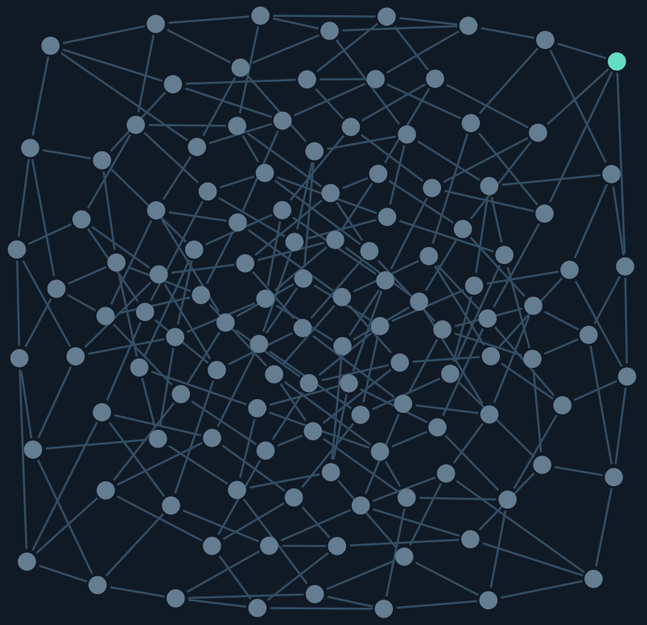
Методы эвристического поиска и машинного обучения для решения головоломок



Классический поиск, CayleyPy Cube
и исследования устойчивости Beam Search

Даниил Дроздов · СПбГУ

14 октября 2026 года

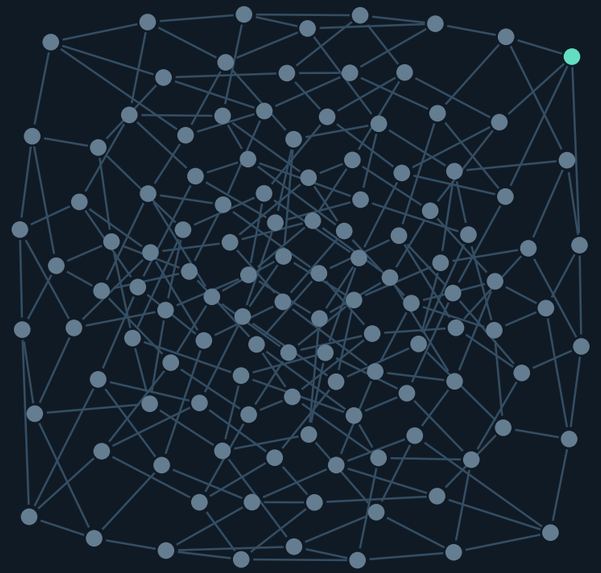
01

Задача и приложения

Состояния, генераторы и короткие пути

Как искать в графе, которого мы не видим?

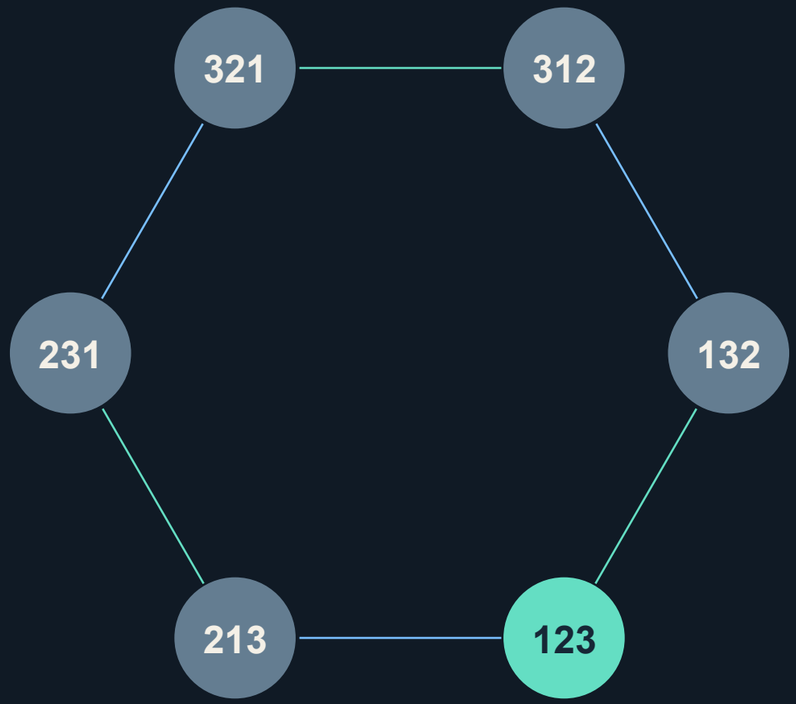
Найти путь



Состояние и операции известны
Соседей можно вычислить

Граф Кэли: состояния и генераторы

$$G = \langle S \rangle \quad x \to xa,\ a \in S$$



Вершины — элементы группы
Рёбра — действие генераторов

Вершины: элементы G · рёбра: действия из S · путь: слово

Перестройки генома: путь между порядками блоков

Две знаковые перестановки

Мышь

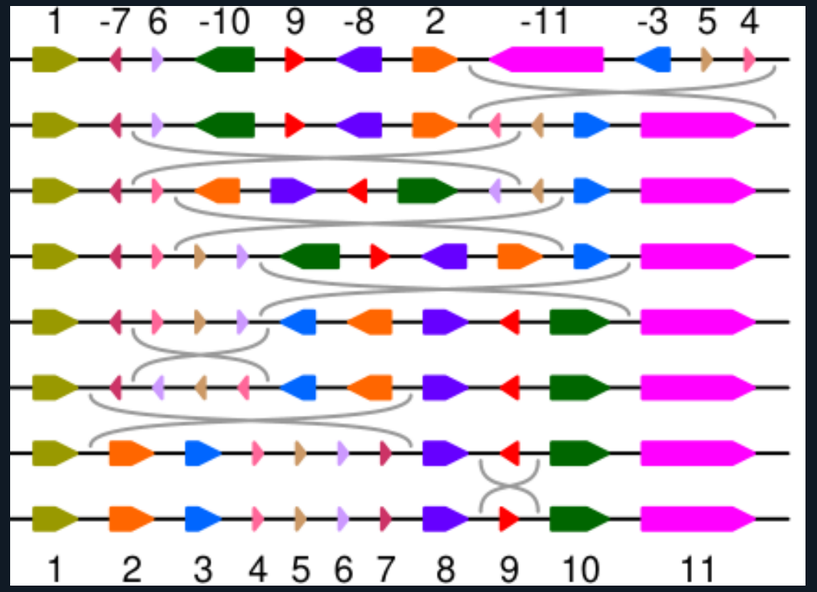


Человек



Номер — общий блок
Знак + или − — его ориентация

Семь последовательных инверсий



Кратчайший сценарий: 7 инверсий

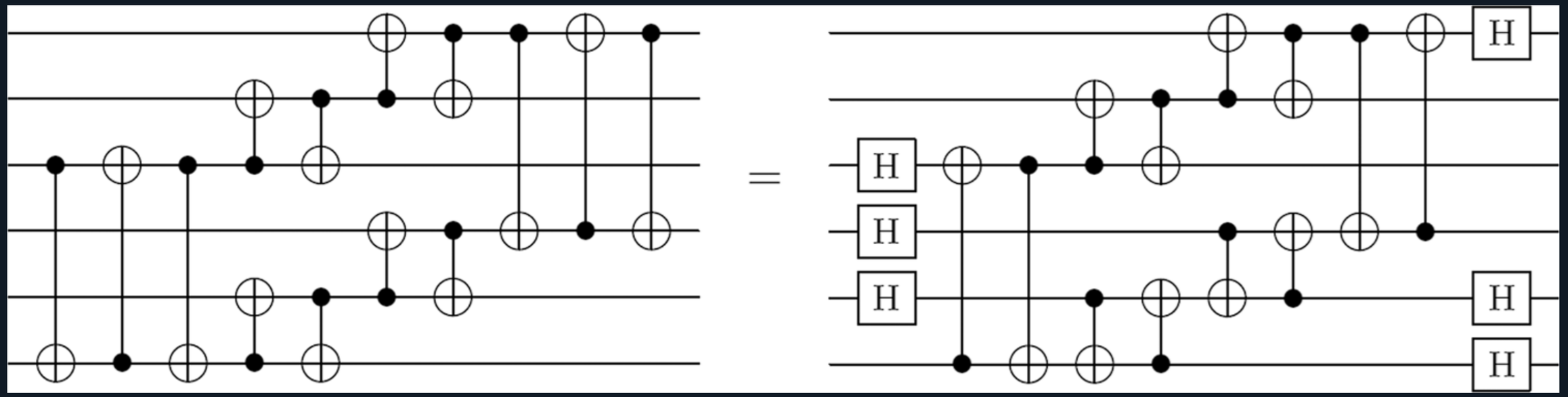
Pevzner и Tesler, 2003 · X-хромосомы · рисунок 2h

Два генома → две знаковые перестановки → 7 инверсий → кратчайший путь

Синтез квантовых схем

Один оператор U — разные схемы. Ищем схему с меньшей стоимостью.

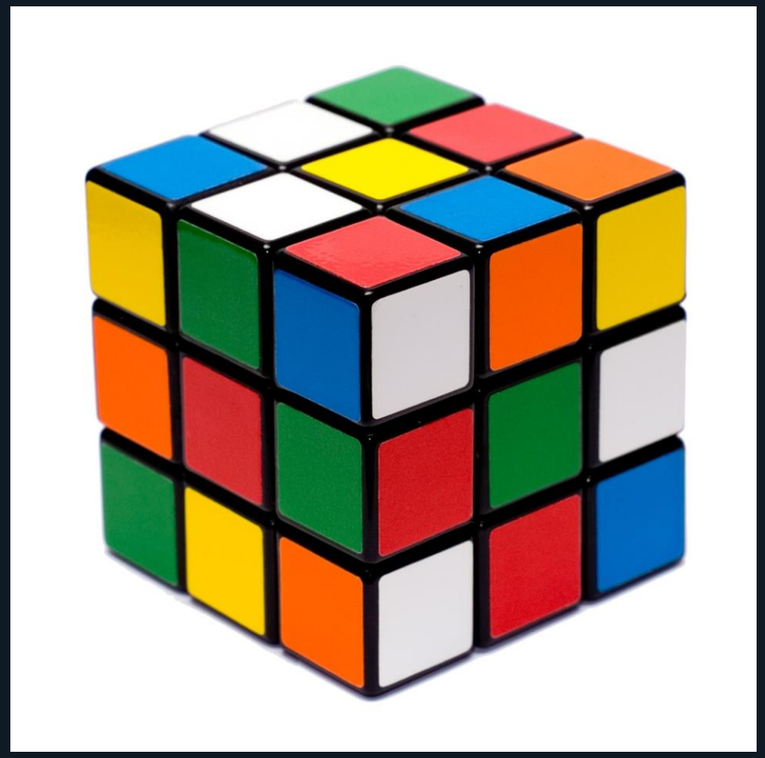
14 CNOT **12 CNOT + однокубитные H**



Оператор → вершина · вентиль → ребро · синтез → поиск короткого пути

Bravyi, Latone, Maslov, 2022 · рисунок 2 · CC BY 4.0

Кубик — наглядная задача для сравнения методов



Наглядное состояние
Обратимые ходы
Проверяемое решение
Огромное пространство

Конфигурация → вершина · поворот → ребро · сборка → путь к цели

Что именно считается одним ходом?

Метрика	Один ход	Для 3×3×3
HTM	90°, 180°, 270° внешней грани	18 действий
QTM	±90° внешней грани	12 действий
Срезы	±90° любого отдельного слоя	другая метрика

Одни состояния + разные генераторы = разные расстояния

43 252 003 274 489 856 000

$$4{,}3 \times 10^{19}$$

состояний стандартного кубика 3×3×3

Даже 1 байт на состояние ≈ 43 эксабайта

02

Классический поиск

Подгруппы, допустимые эвристики и число Бога

Thistlethwaite: сужать пространство

$$G_0 \supset G_1 \supset G_2 \supset G_3 \supset \{e\}$$

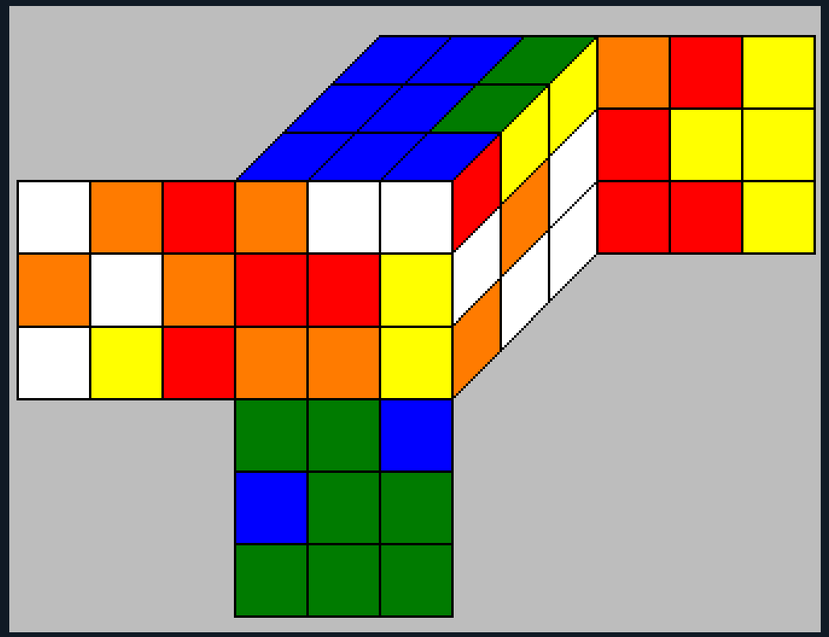
Все состояния → **Ограничение ориентаций** → **Ограничение перестановок** → **Цель**

В среднем < 2 с

Пример реализации dfinnis/Rubik на Go

1981: до 52 ходов HTM · время относится к конкретной современной реализации

Kociemba: две фазы вместо цепочки



Перейти в H → **Решить внутри H**

$$H = \langle U, D, R^2, L^2, F^2, B^2 \rangle$$

Несколько секунд

Python · Raspberry Pi 3

В среднем < 20 ходов HTM · без общей гарантии кратчайшего решения

Без эвристики дерево поиска растёт экспоненциально



Разные пути могут вести в одно состояние

Korf, 1997

Depth	Nodes
1	18
2	243
3	3,240
4	43,254
5	577,368
6	7,706,988
7	102,876,480
8	1,373,243,544
9	18,330,699,168
10	244,686,773,808
11	3,266,193,870,720
12	43,598,688,377,184
13	581,975,750,199,168
14	7,768,485,393,179,328
15	103,697,388,221,736,960
16	1,384,201,395,738,071,424
17	18,476,969,736,848,122,368
18	246,639,261,965,462,754,048

A*: полная оценка и условие оптимальности

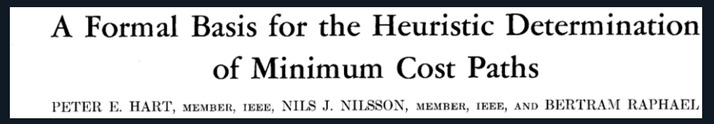
$$f(s) = g(s) + h(s)$$

g(s): стоимость от старта

h(s): оценка до цели

$$0 \leq h(s) \leq d(s,e)$$

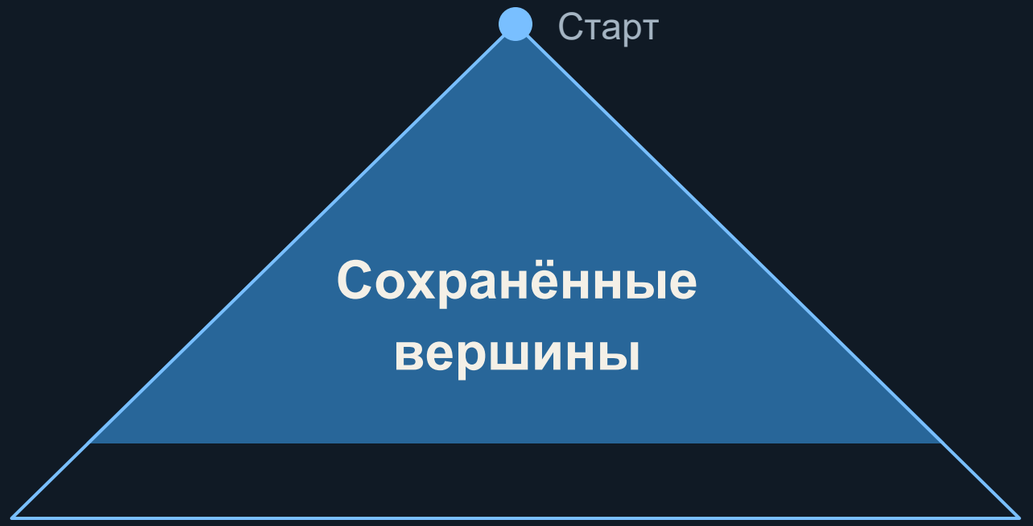
Допустимая эвристика + корректный учёт повторов → оптимальный путь


A Formal Basis for the Heuristic Determination of Minimum Cost Paths

PETER E. HART, MEMBER, IEEE, NILS J. NILSSON, MEMBER, IEEE, AND BERTRAM RAPHAEL

A* и IDA*: сколько нужно хранить

A*: много вершин в памяти



Альтернативы в OPEN
и обработанные состояния

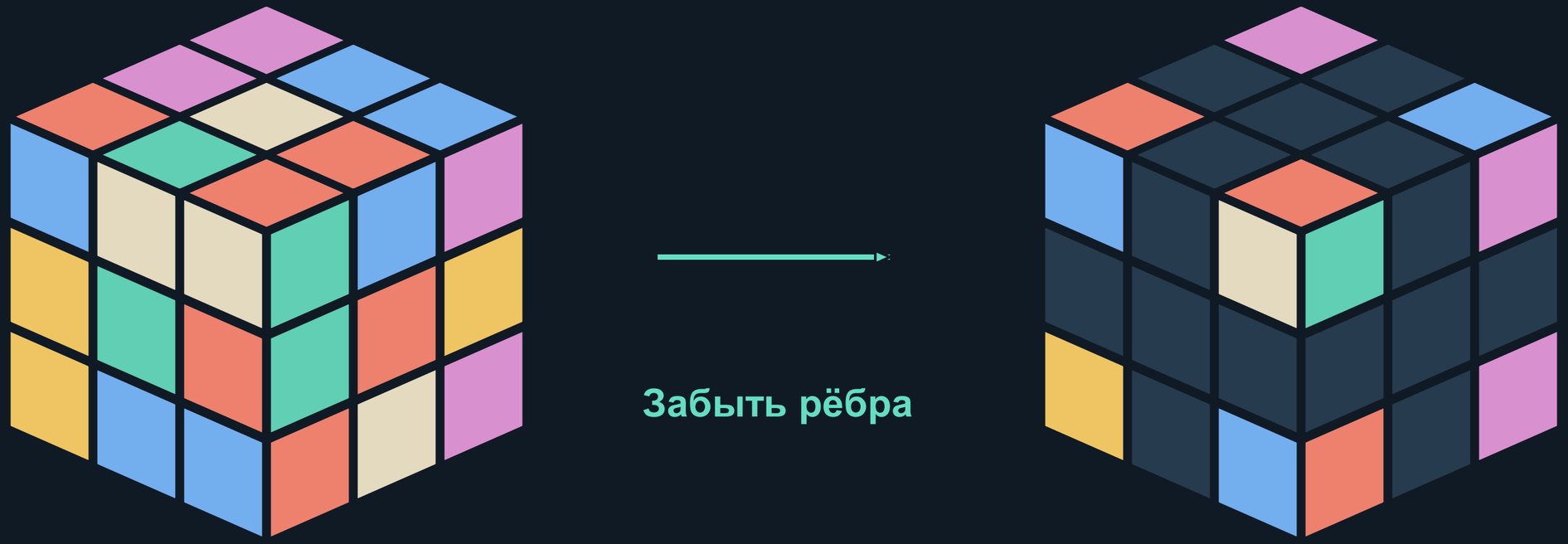
IDA*: текущий путь



Стек пути и служебные данные
Новый порог — новый обход

IDA* посещает много ветвей. Цвет показывает память, а не все посещённые вершины.

Pattern Database: точно решить абстракцию



PDB: таблица даёт доказанную нижнюю оценку

Полное состояние → **Абстракция φ(s)** → **Точное d_a из PDB**

$$h(s) = d_a(\varphi(s), \varphi(e)) \leq d(s, e)$$

Проекция любого полного пути не дороже исходного

Абстракция сохраняет допустимые переходы и не увеличивает их стоимость

Несколько PDB: максимум безопасен

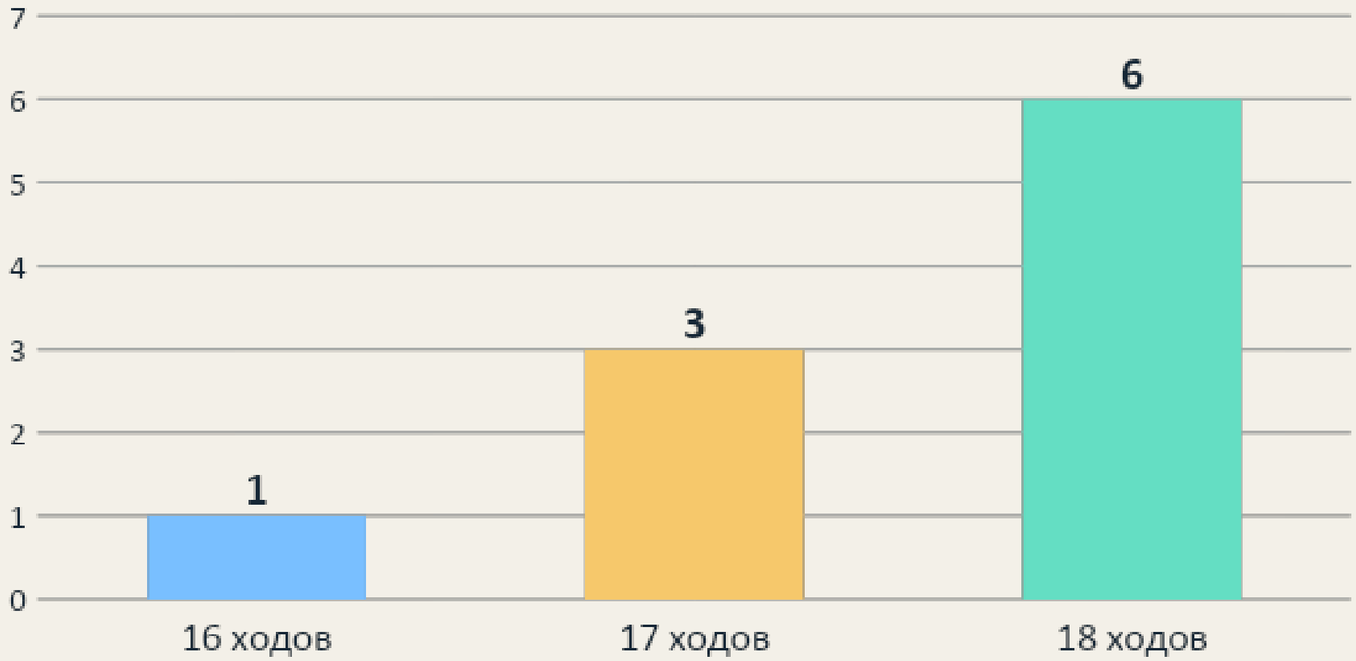
$$h(s) = \max\{ h_1(s), h_2(s), h_3(s) \}$$

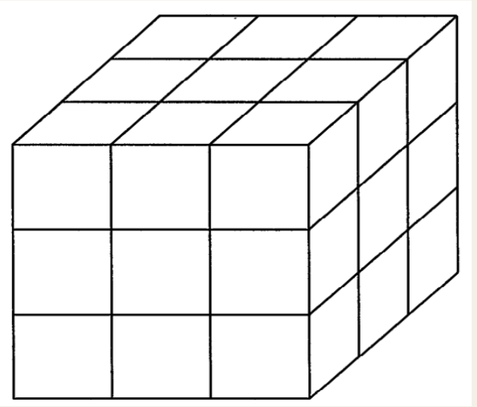
Один ход может менять сразу несколько шаблонов

Сумма требует отдельного разбиения стоимости ходов

Korf: оптимальные решения случайных кубиков

Число задач





Один проход IDA* с PDB

Проверка решений до 16 ходов: менее 4 часов

Проверка решений до 17 ходов: около 2 суток

Sun UltraSPARC Model 1 · 10 случайных задач · HTM · Korf, 1997

Таблица хранит знания, но занимает память

Сильная сторона

Точное расстояние подзадачи

Доказанная допустимость

Цена

Размер таблицы

Выбор абстракции человеком

Сильнее шаблон → больше памяти

Число Бога: от решения к диаметру графа

20

Диаметр графа в HTM

$$\mathrm{diam}(\mathrm{Cay}(G,S_HTM)) = 20$$

1981	1995	2008	2010
52	**29**	**22**	**20**

Стандартный 3×3×3 · HTM · доказанные верхние границы

Почему число Бога равно двадцати

Нижняя граница ≥ 20

Superflip собирается за 20 ходов.
Нужно исключить 19 и меньше.

Reid, 1995: модификация Kociemba и сокращение случаев симметриями.

Для superflip доказано, какие варианты обязательно надо проверить.
Программа проверила их полностью: решений короче 20 нет.

Таблицы расстояний до H существовали раньше.
PDB углов и рёбер Korf — другой выбор, 1997.

Верхняя граница ≤ 20

В один класс Hg входят состояния, у которых совпадают три признака:

Ориентации всех углов
Ориентации всех рёбер
Четыре позиции рёбер среднего слоя

Порядок деталей может различаться.
Одинаковые три признака — один класс.

2010: классы проверяют целиком с IDA*, PDB и Kociemba.
Каждое состояние решается за ≤ 20.

Зачем здесь машинное обучение



Общий подход к обучению и поиску · фото Gerwin Sturm · CC BY-SA 2.0

Размер графа

Перебор и точные PDB
требуют слишком много памяти и времени

От 3×3×3 — к новым задачам

Кубики большего размера
и другие головоломки

ML: учить оценку по переходам и направлять ею поиск

Неизвестные расстояния: пять источников сигнала

Метод	Сигнал обучения	Что учим	Как строим путь
DeepCubeA	Обновление Беллмана	Стоимость до цели	Batch weighted A*
EfficientCube	История перемешивания	Политику действий	Beam Search
CayleyPy Cube	Глубина блуждания	Геометрическую оценку	GPU Beam Search
CayleyPy RL	Награду + TD-оценки	Стоимость состояния	Beam Search
GFlowNet	Баланс + штраф потока	Политики P_F / P_B	P_B + Beam Search

Сигнал обучения → предсказание модели → построение пути

03

CayleyPy Cube

Случайные блуждания, ResMLP и GPU Beam Search

CayleyPy Cube: полный цикл решателя

Обучение

Блуждания → **ResMLP** → **Готовая модель**

От цели получаем пары (s, k)

MSE по глубине обновляет веса

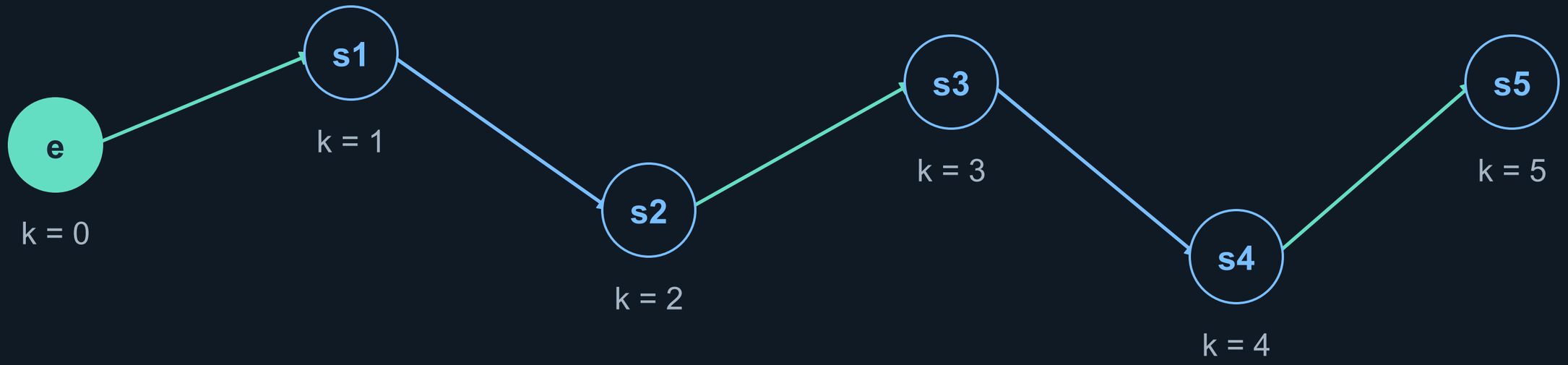
Оценка hθ(s) веса фиксированы

Решение

Старт → **GPU Beam Search** → **Путь к цели**

Модель выдаёт оценку · поиск строит и проверяет последовательность ходов

Dataset из блужданий от цели



Сохраняем (s_k, k)

Новые траектории на каждой эпохе
Non-backtracking: без немедленной отмены хода

Глубина блуждания и кратчайшее расстояние



$$k = 2, \quad d(s_2, e) = 0$$

Запрет немедленного возврата не исключает все циклы

Что предсказывает регрессия глубины

$$h_\theta(s) \to \mathbb{E}[K \mid S = s]$$

Условная средняя глубина появления состояния

Закон генерации данных определяет целевую функцию; допустимость не гарантирована

ResMLP: оценка состояния одним числом

Состояние	→ Linear	→ Residual blocks	→ Оценка
$6N^2$ цветов one-hot	Вектор признаков	Linear, ReLU BatchNorm	Один скаляр $h_\theta(s)$

MSE: предсказание сопоставляется с глубиной k

Inference используется многократно · для другого размера обучается отдельная сеть

GPU: соседи и inference целым batch

Луч B_i → **Генераторы** → **Batch inference** → **Top-W**

W состояний в тензоре

До W·|S| кандидатов

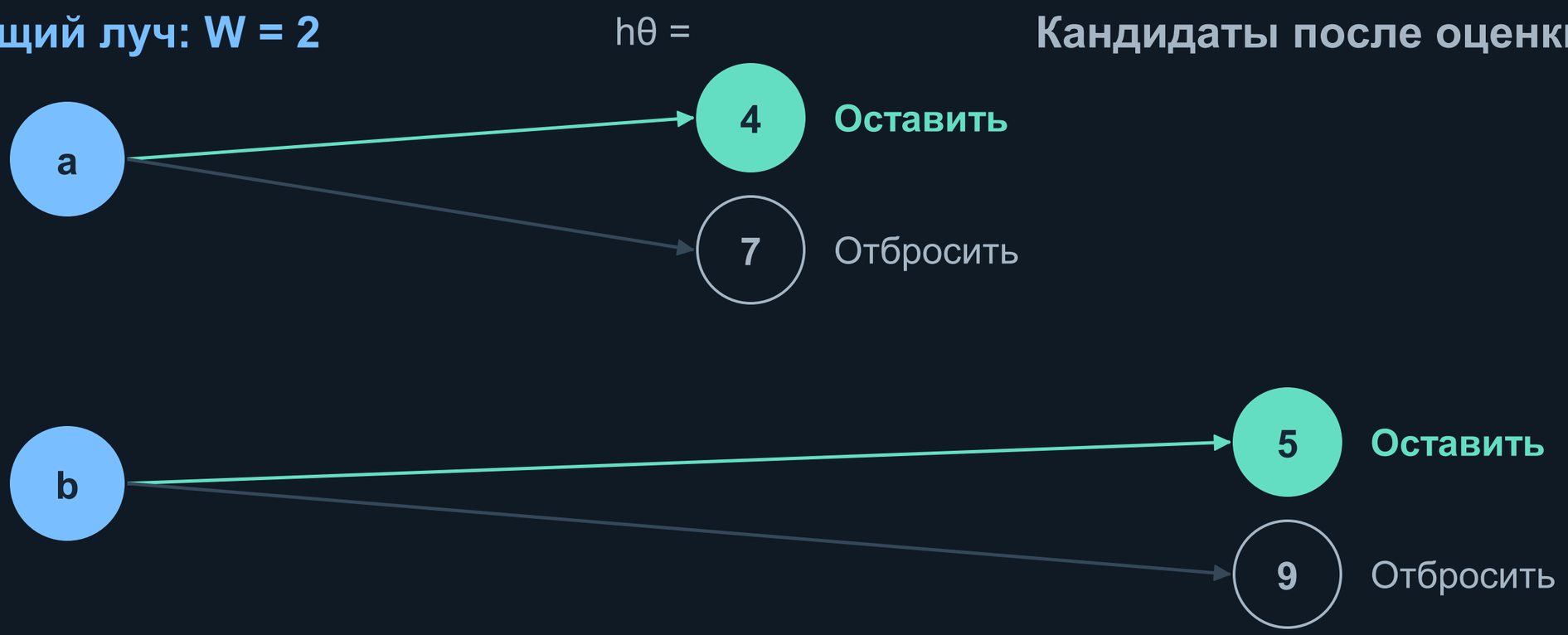
Удаление повторов оценки hθ

Следующий луч и parent pointers

При $W = 2^{24}$ даже один уровень — большое вычисление

Память: состояния, кандидаты, оценки, родители · не только веса модели

Beam Search: оставить W лучших продолжений



Остаются оценки 4 и 5 · потерянные альтернативы не входят в следующий луч

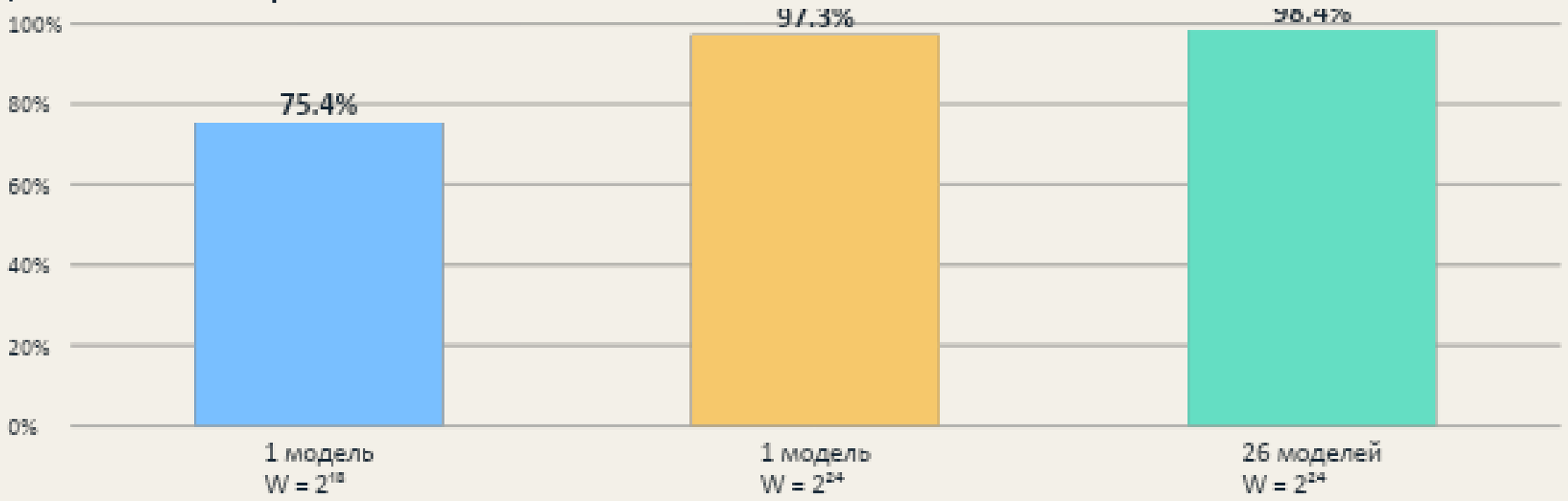
Ансамбль: независимые оценки и разные пути



Пример длин

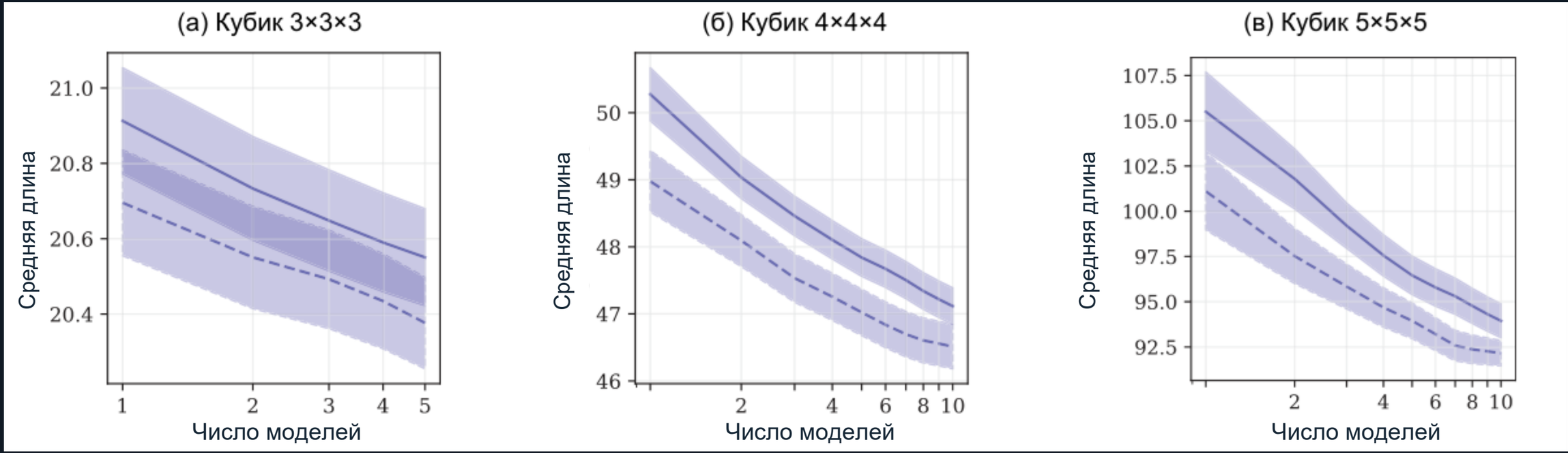
Выбираем кратчайший
из найденных путей

Качество зависит от ширины и числа агентов



1000 задач · классическая QTM · 26 моделей при $W = 2^{24}$

CayleyPy Cube на 4×4×4 и 5×5×5



4×4×4: 53,5 → 46,5

5×5×5: 96,6 → 92,2

Сплошная: случайный ансамбль · пунктир: лучший · тень: ±1 стандартная ошибка
4×4×4 / 5×5×5: UQTM · W = 2^{24} · обучено 29 / 69 моделей

Общий интерфейс для разных графов

Кубики

LRX и Pancake

Другие перестановочные головоломки

Описание среды

Вектор состояния
Генераторы-перестановки
Проверяемая цель

Dataset → ResMLP → Beam Search

Переносится алгоритмический цикл · веса и бюджет подбираются для каждой среды

04

Мои исследования

Dataset, модель и модификации Beam Search

Цель работы и три исследовательские задачи

Повысить success rate при меньшей ширине луча

Данные	Генератор train dataset и согласованность меток
Модель	Качество оценки и цена inference
Поиск	Local cycle prevention, retries и backtracking

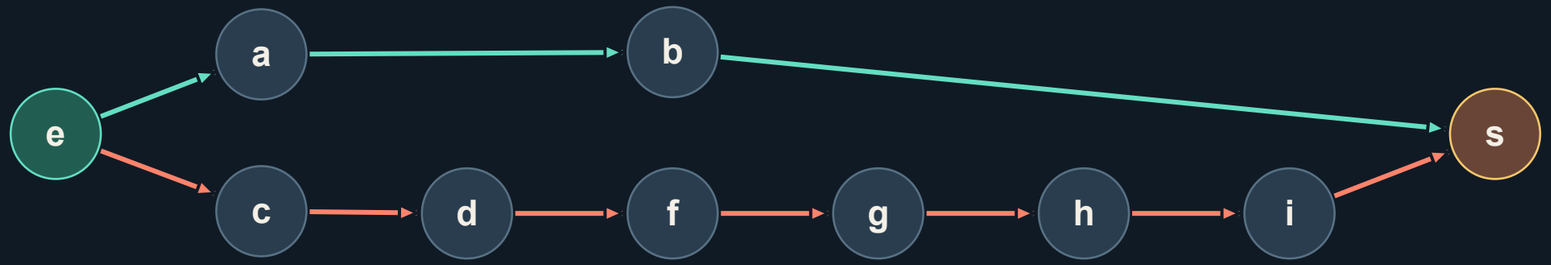
Baseline на стенде: новые random walks + одна MLP + обычный Beam Search

Три размера кубика, один стенд

Кубик	Задачи	Фацеты	Генераторы
3×3×3	82	54	18
4×4×4	43	96	24
5×5×5	19	150	30

Santa 2023 · A100 40 ГБ · EPYC 7742 · ходы отдельных срезов

Одна вершина может иметь разные метки



3

7

Две метки одного s

СБ, СБ-К и фиксированный dataset БД

СБ

Новые random walks на каждой эпохе

Один s может получить разные k

СБ-К

Запрет повторов в пределах эпохи

Контроль обновляется между эпохами

БД

Один фиксированный dataset

Глобальный контроль повторов

БД: около 128 млн пар · 64 эпохи · T = 26 / 45 / 65

Большой dataset: CPU, GPU и тупики траекторий

Память

GPU генерирует batch
CPU хранит большой dataset

Перенос готовых состояний
освобождает память GPU

Тупики генерации

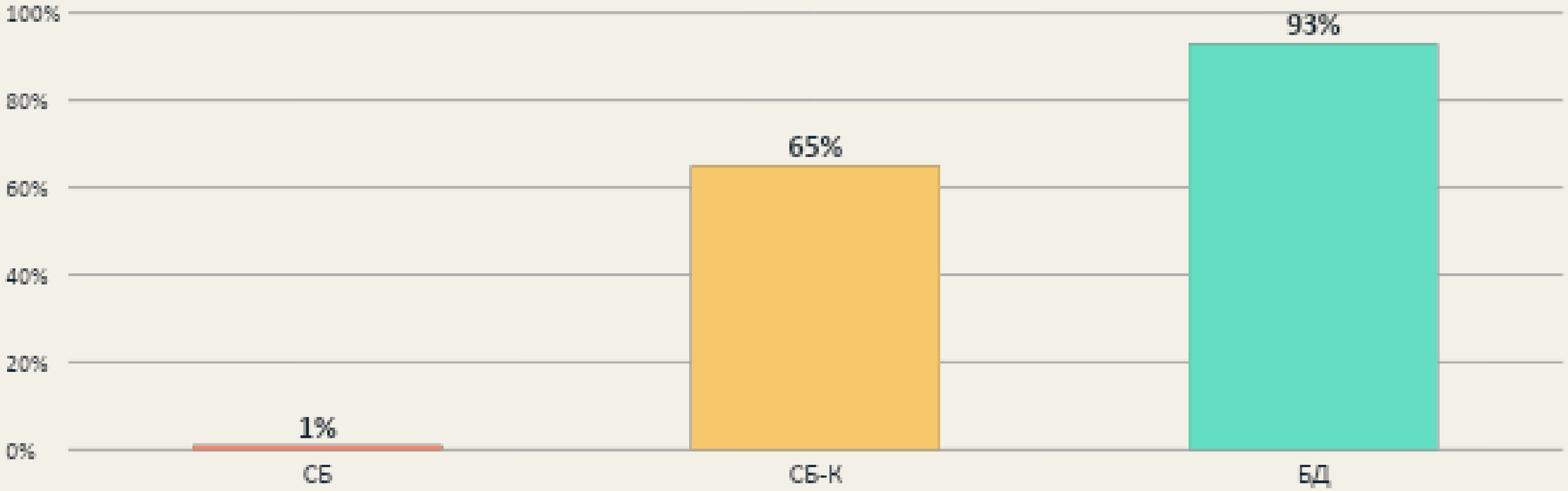
Все соседи уже посещены
Траектория останавливается

Продолжаем другие
доступные траектории

Одна вершина получает одну метку · это ещё не точное кратчайшее расстояние

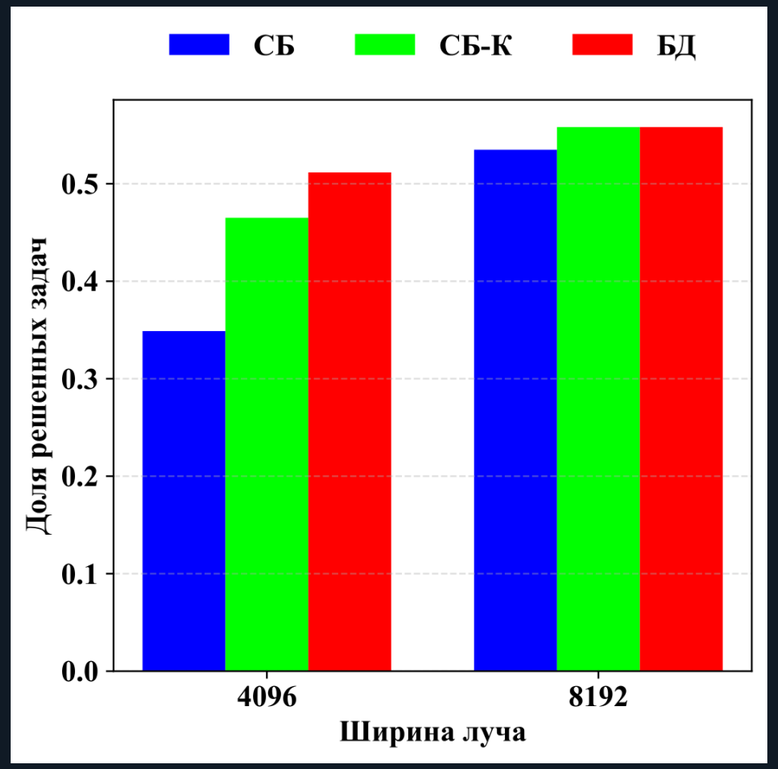
На узком луче разница данных особенно велика

Доля решённых задач



3×3×3 · 82 задачи · W = 128 · MLP

На 4×4×4 данные тоже меняют успех поиска



43 задачи · MLP

Три архитектуры видят разную структуру

MLP

Плоский вектор

≈ 4 млн параметров

GNN / RGCN

Состояние + соседи

≈ 3 млн параметров

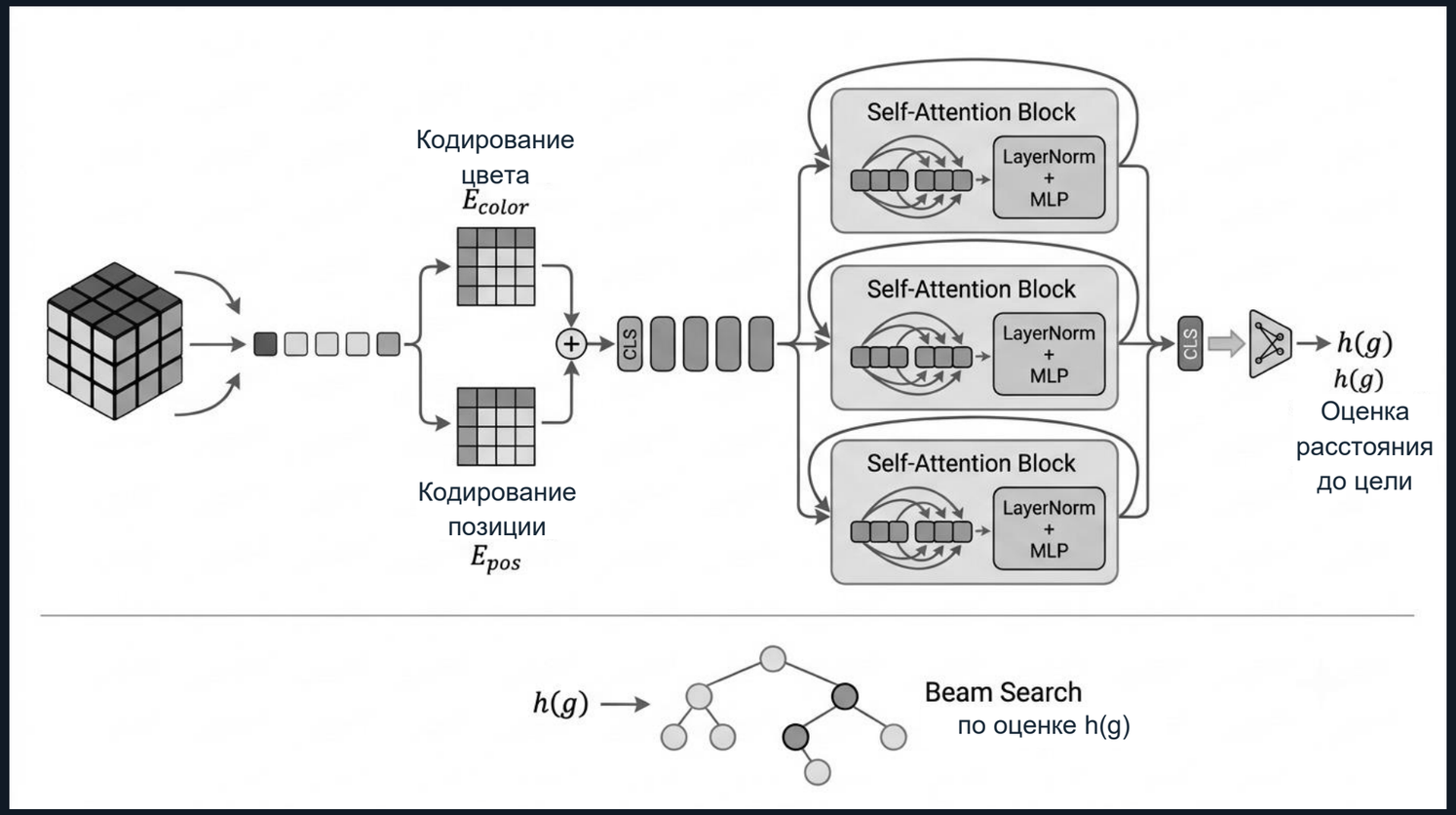
SA

Токены фацетов

≈ 1 млн параметров

Во всех сравнениях архитектур используется один dataset БД

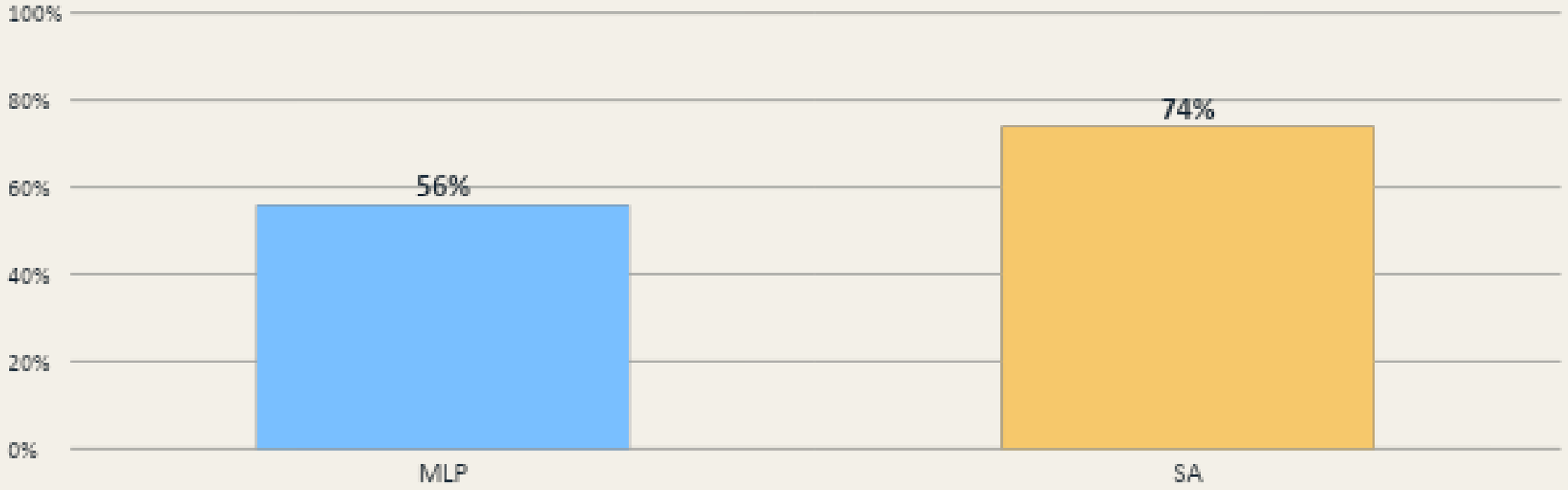
Self-attention связывает фацеты разных граней



4 головы · d_model = 192 · около 1 млн параметров

На 4×4×4 внимание повышает долю успеха

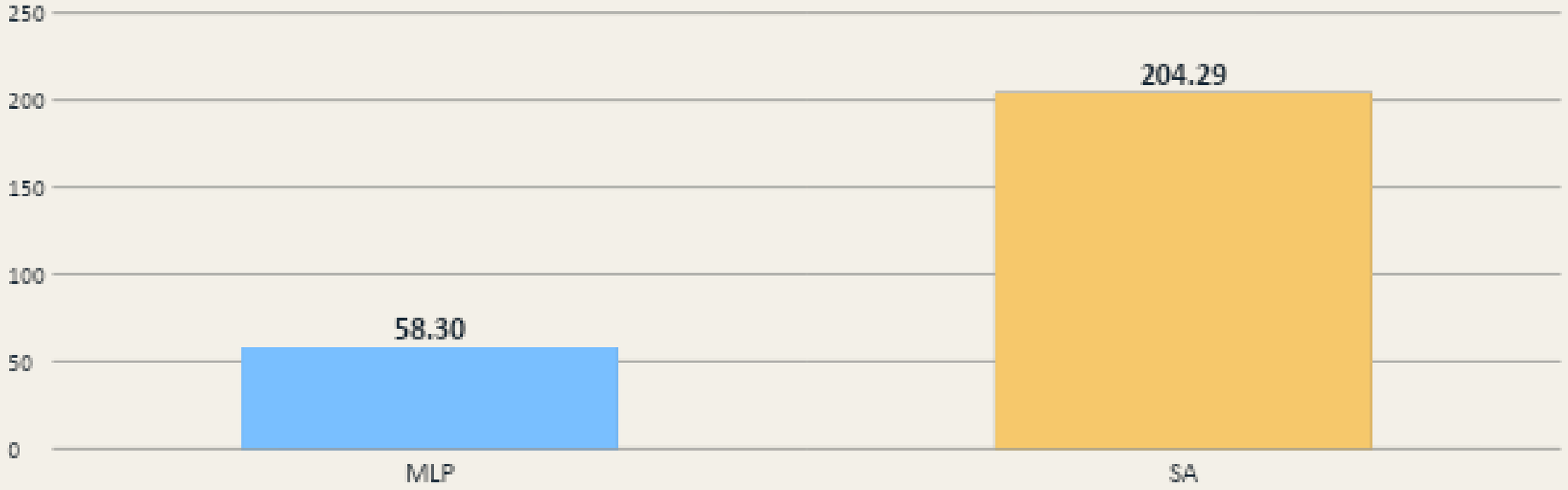
Доля решённых задач



43 задачи · W = 8192

За этот выигрыш приходится платить временем

Среднее время на задачу, с



4×4×4 · 43 задачи · W = 8192

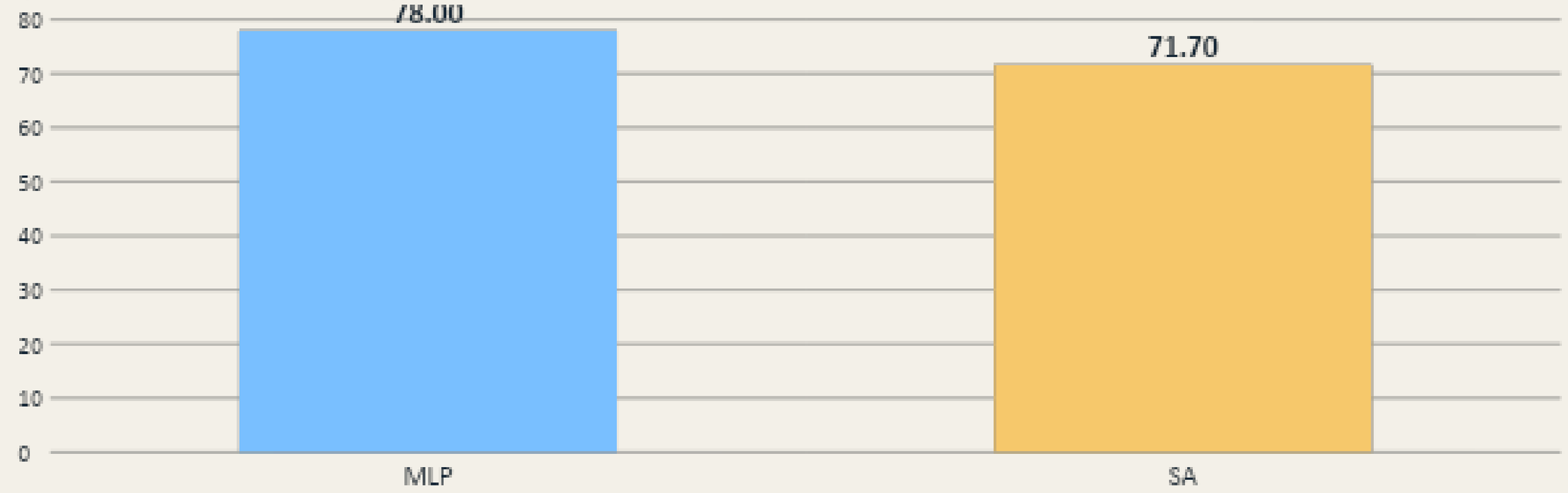
Быстрый MLP оставляет бюджет самому поиску

MLP и GNN: одинаковый успех, разное время

Архитектура	3×3×3 W = 128	3×3×3 W = 1024	4×4×4 W = 4096	4×4×4 W = 8192
MLP	**93%** / **0,04**	**100%** / **0,16**	51% / 2,3	56% / 58,3
GNN	**93%** / **1,3**	**100%** / **5,31**	—	—
SA	**98%** / 0,21	100% / 0,96	53% / 3,97	**74%** / 204,29

В ячейках: доля решённых задач / среднее время, с

У SA меньше средняя длина решения



4×4×4 · W = 8192 · отдельная экспериментальная серия

Beam Search: почему оценки недостаточно?

Конечный луч

Ошибочная оценка вытесняет ветвь

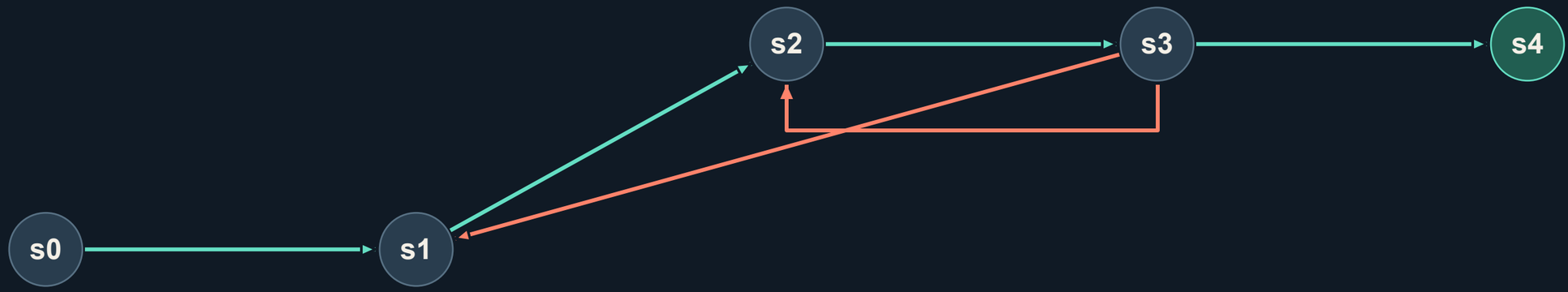
Потерянная ветвь не возвращается

Циклы и повторы

Модель снова выбирает знакомое

Бюджет тратится без прогресса

Первое изменение поиска: помнить недавний путь

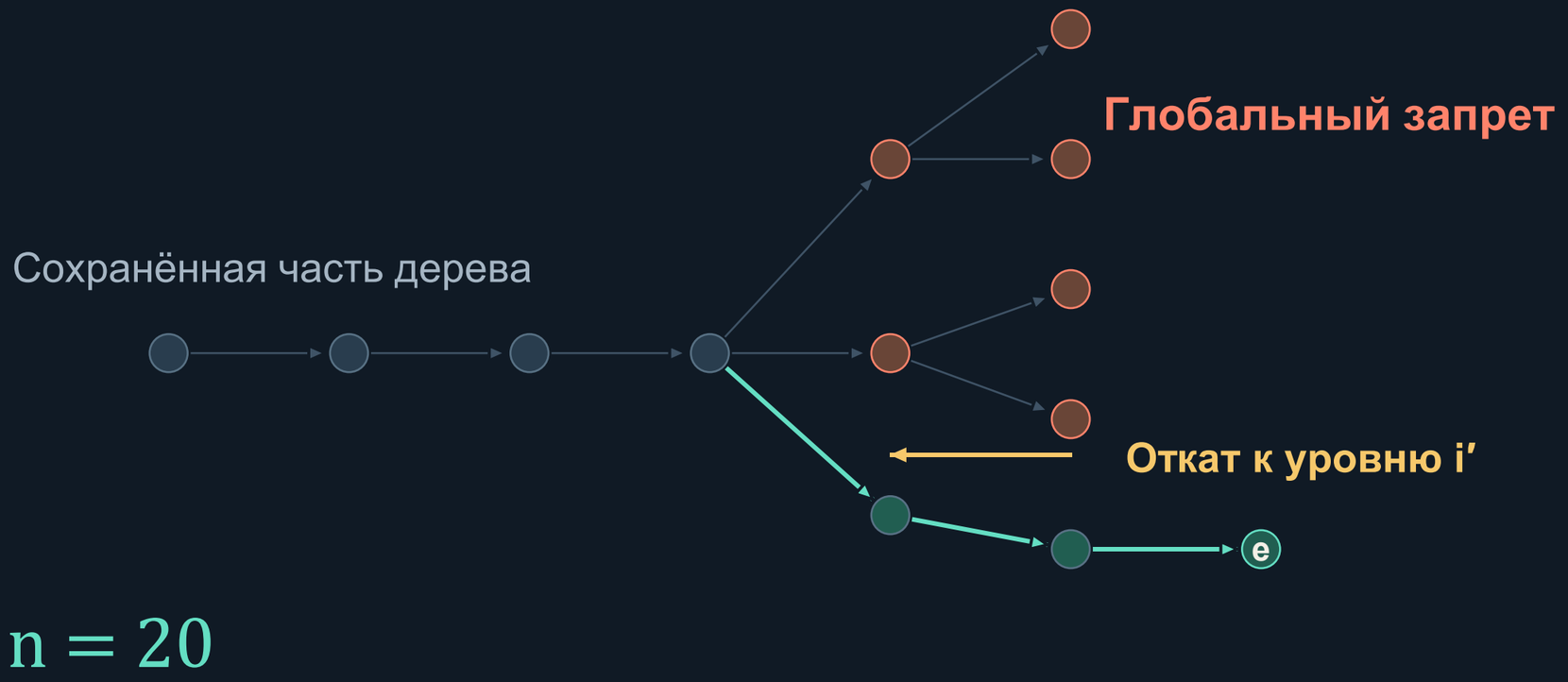


$k = 10$

Возвраты запрещены

Запрет повторов вдоль текущей траектории за последние k шагов

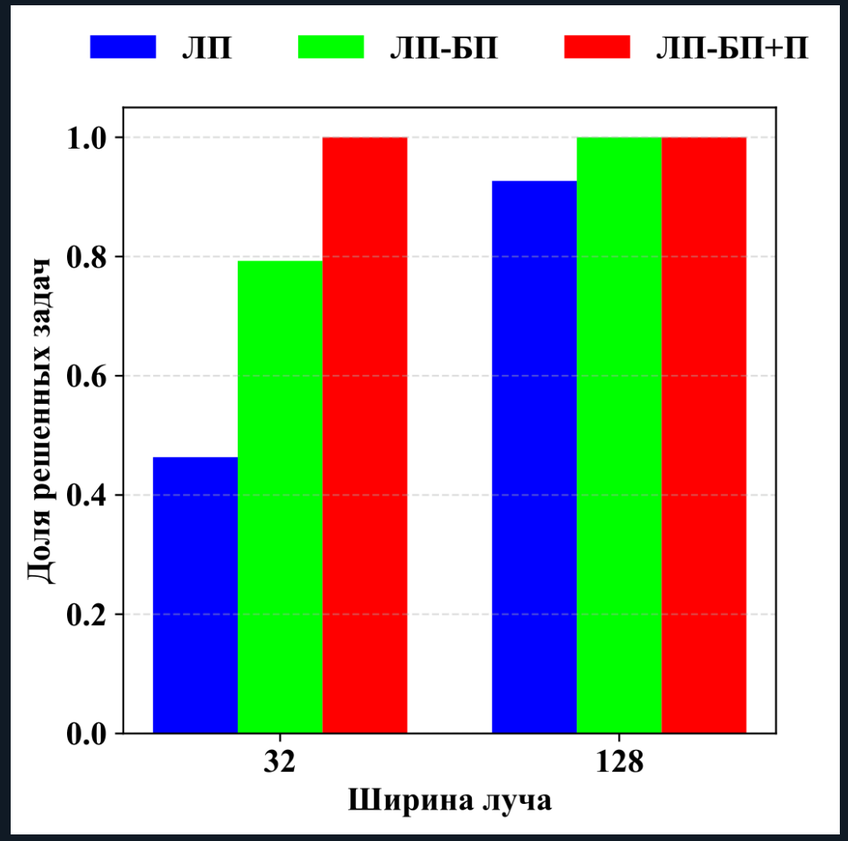
Второе изменение: вернуться после неудачи



$$n = 20$$

Глобальный запрет хвоста → откат к раннему затронутому уровню → пересчёт

На 3×3×3 помогают оба изменения поиска

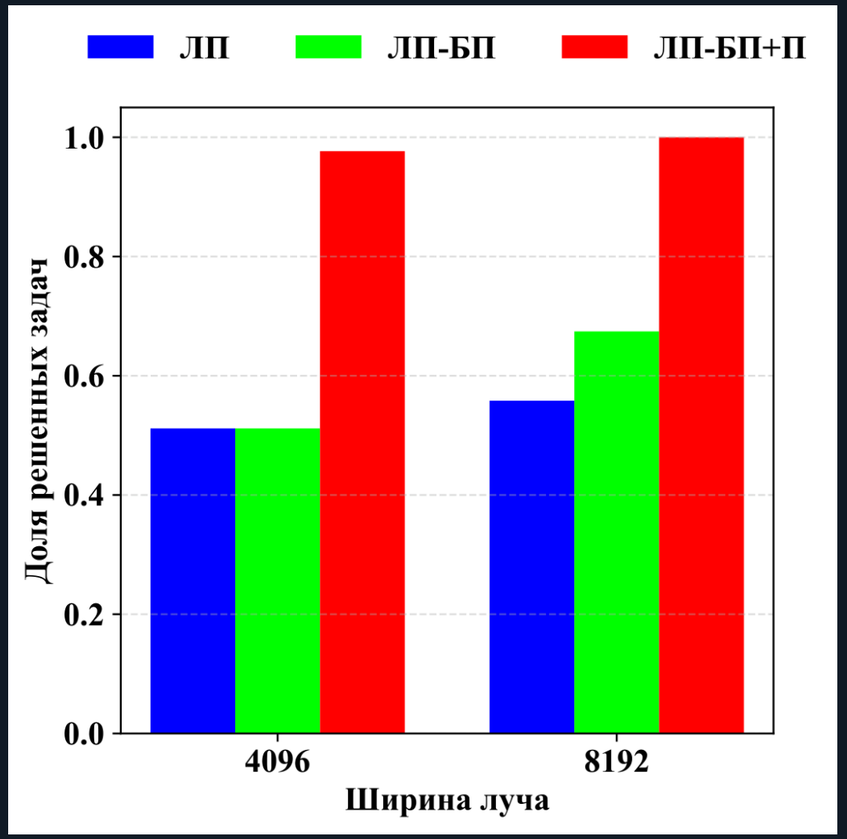


82 задачи · MLP + БД

- ЛП — Beam Search
- ЛП-БП — локальный запрет повторов
- ЛП-БП+П — запрет и повторные попытки

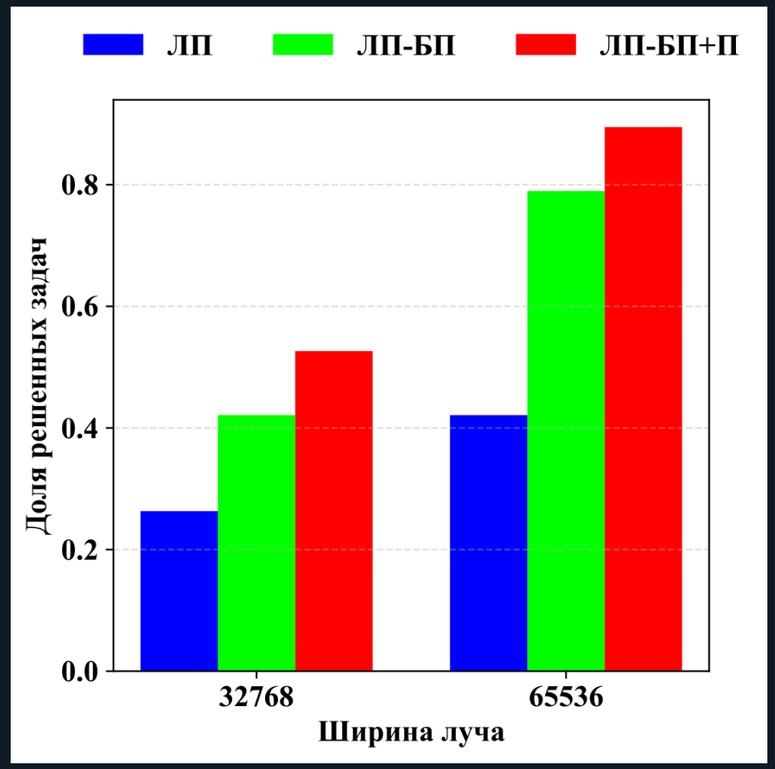
На 4×4×4 особенно важны повторные попытки

- ЛП — Beam Search
- ЛП-БП — локальный запрет повторов
- ЛП-БП+П — запрет и повторные попытки



43 задачи · MLP + БД

На 5×5×5 снова помогают оба механизма



ЛП — Beam Search

19 задач · MLP + БД

ЛП-БП — локальный запрет повторов

ЛП-БП+П — запрет и повторные попытки

ECayleyPy соединяет данные, модель и поиск

Данные

БД: фиксированный dataset с согласованными метками

Модель

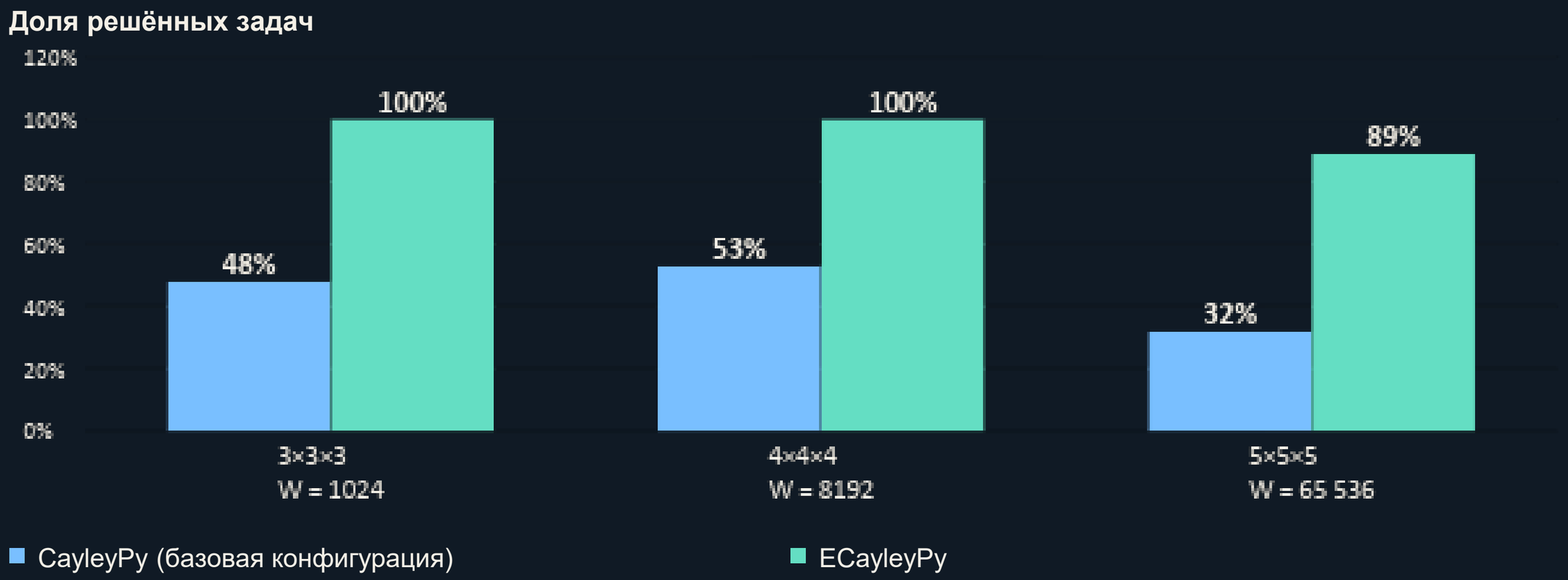
MLP: быстрая оценка состояния

Поиск

Локальная история и повторные попытки с откатом

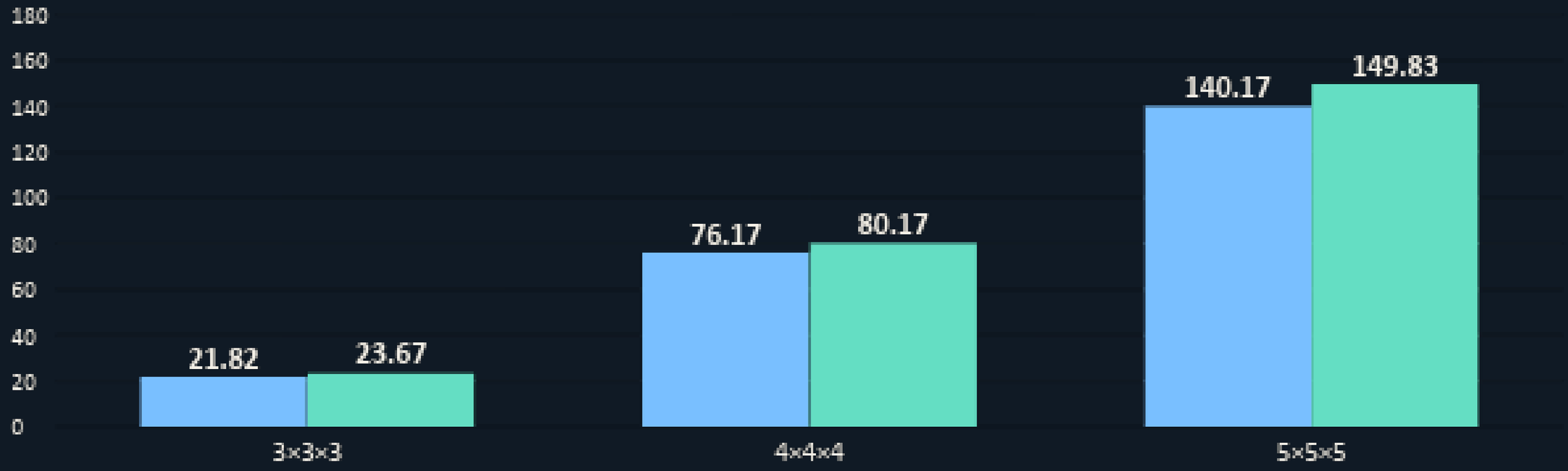
Базовая конфигурация: одна MLP + новые блуждания + обычный Beam Search

Доля успеха растёт на всех трёх размерах



Устойчивость имеет цену в длине пути

Средняя длина решения, ходов



Среднее по задачам, решённым обоими методами

05

Дальнейшие методы и задачи

CayleyPy RL, GFlowNet и следующий масштаб

CayleyPy RL: от глубины к стоимости пути

Warm-up → **TD-цель** → **Дообучение**

Random walks
метка глубины k

1 + min hθ(s′)
по соседям

Цель ограничена
диапазоном [0, k]

Стоимость до цели

$$h(e) = 0$$

$$h(s) = 1 + \min h(s')$$

После обучения путь строит Beam Search · точность и допустимость не гарантированы

CayleyPy RL: что даёт дообучение

LRX на S_{28}: один длинный элемент, 10 независимых запусков

Эпохи warm-up	Эпохи DQN	Доля успеха	Лучший путь	Медиана длины
30	0	70 %	464	650
100	0	70 %	436	516
30	**200**	**80 %**	**382**	**402**
0	230	20 %	434	471

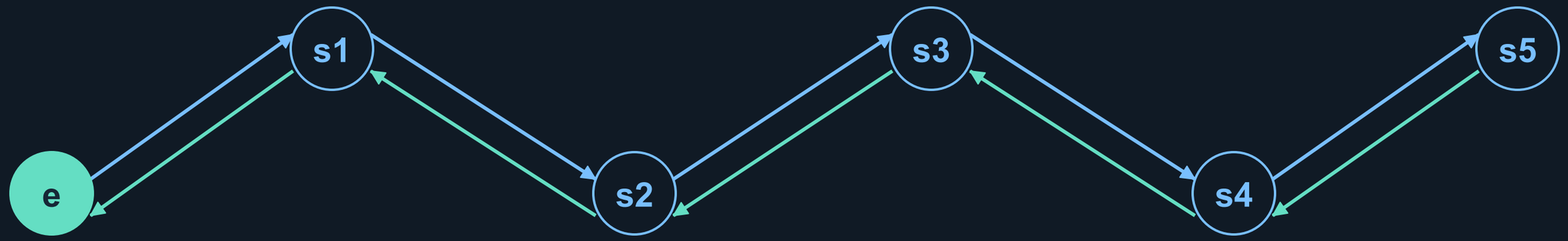
DQN после warm-up: 8 успехов из 10, медиана 402 вместо 650

Chervov и др., CayleyPy RL · таблица 6 · один элемент S_{28} · 10 запусков на строку

GFlowNet: согласовать вероятности траекторий

P_F: от цели

P_B: обратно к цели



Баланс вероятностей

Согласовать один путь
в двух направлениях

Регуляризация потока

Предпочесть меньше посещений
Теорема относится к точному оптимуму

Решение: обратная политика + Beam Search · тесты статьи: до 3×3×3

GFlowNet: сравнение с CayleyPy Cube

3×3×3: при узком луче политика чаще приводит к цели

Ширина W	Доля успеха CayleyPy Cube	Доля успеха GFlowNet	Средняя длина CayleyPy Cube	Средняя длина GFlowNet
1	0 %	47,1 %	—	—
8	**0,1 %**	**98,4 %**	**—**	**—**
64	**68,7 %**	**100 %**	**—**	**25,33**
512	100 %	100 %	24,34	23,49
4096	100 %	100 %	22,44	22,42
262 144	100 %	100 %	21,15	21,24

При W = 262 144 на H200: GFlowNet 1,74 с; CayleyPy Cube 6,19 с

Morozov и др., 2026 · таблица 1, строки для 3×3×3 · QTM · 1000 задач · лимит 100 шагов

Что исследовать дальше

$N \geq 6$

Кубики большего размера

Больше фацетов и генераторов
Сложнее память и inference

Новые среды

Сложные головоломки
и другие графы Кэли

Больше решений за то же время

Одинаковое время и лимит памяти.
Сравнить долю решённых задач
и длины найденных путей.

Проверить большие задачи, новые среды и качество решений при одинаковых ограничениях

Обучаемая эвристика и устойчивый поиск

Данные

Частичная информация
о геометрии графа

Модель

Оценка состояния,
которую можно быстро вычислить

Поиск

Конкретный путь
из несовершенных оценок



Найденный путь

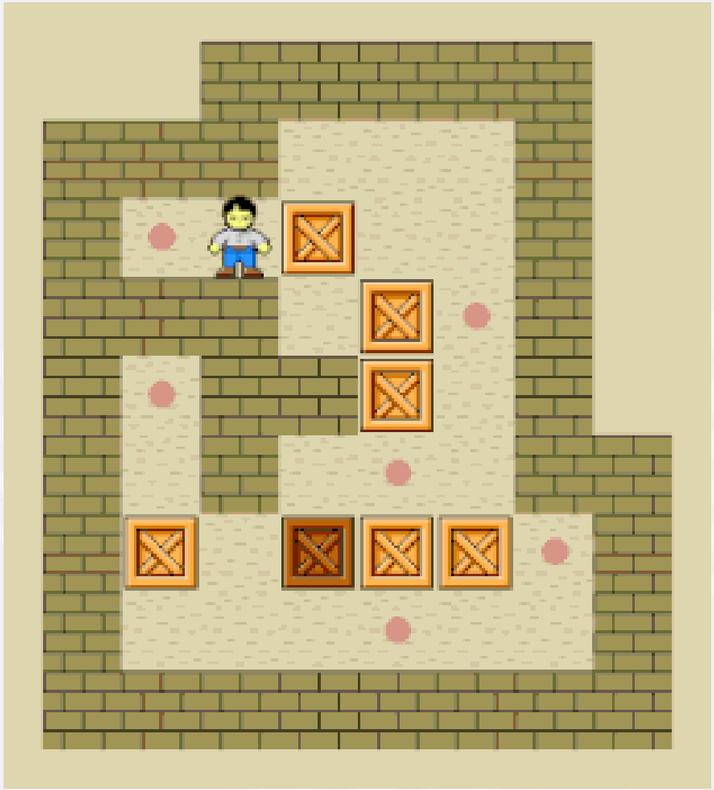
Методы эвристического поиска и машинного обучения для решения головоломок

Про Sokoban и Pushworld

Ананикян А.
Лаборатория Маркова

Sokoban

- Hiroyuki Imabayashi, 1981
- Thinking Rabbit, 1982
- В начале – пара десятков задач
- Сейчас
  - десятки тысяч ручных
  - миллионы сгенерированных
- NP-трудный, 1996; Dor & Zwick
- PSPACE-полный, 1997; Culberson
- Два основных направления для решения компьютером



Sokoban: алгоритмы

≈ 2003

Ken’ichiro Takahashi starts Takaken

2007

Takaken — among the leaders

January 2013

Florent Diedler starts Sokolution

December 2016

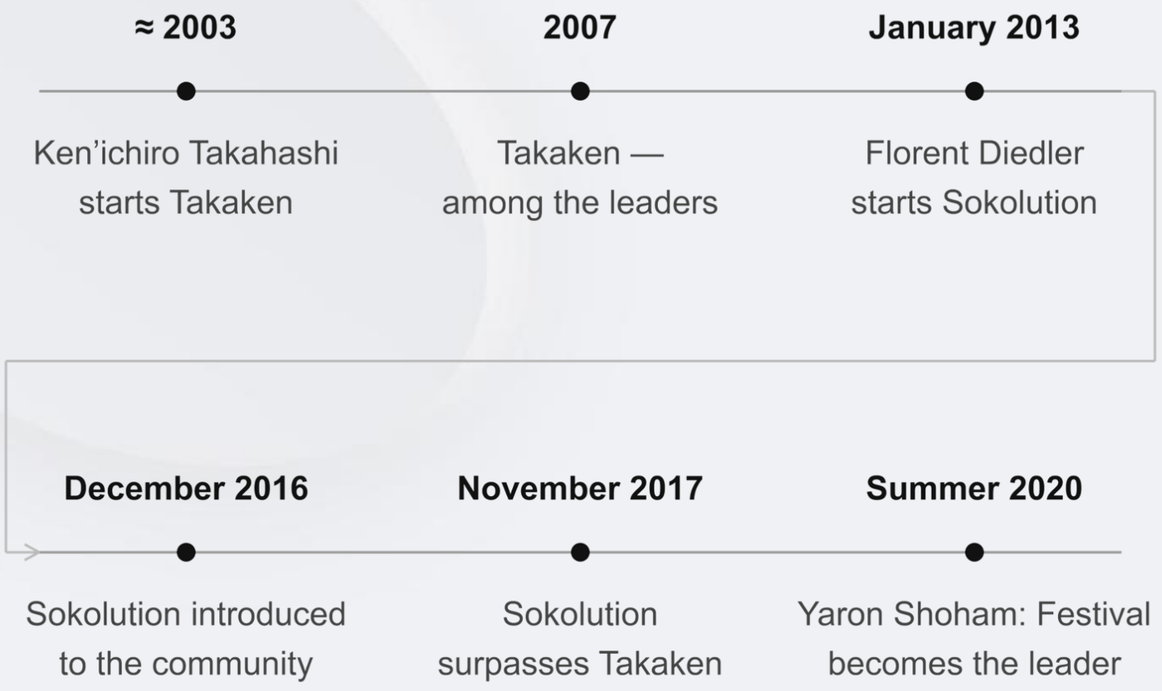
Sokolution introduced to the community

November 2017

Sokolution surpasses Takaken

Summer 2020

Yaron Shoham: Festival becomes the leader



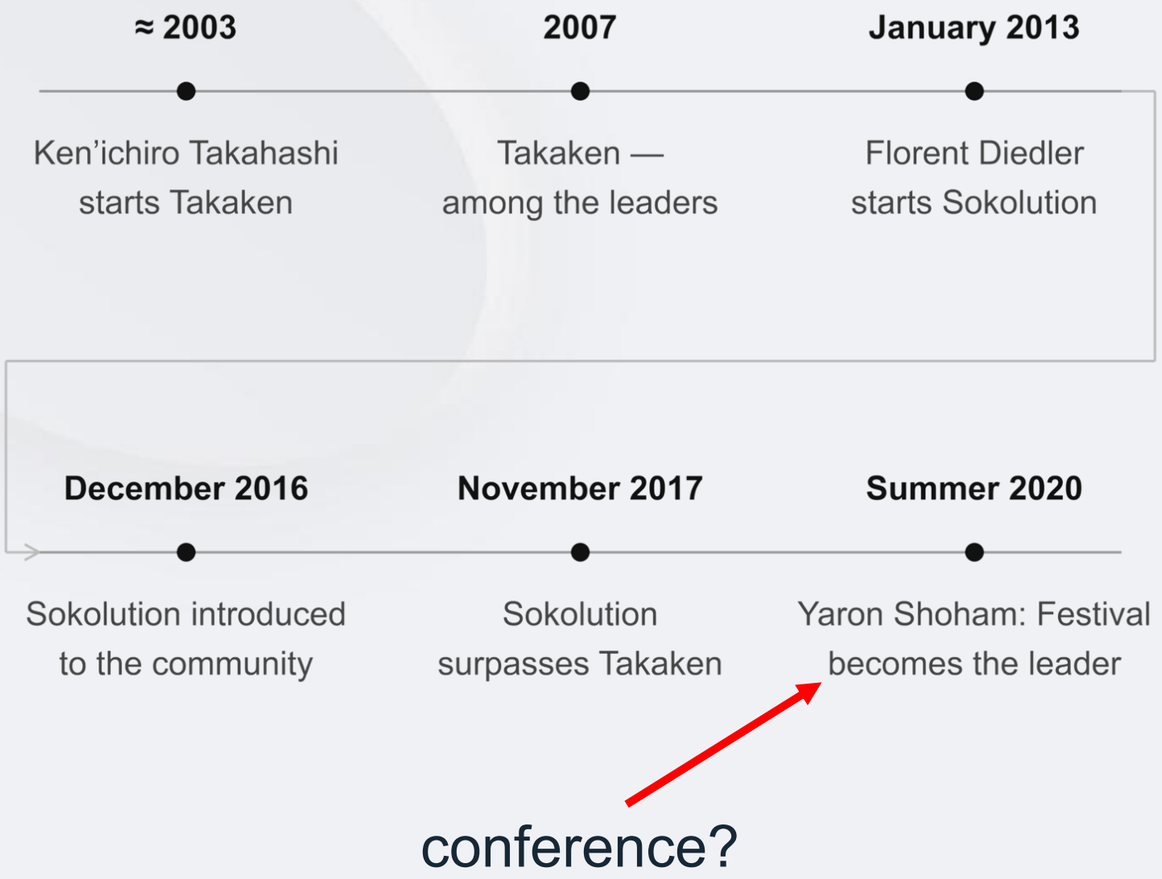
Conclusion



I worked on Sokolution since January 2013, the project is almost finished because I don't want to have a **burn-out**. It was really hard to be focused on it during the last **5 years** working sometimes **until 3 a.m** to debug a new feature or waiting for the end of a bench. It was a real pleasure to work on this subject and exchange information with Brian and Matthias.

Even if I have a lack of self-confidence, I really don't realize that I have created the best solver of the world in the Sokoban world. I think I am proud of me even if very few people know this domain.

That's all ! I revealed all features inside Sokolution. I hope it will help other solvers to be more powerful and beat Sokolution :)



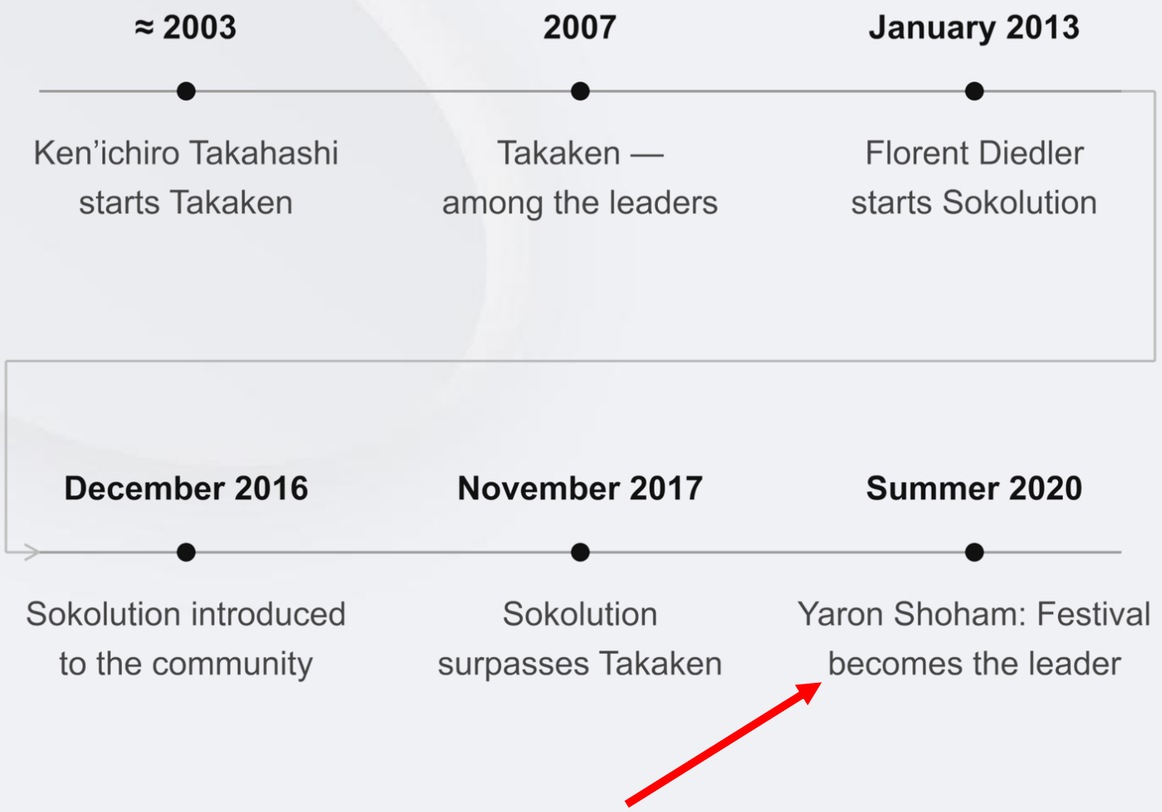
Conclusion



I worked on Sokolution since January 2013, the project is almost finished because I don't want to have a **burn-out**. It was really hard to be focused on it during the last **5 years** working sometimes **until 3 a.m** to debug a new feature or waiting for the end of a bench. It was a real pleasure to work on this subject and exchange information with Brian and Matthias.

Even if I have a lack of self-confidence, I really don't realize that I have created the best solver of the world in the Sokoban world. I think I am proud of me even if very few people know this domain.

That's all ! I revealed all features inside Sokolution. I hope it will help other solvers to be more powerful and beat Sokolution :)



Conference on Games

Conclusion



I worked on Sokolution since January 2013, the project is almost finished because I don't want to have a **burn-out**. It was really hard to be focused on it during the last **5 years** working sometimes **until 3 a.m** to debug a new feature or waiting for the end of a bench. It was a real pleasure to work on this subject and exchange information with Brian and Matthias.

Even if I have a lack of self-confidence, I really don't realize that I have created the best solver of the world in the Sokoban world. I think I am proud of me even if very few people know this domain.

That's all ! I revealed all features inside Sokolution. I hope it will help other solvers to be more powerful and beat Sokolution :)

Sokoban: алгоритмы

≈ 2003 — 2007 — January 2013

Showing 1–1 of 1 results for author: Shoham, Yaron

Search v0.5.6 released 2020-02-24

Searching in archive **cs**. Search in all archives.

Shoham, Yaron | Author(s) | Search

Show abstracts | Hide abstracts

Advanced Search

50 results per page. Sort results by Announcement date (newest first) Go

1. arXiv:2105.01904 [pdf, other] cs.LG cs.AI

Solving Sokoban with forward-backward reinforcement learning

Authors: Yaron Shoham, Gal Elidan

Abstract: Despite seminal advances in reinforcement learning in recent years, many domains where the rewards are sparse, e.g. given only at task completion, remain quite challenging. In such cases, it can be beneficial to tackle the task both from its beginning and end, and make the two ends meet. Existing approaches that do so, however, are not effective in the common scenario where the strategy needed nea... ▽ More

Submitted 22 May, 2021; **v1** submitted 5 May, 2021; **originally announced** May 2021.

Comments: To be published in SoCS 2021

ACM Class: I.2.6; I.2.8

Sokolution introduced to the community

Sokolution surpasses Takaken

Yaron Shoham: Festival becomes the leader

Conference on Games

Conclusion

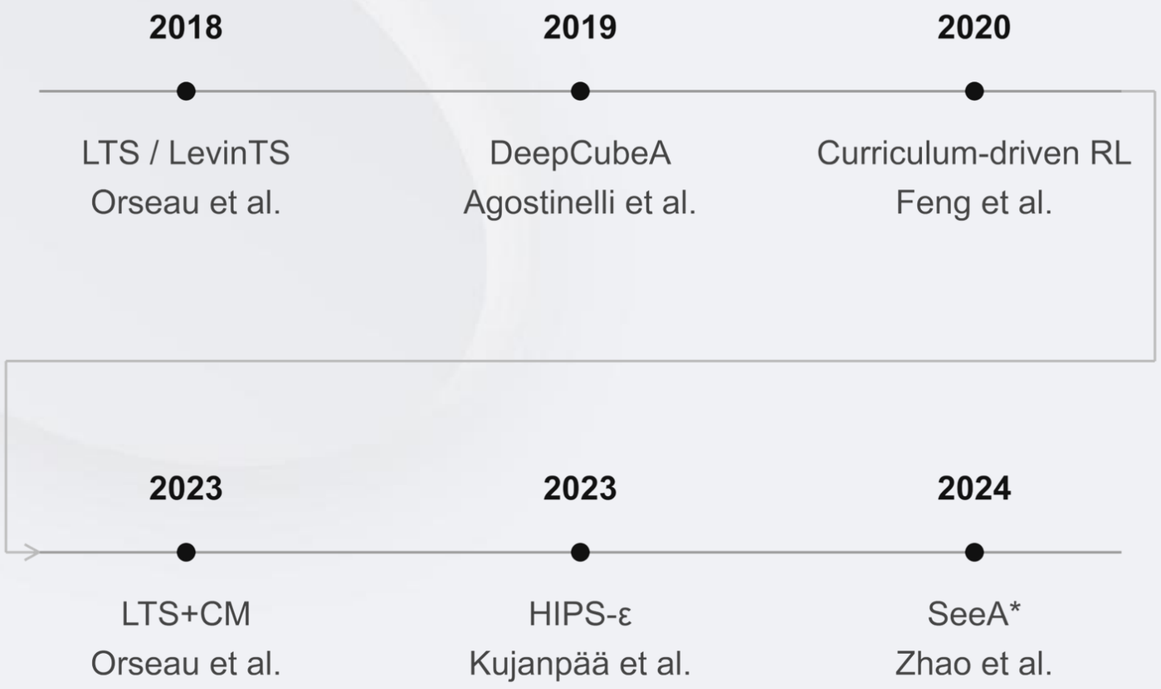


I worked on Sokolution since January 2013, the project is almost finished because I don't want to have a **burn-out**. It was really hard to be focused on it during the last **5 years** working sometimes **until 3 a.m** to debug a new feature or waiting for the end of a bench. It was a real pleasure to work on this subject and exchange information with Brian and Matthias.

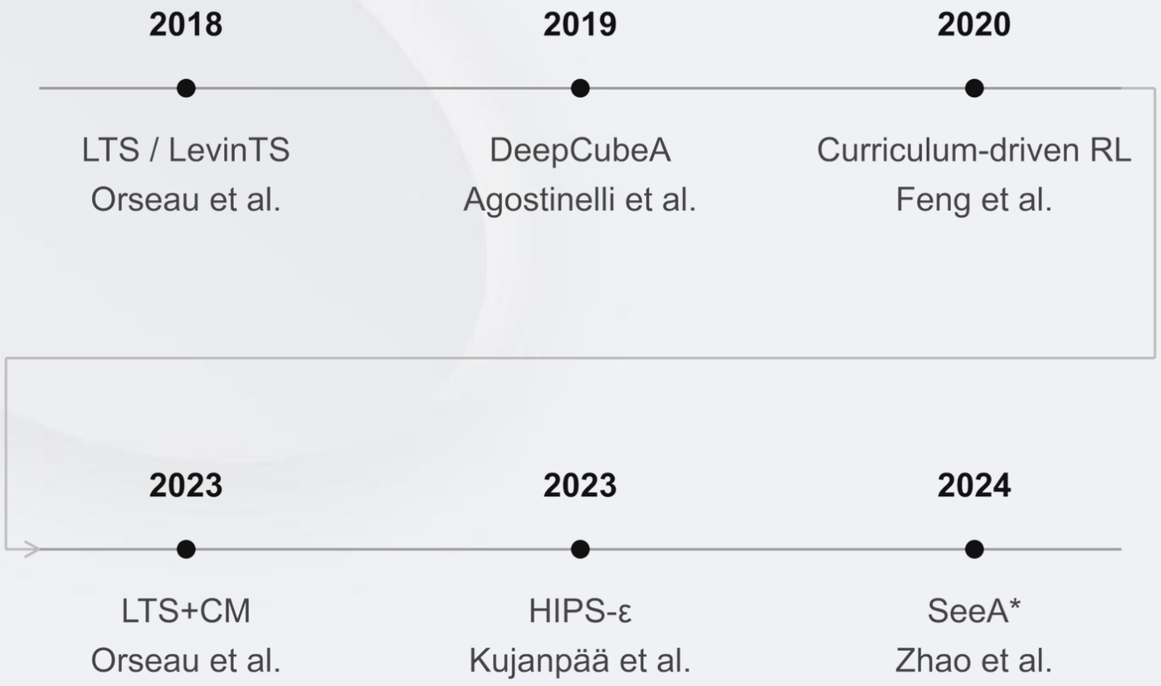
Even if I have a lack of self-confidence, I really don't realize that I have created the best solver of the world in the Sokoban world. I think I am proud of me even if very few people know this domain.

That's all ! I revealed all features inside Sokolution. I hope it will help other solvers to be more powerful and beat Sokolution :)

Sokoban: ML

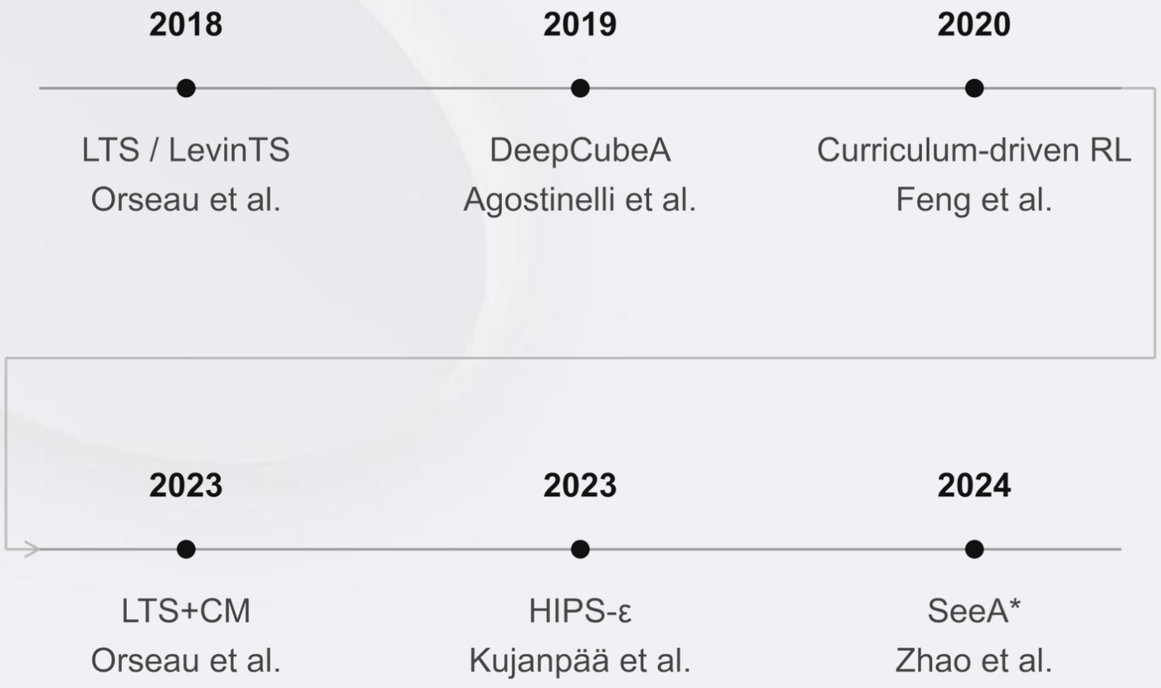

2018
2019
2020
LTS / LevinTS
Orseau et al.
DeepCubeA
Agostinelli et al.
Curriculum-driven RL
Feng et al.
2023
2023
2024
LTS+CM
Orseau et al.
HIPS-ε
Kujanpää et al.
SeeA*
Zhao et al.

Sokoban: ML



Улучшим SeeA*!

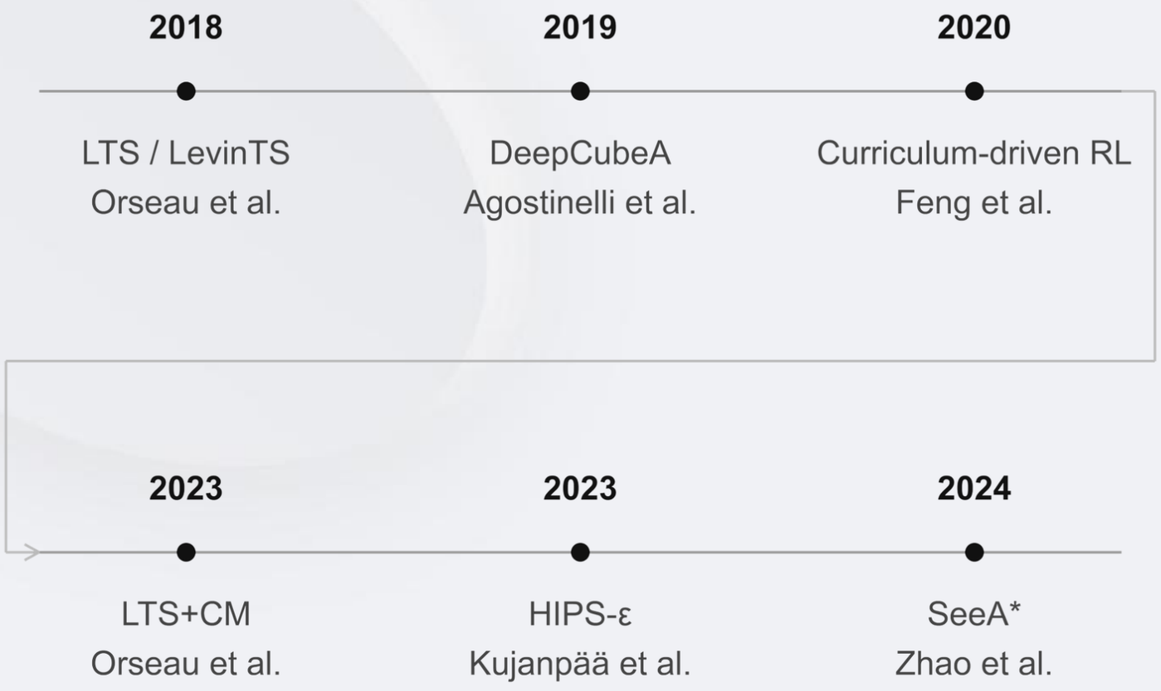
Sokoban: ML



Улучшим SeeA*!

- Результаты не воспроизводятся

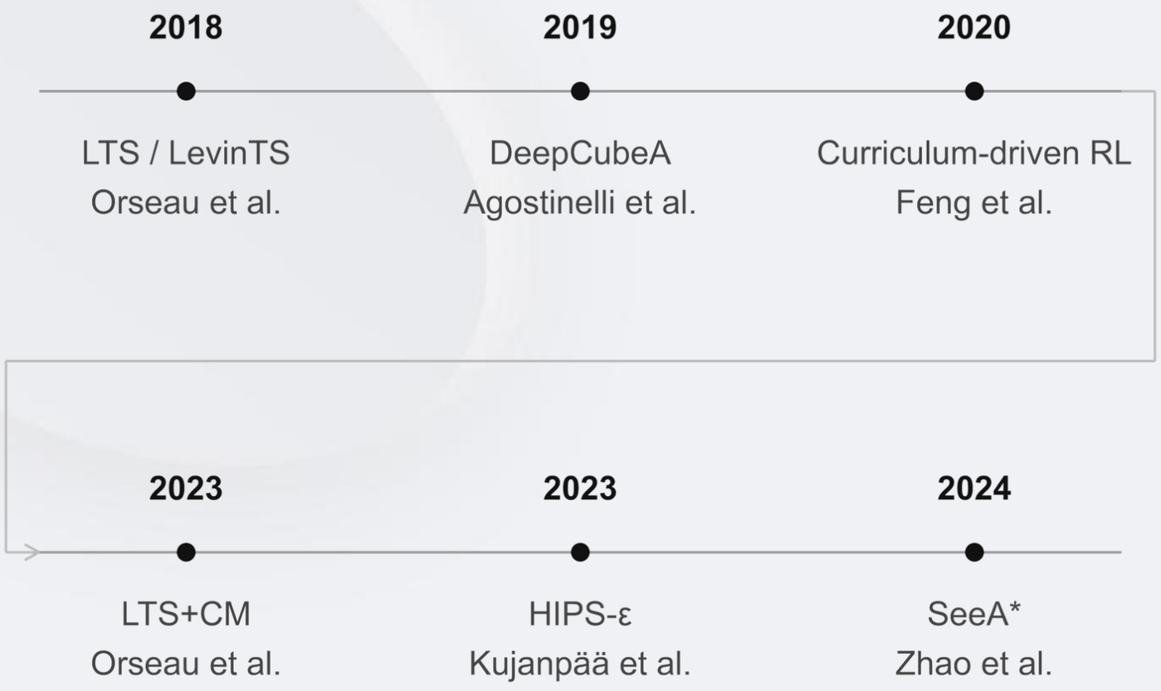
Sokoban: ML



Улучшим SeeA*!

- Результаты не воспроизводятся
- Мы такого не боимся. Чо там с кодом?

Sokoban: ML

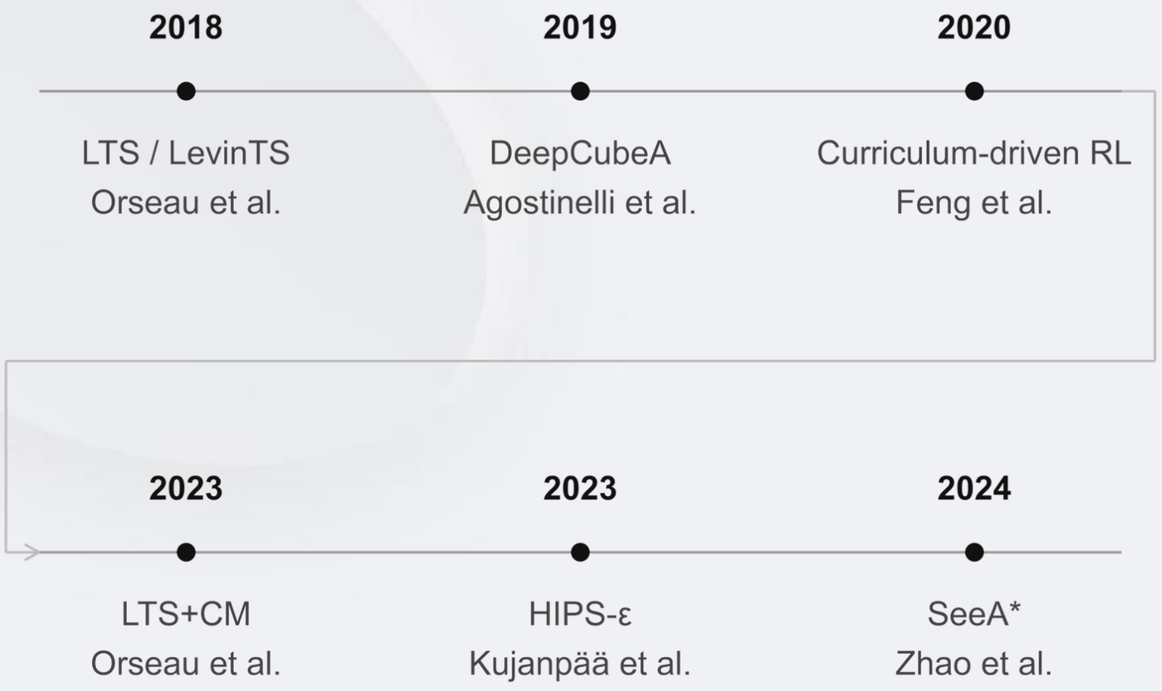


Улучшим SeeA*!

- Результаты не воспроизводятся
- Мы такого не боимся. Чо там с кодом?

```
def get_f_cost(self, child, g, predicted_h):
    if self._use_learned_heuristic and self._use_heuristic:
        return self._w * max(predicted_h, child.heuristic_value()) + g
    if self._use_learned_heuristic:
        return self._w * predicted_h + g
    if self._use_heuristic:
        return self._w * child.heuristic_value() * g
    return g
```

Sokoban: ML



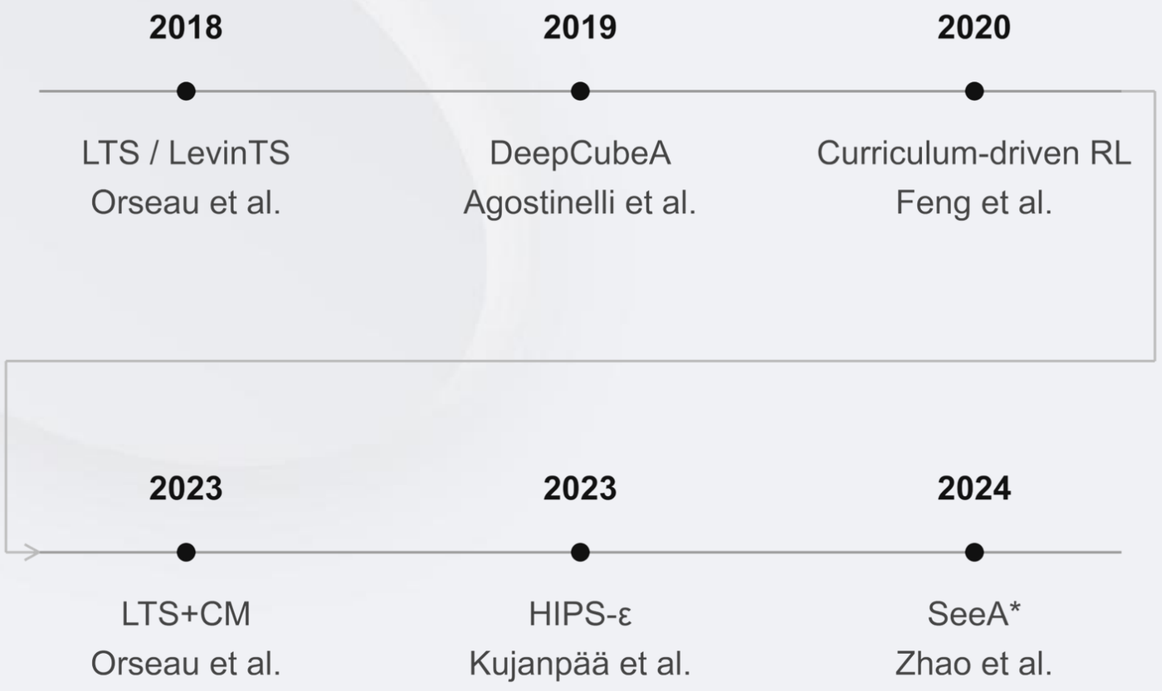
Улучшим SeeA*!

- Результаты не воспроизводятся
- Мы такого не боимся. Чо там с кодом?

```
def get_f_cost(self, child, g, predicted_h):
    if self._use_learned_heuristic and self._use_heuristic:
        return self._w * max(predicted_h, child.heuristic_value()) + g
    if self._use_learned_heuristic:
        return self._w * predicted_h + g
    if self._use_heuristic:
        return self._w * child.heuristic_value() * g
    return g
```

- Ничего, просто залили не тот код. Мб поправить?

Sokoban: ML



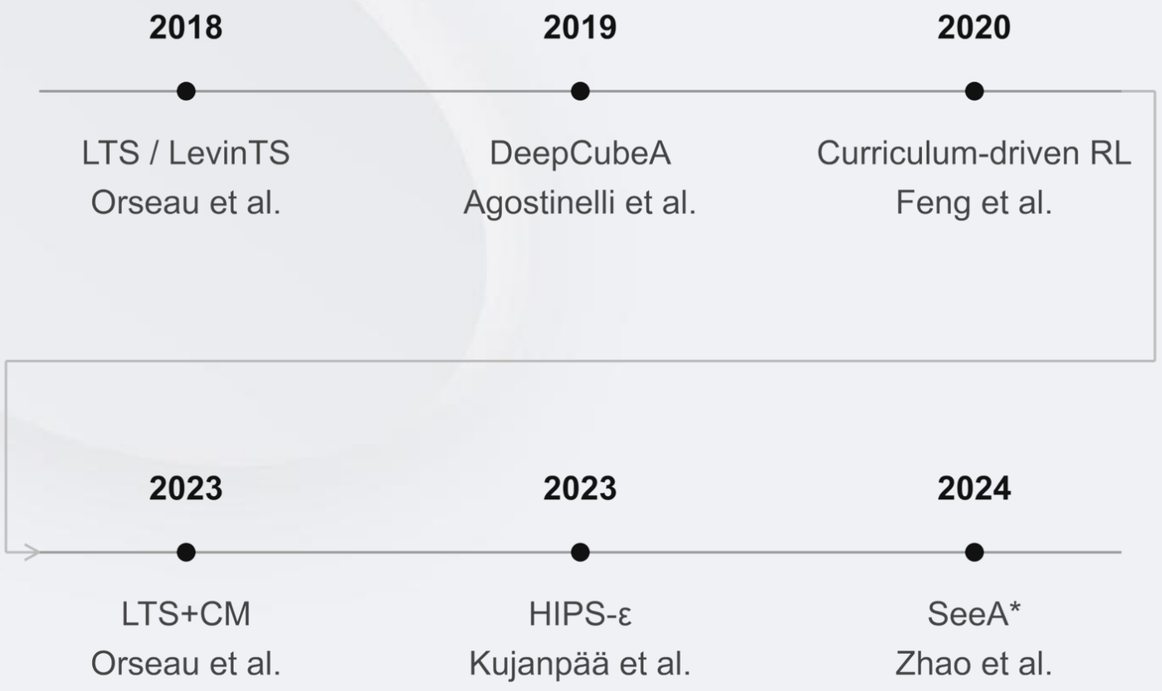
Улучшим SeeA*!

- Результаты не воспроизводятся
- Мы такого не боимся. Чо там с кодом?

```
def get_f_cost(self, child, g, predicted_h):
    if self._use_learned_heuristic and self._use_heuristic:
        return self._w * max(predicted_h, child.heuristic_value()) + g
    if self._use_learned_heuristic:
        return self._w * predicted_h + g
    if self._use_heuristic:
        return self._w * child.heuristic_value() * g
    return g
```

- Ничего, просто залили не тот код. Мб поправить?
- Результаты ухудшились

Sokoban: ML



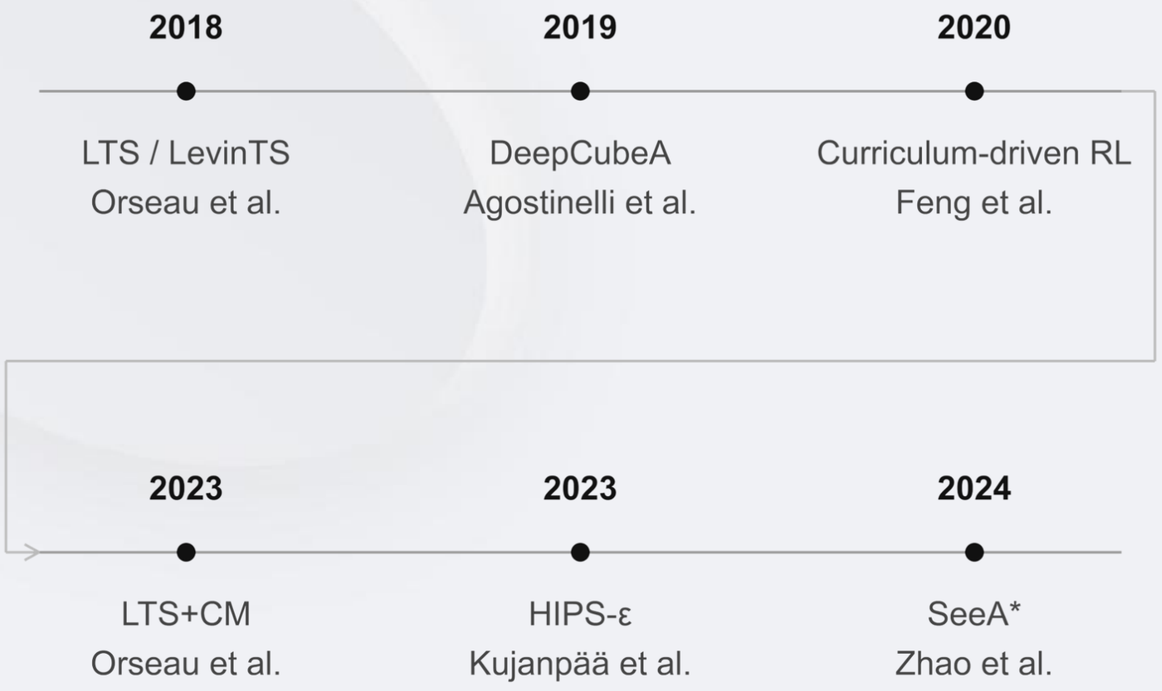
Улучшим SeeA*!

- Результаты не воспроизводятся
- Мы такого не боимся. Чо там с кодом?

```
def get_f_cost(self, child, g, predicted_h):
    if self._use_learned_heuristic and self._use_heuristic:
        return self._w * max(predicted_h, child.heuristic_value()) + g
    if self._use_learned_heuristic:
        return self._w * predicted_h + g
    if self._use_heuristic:
        return self._w * child.heuristic_value() * g
    return g
```

- Ничего, просто залили не тот код. Мб поправить?
- Результаты ухудшились
- А если написать A* по-петербургски?

Sokoban: ML



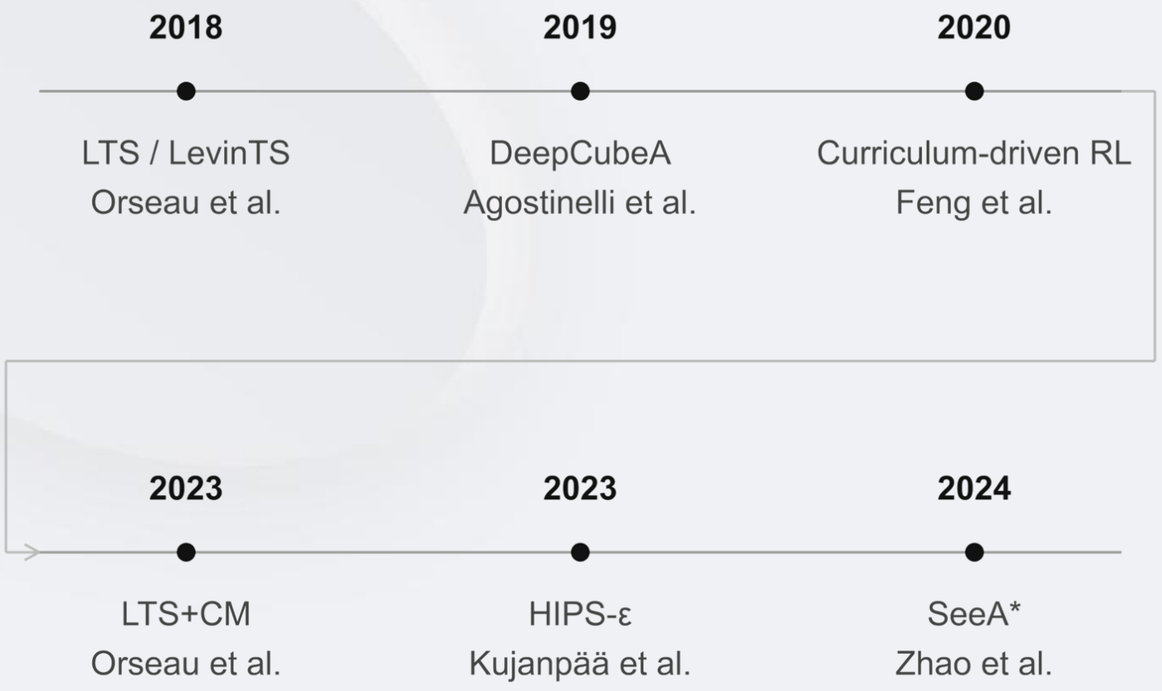
Улучшим SeeA*!

- Результаты не воспроизводятся
- Мы такого не боимся. Чо там с кодом?

```
def get_f_cost(self, child, g, predicted_h):
    if self._use_learned_heuristic and self._use_heuristic:
        return self._w * max(predicted_h, child.heuristic_value()) + g
    if self._use_learned_heuristic:
        return self._w * predicted_h + g
    if self._use_heuristic:
        return self._w * child.heuristic_value() * g
    return g
```

- Ничего, просто залили не тот код. Мб поправить?
- Результаты ухудшились
- А если написать A* по-петербургски?
- Результаты улучшились

Sokoban: ML



Улучшим SeeA*!

- Результаты не воспроизводятся
- Мы такого не боимся. Чо там с кодом?

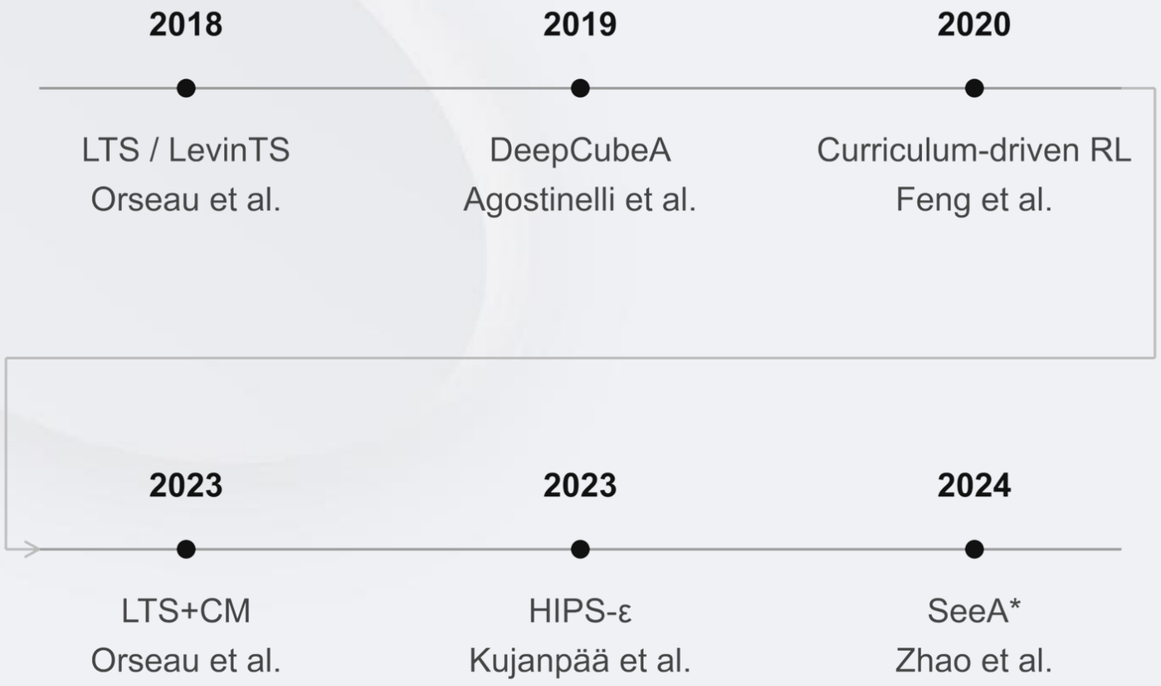
```
def get_f_cost(self, child, g, predicted_h):
    if self._use_learned_heuristic and self._use_heuristic:
        return self._w * max(predicted_h, child.heuristic_value()) + g
    if self._use_learned_heuristic:
        return self._w * predicted_h + g
    if self._use_heuristic:
        return self._w * child.heuristic_value() * g
    return g
```

- Ничего, просто залили не тот код. Мб поправить?
- Результаты ухудшились
- А если написать A* по-петербургски?
- Результаты улучшились

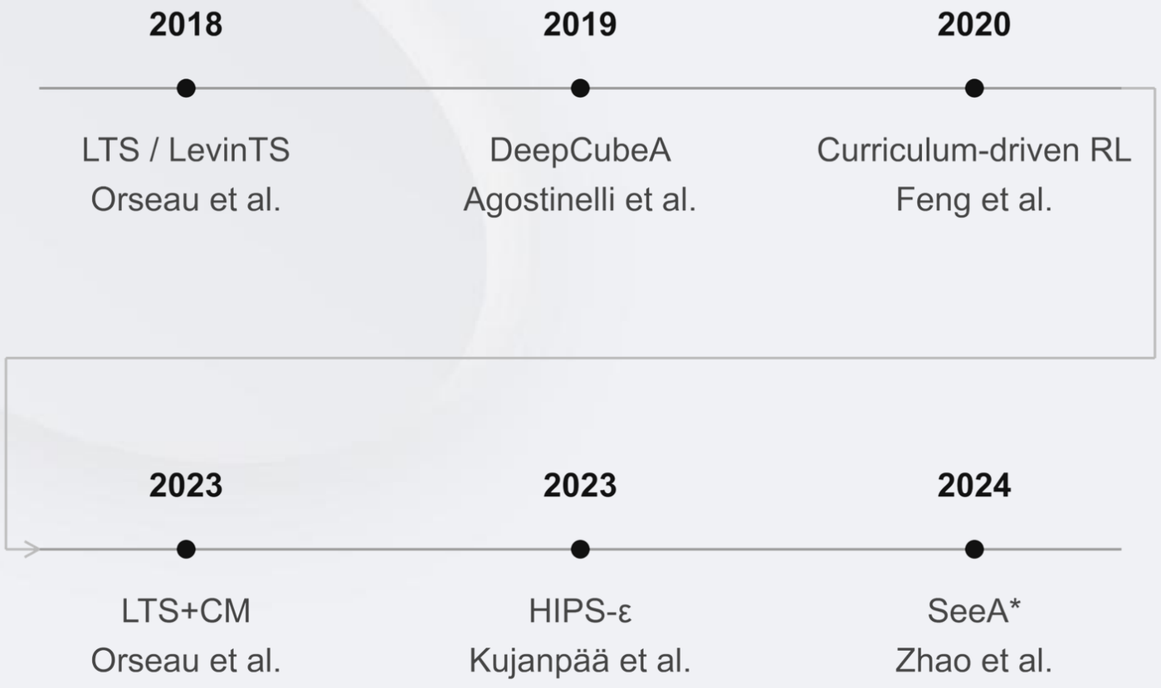
В 20 РАЗ!

Sokoban: ML

~~Улучшим SeeA*!~~

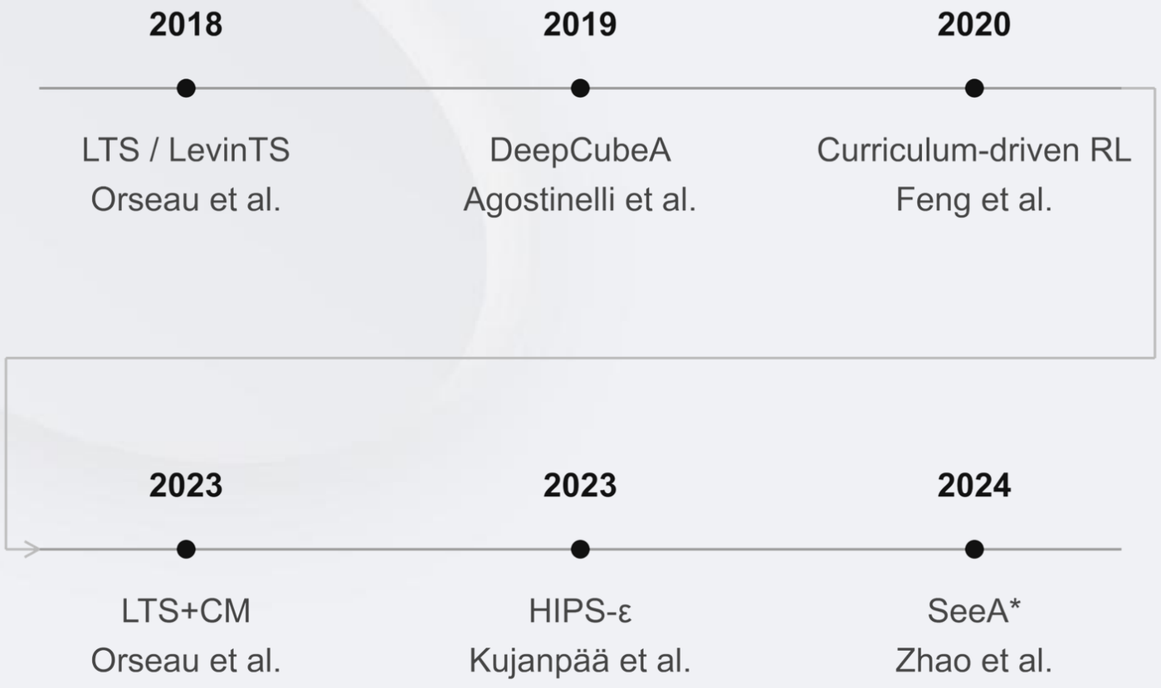


Sokoban: ML



~~Улучшим SeeA*!~~
~~Улучшим LTS~~

Sokoban: ML



~~Улучшим SeeA*!~~
~~Улучшим LTS~~
Улучшим HIPS-ε?

Sokoban: ML

2018
LTS / LevinTS
Orseau et al.

2019
DeepCubeA
Agostinelli et al.

2020
Curriculum-driven RL
Feng et al.

2023
LTS+CM
Orseau et al.

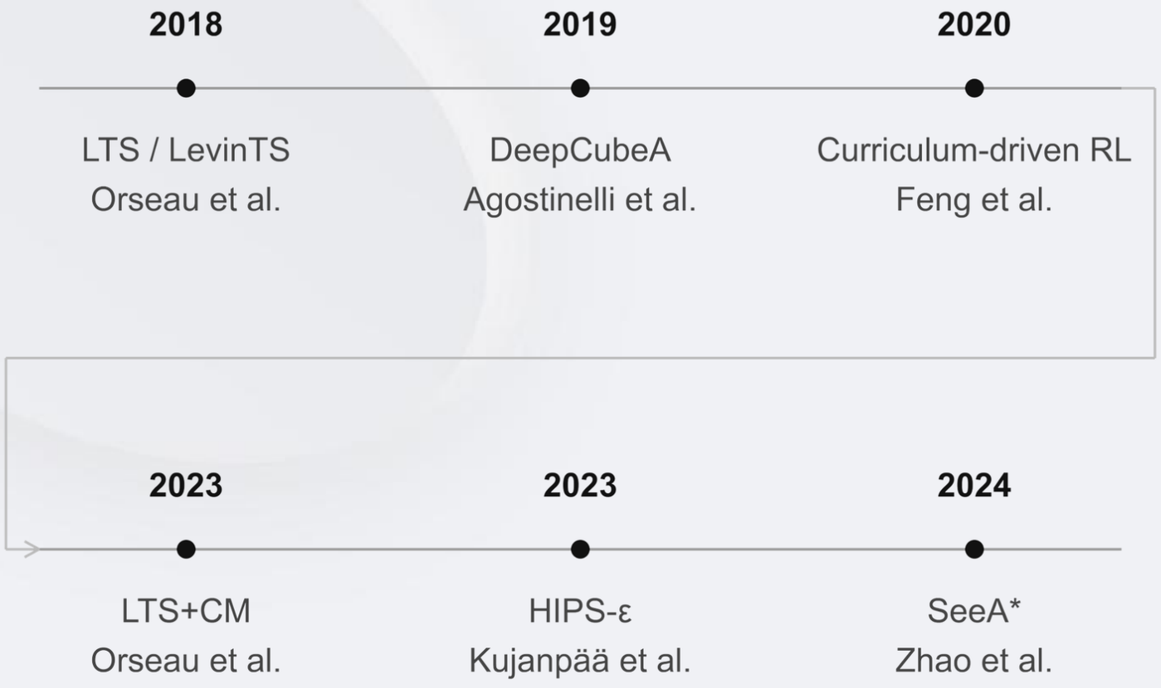
2023
HIPS-ε
Kujanpää et al.

2024
SeeA*
Zhao et al.

~~Улучшим SeeA*!~~
~~Улучшим LTS~~
Улучшим HIPS-ε?



Sokoban: ML



Conclusion



I worked on Sokoban since September 2025, the project is almost finished because I don't want to have a **burn-out**. It was really hard to be focused on it during the last **12 weeks** working sometimes **until 3 a.m** to fix one more A* realisation or waiting for the end of a bench. It was a real pleasure to work on this subject and exchange information with Daniel and Konstantin.

Even if I lack self-confidence, I am now confident that removing ML makes a Sokoban solver twenty times faster. I think I am proud of myself, even if the other solvers remain undefeated because they never ran.

That's all ! I revealed all features inside Sokoban ML-solvers. I hope it will help other solvers to be more powerful and beat Sokoban :)

Sokoban: ML

2018
LTS / LevinTS
Orseau et al.

2019
DeepCubeA
Agostinelli et al.

2020
Curriculum-driven RL
Feng et al.

2023
LTS+CM
Orseau et al.

2023
HIPS-ε
Kujanpää et al.

2024
SeeA*
Zhao et al.

conferences?

Conclusion

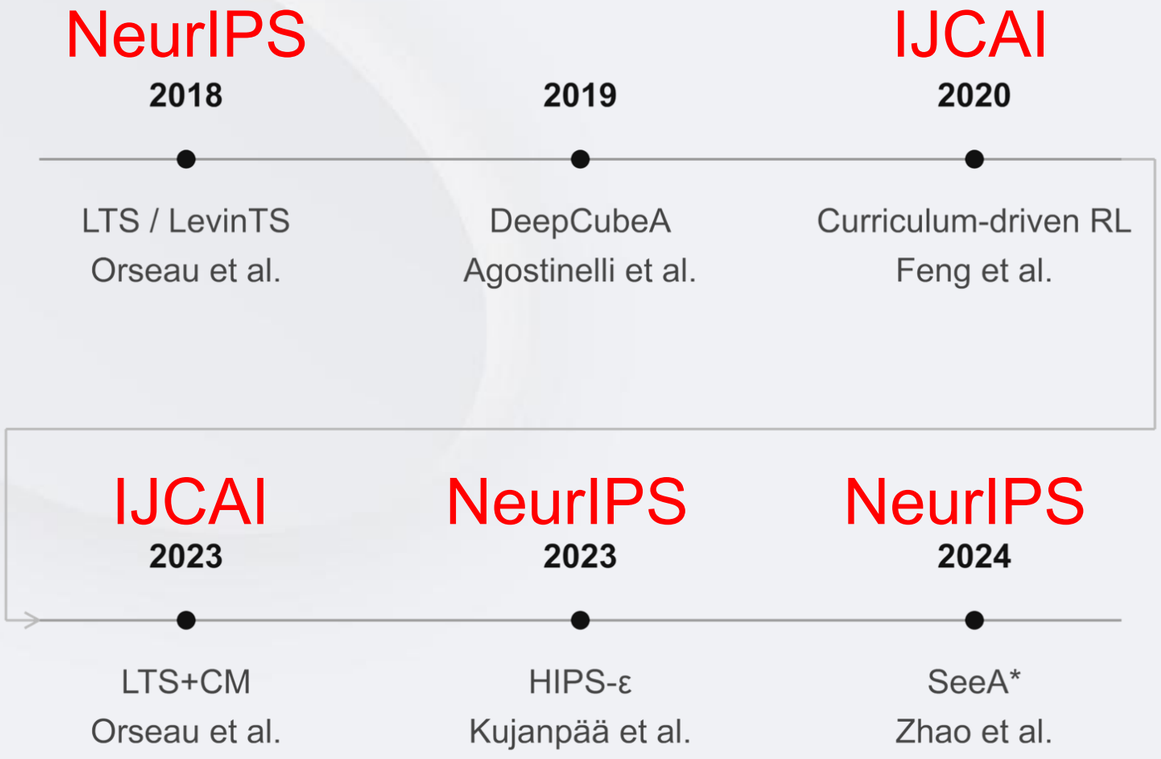


I worked on Sokoban since September 2025, the project is almost finished because I don't want to have a **burn-out**. It was really hard to be focused on it during the last **12 weeks** working sometimes **until 3 a.m** to fix one more A* realisation or waiting for the end of a bench. It was a real pleasure to work on this subject and exchange information with Daniel and Konstantin.

Even if I lack self-confidence, I am now confident that removing ML makes a Sokoban solver twenty times faster. I think I am proud of myself, even if the other solvers remain undefeated because they never ran.

That's all ! I revealed all features inside Sokoban ML-solvers. I hope it will help other solvers to be more powerful and beat Sokoban :)

Sokoban: ML



conferences?

Conclusion



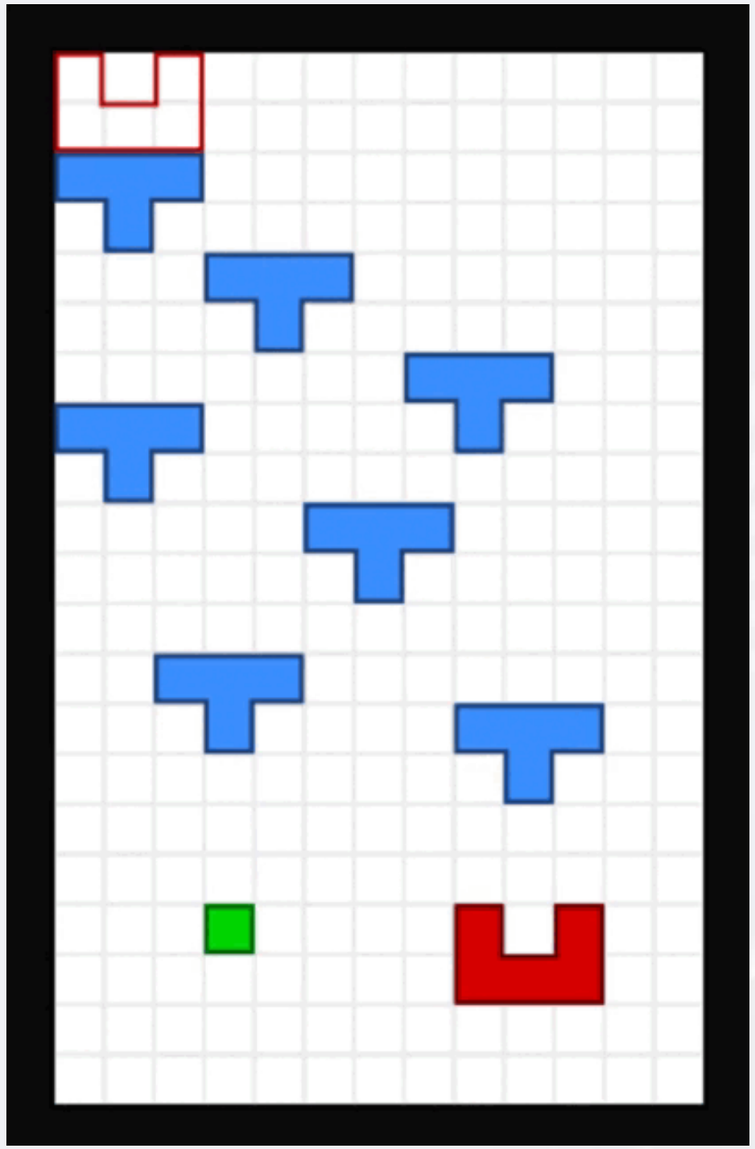
I worked on Sokoban since September 2025, the project is almost finished because I don't want to have a **burn-out**. It was really hard to be focused on it during the last **12 weeks** working sometimes **until 3 a.m** to fix one more A* realisation or waiting for the end of a bench. It was a real pleasure to work on this subject and exchange information with Daniel and Konstantin.

Even if I lack self-confidence, I am now confident that removing ML makes a Sokoban solver twenty times faster. I think I am proud of myself, even if the other solvers remain undefeated because they never ran.

That's all ! I revealed all features inside Sokoban ML-solvers. I hope it will help other solvers to be more powerful and beat Sokoban :)

PushWorld

- Google DeepMind, 2023; Kansky et al.
- 5 наборов задач:
  - Level 0 (15.4k)
  - Level 1–4 (68/74/67/14)
- Согласно DeepMind, лучше Sokoban:
  - разные формы объектов
  - подвижные препятствия
  - планирование действий
- Асимптотически – все еще Sokoban
- Те же два направления, но нет сформированных сообществ

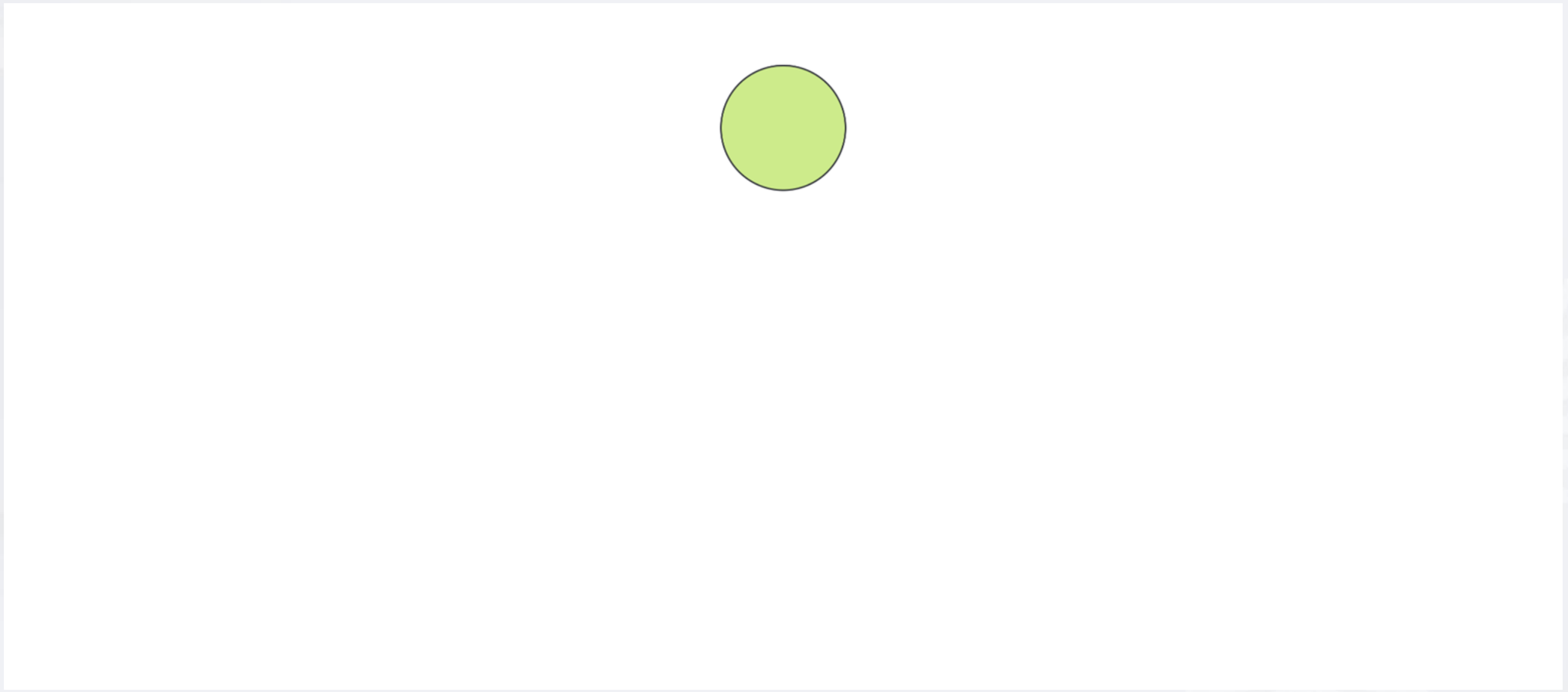


Литературный обзор

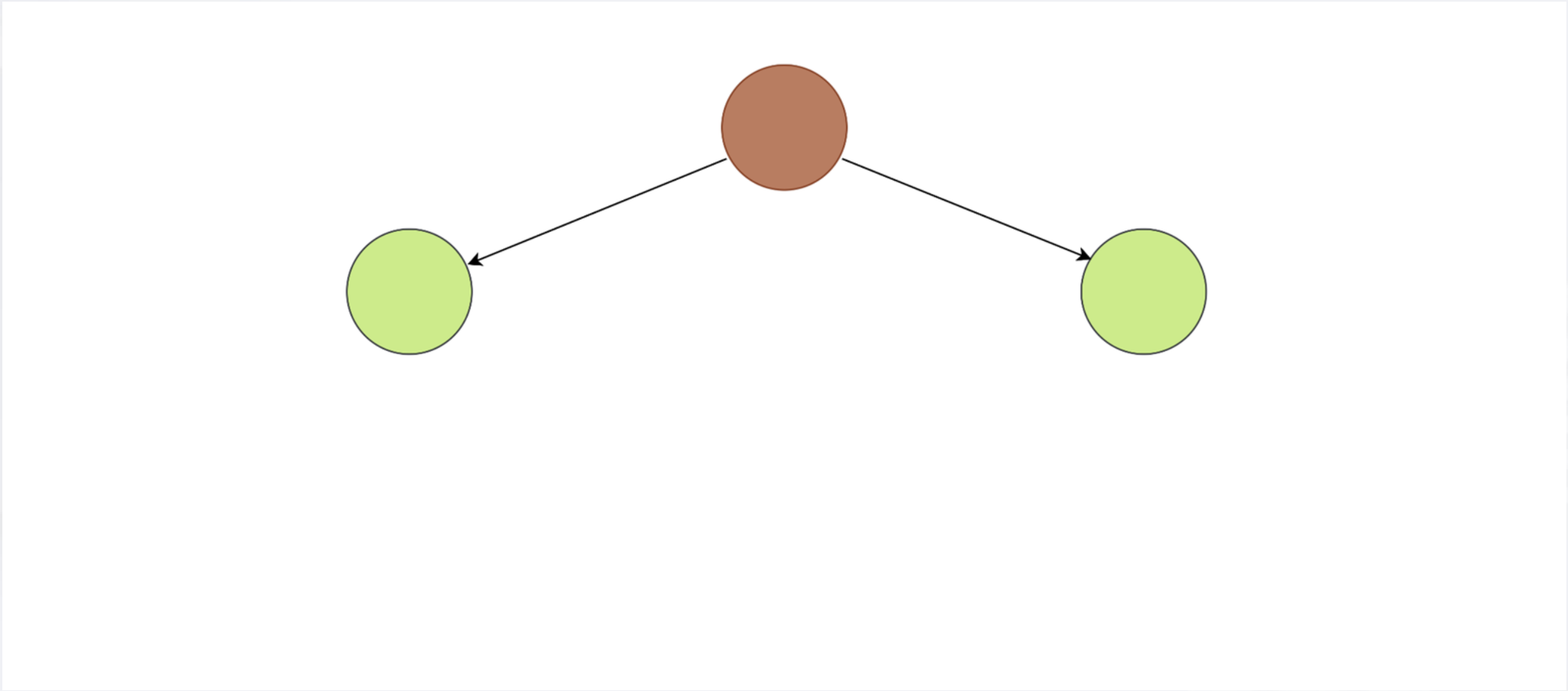
- 1 article*, как один из бенчмарков (ICML 2026)
- 2 preprints. В работе Deepmind формулируется задача, проводятся эксперименты с решателями на C/C++ и RL-агентами. Вторая работа содержит PushWorld как один из бенчмарков.
- 1 bachelor thesis. Улучшаются результаты RL-агентов.
- 1 master thesis. Обучается модель, и производится сравнение с одним из C-решателей из оригинальной работы Deepmind.

*выросла из master thesis

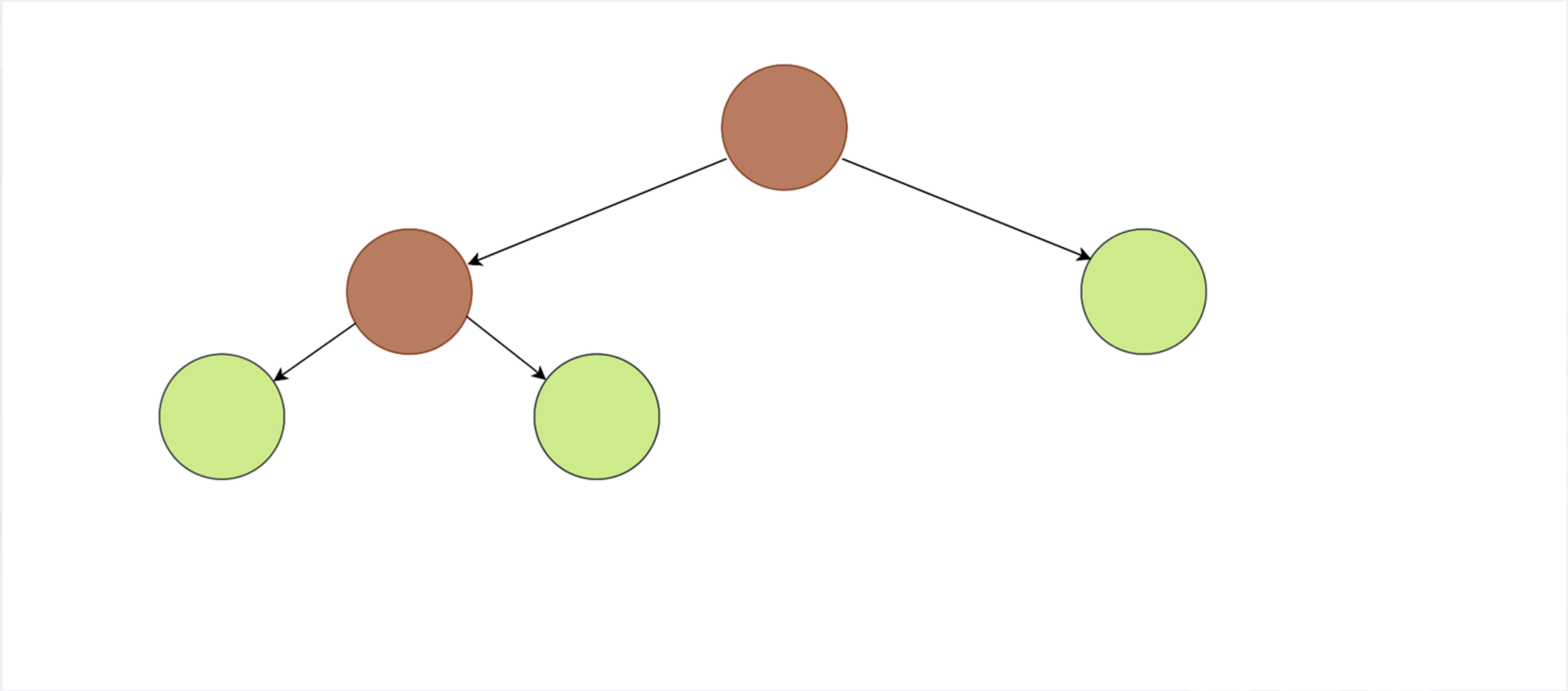
Best-first search



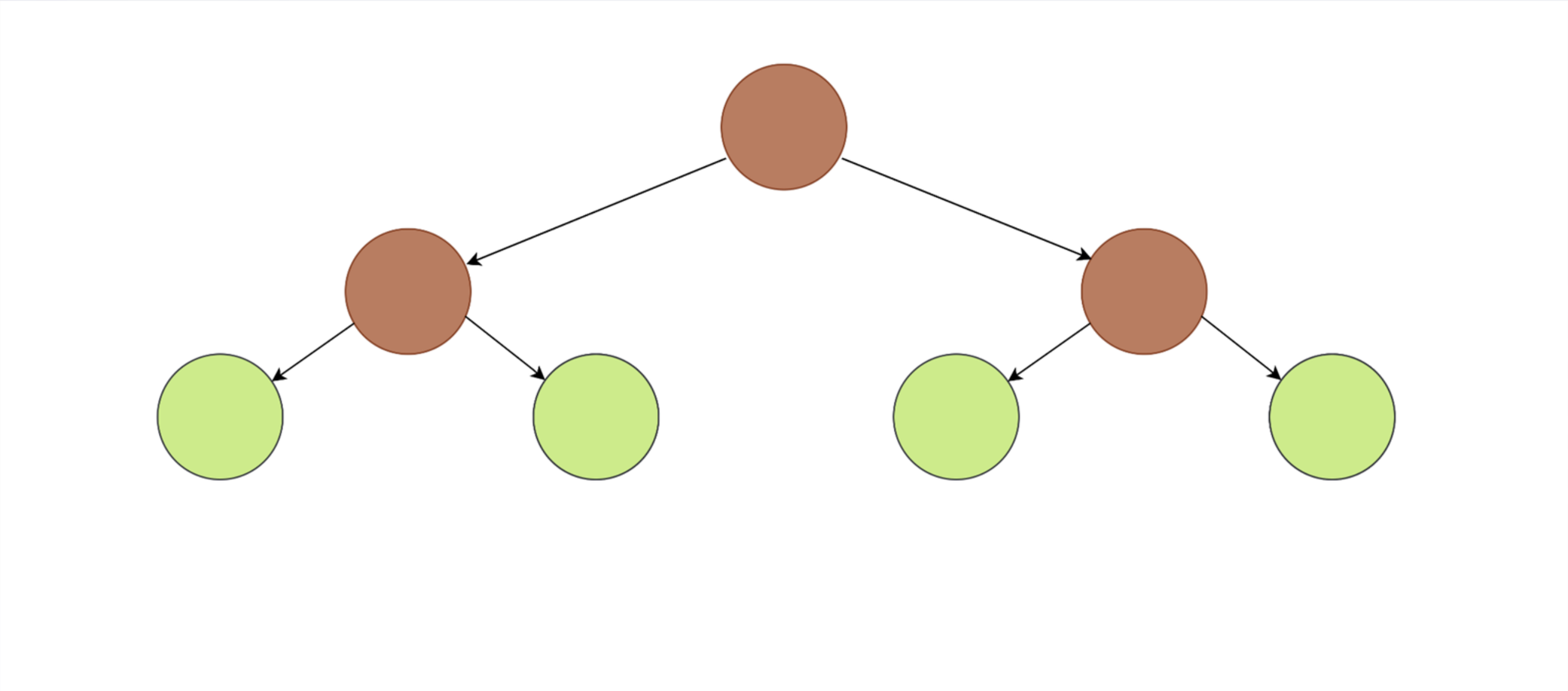
Best-first search



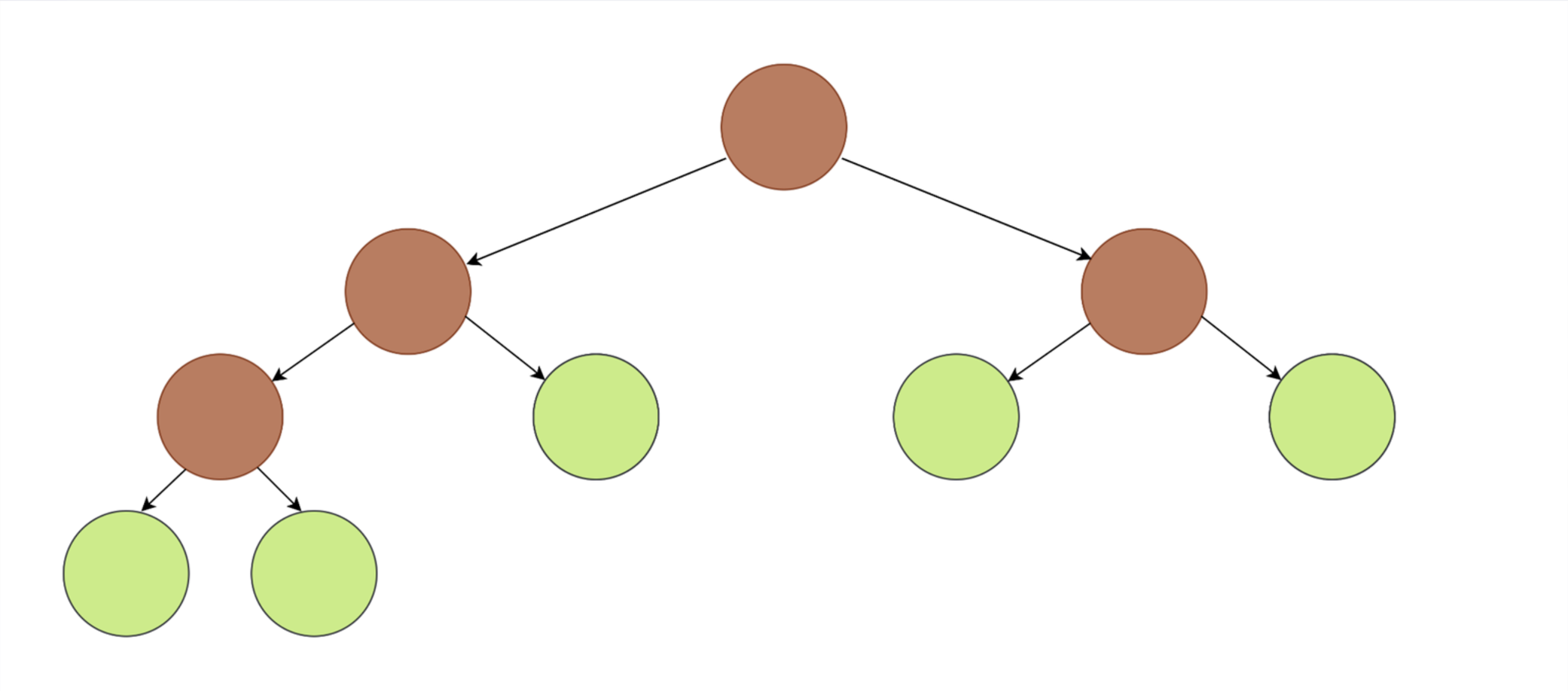
Best-first search



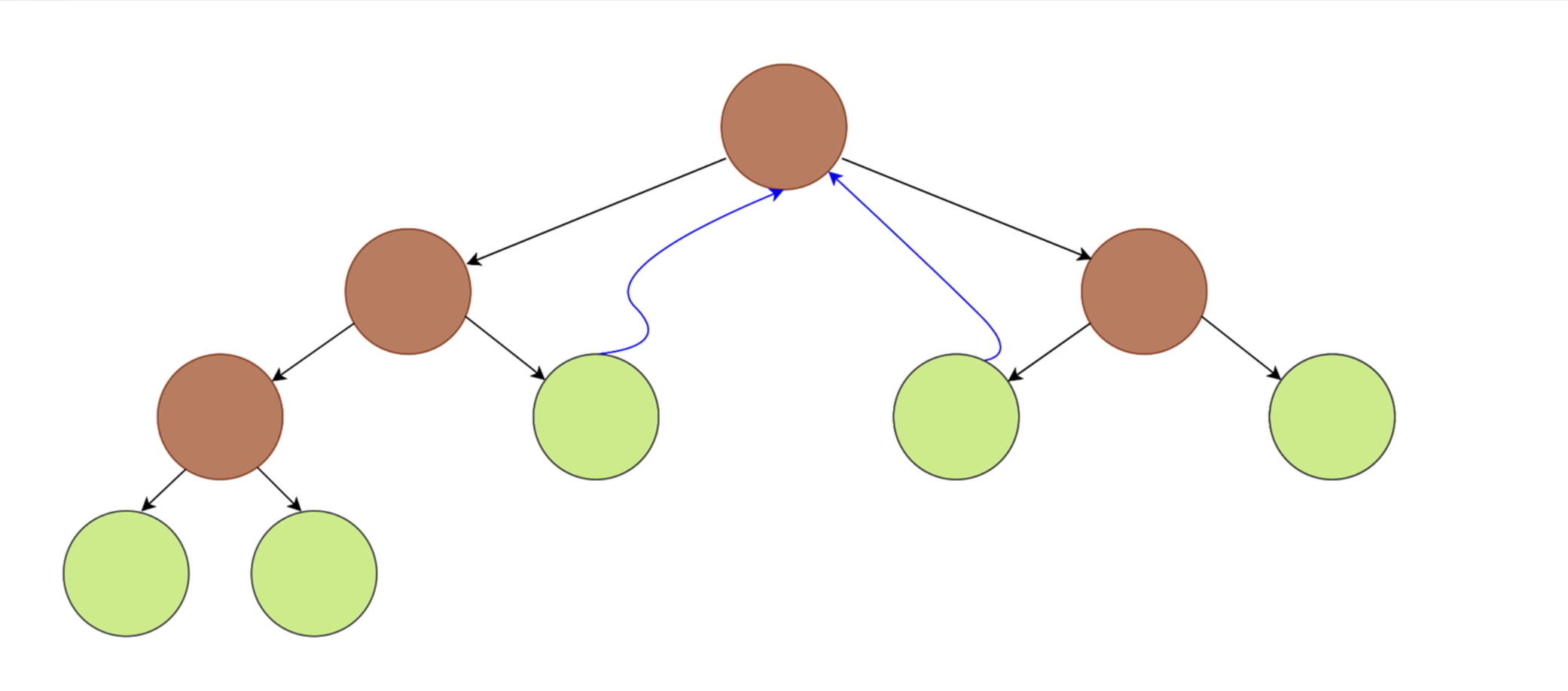
Best-first search



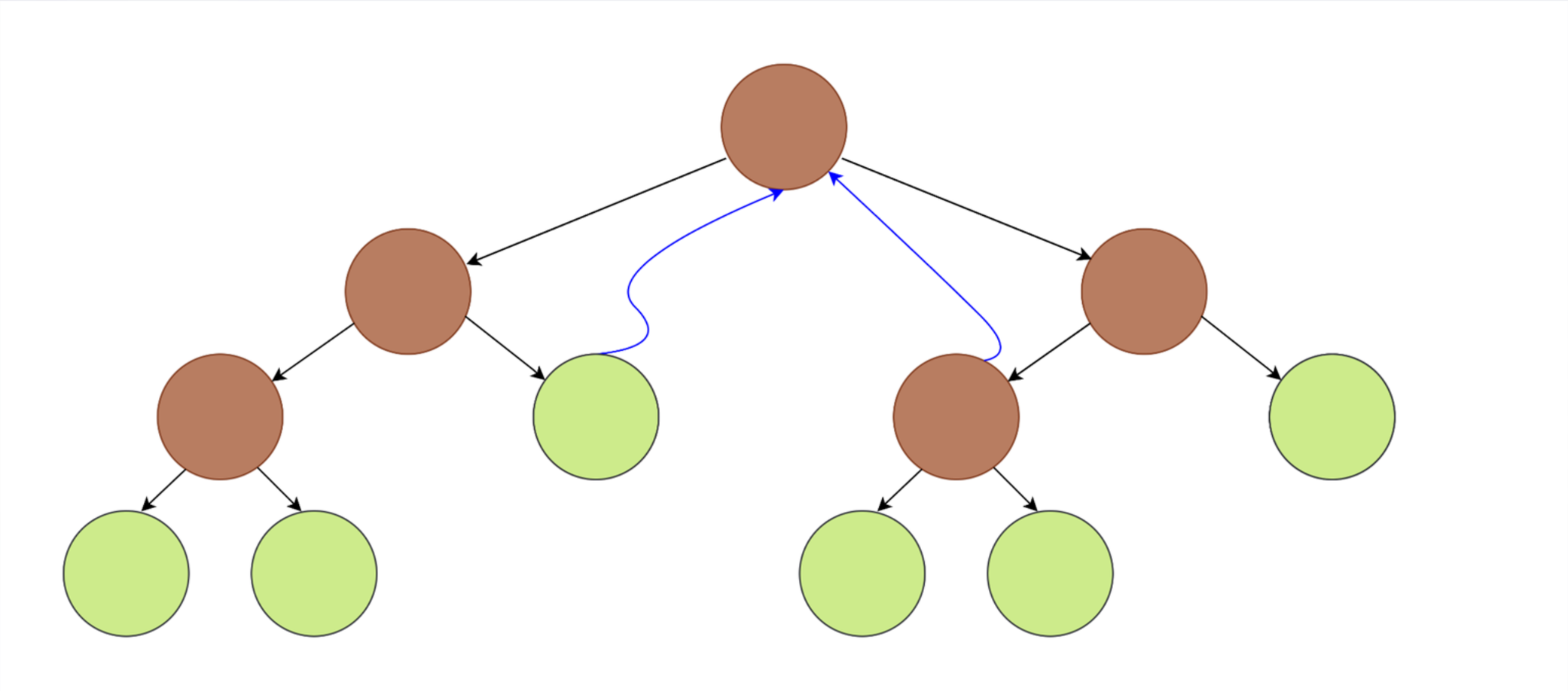
Best-first search



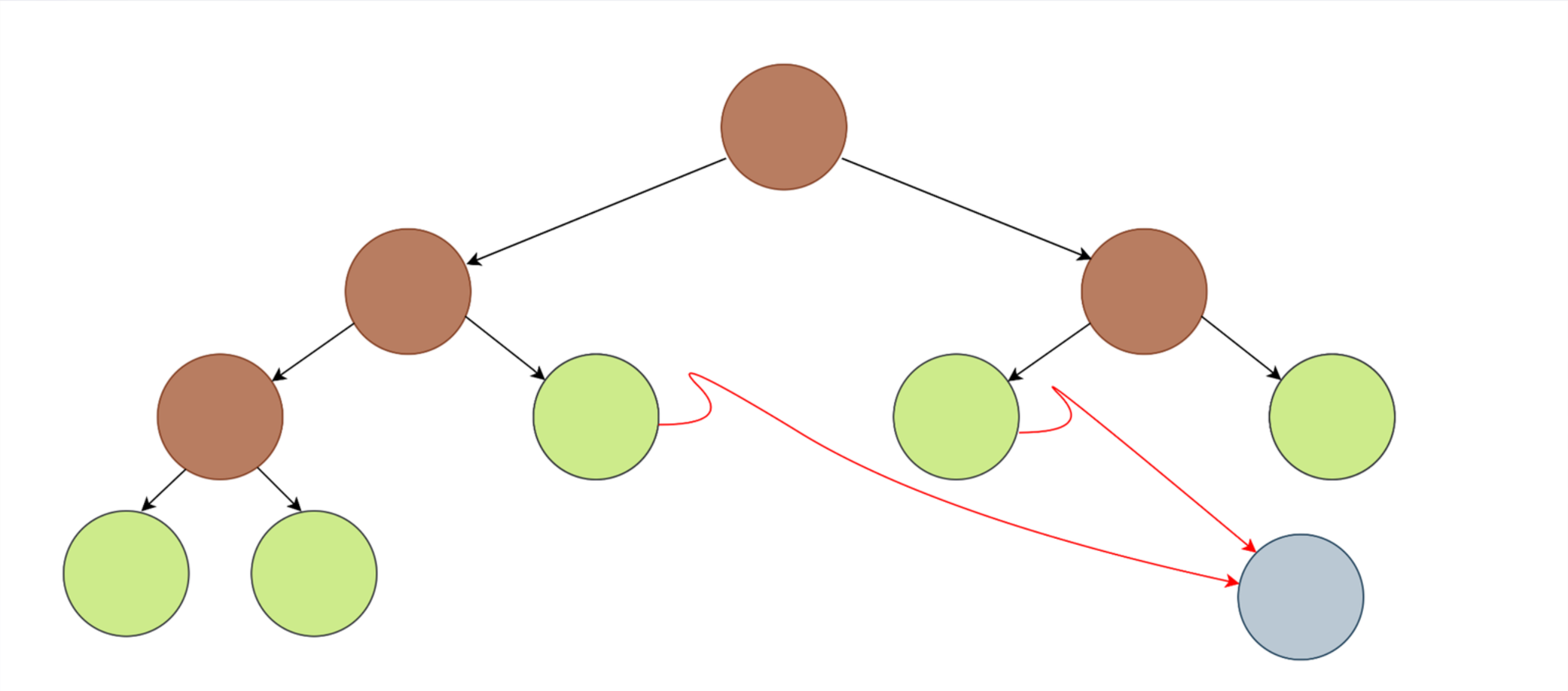
Best-first search: root distance



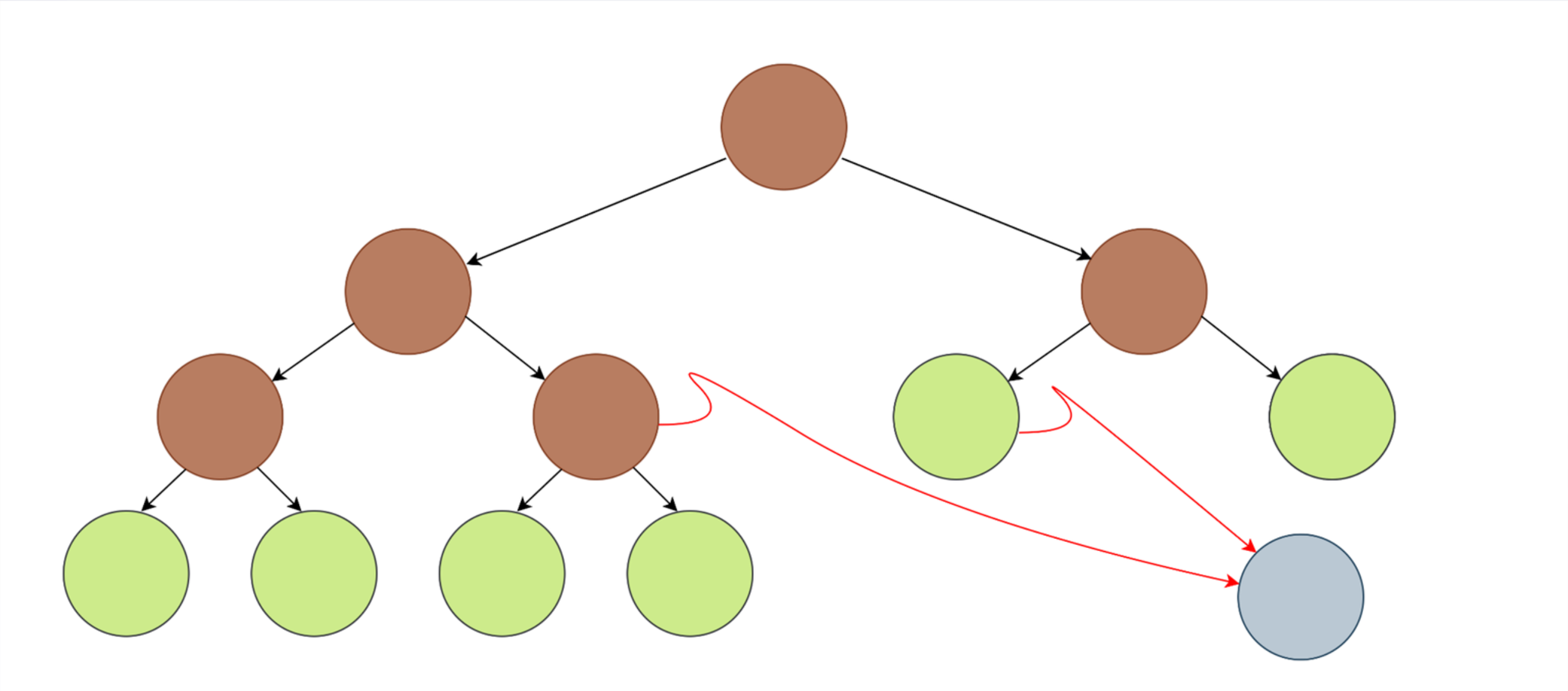
Best-first search: root distance



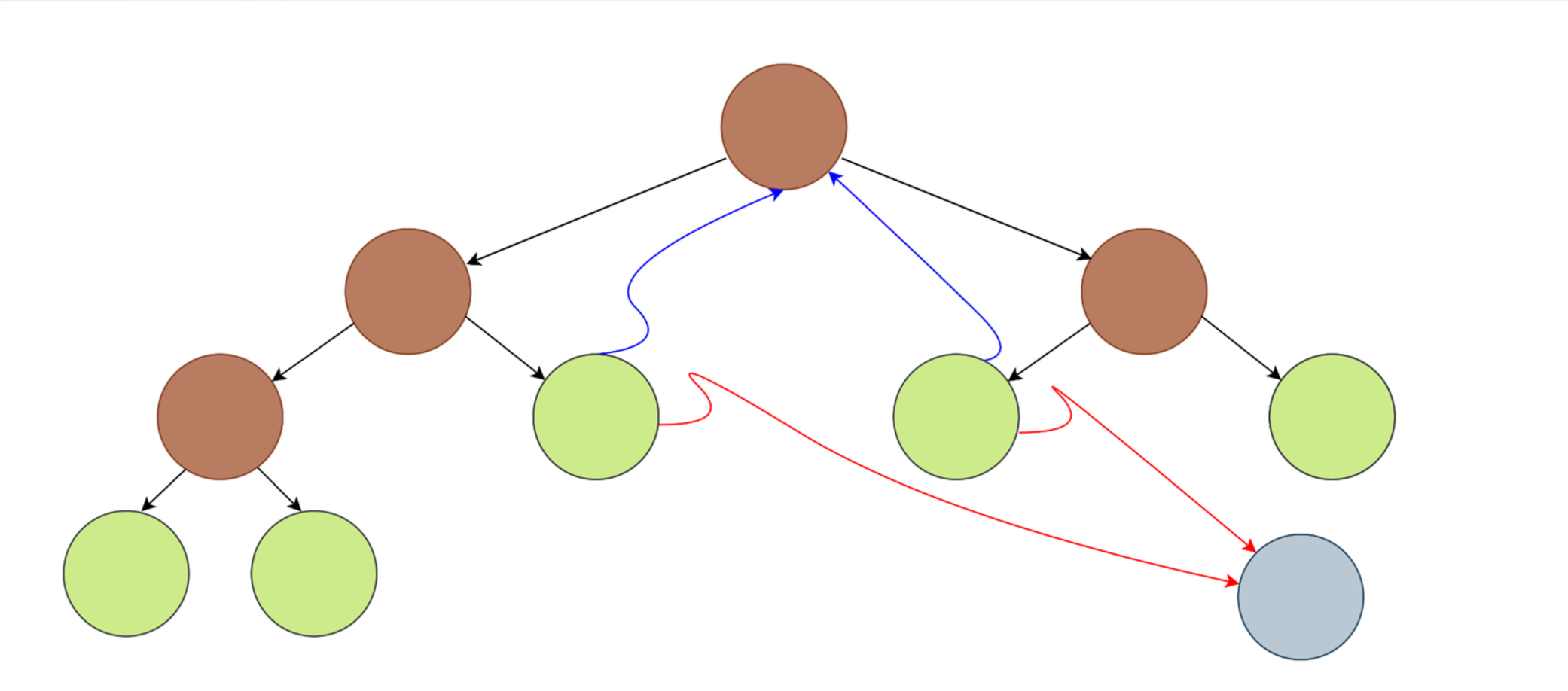
Best-first search: goal distance



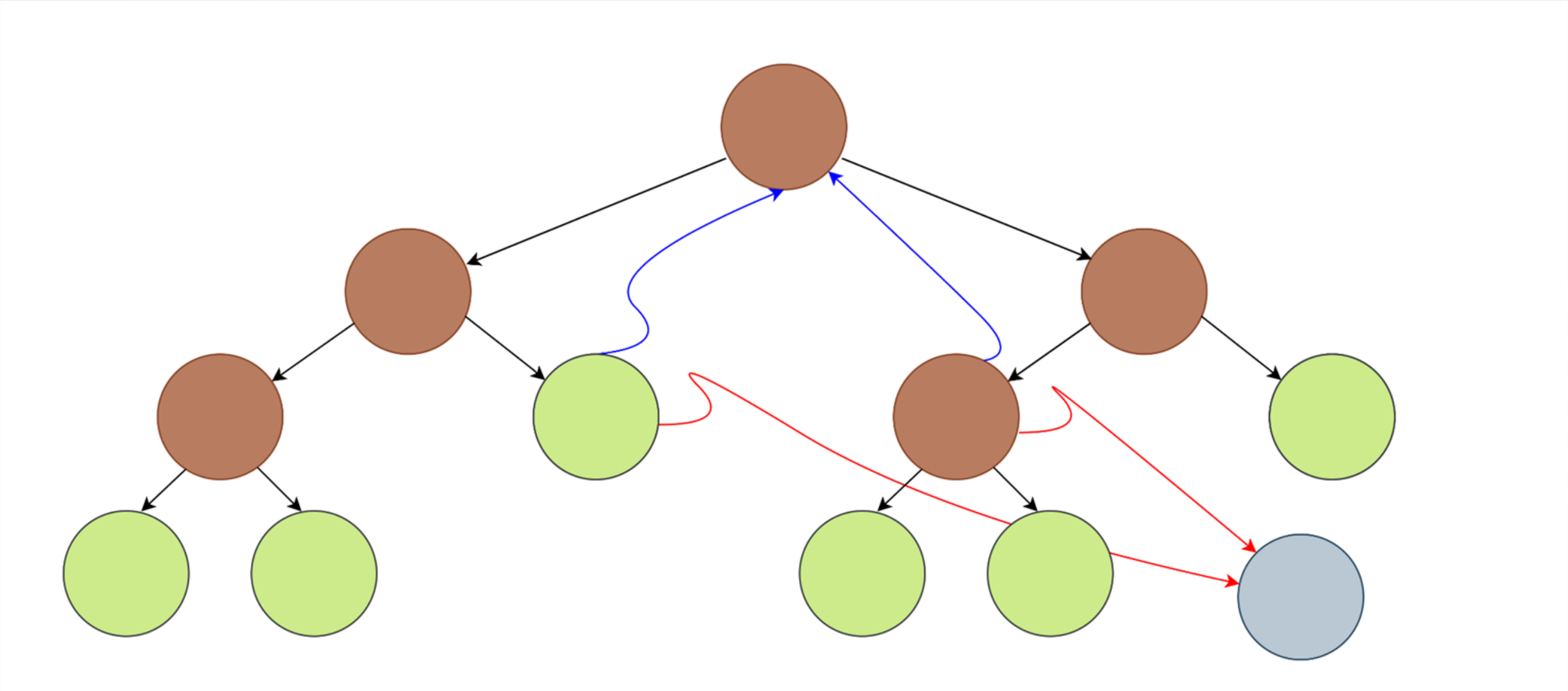
Best-first search: goal distance



Best-first search: balance

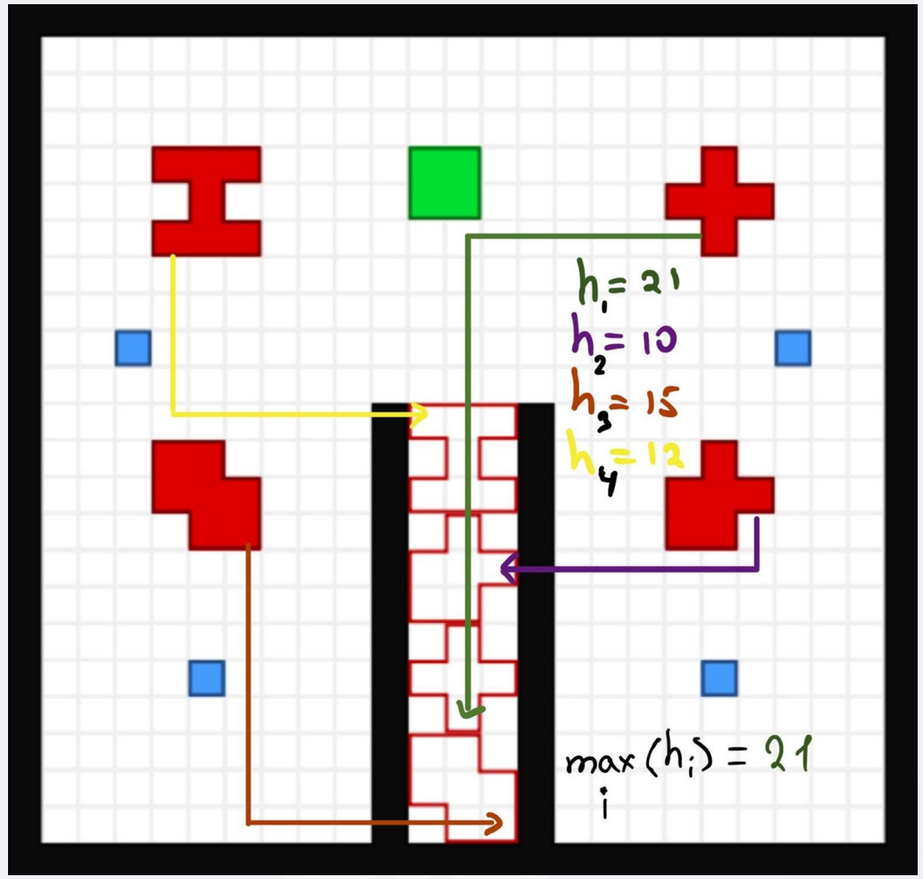


Best-first search: balance



В итоге

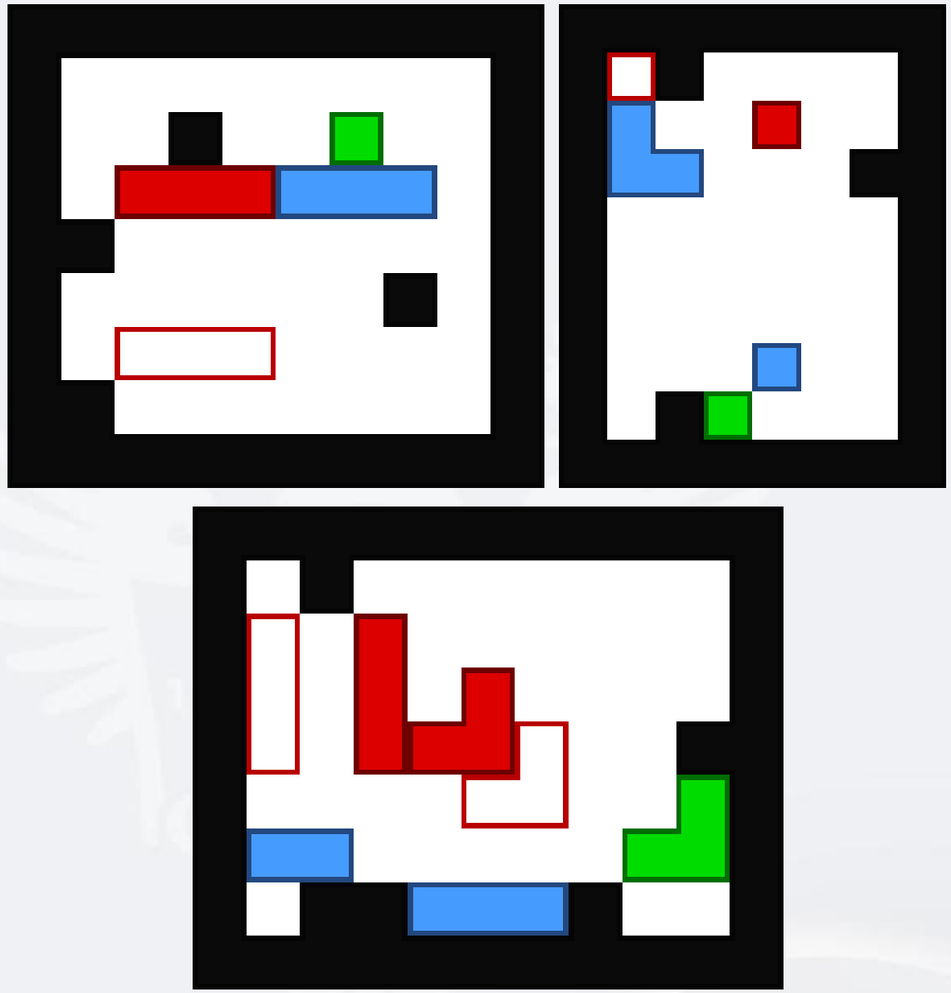
1. Dijkstra: на очередном шаге
    1. выбор argmin g(n) среди листьев.
       Если это t, то закончили.
    2. добавление потомков-листьев n
2. A*: на очередном шаге /-/, но argmin f(n).
   Откуда берется h?
3. WA* – вариация A*, где
   f(n) := g(n) + w * h(n)



max manh heuristic

Модель

- Вход: многоканальное (C x H x W) представление состояния задачи PushWorld
- 4 сверточных слоя: C→64→64→128→128, ядра 3x3, активация ReLU
- ≈ 280к параметров
- Softplus для неотрицательного предсказания числа шагов
- Обучение на авторском датасете Level 0.



Задания Level 0

Level 0

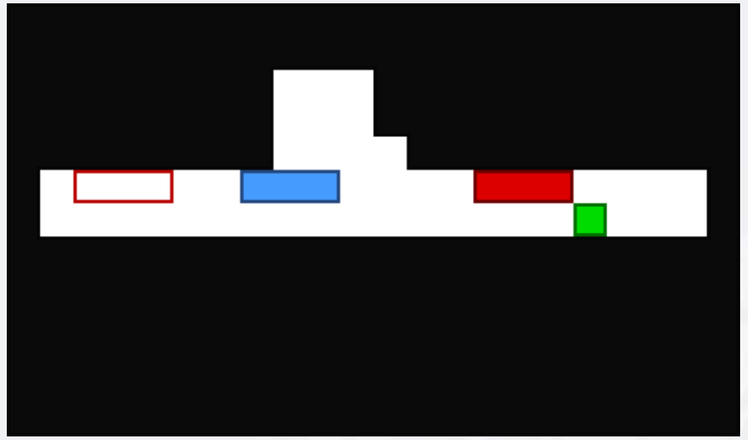
Algorithm	Solved (из 1400)	Avg exp	Avg time, s	Avg Len
A* + max	1386	960.96	0.055	**11.40**
A* + sum	1392	644.96	0.033	**11.40**
A* + nn	1380	**147.82**	0.126	11.58
WA*(5) + max	**1398**	292.63	**0.016**	11.68

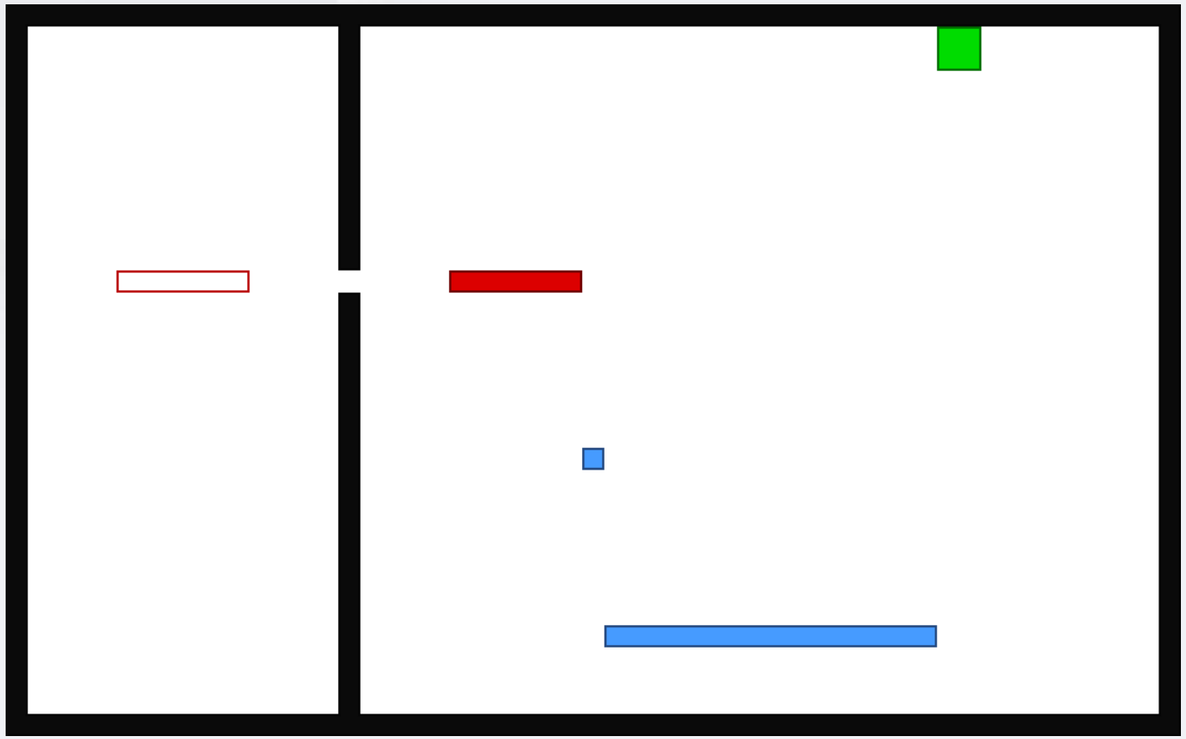
Level 1

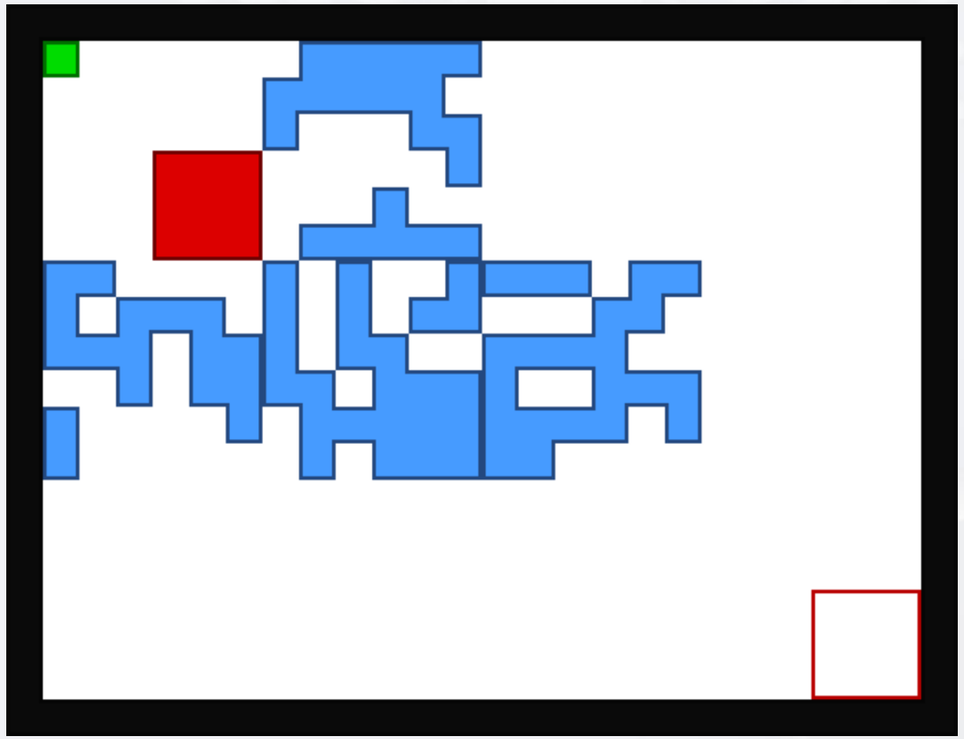
Algorithm	Solved (из 68)	Avg exp	Avg time, s	Avg Len
A* + max	50	2685.51	0.172	**21.10**
A* + sum	53	2424.05	0.159	**21.10**
A* + nn	39	**1059.23**	0.826	21.51
WA*(5) + max	**56**	1861.28	**0.097**	22.74

Улучшаем датасет









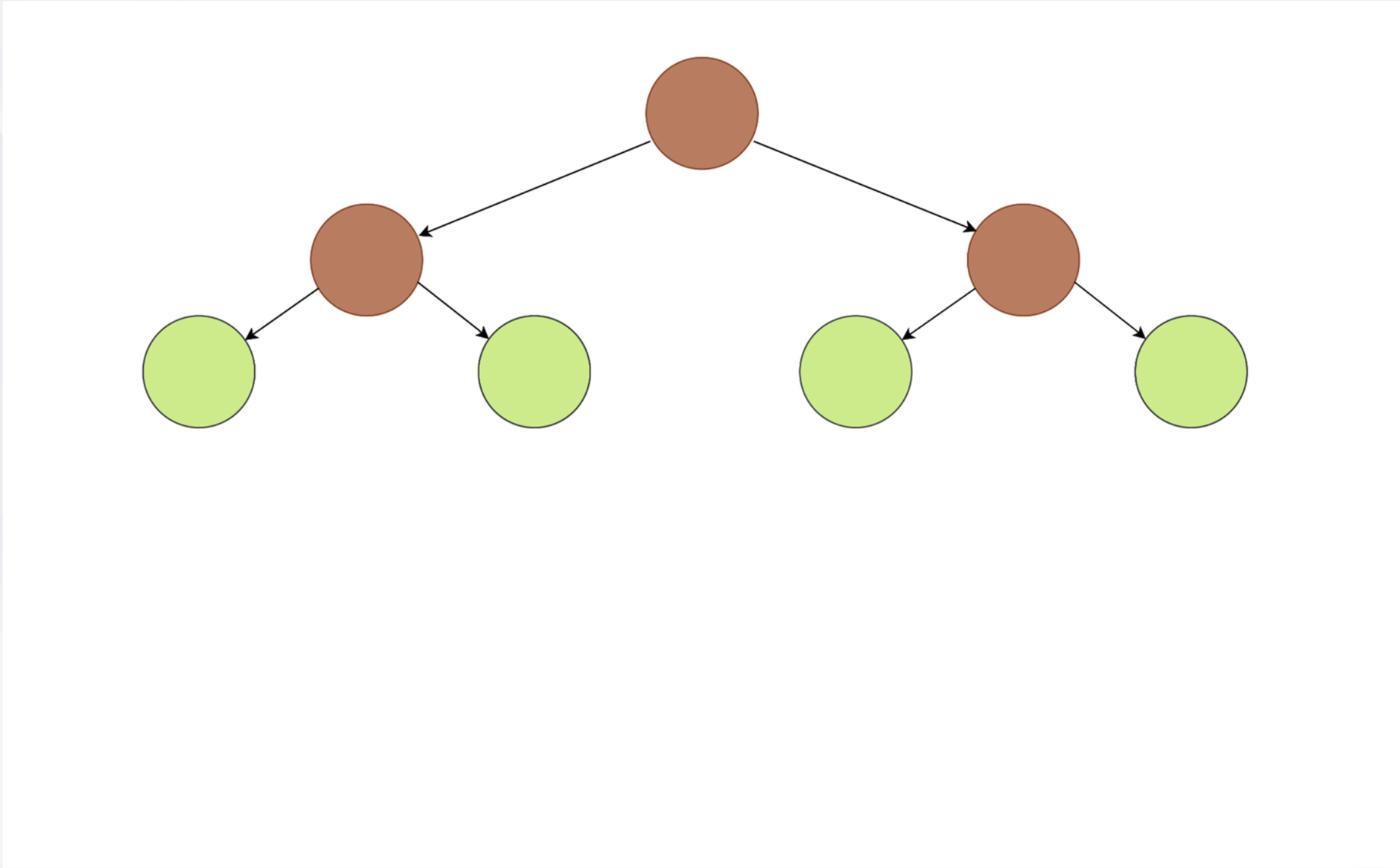
Level 1. Напоминание

Algorithm	Solved (из 68)	Avg exp	Avg time, s	Avg Len
A* + sum	53	2424.05	0.159	**21.10**
WA*(5) + max	**56**	1861.28	**0.097**	22.74
A* + nn	39	**1059.23**	0.826	21.51

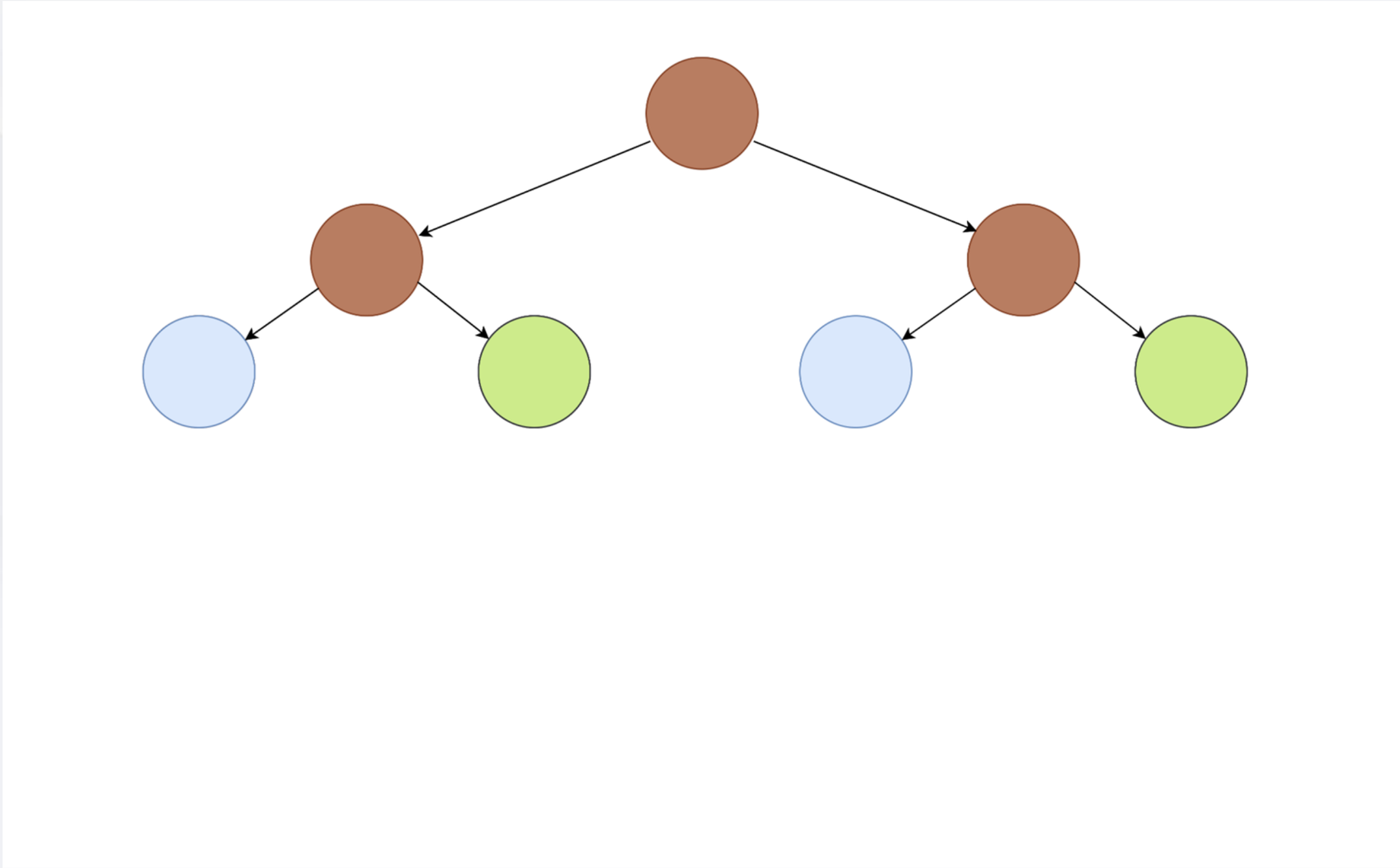
Level 1. new nn

Algorithm	Solved (из 68)	Avg exp	Avg time, s	Avg Len
A* + sum	53	2433.39	0.160	**20.97**
WA*(5) + max	**56**	1595.31	**0.080**	22.13
A* + nn	39	1019.63	0.762	21.39
A* + new nn	45	**675.42**	0.583	21.21

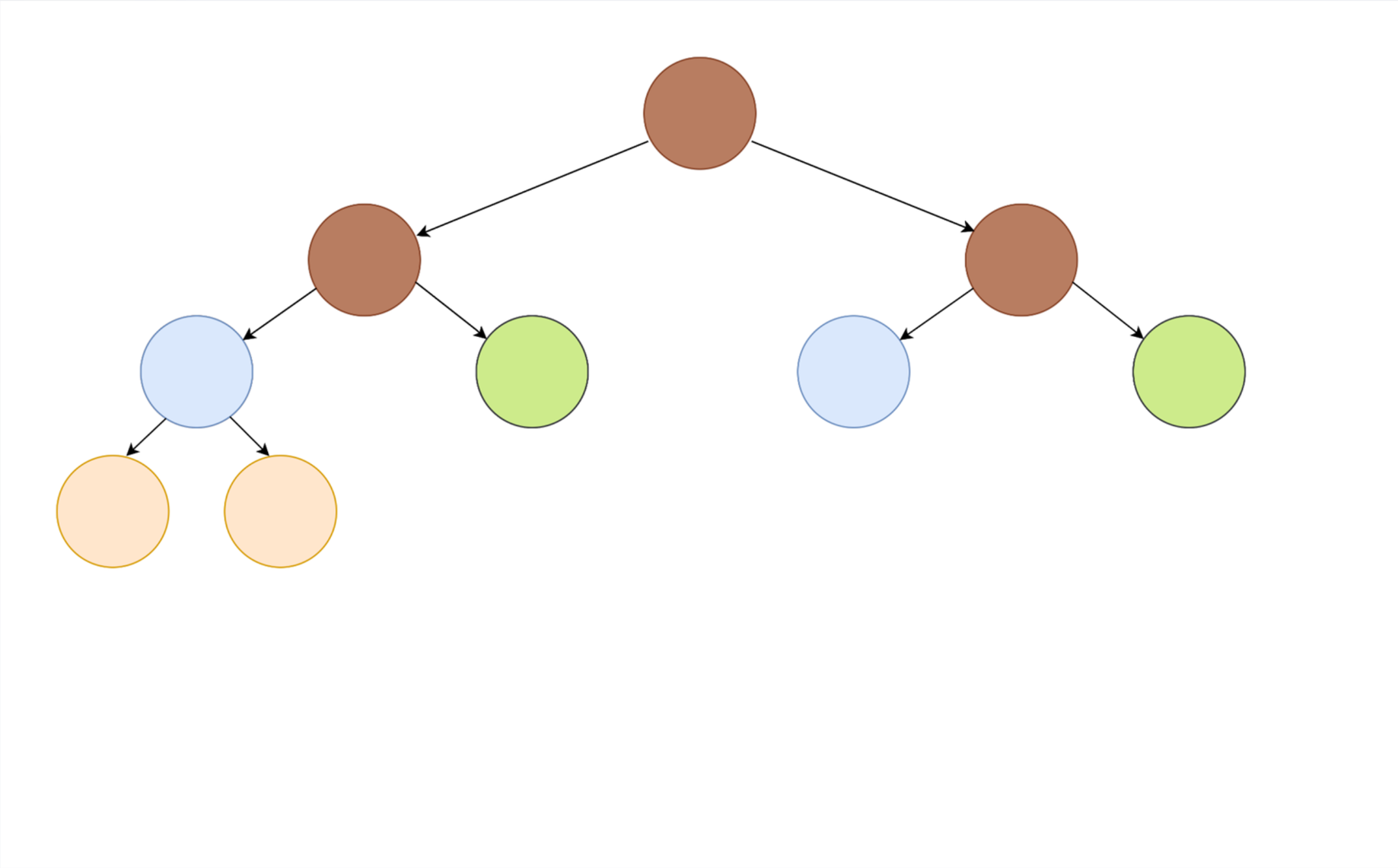
Qnn



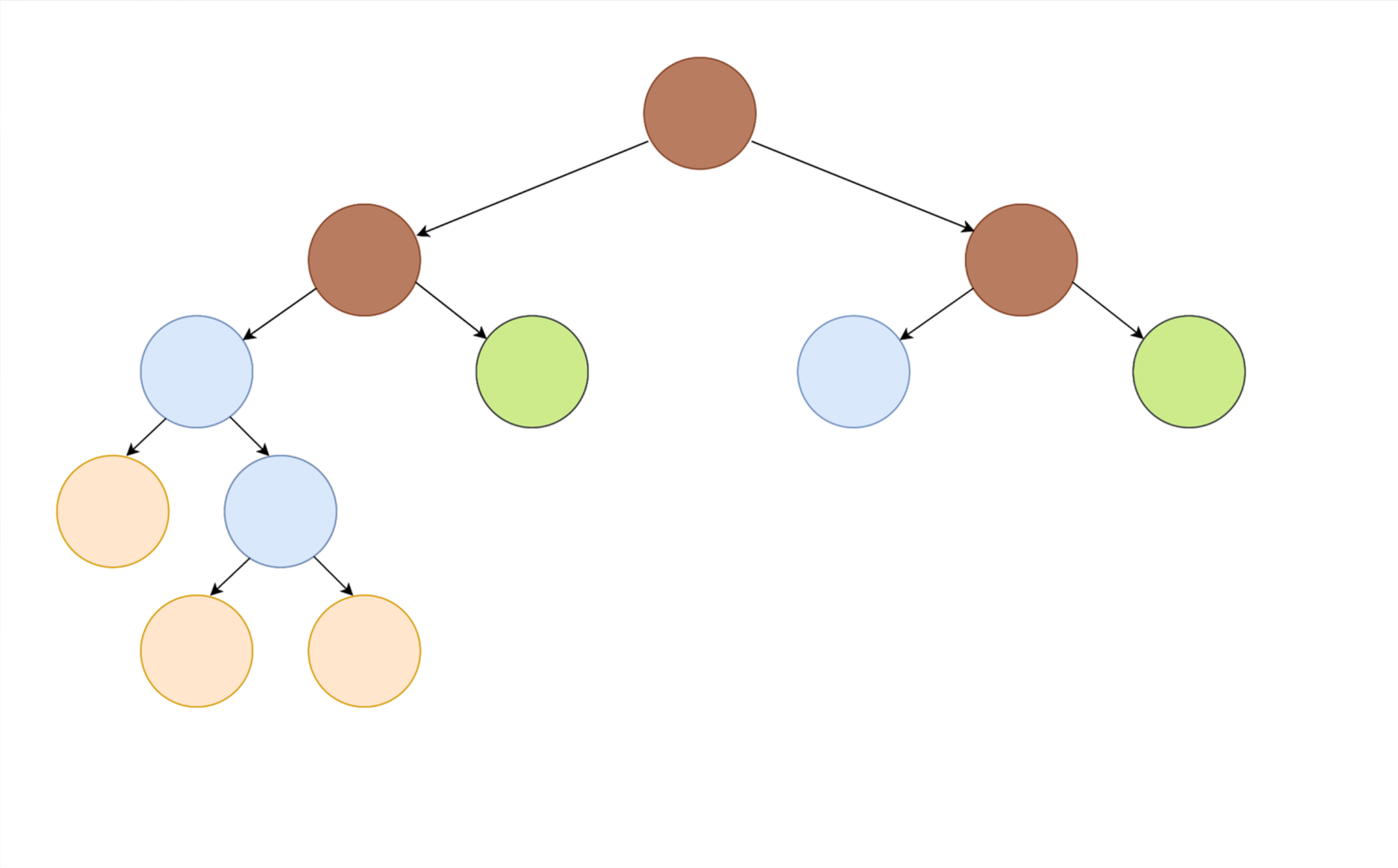
Qnn: manhattan exploration



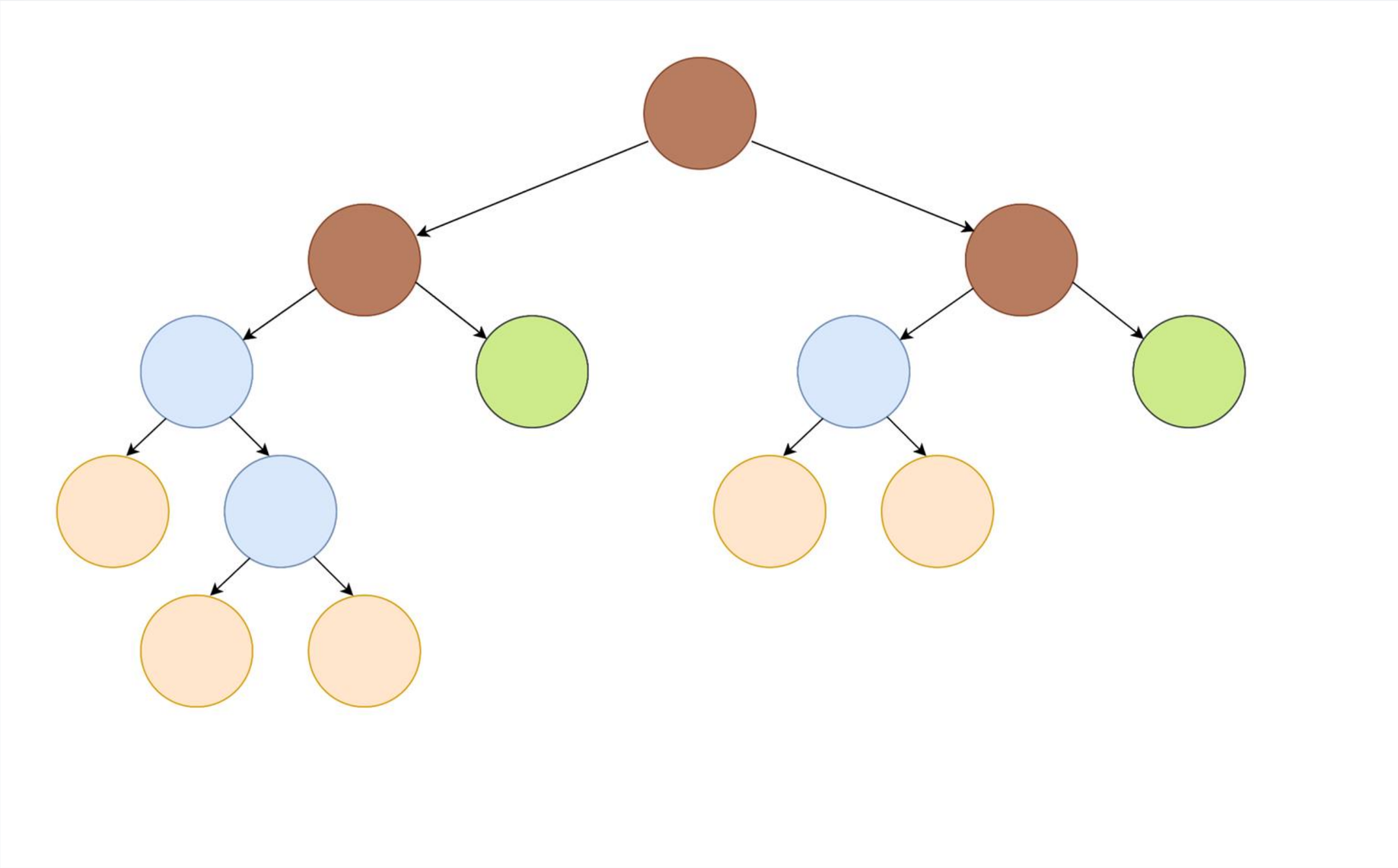
Qnn: manhattan exploration



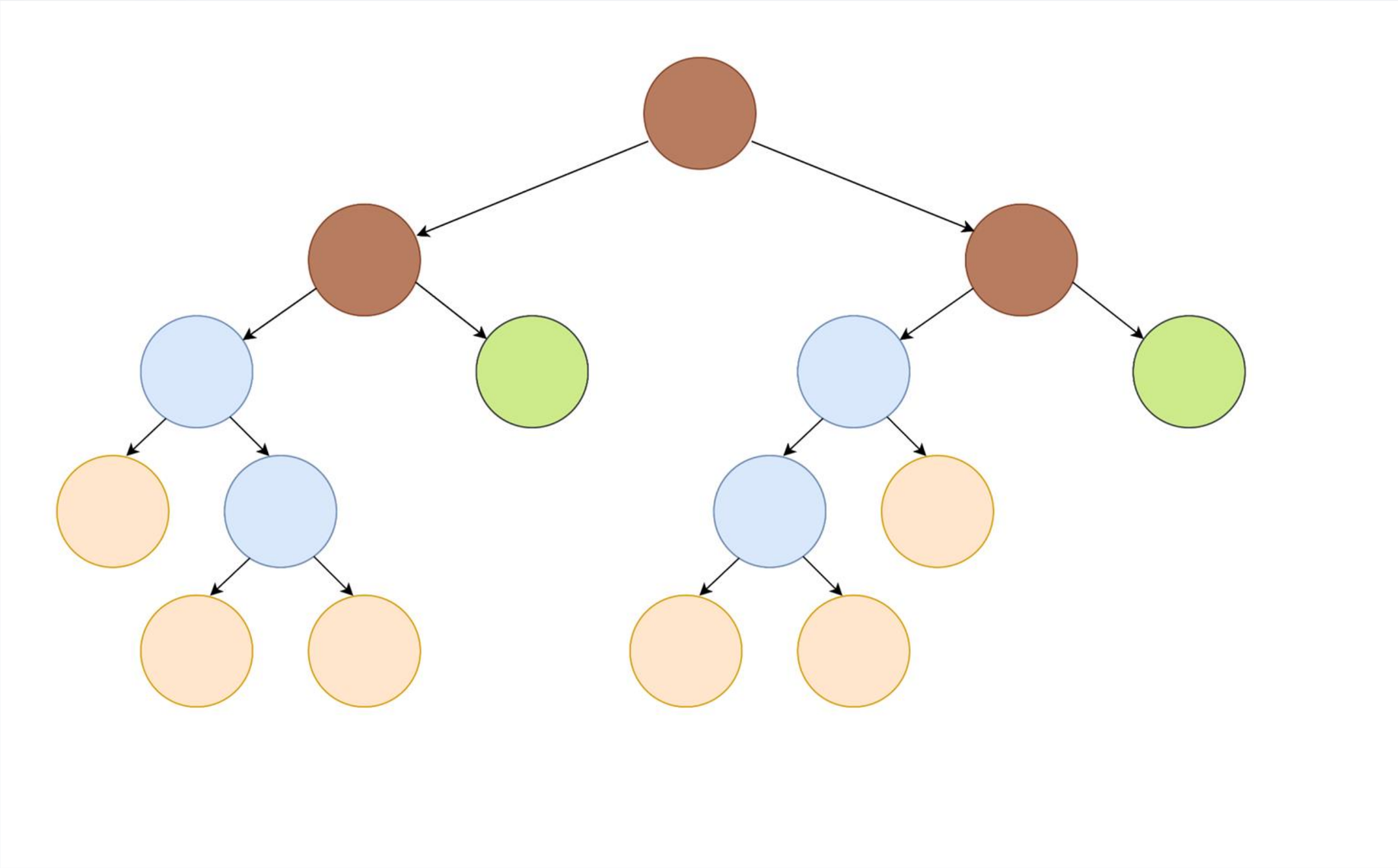
Qnn: manhattan exploration



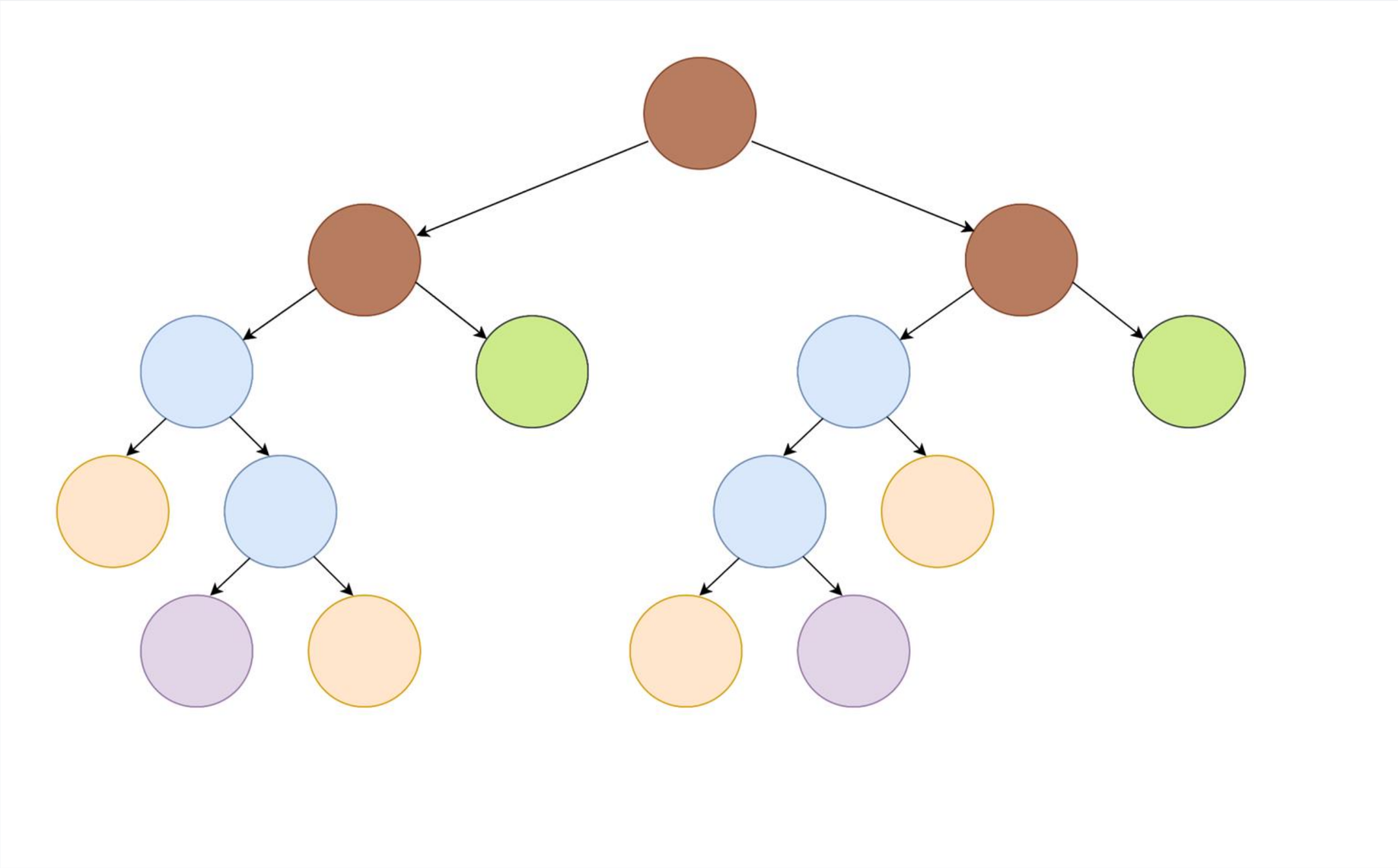
Qnn: manhattan exploration



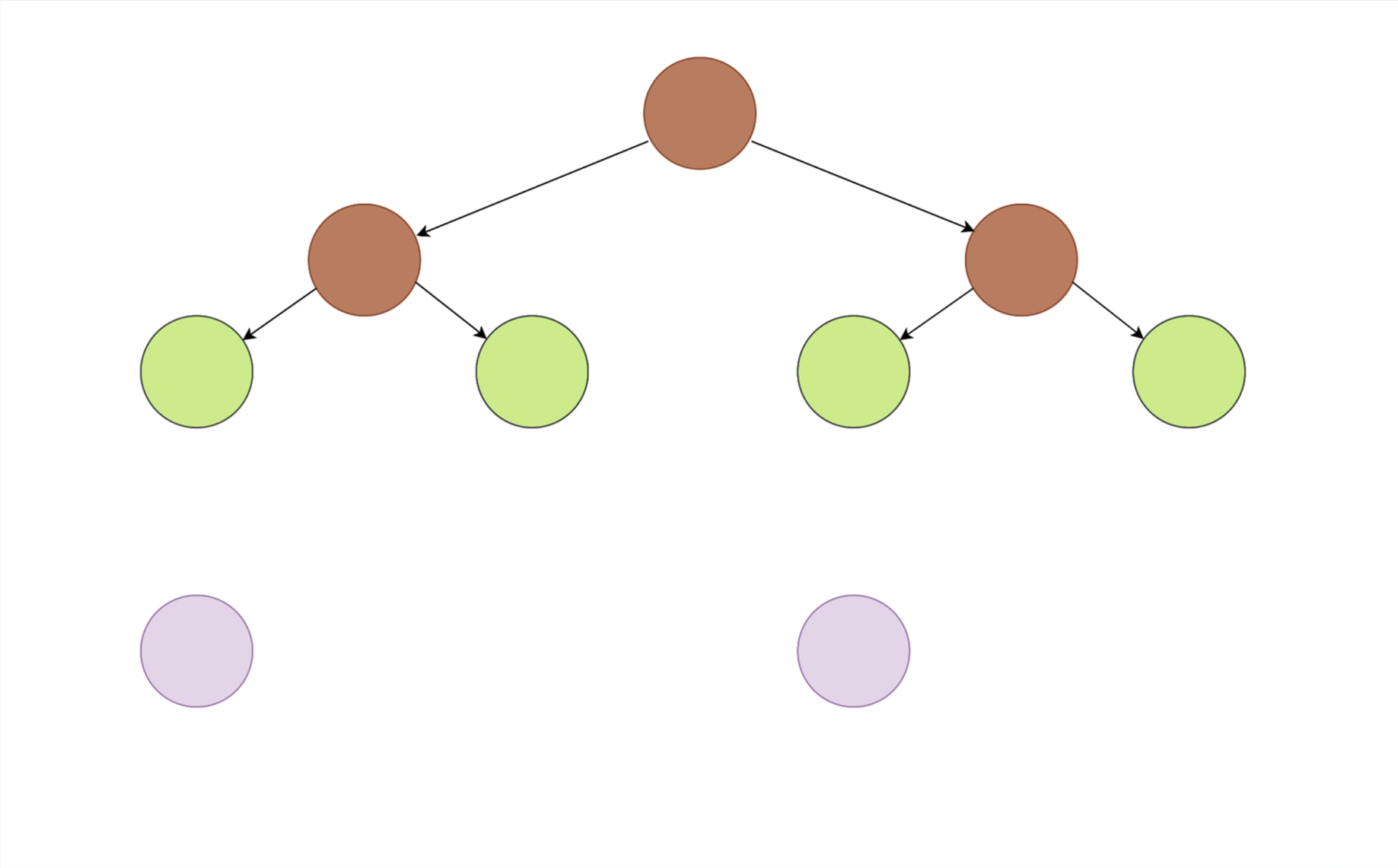
Qnn: manhattan exploration



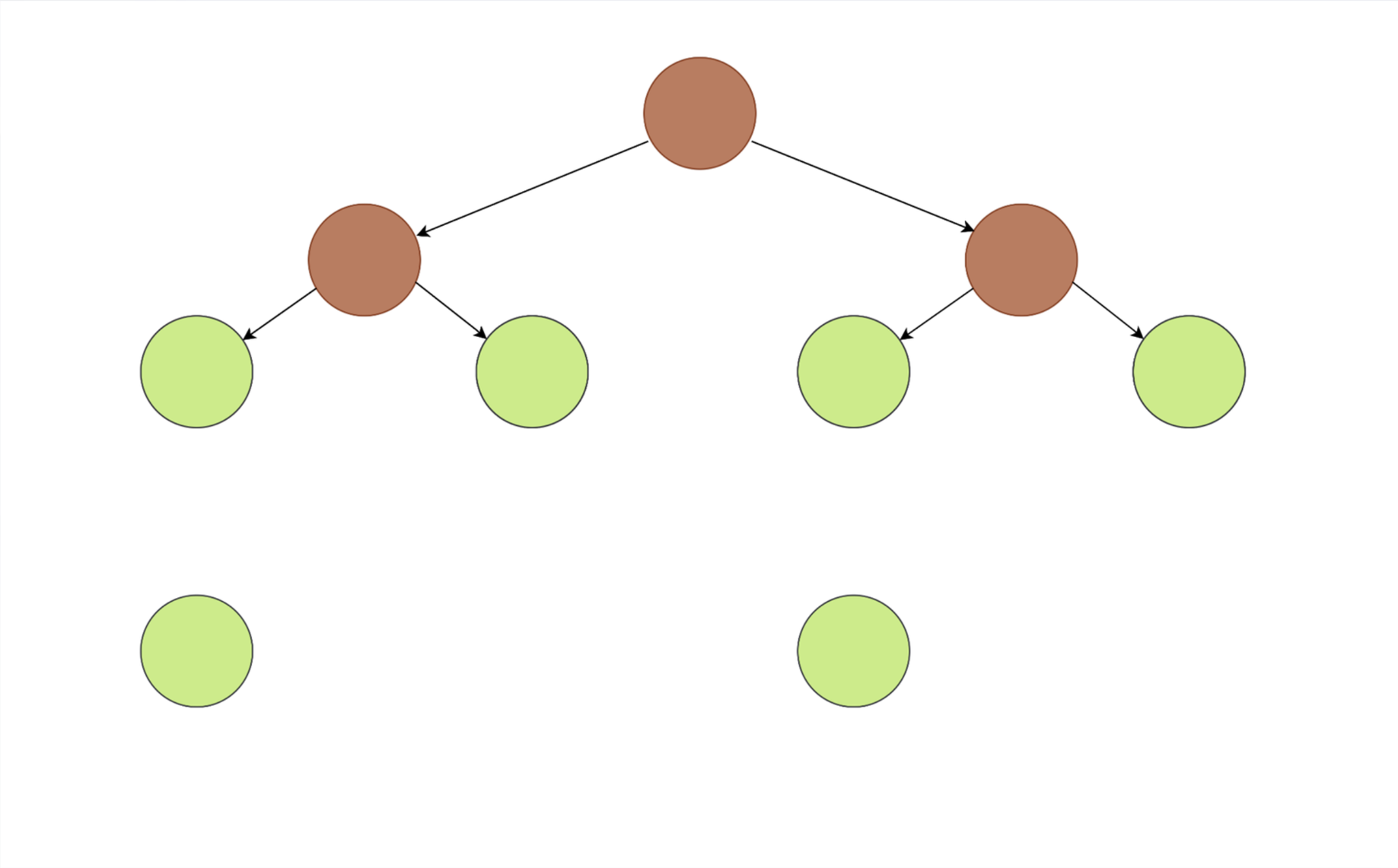
Qnn: manhattan exploration



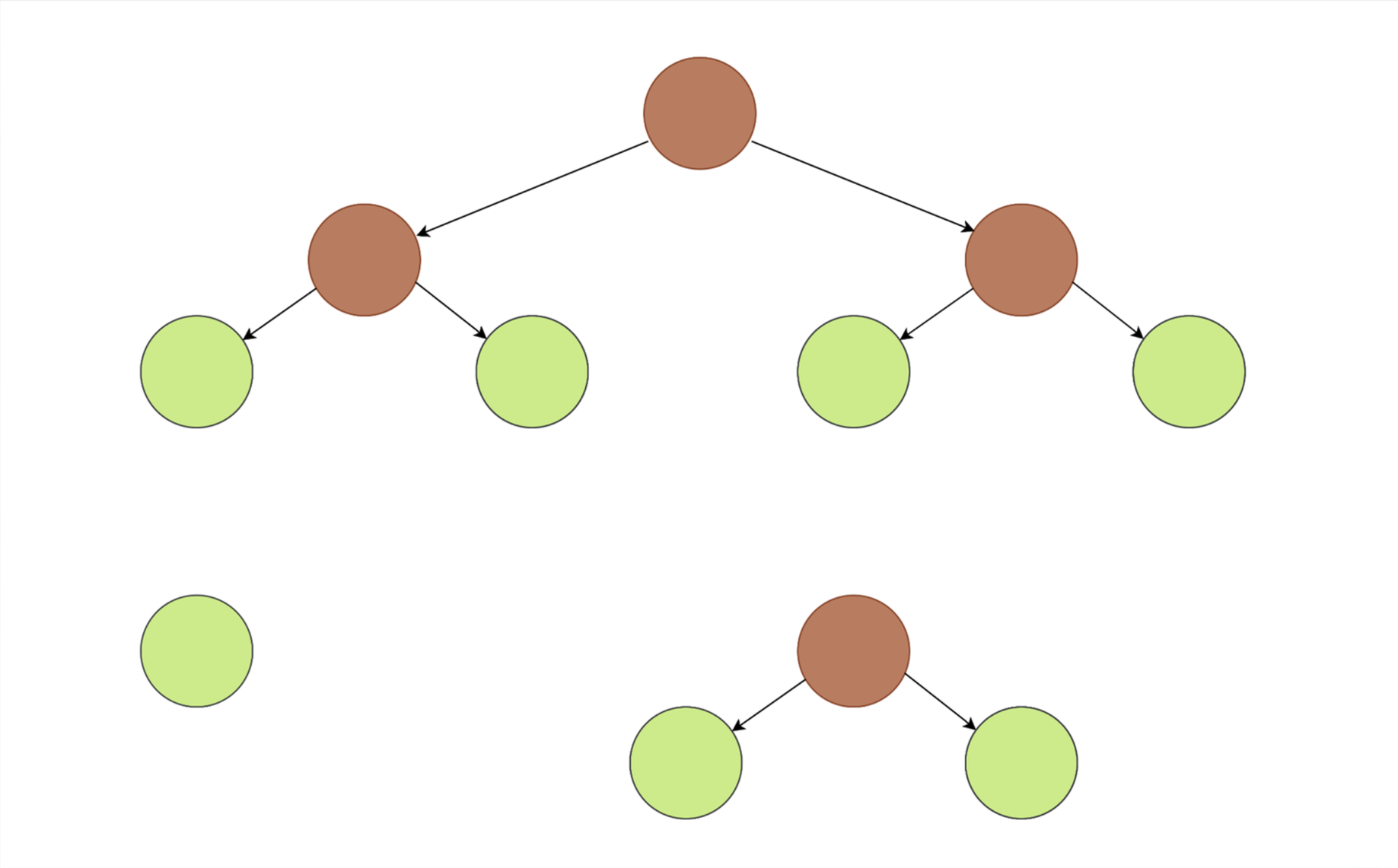
Qnn: manhattan exploration



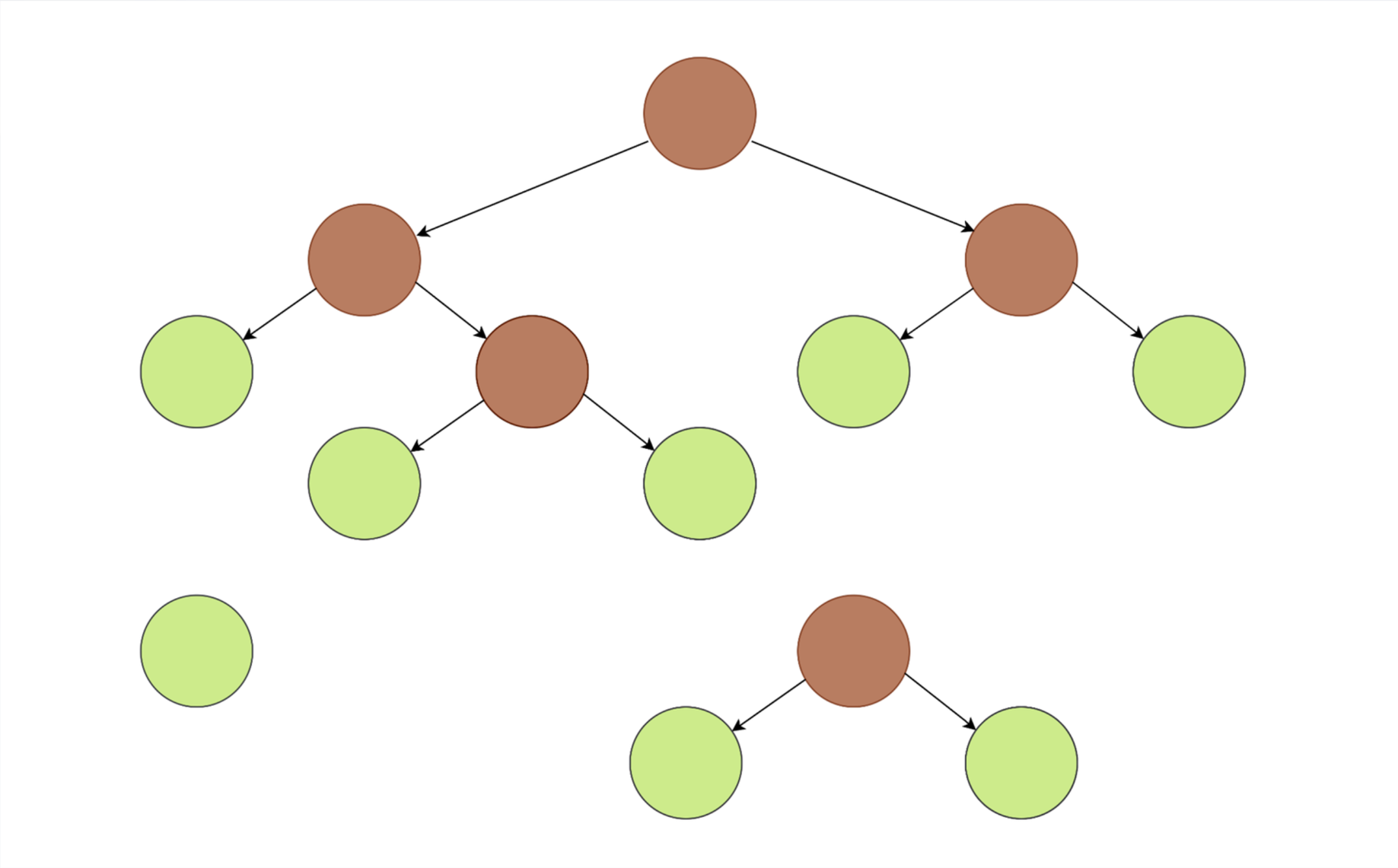
Qnn: neural exploitation



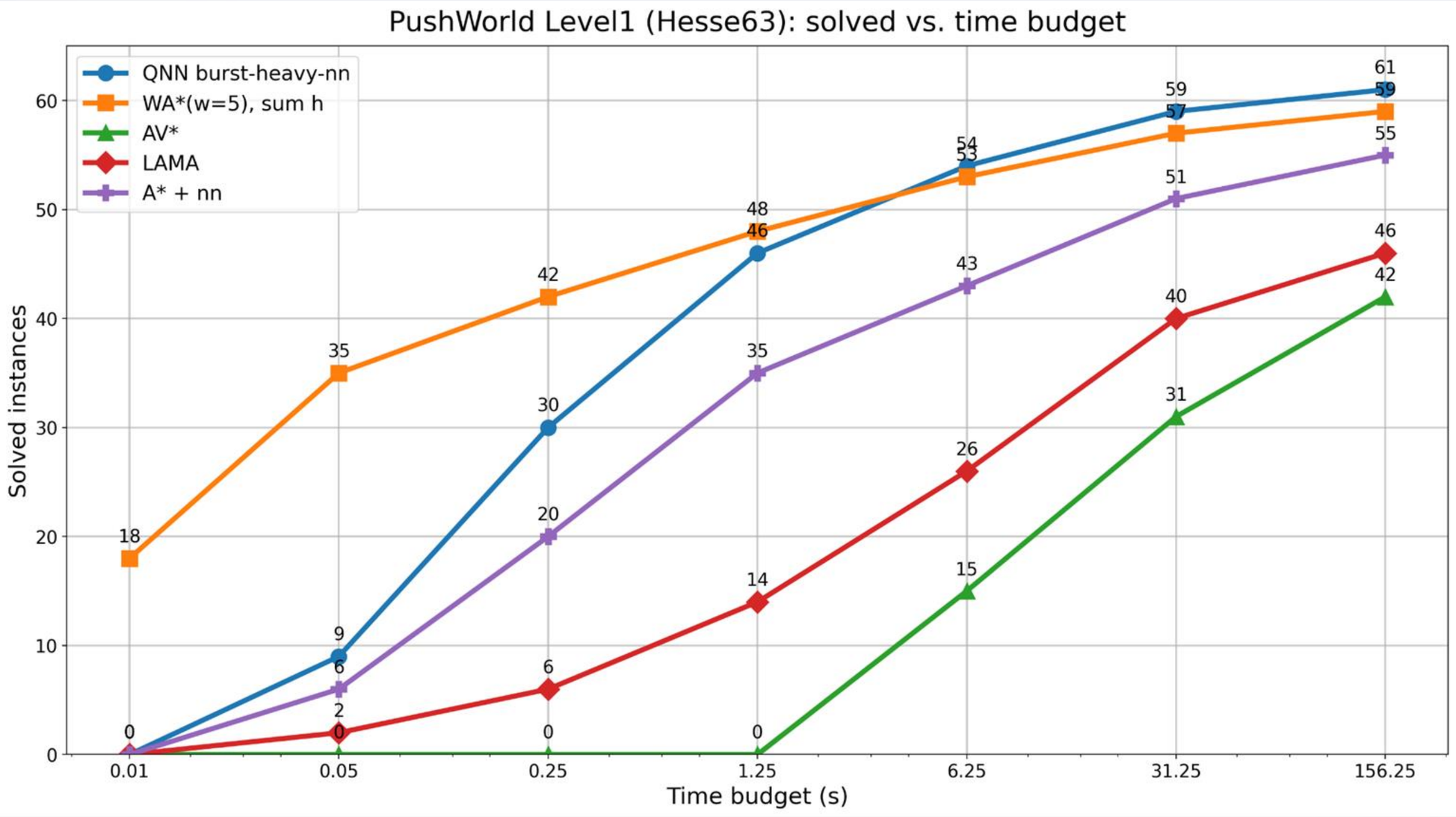
Qnn: neural exploitation



Qnn: neural exploitation



Level 1. Qnn и все-все-все



Выводы

- С ML-решателями Sokoban не все в порядке
- Есть опасения, что PushWorld тоже будет “конференционной лазейкой”
- Гики рулят