Методы и алгоритмы эвристического поиска

Jump Point Search и его модификации

Исполнители: Васильев Е.С., Денисов Н.В.

СПбГУ, 2025/2026

Актуальность эвристических алгоритмов поиска

Робототехника

Навигация и планирование движений автономных роботов в реальном времени, обход препятствий на производстве и в быту.

Игровая индустрия

Построение путей для неигровых персонажей (NPC), оптимизация перемещений в игровом мире, создание реалистичного поведения.

Беспилотный транспорт

Разработка маршрутов для беспилотных автомобилей и дронов с учетом дорожной обстановки, пробок и безопасности.

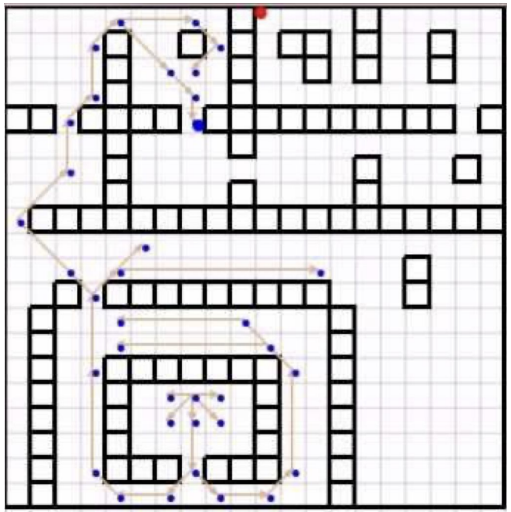
Склады и логистика

Оптимизация перемещения товаров на складах, эффективное планирование маршрутов доставки, сокращение времени выполнения заказов.

Постановка задачи: поиск оптимального пути (неформально)

Задача поиска пути на сетке является фундаментальной для многих приложений. Мы стремимся найти наилучший маршрут между двумя точками, учитывая особенности среды.

- **Входные данные:** Прямоугольная сетка с клетками, которые могут быть либо проходимыми, либо препятствиями. Сетка может быть 4-связной (только по сторонам) или 8-связной (по сторонам и диагоналям).
- **Взвешенный случай:** В более сложных сценариях разные типы местности (например, дорога, трава, болото) имеют разные стоимости прохождения, что влияет на общую стоимость маршрута.
- **Исходные данные:** Заданы стартовая точка **s** и целевая точка **t**.
- **Цель:** Найти маршрут из **s** в **t** с минимальной суммарной стоимостью. При этом важны не только оптимальность маршрута, но и скорость его нахождения.



Постановка задачи: поиск оптимального пути (формально)

Граф-сетка

- Сетка **H × W**
- Каждая клетка:
  - Проходимая
  - Препятствие
- Граф **G=(V,E)**:
  - V — множество проходимых клеток
  - E — допустимые переходы между соседними клетками (4- или 8-связность)

Стоимость переходов

- Для каждой клетки с задана **стоимость местности** t(c)>0

Uniform:

- Все проходимые клетки: t(c)=1
- Стоимость шага: 1 (ортогональ), √2 (диагональ)

Weighted:

- Клетки имеют разные стоимости t(c) (типы местности)
- Стоимость шага s->d:
  - |<s,d>|=|m|*t
  - |m| - длина шага (1 или √2)
  - t - взвешенное среднее стоимости клеток, пересекаемых шагом

Задачи и критерии

Найти путь (последовательность вершин от старта s из V до цели t из V) и минимизировать его суммарную стоимость при заданных стартовой и конечной вершинах.

Критерии успеха: оптимальность маршрута и **высокая скорость** работы алгоритма.

Базовый подход: A* и его ограничения

Алгоритм A* широко используется для поиска кратчайших путей благодаря своей оптимальности и простоте. Однако на больших сетках он сталкивается с рядом проблем, снижающих его эффективность.

Проблемы алгоритма A*

- **Симметричные пути:** A* исследует множество эквивалентных путей, ведущих к одной и той же вершине с одинаковой стоимостью. Это приводит к избыточной обработке узлов и замедляет поиск.
- **Высокий branch фактор:** у каждой вершины до 8 соседей и мы должны рассматривать каждую из них и обновлять очередь, что не является дешевой операцией.
- **Требования к памяти:** Необходимость хранить все посещенные узлы в структурах данных (например, CLOSED-список) значительно увеличивает потребление памяти, особенно на больших картах.

Jump Point Search (JPS): Основная идея (Harabor & Grastien 2011)

Jump Point Search (JPS) был разработан для ускорения A* путем устранения избыточных симметрий в сетке, сохраняя при этом оптимальность найденного пути.

Pruning

Локальное отбрасывание "симметричных" соседей, которые не могут лежать на уникальном оптимальном пути. Это позволяет значительно сократить число исследуемых направлений.

Jumping (перепрыгивание)

Вместо пошагового перемещения JPS "перепрыгивает" через блоки проходимых узлов до jump point. В OPEN попадают только jump point, что существенно сокращает количество операций с очередью.

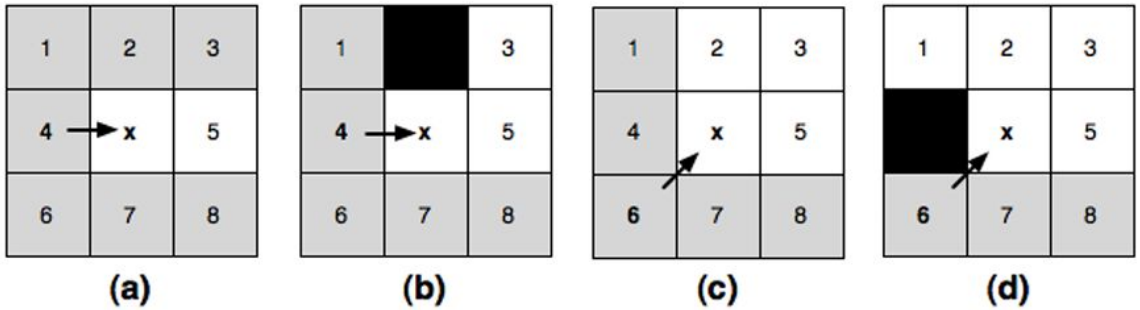
Ключевые понятия

- **Natural neighbours:** Соседи, которые остаются после pruning без препятствий.
- **Forced neighbours:** Соседи, которые должны быть рассмотрены из-за наличия препятствий, влияющих на оптимальный путь.
- **Jump points:** Узлы, где поиск может "ветвиться", либо где находится целевая точка. Это стратегические точки для продолжения поиска.

Pruning в JPS

- Рассматриваем узел x и родителя p(x)
- Смотри на 3x3 окрестность вокруг x
- Natural neighbour
  - Направления, которые остались бы без препятствий
  - Задаются только направлением из p в x
- Forced neighbour
  - Направления, которые стали необходимыми из-за препятствий
  - Иначе оптимального пути могло бы не существовать

Итог:
– сильно уменьшается число направлений, в которых продолжаем поиск
– при этом остальные оптимальные пути не теряются

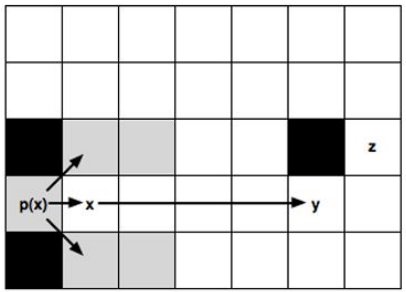


Jumping в JPS

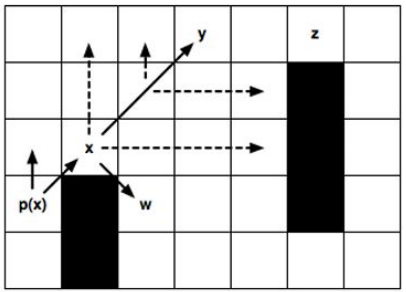
- Для каждого выбранного направления d
  - Идем от x по d
  - пока не:
    i. уперлись в стену
    ii. пришли в цель
    iii. встретили клетку с соседствующей FN
- Текущая вершина в ii и iii случаях - jump point
- Добавляем в OPEN

- Для диагонального d = h + w чуть хитрее
- Кроме перечисленного слева:
- Запускаем jump(n, h) и jump(n, w)
- Если вернуло не null, то помечаем как jump point

Эффект: уменьшается количество операций с очередью



(a)



(b)

JPSW объяснение

Pruning: (orthogonal last)

Дейкстра — сортировка по (g, |last_step|)

Jumping:

Если несколько типов клеток в окрестности 3x3, то это Jump Point

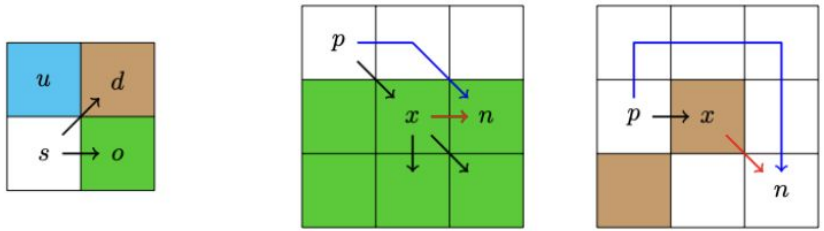


Figure 1: White has cost 1, green 2, brown 10, and cyan 0.1. (Left): The diagonal move $\langle s, d \rangle$ has cost $13.1/4\sqrt{2}$, and the orthogonal move $\langle s, o \rangle$ has cost 1.5. (Middle) and (Right): The blue path proves that the red move is unnecessary.

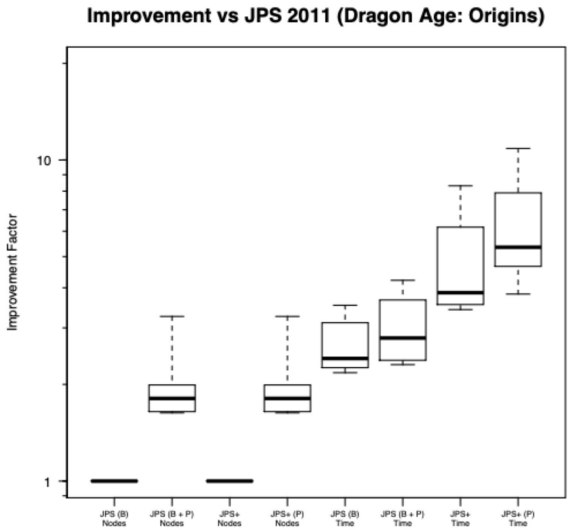
План экспериментов

1. Задачи
   a. Путь минимальной стоимости из s в t на Uniform Grid
   b. Путь минимальной стоимости из s в t на Weighted Grid
2. Данные
   a. Часть сетов из 2D Pathfinding Benchmarks (maze, random, room, terrain)
3. Бейзлайны
   a. A*
4. Метрики
   a. Время работы (предподсчет, онлайн)
   b. Затраты по памяти (предподсчет, онлайн)
5. Графики
   a. A* vs JPS в задаче Uniform Grid. По оси X — алгоритм, по оси Y — значение метрики, усредненное по 100 запускам, с доверительными интервалами. По одному графику на каждую метрику
   b. A* vs JPSW с теми же графиками.

Эксперименты:

1. JPS
2. JPSW

Ожидаемые графики



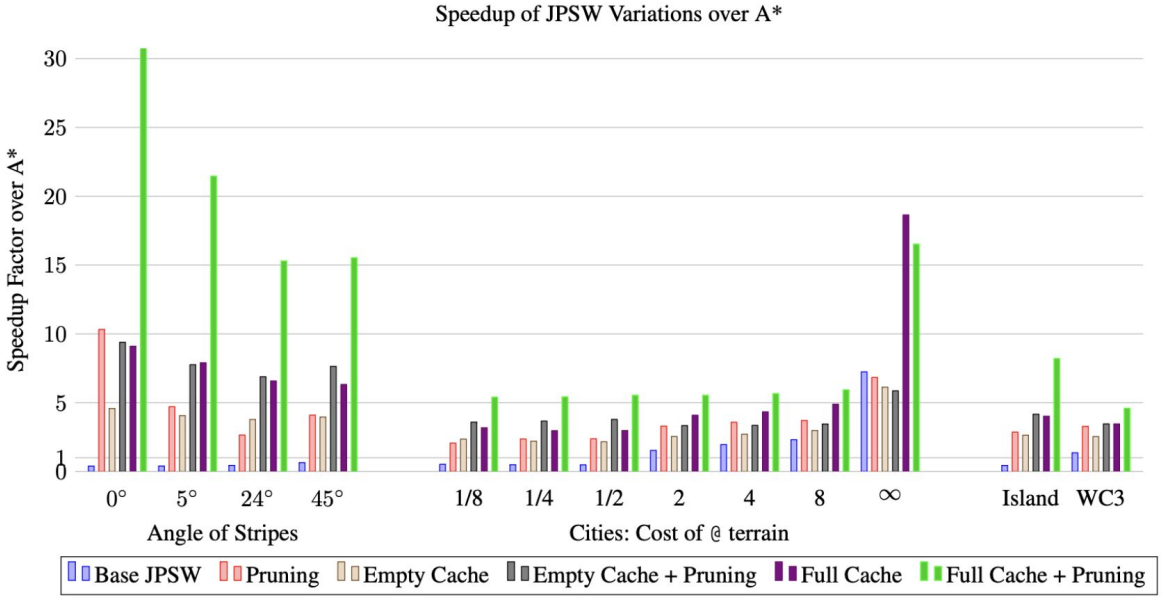
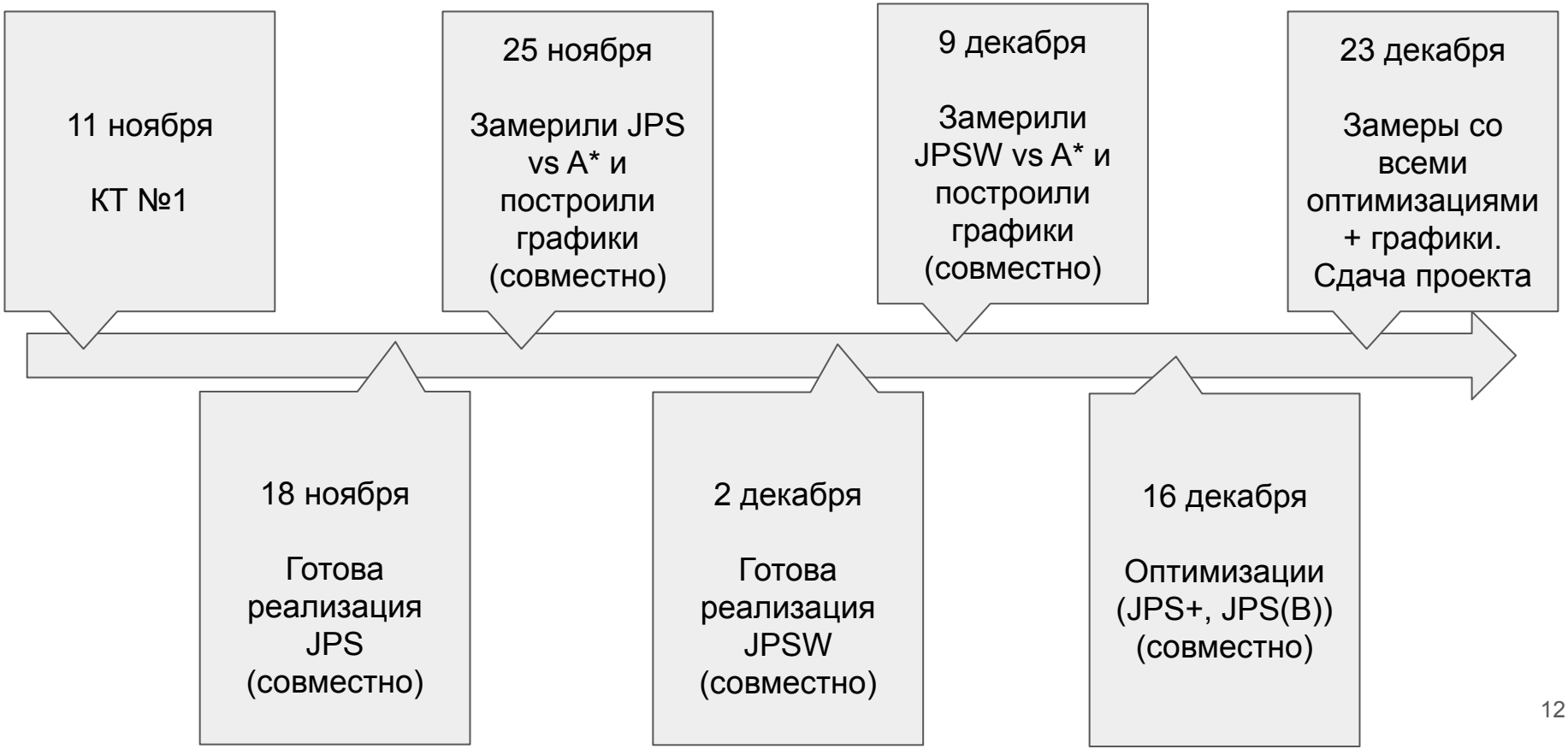


Figure 6: Speedups over A* for the Stripes, City Maps and Multi-terrain Maps for the various configurations

План работ


11 ноября
КТ №1
18 ноября
Готова реализация JPS (совместно)
25 ноября
Замерили JPS vs A* и построили графики (совместно)
2 декабря
Готова реализация JPSW (совместно)
9 декабря
Замерили JPSW vs A* и построили графики (совместно)
16 декабря
Оптимизации (JPS+, JPS(B)) (совместно)
23 декабря
Замеры со всеми оптимизациями + графики. Сдача проекта