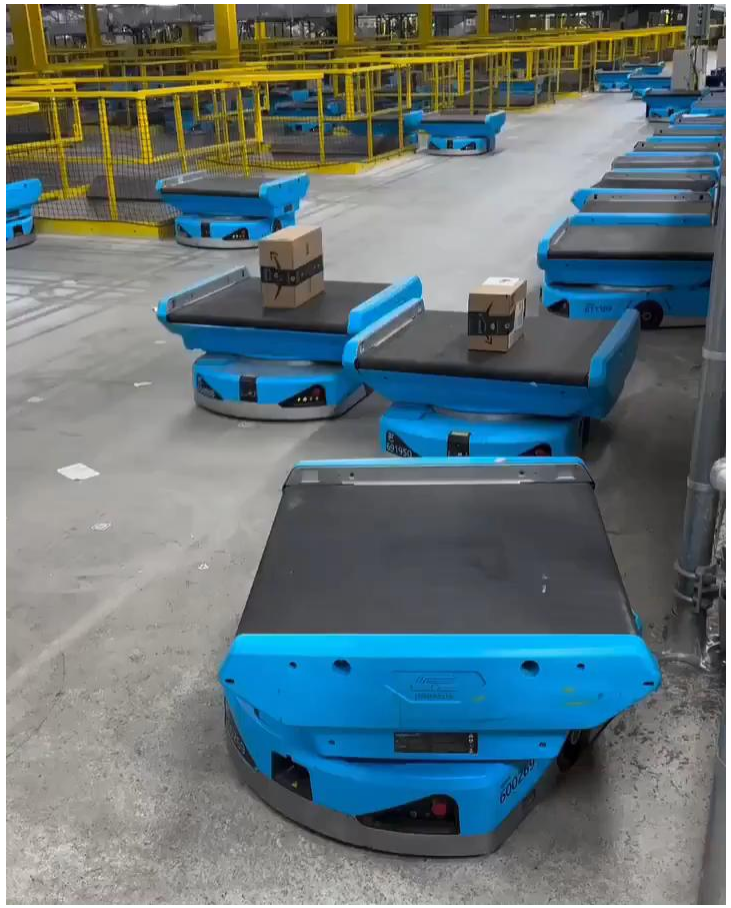
(each agent)

1. plans the shortest path to the goal
   ignoring the other agents
2. picks the first action of that plan

Почему бы просто каждому агенту не планировать A*?

Это не работает =(





goals achieved by the red agent = 0
goals achieved by all agents = 36
Throughput (512 steps, 32 agents) = 0.07

How about ‘pure’ reinforcement learning (RL)?

→ Non-stationary environment
→ Sparse reward
→ Hard combinatorial structure of the solution space

→ **Slow, unstable learning (costly?)**
→ **Low generalization**

Planning and Learning …

… is all you need

(to solve decentralized MAPF efficiently)

FOLLOWER

The Thirty-Eighth AAAI Conference on Artificial Intelligence (AAAI-24)

Learn to Follow: Decentralized Lifelong Multi-Agent Pathfinding via Planning and Learning

Alexey Skrynnik[1,2], **Anton Andreychuk**[1], **Maria Nesterova**[2,3], **Konstantin Yakovlev**[2,1], **Aleksandr Panov**[1,3]
[1]AIRI, Moscow, Russia
[2]Federal Research Center for Computer Science and Control of Russian Academy of Sciences, Moscow, Russia
[3]MIPT, Dolgoprudny, Russia
skrynnikalexey@gmail.com, andreychuk@airi.net, minesterova@yandex.ru, yakovlev@isa.ru, panov@airi.net

Abstract

Multi-agent Pathfinding (MAPF) problem generally asks to find a set of conflict-free paths for a set of agents confined to a graph and is typically solved in a centralized fashion. Conversely, in this work, we investigate the decentralized MAPF setting, when the central controller that possesses all the information on the agents' locations and goals is absent and the agents have to sequentially decide the actions on their own without having access to the full state of the environment. We focus on the practically important lifelong variant of MAPF, which involves continuously assigning new goals to the agents upon arrival to the previous ones. To address this complex problem, we propose a method that integrates two complementary approaches: planning with heuris-

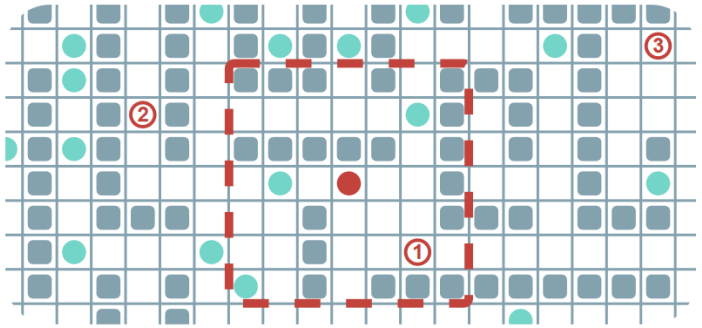


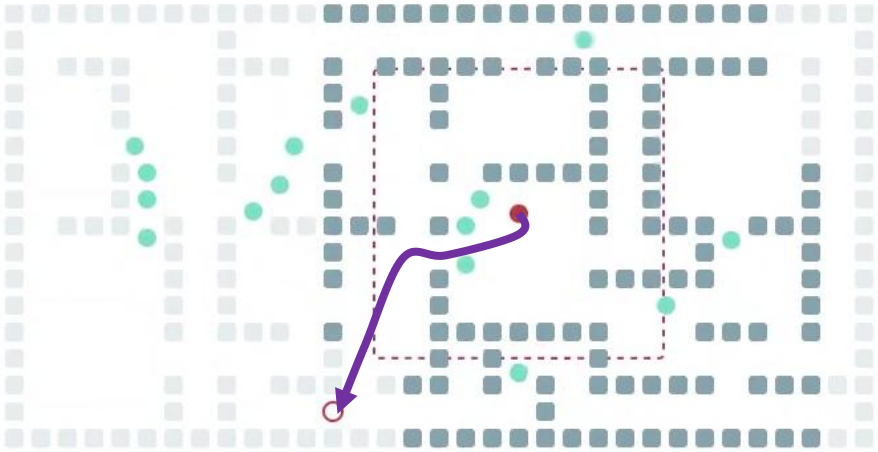
Figure 1: An example of a decentralized LMAPF instance. Agents are depicted as filled circles. The dashed line illus-

https://ojs.aaai.org/index.php/AAAI/article/view/29704/31207

Learn to Follow: Decentralized Lifelong Multi-Agent Pathfinding via Planning and Learning

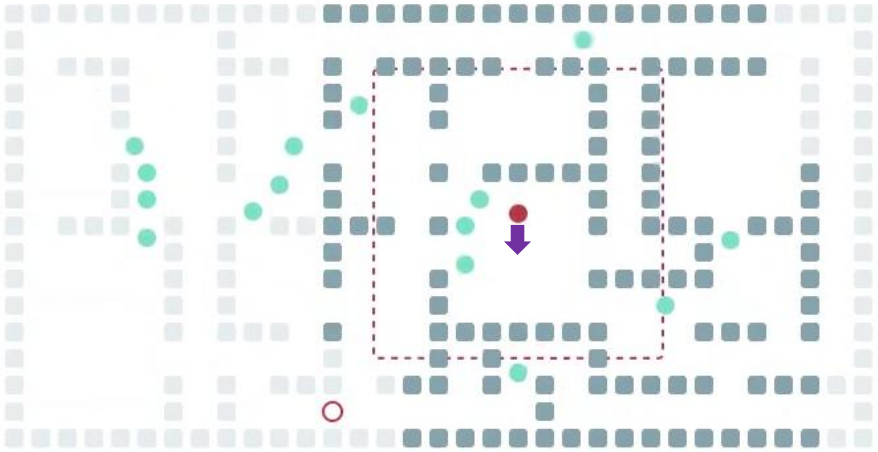
Two types of behavior:

- Moving towards the goal (**long-range planning**)
- Avoiding the other agents and/or yielding to them (**short term decision making**)



A* (heuristic search)

+



PPO (reinforcement learning)

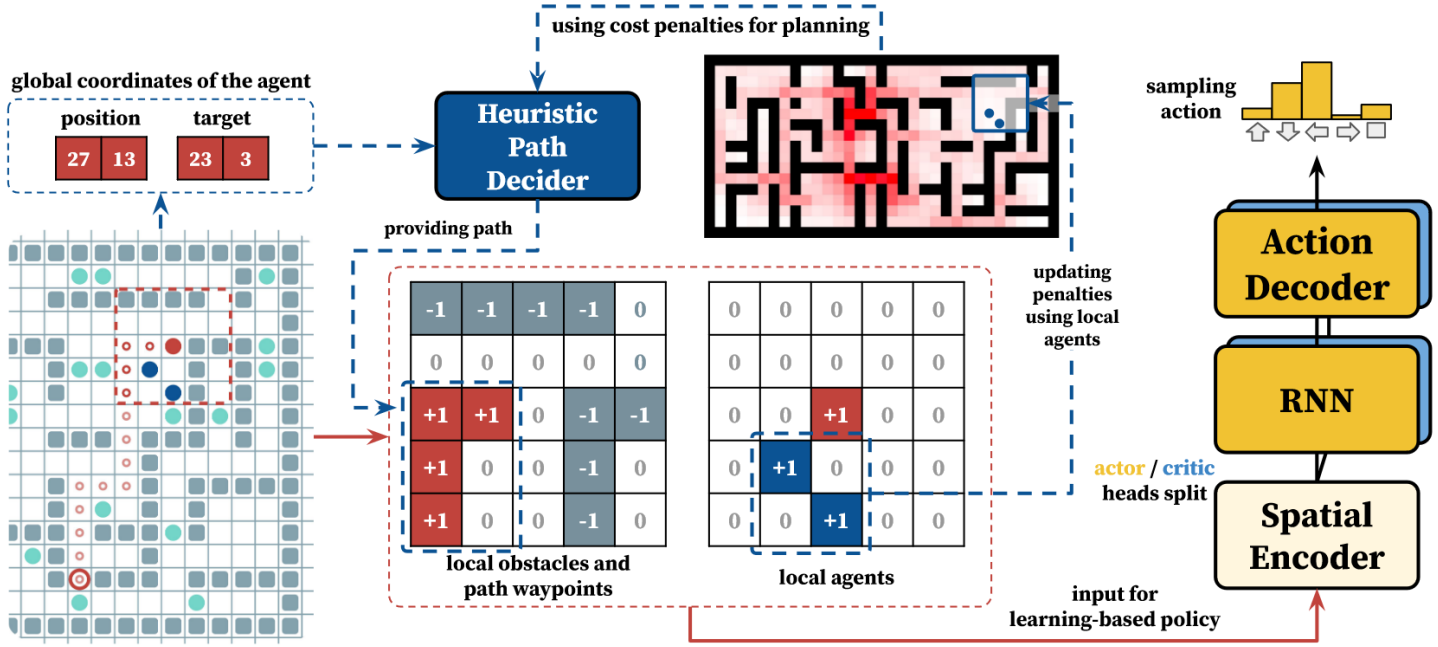
Learn to Follow: Decentralized Lifelong Multi-Agent Pathfinding via Planning and Learning

Planning

- A*
- Other agents are not taken into account
- Penalizing frequently used cells

(Reinforcement) Learning

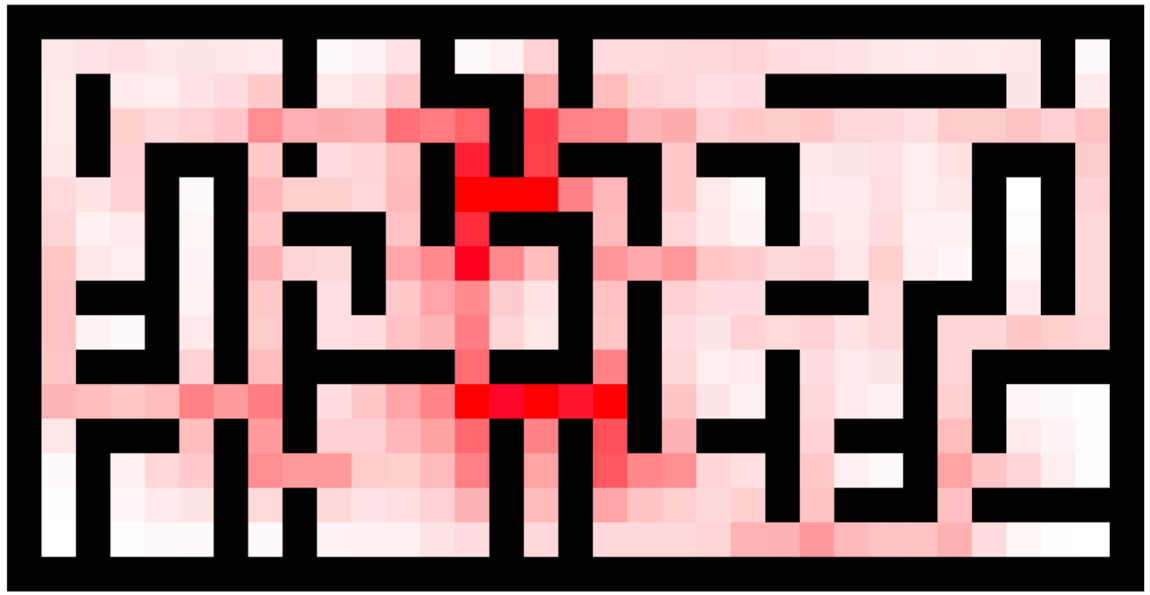
- PPO
- Local observation (path segment, other agents, obstacles)
- Decentralized (fast) learning
- Simplistic reward (achieve a waypoint = reward)



Learn to Follow: Decentralized Lifelong Multi-Agent Pathfinding via Planning and Learning

Planning

- A*
- Other agents are not taken into account
- **Penalizing frequently used cells**

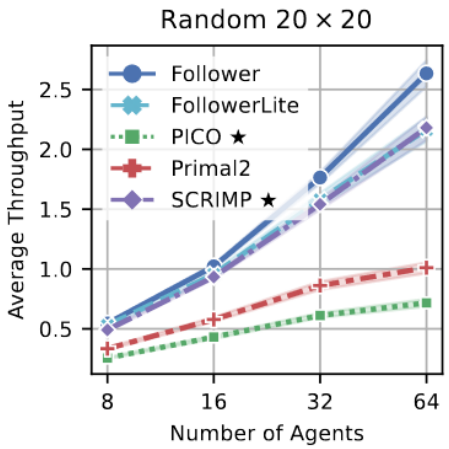


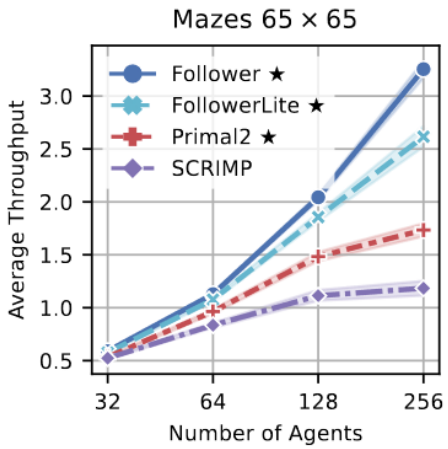
$$cost(c,t) = \underbrace{\frac{\max_{c' \in V}(avg\_cost(c'))}{avg\_cost(c)}}_{\textbf{static cost}} + \underbrace{\sum_{t' \in [0,t]} AgentAtCell(c,t')}_{\textbf{dynamic cost}}$$

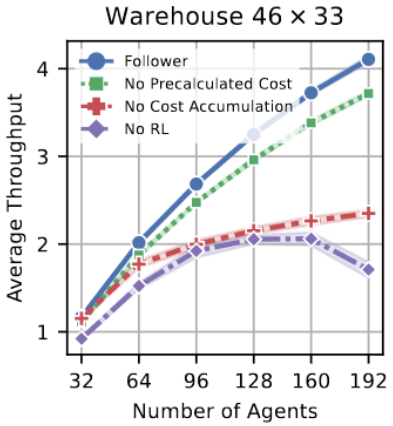
$$avg\_cost(c) = \sum_{c' \in V_{free}(c)} \frac{path\_cost(c,c')}{|V_{free}(c)|},$$

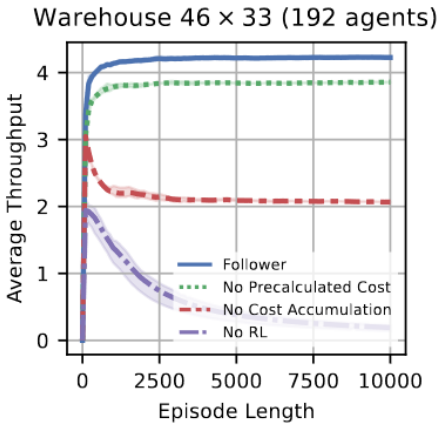
$V_{free}(c)$ – denotes the vertices reachable from c

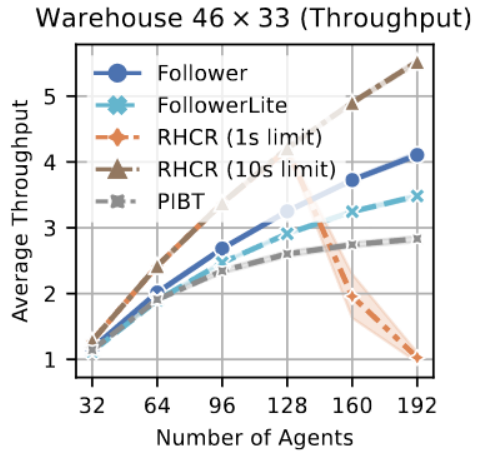
Learn to Follow: Decentralized Lifelong Multi-Agent Pathfinding via Planning and Learning

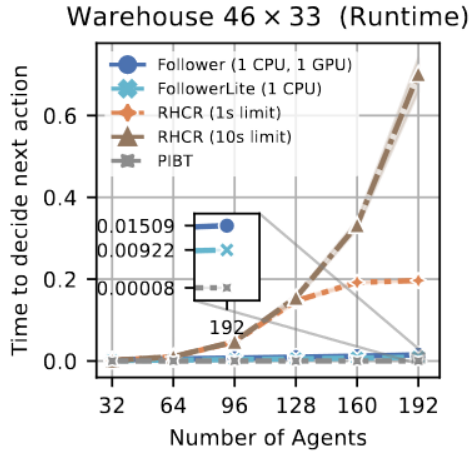










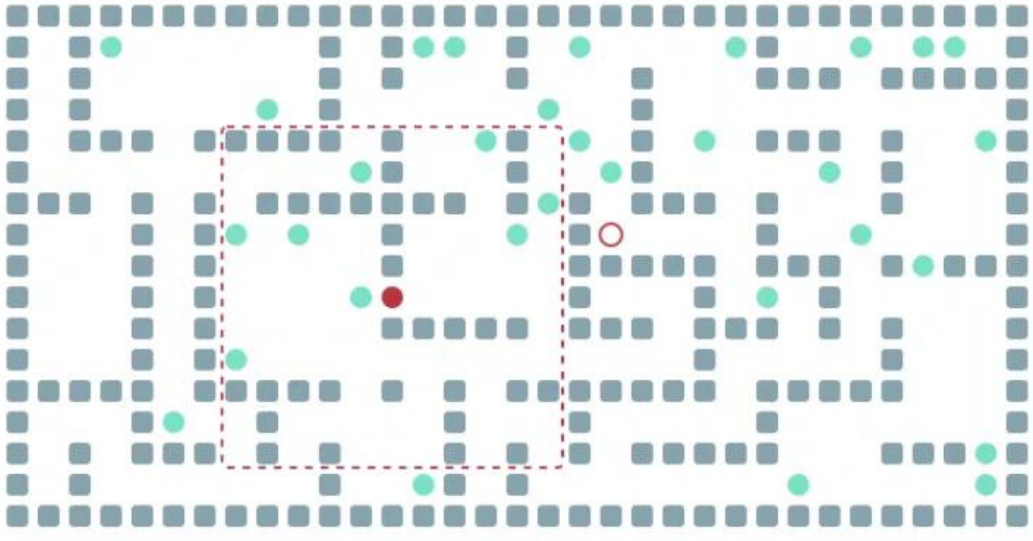


Empirical Results

- Outperform (learnable) competitors in terms of solution quality and ability to generalize
- Better scalability compared to centralized planners
- **Both components — planning and learning — are crucial**

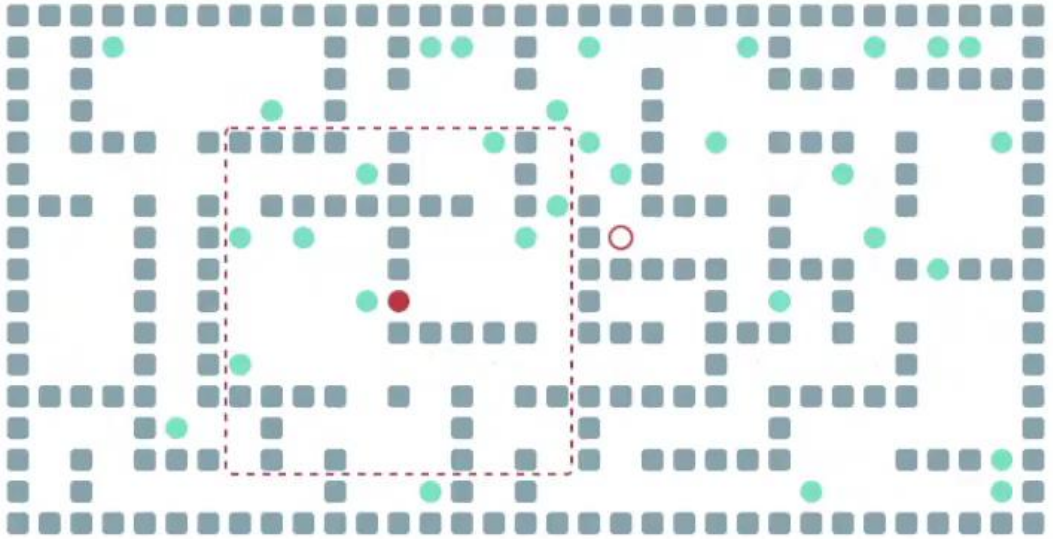
Learn to Follow: Decentralized Lifelong Multi-Agent Pathfinding via Planning and Learning

Planning only



goals achieved by the red agent = 0
goals achieved by all agents = 36
Throughput (512 steps, 32 agents) = 0.07

Planning + Learning



goals achieved by the red agent = 10
goals achieved by all agents = 330
Throughput (512 steps, 32 agents) = 0.64

SWITCHER

When to Switch: Planning and Learning for Partially Observable Multi-Agent Pathfinding

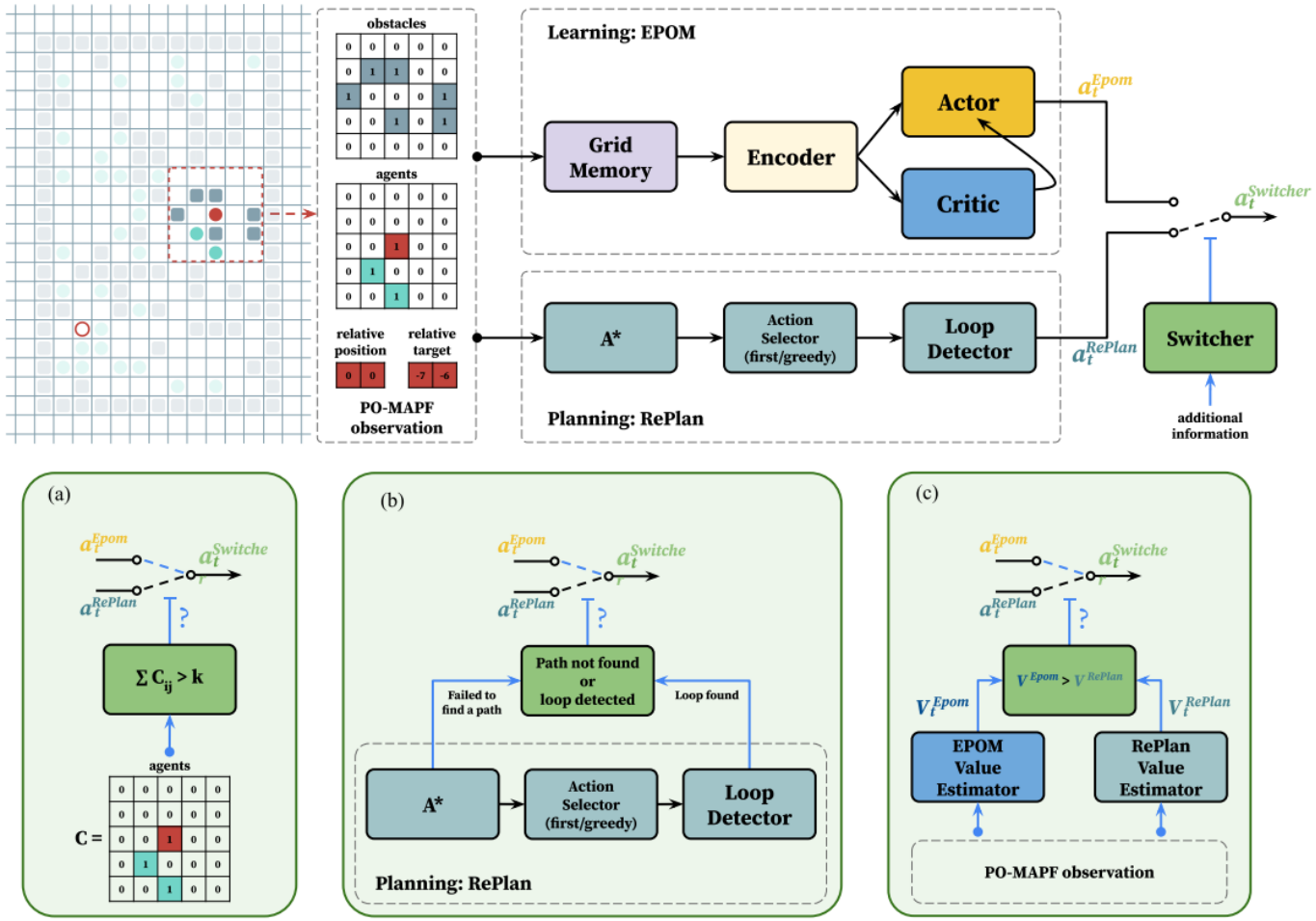
Alexey Skrynnik, Anton Andreychuk, Konstantin Yakovlev, and Aleksandr I. Panov, *Member, IEEE*

***Abstract*— Multi-agent pathfinding (MAPF) is a problem that involves finding a set of non-conflicting paths for a set of agents confined to a graph. In this work, we study a MAPF setting, where the environment is only partially observable for each agent, i.e., an agent observes the obstacles and other agents only within a limited field-of-view. Moreover, we assume that the agents do not communicate and do not share knowledge on their goals, intended actions, etc. The task is to construct a policy that maps the agent's observations to actions. Our contribution is multifold. First, we propose two novel policies for solving**

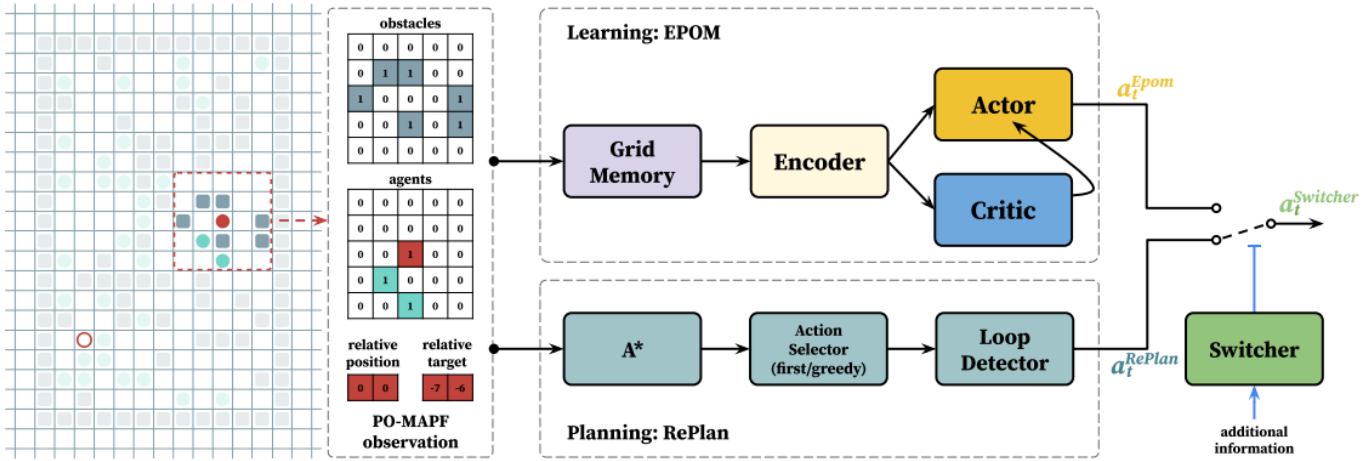
γ	Discount factor, value between 0 and 1, determines the importance of future rewards.
π	Decision-making policy, maps states to actions.
$\mathcal{G}$	Expected discounted return, expected sum of discounted rewards over time.
θ	Set of parameters of a neural network, defines the network's behavior.
h_t	Hidden state of the neural network, calculated

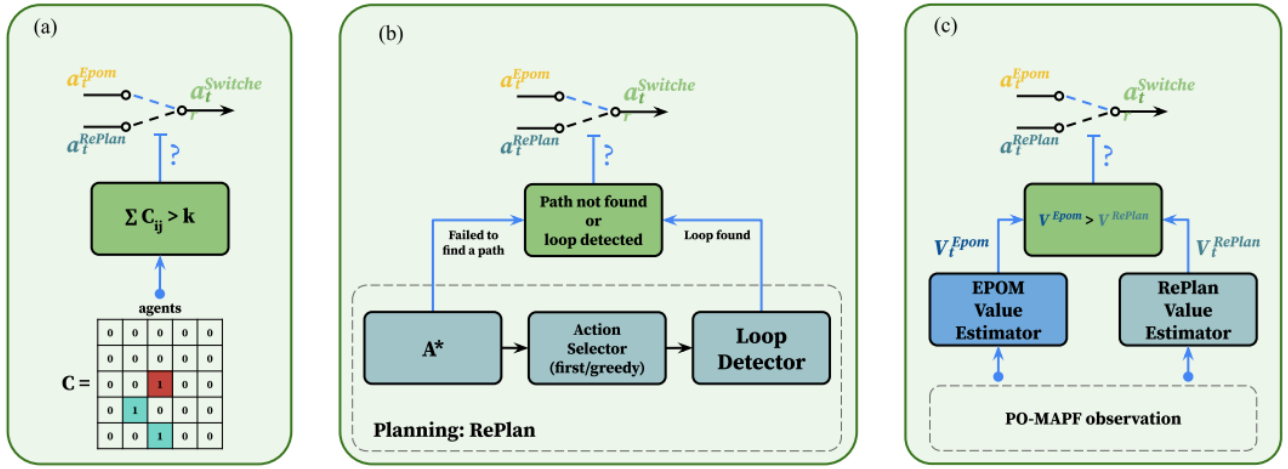
https://ieeexplore.ieee.org/document/10236574

When to Switch: Planning and Learning for Partially Observable Multi-Agent Pathfinding



When to Switch: Planning and Learning for Partially Observable Multi-Agent Pathfinding

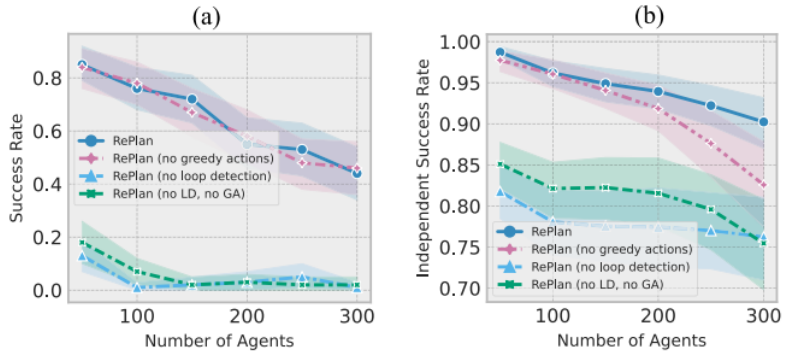




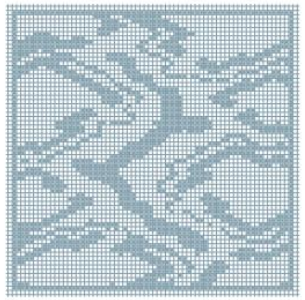
Algorithm 1 High-Level Pseudocode for the Search-Based Policy Incorporating LD and GAs: RePlan

Input: o — observation; map — map; pos — current position of the agent; $goal$ — position of the goal; $hist$ — sequence of the already performed actions. At the beginning of the episode, $plan = map = \emptyset$.
Output: a — action to perform at the current timestep; updated $hist$ and map.

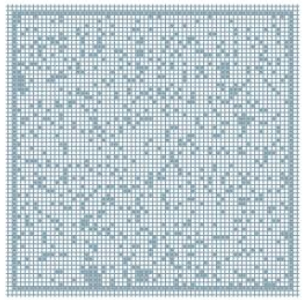
map:= MapUpdate(map, o);
$plan$:= A*(map, pos, $goal$);
if $plan \neq \emptyset$ **then**
  a:= GetFirstAction($plan$);
else
  a:= GetGreedyAction(pos, map);
if *DetectLoop*(a, $plan$) **then**
  With p_{wait} probability **return** wait;
$hist$:= $hist$ + a;
return a, map, $hist$



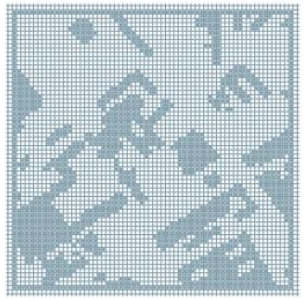
When to Switch: Planning and Learning for Partially Observable Multi-Agent Pathfinding



(a)



(e)



(i)

SUCCESS RATES OF THE EVALUATED PO-MAPF SOLVERS WITH RESPECT TO DIFFERENT MAP TYPES. IN ADDITION TO THE SUCCESS RATES, WE INCLUDE THE STANDARD DEVIATION COMPUTED FOR EACH MAP AND NUMBER OF AGENTS, WHICH IS THEN AVERAGED ACROSS ALL INSTANCES

agent	mazes	random	sc1	street	wc3	average success rate
RePlan	92.41% ±16.07	66.72% ±20.84	53.29% ±18.06	87.1% ±19.88	55.51% ±28.71	69.72% ±19.66
EPOM	80.38% ±31.24	52.81% ±22.66	17.26% ±15.25	52.94% ±40.03	35.31% ±17.28	45.95% ±23.98
Assistant Switcher	97.36% ±14.73	**77.84%** ±13.13	**67.1%** ±17.06	**95.1%** ±12.27	82.97% ±14.53	**81.71%** ±14.75
Heuristic Switcher	97.18% ±5.53	73.98% ±7.16	43.73% ±13.75	78.19% ±20.54	43.73% ±12.96	66.92% ±11.28
Learnable Switcher	**99.39%** ±3.11	75.44% ±5.87	55.16% ±14.83	94.17% ±15.61	**83.78%** ±8.71	77.88 % ±9.66

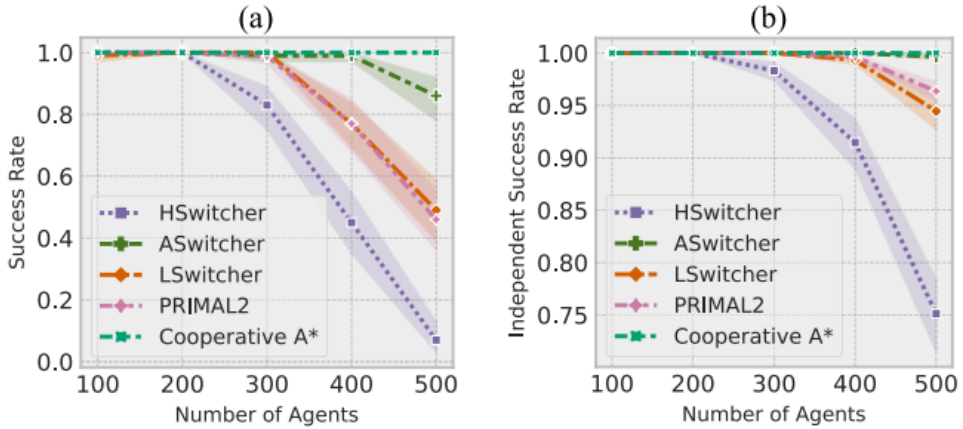


Fig. 8. Comparison of switchers with other approaches on the mazes maps. Cooperative A* has access to the full state of the environment, so it shows the best results, solving all the presented instances. The best algorithm among those working in the PO-MAPF setting is ASwitcher. The shaded area represents the 95% confidence intervals. (a) Success rate. (b) Independent success rate.

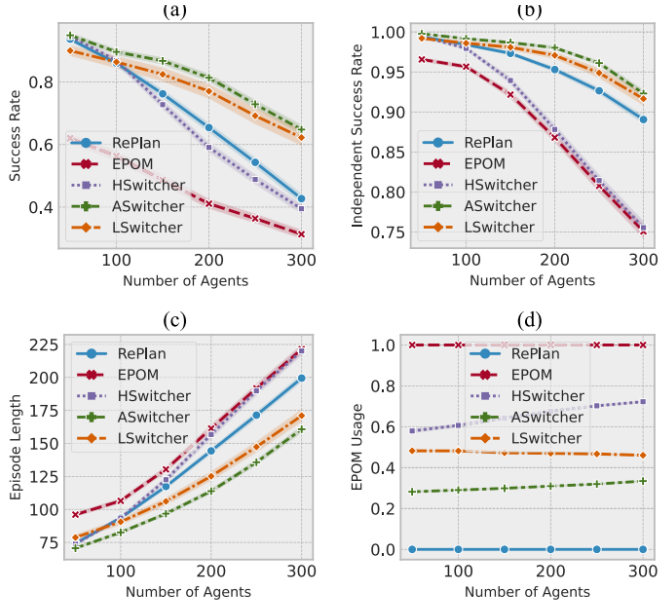
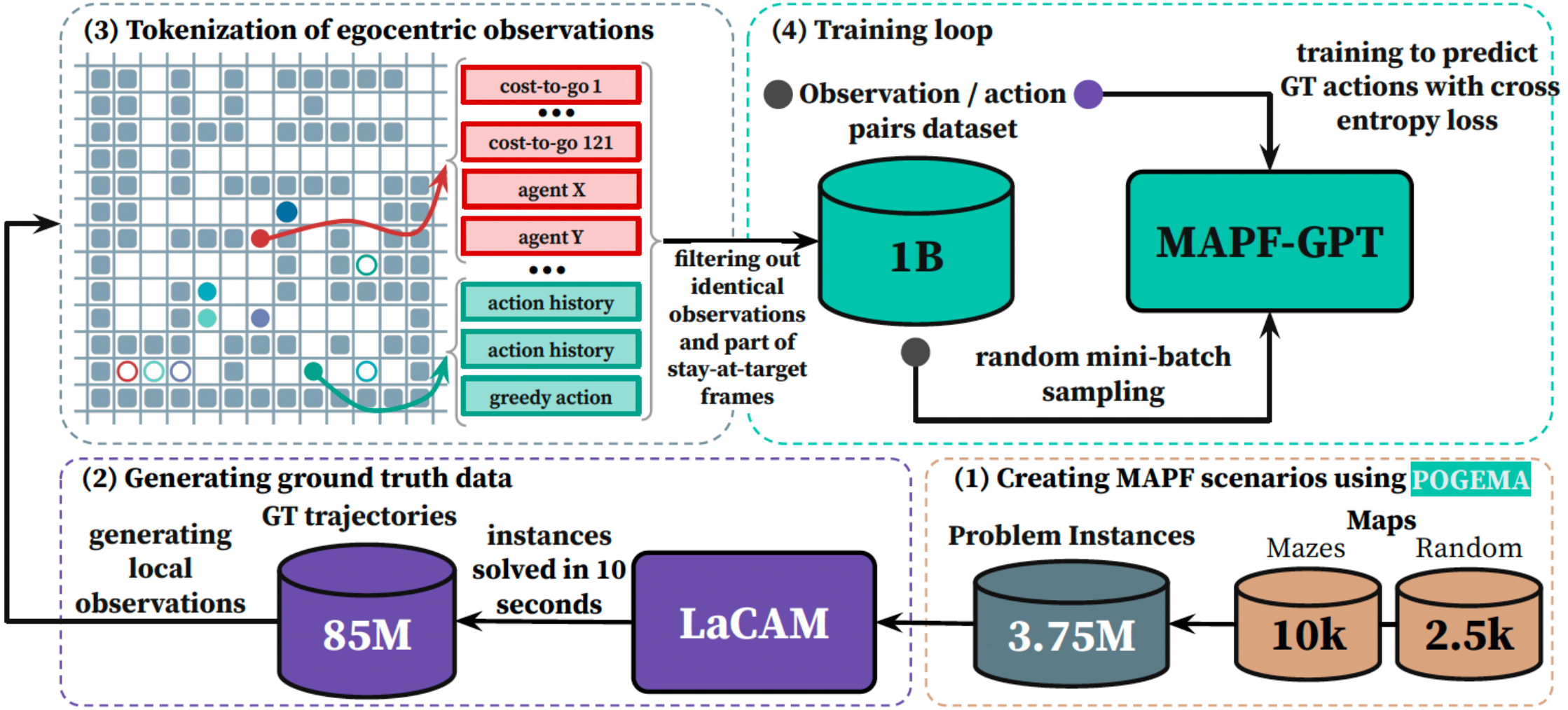


Fig. 6. (a) Success rate, (b) independent success rate, (c) episode length, and (d) EPOM usage per each number of agents averaged over all the evaluated instances. The shaded area reports confidence intervals 95%.

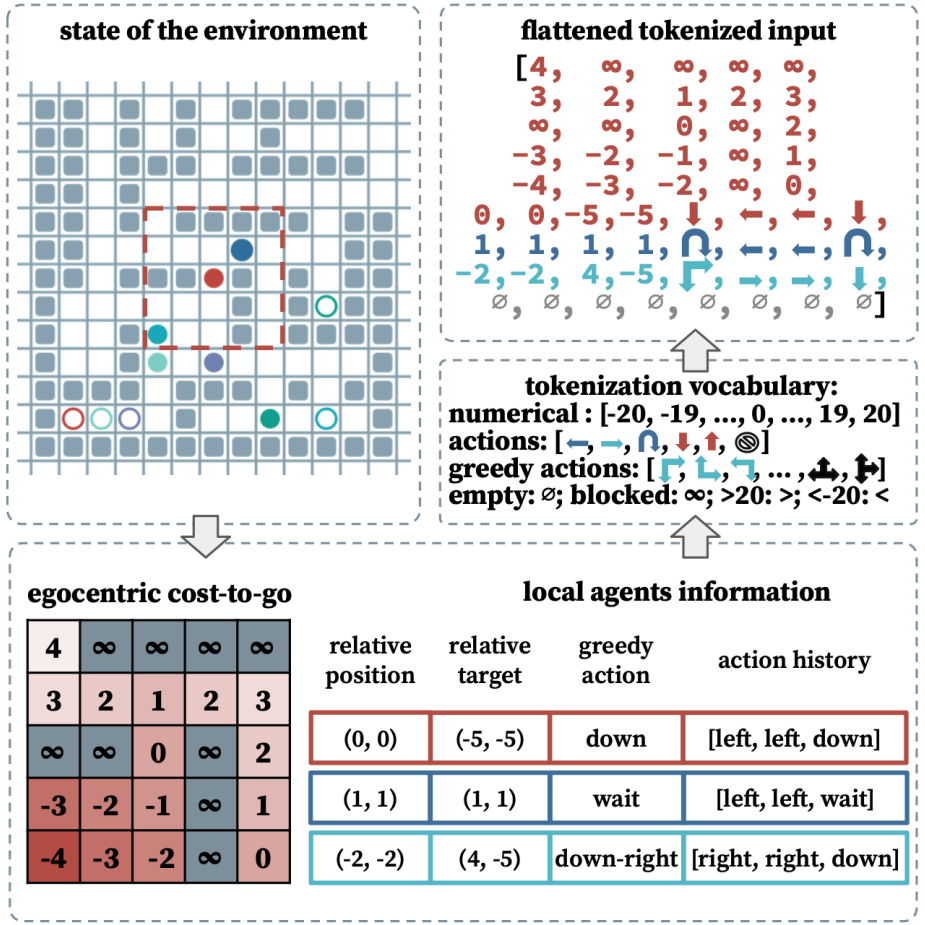
MAPF-GPT: Imitation Learning for Multi-Agent Pathfinding at Scale



MAPF-GPT: Imitation Learning for Multi-Agent Pathfinding at Scale

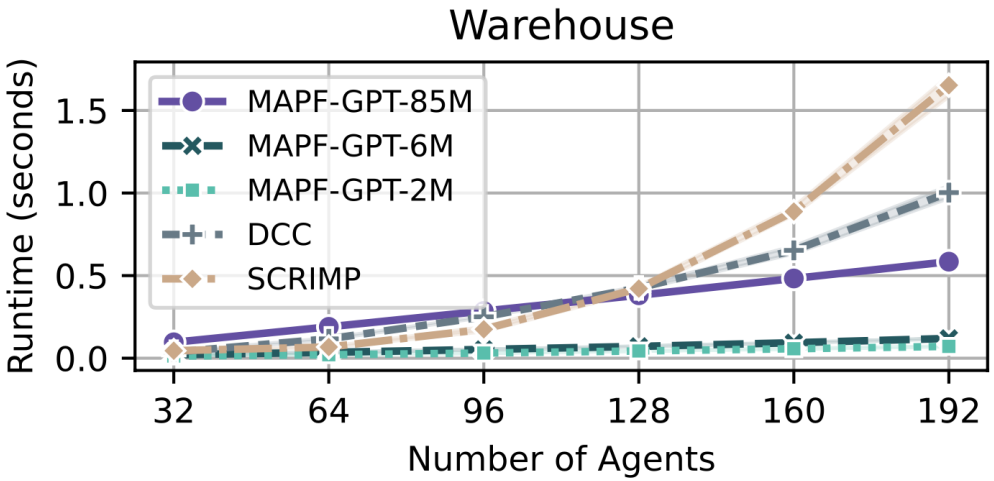
Tokenization

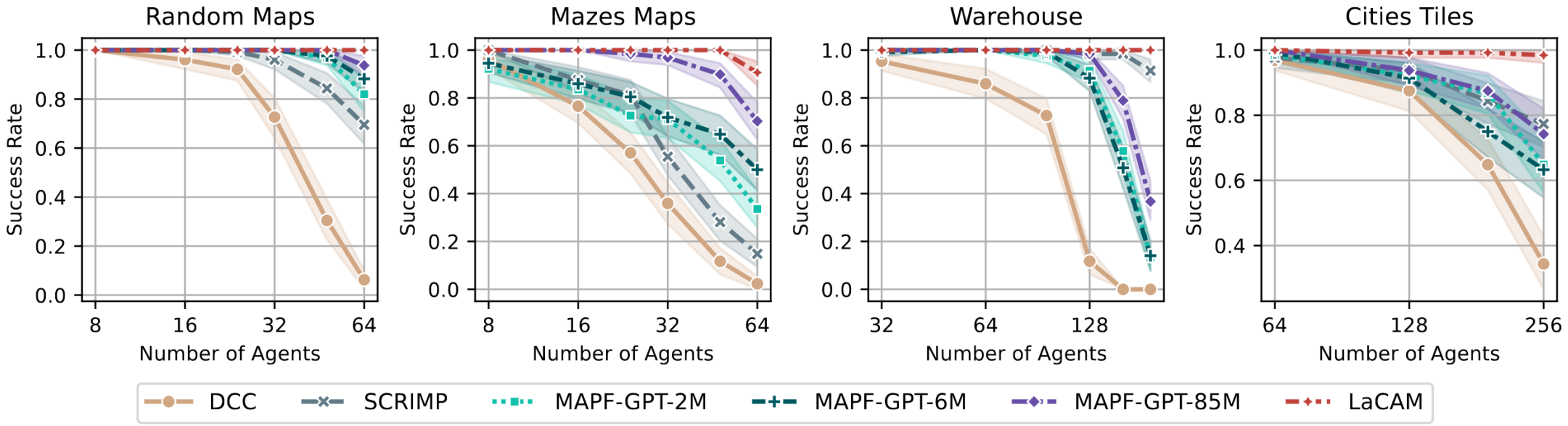
- The input consists of 256 tokens, but the example shown uses fewer tokens for clarity and visibility.
- The local observation of an agent at a certain time step while following its (expert) path is composed of two parts: egocentric cost-to-go in local 11×11 field of view and data about the agent itself and the nearby ones.
- The information about each agent is encoded via 10 tokens: 2 for the current position, 2 for goal location, 5 for action history, and 1 for the next greedy action.



MAPF-GPT: Imitation Learning for Multi-Agent Pathfinding at Scale

→ **SOTA performance with pure imitation learning**





MAPF-GPT Идеи для развития

Доучивать на сложных сценариях (уже реализовано, подано на IROS 2025, статья принята)

Эволюционное обучение aka AlphaEvolve

Обучение без учителя для проектирования окружений в устойчивом многоагентном поиске пути

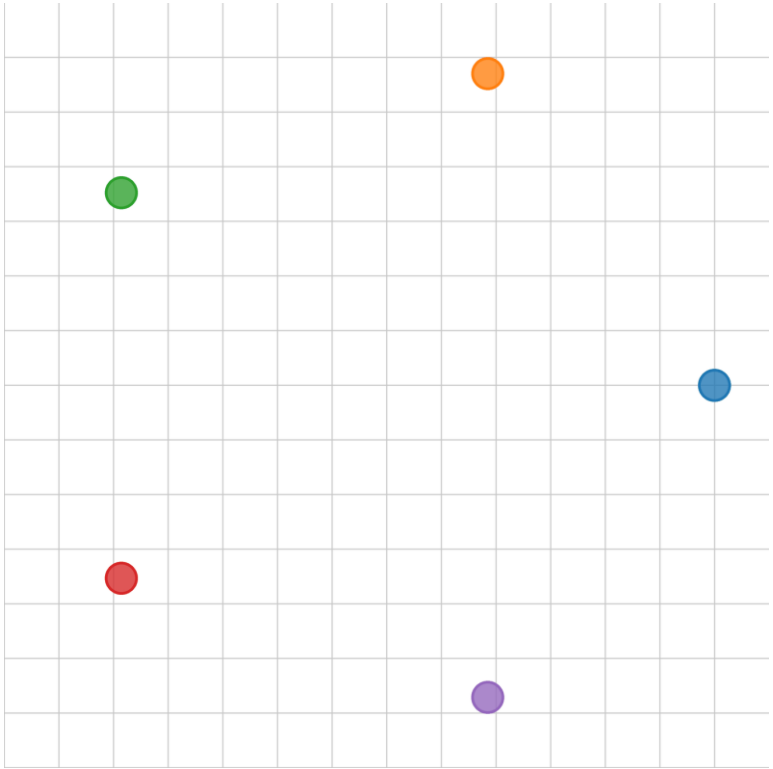
Этот проект предлагает замкнутую схему, которая объединяет генератор окружений в стиле alpha-evolve с обучением крупномасштабного трансформера для задачи поиска пути несколькими агентами (MAPF). Генератор создаёт разнообразные и всё более сложные карты, эксперты-решатели предоставляют демонстрации, а трансформер обучается на этих траекториях. Слабые стороны модели, выявленные в ходе обучения, направляют генератор на эволюцию новых сложных карт, что обеспечивает адаптивность и полноту датасета.

В этом итеративном цикле проектирование карт и обучение стратегии тесно интегрированы: генератор непрерывно создаёт окружения, выявляющие сбои, а трансформер совершенствуется, обучаясь на них. Этот процесс даёт модель, способную обобщать на новые карты, более высокие плотности агентов и сложные задачи координации, устраняя необходимость в создании вручную разработанных тестов.

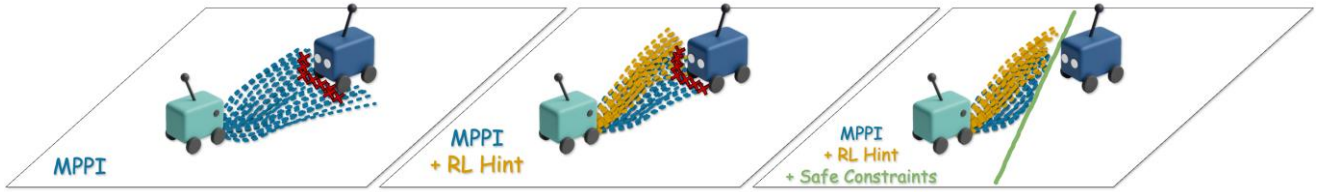
Задачи:
- Реализовать генератор карт в стиле alpha-evolve и связать его с созданием сценариев.
- Интегрировать существующий решатель MAPF для получения экспертных траекторий (например, LaCAM*).
- Построить конвейер для датасета и обучить крупномасштабный трансформер.
- Замкнуть цикл: эволюционировать карты на основе ошибок модели.
- Провести оценку на стандартных бенчмарках MAPF.

Децентрализованный MAPF в непрерывной среде

Есть авторское опубликованное решение на классических оптимизационных алгоритмах (стохастическая оптимизация, **MPPI**)



Сейчас добавляем в него RL



Получается неплохо (даже с наивной комбинацией)

Есть ещё много идей, как комбинировать => вы можете присоединиться к этому проекту

Немного оффтопик: Spatial Intelligence

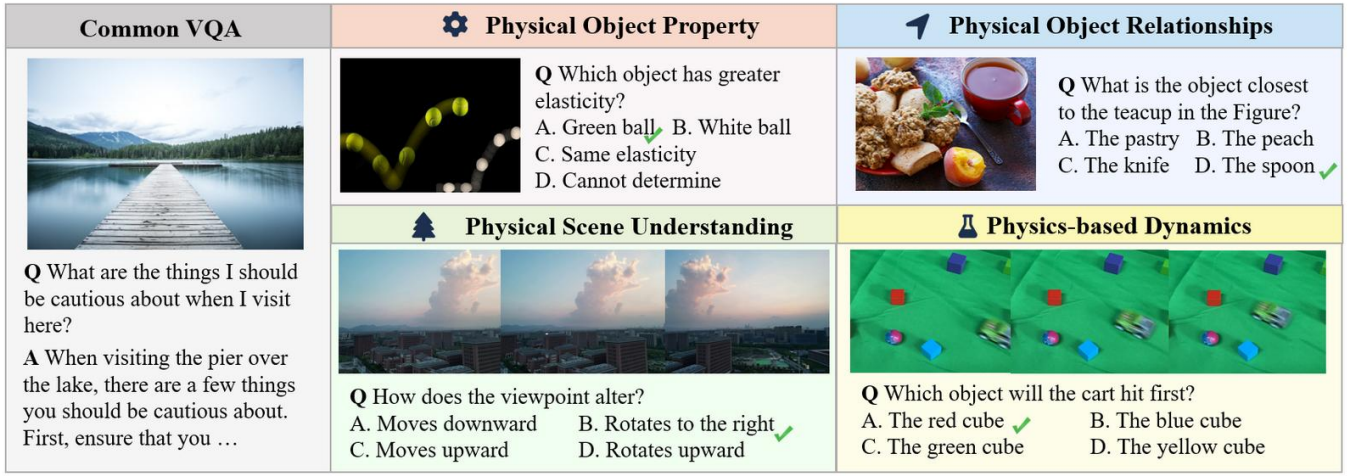
https://physbench.github.io/

PHYSBENCH

Benchmarking and Enhancing VLMs for Physical World Understanding

Wei Chow[1*], Jiageng Mao[1*], Boyi Li[2], Daniel Seita[1], Vitor Guizilini[3], Yue Wang[1]
[1]University of Southern California, [2]UC Berkeley,[3]Toyota Research Institute
*Equal Contribution

ICLR 2025



Common VQA tasks typically involve questions about visual content and general knowledge.
PHYSBENCH emphasizes understanding the physical world, encompassing 4 dimensions.

#	Model	Method	Source	Date	ALL
-	Human Performance	-	Link	2024-08-15	95.87
25	MolmoE-72B	merge	Link	2024-12-15	48.67
1	InternVL-Chat1.5	merge	Link	2024-08-15	47.51
24	MolmoE-7B-D	merge	Link	2024-12-15	45.18
28	MiniCPM2.6	merge	Link	2024-12-15	44.89
30	Aquila-VL	merge	Link	2024-12-15	43.22
29	Xinyuan-VL	merge	Link	2024-12-15	43.09
2	LLaVA1.6-vicuna	merge	Link	2024-08-15	42.28
22	MolmoE-1B	merge	Link	2024-12-15	40.63
3	LLaVA-1.5-13B	merge	Link	2024-08-15	40.45
23	MolmoE-7B	merge	Link	2024-12-15	40.41
4	LLaVA-1.5-7B	merge	Link	2024-08-15	40.09

Можно попробовать «ворваться в лидерборд» с небольшими (1B-4B) моделями

Немного оффтопик: архитектуры с памятью

Recurrent Memory Transformer

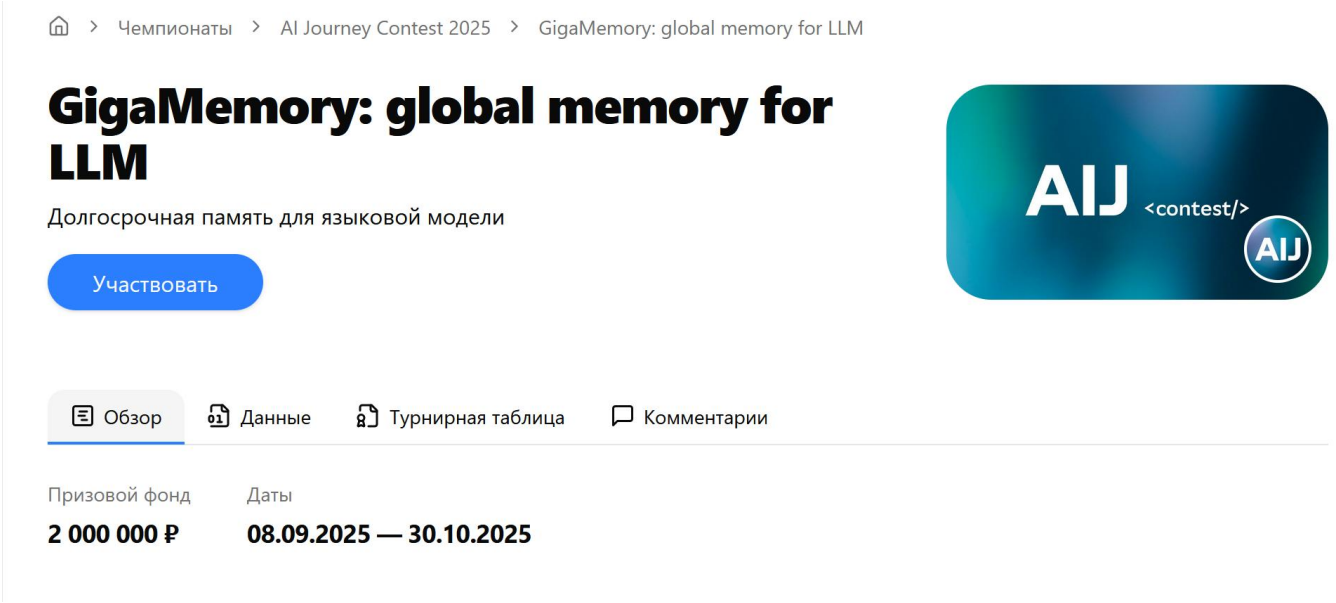
Aydar Bulatov[1]
bulatov.as@phystech.edu

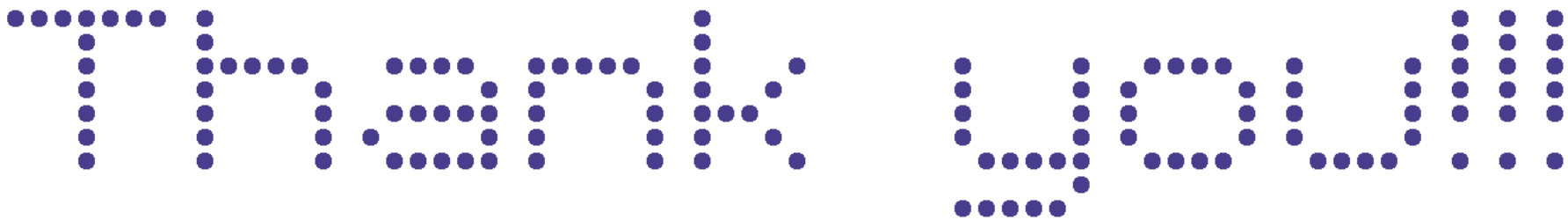
Yuri Kuratov[1,2]
yurii.kuratov@phystech.edu

Mikhail S. Burtsev[1,2]
burtcev.ms@mipt.ru

[1]Neural Networks and Deep Learning Lab,
Moscow Institute of Physics and Technology, Dolgoprudny, Russia
[2]AIRI, Moscow, Russia

https://dsworks.ru/en/champ/aij25-memory




Thank you!!