Методы и алгоритмы эвристического поиска

Jump Point Search и его улучшения

КТ1 по проектной работе • СПбГУ • 2025/2026

Где применяются алгоритмы поиска пути?

Робототехника

Автономная навигация роботов в складских помещениях, на производстве и в общественных пространствах

Беспилотный транспорт

Планирование маршрутов для автономных автомобилей в городской среде и на магистралях

Логистика

Оптимизация перемещений в распределительных центрах и управление складскими операциями

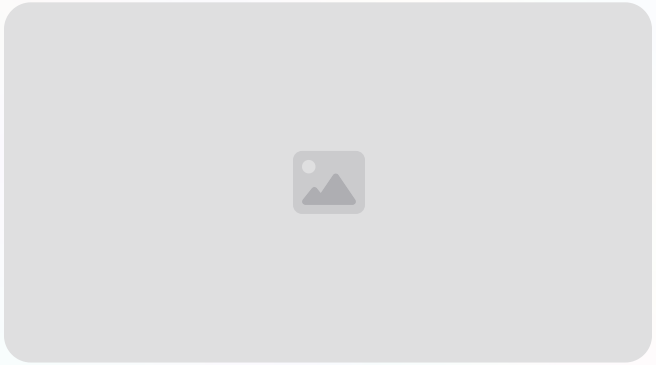
Игровая индустрия

Интеллектуальное поведение NPC персонажей и стратегическое планирование в реальном времени

Постановка задачи

Неформальная постановка

Необходимо найти кратчайший путь на решетке (grid map) между двумя точками, избегая препятствий. При этом важна не только оптимальность, но и **скорость поиска**.



Формальная постановка

Дано:

- Граф $G = (V, E)$, представляющий решетку
- Стартовая вершина $s \in V$
- Целевая вершина $t \in V$
- Весовая функция $w: E \to \mathbb{R}^+$

Найти: путь $P = (s, v_1, v_2, \ldots, t)$, минимизирующий $\sum w(e_i)$

Критерий: оптимальность + минимальное время работы

Проблема классического A*

Избыточность расширений

A* исследует множество симметричных путей, которые приводят к одной и той же точке с одинаковой стоимостью

Низкая производительность

На больших открытых пространствах алгоритм тратит значительное время на обработку узлов, лежащих на эквивалентных путях

Большие затраты памяти

Необходимость хранить все посещенные узлы в открытом и закрытом списках увеличивает требования к памяти

Jump Point Search: основная идея

Ключевой принцип

Вместо расширения каждой клетки «перепрыгиваем» через промежуточные узлы, останавливаясь только в **jump-точках** — местах, где:

- Достигнута цель
- Есть форсированный сосед (из-за препятствия)
- Найдена точка поворота на диагональном пути

Промежуточные узлы не добавляются в open list, что радикально сокращает число операций

Эволюция JPS: четыре улучшения

JPS (2011)

Оригинальный онлайн-алгоритм с рекурсивными прыжками на равновесных решетках

JPS (B)

Блочное сканирование строк через битовые операции — ускорение в 2-4 раза

JPS+

Предобработка: заранее сохраняем jump-точки для каждой клетки и направления

JPSW

Обобщение на взвешенные решетки с лексикографическим порядком

Технические детали улучшений

1

Блочное сканирование (JPS B)

Чтение целых машинных слов вместо проверки узлов по одному. Побитовые операции быстро находят «остановки» — повороты, тупики, форсированные соседи.

2

Предобработка (JPS+)

Для каждой клетки и 8 направлений заранее сохраняется первая достижимая jump-точка. Доступ к преемникам за O(1). Требует несколько МБ памяти.

3

Улучшенное отсечение

Промежуточные jump-точки (не у препятствий) не расширяются — их прямые преемники генерируются сразу.

4

JPSW: взвешенные решетки

Лексикографический тай-брейк (g, |последний ход|) + прореживание через мини-Дейкстру в 3×3 окне. Diagonal Branch Pruning против overscan.

Пример работы алгоритма

01

Старт из точки S

Определяем естественных соседей в направлении к цели T

02

Диагональный прыжок

Двигаемся по диагонали, проверяя ортогональные направления

03

Обнаружение форсированного соседа

Встречаем препятствие — это jump-точка, добавляем в open list

04

Продолжение до цели

Повторяем процесс, пока не достигнем T

План экспериментального исследования

Входные данные

- Benchmark-карты: Dragon Age, StarCraft, Baldur's Gate
- Синтетические решетки (размеры 512×512, 1024×1024)
- Multi-terrain карты с весами 1-5
- Городские карты (Moving AI Lab)

По 1000 случайных запросов на каждую карту

Baseline алгоритмы

- Классический A* (8-связная решетка)
- Canonical A*
- HPA* (иерархический метод)
- Subgoal Graphs

Метрики

- Время поиска (мс)
- Число расширенных узлов
- Объем памяти (для предобработки)
- Время предобработки (для JPS+)
- Speedup относительно A*

Ожидаемые результаты: ускорение JPS в 5-10× над A*, JPS+ до 15× при приемлемых затратах памяти (10-50 МБ)

Распределение работ в команде

1

Недели 1-2: Изучение теории

Все участники: детальный разбор статей, формализация алгоритмов

2

Недели 3-4: Реализация

Участник 1: JPS базовый + JPS(B)

Участник 2: JPS+ и предобработка

Участник 3: JPSW и взвешенные решетки

3

Недели 5-6: Тестирование

Участник 1: генерация тестовых данных

Участники 2-3: baseline алгоритмы, сбор метрик

4

Неделя 7: Анализ

Все участники: построение графиков, написание отчета, подготовка финальной презентации

Ключевые вехи

- **Неделя 2:** Псевдокод всех алгоритмов готов
- **Неделя 4:** Рабочие прототипы
- **Неделя 6:** Завершение экспериментов
- **Неделя 7:** Защита проекта