

# Построение параллельных алгоритмов для решения задачи быстрогодействия с фазовыми ограничениями

студент 5 курса К. Ю. Егоров  
научный руководитель — к.ф-м.н., доцент И. В. Востриков

Кафедра системного анализа

28 мая 2022 г.

# Постановка задачи

Рассмотрим задачу

$$\dot{x} = f(t, x, u), \text{ где } x = [x_1, x_2, \dot{x}_1, \dot{x}_2].$$

Фазовые ограничения

$$[x_1, x_2] \in \Omega \subseteq \mathbb{R}^2.$$

Минимизируем функционал

$$J(u) = \int_{t_0}^{t_1} g(x, u) dt \rightarrow \min_u.$$

Необходимо решать уравнение Гамильтона–Якоби–Беллмана

$$\min_{u \in U} \left\{ g(x, u) + \sum_{i=1}^n \frac{\partial V(x)}{\partial x_i} f_i(x, u) \right\} = 0.$$

# Дискретизация задачи

Введем равномерную сетку

$$\Xi = \left\{ (x_i, y_j) \in \Pi, x_i = \varepsilon \frac{i}{N}, y_j = \varepsilon \frac{j}{M} \right\}, \text{ где } \Omega \subseteq \Pi.$$

Возможные переходы для  $(i, j)$

$$\text{possible}(i, j) = \{(i, j) \mid i \in \{i, i \pm 1\}, j \in \{j, j \pm 1\}, (x_i, y_j) \in \Omega\}$$

*Предложение.* Время перехода не зависит от предыдущей скорости и направления. Обозначим это время за

$$d_{i_{\text{from}}, j_{\text{from}}}(i_{\text{to}}, j_{\text{to}}).$$

Неявное матричное уравнение

$$V_{i,j} = \min_{(\hat{i}, \hat{j}) \in \text{possible}(i,j)} \{d_{i,j}(\hat{i}, \hat{j}) + V_{\hat{i}, \hat{j}}\},$$

$$V_{i_1, j_1} = 0.$$

# Алгоритм Беллмана–Форда

- 1 Изначально все узлы, кроме целевого, промаркированы значением  $+\infty$ , а целевой — нулем;
- 2 Необходимо провести  $(NM - 1)$  итерации алгоритма;
- 3 На каждой итерации происходит полный проход по сетке, далее  $(i, j)$  — позиция при проходе;
- 4 Для каждой вершины из возможного множества  $(\hat{i}, \hat{j}) \in \text{possible}(i, j)$  происходит перемаркировка: в случае, если  $V(i, j) > d_{i,j}(\hat{i}, \hat{j}) + V(\hat{i}, \hat{j})$ , перемаркируем

$$V(i, j) \leftarrow d_{i,j}(\hat{i}, \hat{j}) + V(\hat{i}, \hat{j}).$$

## Замечание

Оптимальная траектория не может иметь более  $(NM - 1)$  перемещения. Таким образом алгоритм действительно ищет кратчайший путь.

# Параллельный алгоритм Беллмана–Форда

На центральном процессоре

- Разбиваем сетку на каждой итерации на  $C$  (число ядер) подсеток, считающихся параллельно
- Синхронизацию при обработке последнего ряда подсетки осуществляем массивом мьютексов

На графическом процессоре

- Каждый узел считается на отдельном процессоре
- Синхронизация осуществляется за счет того, что проводится не один проход по матрице, а восемь: отдельно для каждого возможного направления движения

На многонодной установке

- Одна мастер программа,  $LC_l$  (совокупное число ядер) вычислителей (сервисов, общающихся по http)
- Синхронизация осуществляется за счет передачи мастером минимального инкремента каждому вычислителю

# Алгоритм Дейкстры

- 1 На начало алгоритма в множестве граничных узлов находится только целевой узел.
- 2 На каждой итерации алгоритма из множества граничных узлов выбирается узел с минимальной маркировкой.
- 3 Этот узел добавляется в множество обработанных узлов и удаляется из граничного множества.
- 4 Затем из возможного множества выбираются узлы, которые не находятся в множестве обработанных узлов. Эти узлы добавляются в множество граничных вершин, маркировка таких вершин обновляется.

Алгоритм останавливается в случае, если на некоторой итерации алгоритма был выбран начальный узел  $(i_0, j_0)$ .

# Параллельный алгоритм Дейкстры

Распараллеливанию подвергается самый алгоритмически сложная часть алгоритма — поиск минимума в граничном множестве. Для этого граничное множество представляется в виде  $C$  (число доступных ядер процессора) хэш-таблиц. Запись нового узла осуществляется в наименее полную таблицу.

- Для программы на центральном процессоре не требуется дополнительная синхронизация;
- Нет нативного метода, который мог бы поддержать графический процессор, так как взаимодействие с ним осуществляется копированием последовательных страниц памяти. Это потребовало бы использование массива для хранения граничного множества, и в конечном итоге ускорение было бы скомпенсировано операцией удаления из массива;
- Для многонодной установки минимальная инкрементальная информация — это выбранный узел, и узлы, добавленные в граничное множество.

Программы написаны на языке Go

- go routines;
- компилируемый;
- сборка мусора.

Помимо основных алгоритмов написаны:

- генератор ландшафта, использующий перлиновый шум;
- визуализатор пути, использующий OpenGL.

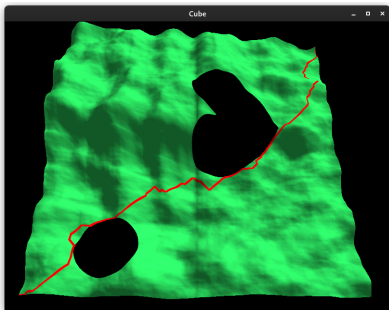
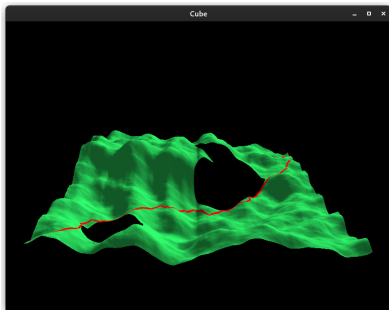
Программы для многонодной установки

- поставляются в виде двух docker-образов: для мастера и вычислителя;
- тестировалось только на Kubernetes кластере;
- требуют ручного применения Kubernetes манифестов.



# Пример 1

Алгоритм Дейкстры из (1, 1) в (1000, 1000)

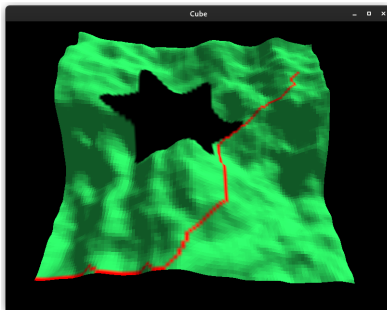
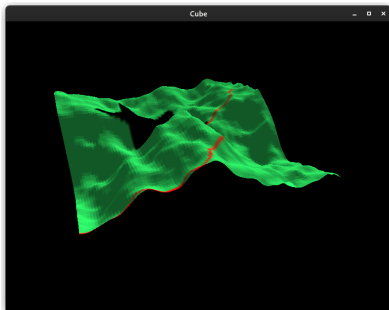


Время перехода

$$d_{i,j}(\hat{i}, \hat{j}) = 100 \cdot (\text{height}(\hat{i}, \hat{j}) - \text{height}(i, j))_+ + 5.$$

## Пример 2

Алгоритм Беллмана–Форда из (1, 1) в (90, 90)



Время перехода

$$d_{i,j}(\hat{i}, \hat{j}) = 100 \cdot (\text{height}(\hat{i}, \hat{j}) - \text{height}(i, j))_+ + 5.$$

# Сравнение времени работы

- ① Беллман Р., Дрейфус С. *Прикладные задачи динамического программирования*. М.: Наука, 1965.
- ② Shu-Xi, Wang. *The Improved Dijkstra's Shortest Path Algorithm and Its Application*. Procedia Engineering. 29. 1186-1190. 2012.
- ③ Glabowski, M., Musznicki, B. *Review and Performance Analysis of Shortest Path Problem Solving Algorithms*. International Journal On Advances in Software. 7. 20-30. 2014.
- ④ Ведякова А. О., Милованович Е. В. *Методы теории оптимального управления*. М.: Редакционно-издательский отдел Университета ИТМО. Санкт-Петербург, 2021.