Исследование эффективности вычисления надежного кратчайшего пути с использование GPU

А.А. Агафонов[1], А.И. Максимов[1], А.А. Бородинов[1]

[1]Самарский национальный исследовательский университет им. академика С.П. Королева, Московское шоссе 34А, Самара, Россия, 443086

Аннотация. В работе рассматривается задача нахождения надежного кратчайшего пути в стохастической транспортной сети, максимизирующего вероятность прибытия в пункт назначения в течение заранее определенного промежутка времени (бюджета поездки). Существующие алгоритмы решения задачи показывают хорошие результаты с точки зрения качества построенного маршрута, однако имеют большую вычислительную сложность, что не позволяет использовать их в практических приложениях для нахождения маршрутов движения в режиме реального времени. В работе представлена стратегия параллелизации алгоритма нахождения надежного пути с использованием CUDA GPU. Экспериментальные исследования, проведенные на транспортной сети города Самары, показывают предложенный подход позволяет сократить время вычислений в среднем в 5 раз.

1. Введение

Задача нахождения маршрута движения продолжает оставаться одной из актуальных проблем в транспортных системах. Существующие навигационные системы и сервисы построения маршрутов движения рассматривают транспортные сети как детерминированные и не учитывают стохастические свойства транспортных потоков при выборе маршрутов движения. В то же время, время прохождения дорожных сегментов может сильно варьироваться в зависимости от дня недели, времени дня, погодных условий, общественных событий и многих других факторов [1]. Однако учет не только среднего времени, но и дисперсии движения, то есть надежности маршрута, делает задачу поиска оптимального маршрута вычислительно сложной.

Исследовательские работы, напротив, изучают алгоритмы маршрутизации в стохастических и зависящих от времени транспортных сетях [2, 3, 4]. Существует несколько постановок задачи нахождения надежного пути в зависимости от используемого критерия оценки:

1. Модели с наименьшим ожидаемым временем движения (Least Expected Time — LET) [5, 6], в которых для сравнения возможных путей в качестве критерия оценки рассматривается ожидаемое время прохождения дорожных сегментов.
2. Модели α-надежного пути, минимизирующие интервал времени, необходимого для обеспечения прибытия в конечную вершину (пункт назначения) к выбранному моменту времени с заданной вероятностью α [7, 4, 8].
3. Модели с наибольшей надежностью движения [9, 3], максимизирующие вероятность прибытия в конечную вершину (пункт назначения) в течение заранее определенного

интервала времени (бюджета поездки). Часто эта проблема обозначается как SOTA (Stochastic On-Time Arrival).

В данной работе задача нахождения надежного кратчайшего пути рассматривается в следующей постановке: определить оптимальную стратегию навигации, максимизирующую вероятность прибытия в пункт назначения за выбранный бюджет поездки (в течение заранее определенного интервала времени).

В [3] предложен алгоритм точного решения SOTA-проблемы. Одним из этапов алгоритма является вычисление свертки, что является основной вычислительно сложной задачей. В общем виде свертка не может быть вычислена аналитически, и поэтому требуется схема дискретной аппроксимации. В работе [10] представлена модификация решения [3], учитывающая актуальную и прогнозную информацию о транспортных потоках в сети. Указанные решения обладают большим временем вычисления маршрута движения и не могут быть использованы в практических приложениях в режиме реального времени. Как следствие, несколько исследований было посвящено задаче ускорения работы алгоритма или разработке аппроксимационных алгоритмов поиска решения.

В статье [11] авторы представили несколько методов ускорения алгоритма решения SOTA-проблемы, включая усовершенствованные алгоритмы вычисления свертки с помощью быстрого преобразования Фурье и алгоритмы вычисления свертки с нулевой задержкой, а также методы определения оптимального порядка вычисления стратегии навигации. В [12] описана эвристика поиска адаптивного маршрута движения в стохастической сети. Представленный метод позволяет обеспечить наиболее вычислительно эффективную стратегию нахождения пути для общих распределений вероятностей. В [13] рассмотрены стохастические варианты двух методов предварительной обработки графа для решения задачи нахождения детерминированного кратчайшего пути, которые можно адаптировать к проблеме SOTA. Стратегия распараллеливания задачи с использованием графического процессора предложена в [14]. В [15] авторы предложили использовать устойчивые распределения Леви для описания времени прохождения дорожных сегментов, что позволяет перейти от операции вычисления свертки для определения надежности пути к пересчету параметров плотности распределения и значительно сокращает время исполнения алгоритма, однако находит путь с увеличенным временем движения.

В данной работе предложена стратегия параллелизации алгоритма нахождения надежного пути с использованием CUDA GPU. Работа организована следующим образом. Во втором разделе приводятся основные обозначения, постановка задачи и описание алгоритма. Стратегия параллелизации представлена в третьем разделе. В четвертом разделе описаны постановка и результаты экспериментальных исследований. В завершение работы представлены заключение и возможные направления дальнейших исследований.

2. Постановка задачи

Улично-дорожную сеть будем рассматривать в виде ориентированного графа $G = (N, A, P)$, где N — множество вершин графа, $|N|$ — количество вершин, A — множество ребер, $|A|$ — количество ребер, P — вероятностное описание времени прохождения ребер графа (т.е. сегментов транспортной сети).

Вес каждого ребра графа $(i, j) \in A$ (т.е. время прохождения дорожного сегмента) будем задавать с помощью случайной величины $T_{ij}(\tau)$ с зависящей от времени плотностью вероятности $p_{ij}^{\tau}(t)$.

Обозначим конечную вершину (пункт назначения) как $d \in N$, интервал времени, допустимый для достижения конечной вершины (т.е. бюджет поездки) как T. Оптимальная стратегия навигации определяется как стратегия максимизации вероятности прибытия в конечную вершину $d \in N$ из начальной (текущей) вершины $o \in N$ при наличии временного бюджета T.

Пусть $u_i(t)$ — вероятность прибытия в конечную вершину d из вершины i за время, не превышающее t. Тогда оптимальная стратегия навигации может быть сформулирована следующим образом:

$$\begin{aligned} u_i^\tau(t) = \max_{j \in N \wedge (i,j) \in A} \int_0^t p_{ij}^\tau(\theta) u_j^{\tau+\theta}(t-\theta) d\theta, \\ \forall i \in N \setminus \{d\}, t \in [0,T], \tau \geq 0, \\ u_d^\tau(t) = 1, t \in [0,T], \tau \geq 0. \end{aligned} \tag{1}$$

Для решения проблемы (1) в работе [3] был предложен дискретный алгоритм, который в виде псевдокода может быть записан следующим образом (Алгоритм 1).

Алгоритм 1: Дискретный алгоритм решения SOTA

Шаг 0. Инициализация
$k = 0$
$u_i^k(x) = 0, \quad \forall i \in N,\ i \neq d,\ x \in \mathbb{N},\ 0 \leq x \leq \frac{T}{\Delta t}$
$u_d^k(x) = 1, \quad x \in \mathbb{N},\ 0 \leq x \leq \frac{T}{\Delta t}$

Шаг 1. Обновление
for $k = 1, 2, \cdots, L$ **do**
  $\tau^k = k\delta$
  $u_d^k(x) = 1, \quad x \in \mathbb{N}, 0 \leq x \leq \frac{T}{\Delta t}$
  $u_i^k(x) = u_i^{k-1}(x)$
    $\forall i \in N, i \neq d, (i,j) \in A, x \in \mathbb{N}, 0 \leq x \leq \frac{\tau^k - \delta}{\Delta t}$
  $u_i^k(x) = \max_j \sum_{h=0}^{x} p_{ij}(h) u_j^{k-1}(x-h)$
    $\forall i \in N, i \neq d, (i,j) \in A, x \in \mathbb{N},$
    $\frac{(\tau^k - \delta)}{\Delta t} + 1 \leq x \leq \frac{\tau^k}{\Delta t}$
end

В алгоритме Δt — интервал дискретизации, δ — минимальное время прохождения дорожного сегмента в сети.

Тогда выбор следующей вершины j в графе дорожной сети (и, соответственно, следующего дорожного сегмента) с учетом оставшегося бюджета поездки t и вычисленного массива вероятностей прибытия $u_i(x)$ производится следующим образом:

$$j = \arg\max_{i \in N} u_i(t). \tag{2}$$

Описание стратегии параллелизации алгоритма с использованием графического процессора представлена в следующем разделе.

3. Методология

Для снижения времени работы алгоритма поиска надежного кратчайшего пути предлагается выполнить его реализацию на графическом акселераторе с использованием CUDA.

3.1. CUDA

CUDA (Compute Unified Device Architecture) — программно-аппаратная архитектура параллельных вычислений, разработанная компанией Nvidia, позволяющая использовать графические процессоры для вычислений общего назначения.

Модель вычислений представлена на рисунке 1. Хост-компьютер передает данные в память устройства и вызывает специальную функцию, называемую ядром. При вызове функции-ядра задаются два параметра: число блоков (blocks) и число потоков в блоке (threads). Каждый поток выполняет один и тот же набор команд, но с разными элементами данных. Потоки в рамках одного блока могут обмениваться результатами вычислений с использованием механизма общей памяти.

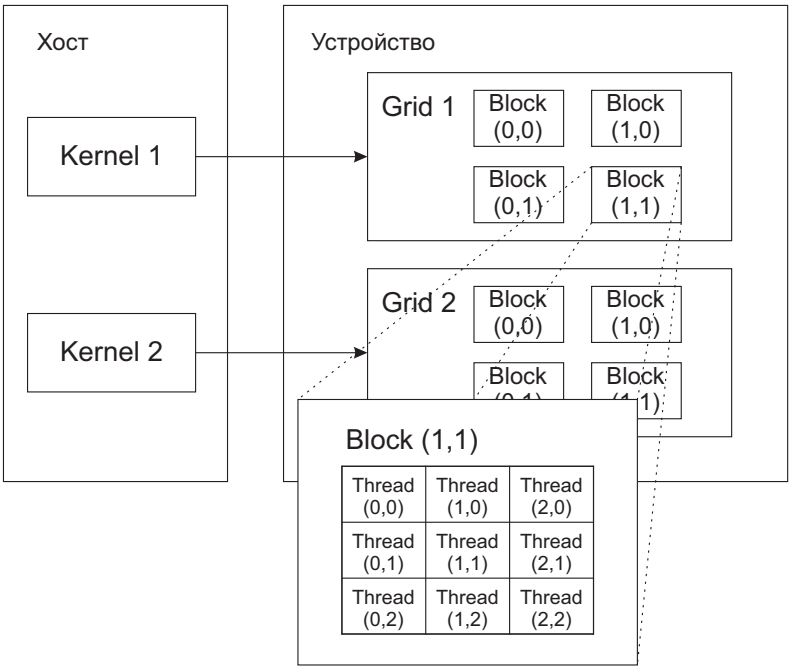

Рисунок 1. Модель вычислений.

3.2. Параллельный алгоритм
Введем обозначение

$$u_{ij}(x) = \sum_{h=0}^{x} p_{ij}(h) u_j(x-h), \tag{3}$$

то есть $u_{ij}(x)$ — вероятность достижения конечной вершины d из вершины i за время x при движении по ребру (i, j).

Алгоритм 1 должен выполняться последовательно по времени (цикл по переменной k), однако расчет вероятностей прибытия $u_i^k(x)$ может быть выполнен параллельно. Реализация этого процесса в CUDA состоит из вызова функции-ядра (Алгоритм 2), рассчитывающей вероятности достижения конечной вершины из каждой вершины графа на заданном временном шаге.

В алгоритме $threadIdx$ - индекс потока, $blockDim$ - размерность блока, $blockIdx$ - индекс блока обработки.

Таким образом, расчет вероятностей достижения вершины назначения за заданный интервал времени будет выполняться параллельно для каждой вершины графа в отдельном потоке графического акселератора.

Алгоритм 2: Параллельный алгоритм

for $k = 1, 2, \cdots, L$ **do**
  process«N_c, T_c»(k)
end

function `process(`k`):`
  `/* рассчитать индекс обрабатываемой вершины` x `*/`
  $x = threadIdx.x + blockDim.x * blockIdx.x$;
  загрузить $p(1), \cdots, p(t)$ и соответствующие им u из памяти ;
  рассчитать $u_{ix}^k \forall (i, x) \in A$;
  рассчитать $u_x^k = \max_i u_{ix}^k$;
return

4. Экспериментальные исследования

4.1. Набор данных

Экспериментальные исследования модели проводились для крупномасштабной улично-дорожной сети г. Самары, состоящей из 47274 дорожных сегментов и 18582 вершин.

Для сравнения времени работы базовой и параллельной реализаций алгоритма нахождения надежного кратчайшего пути были выбраны 6 пар различных вершин отправления-прибытия на графе дорожной сети, после чего задача навигации решалась для каждой пары вершин и различных дней недели, времени начала движения и бюджета поездки. Вершины были выбраны таким образом, чтобы среднее время поездки составляло от 15 до 60 минут. Всего было проведено 6300 экспериментов.

Характеристики используемой ПЭВМ: процессор Intel Core i7-9700K 3.60 GHz, оперативная память 64 ГБ, графический акселератор GeForce RTX 2080 Ti. Среднее время работы алгоритмов представлено в таблице 1.

Таблица 1. Среднее время работы алгоритма.

	Базовый алгоритм	CUDA
Время работы, с	20.58	3.88

Реализация алгоритма на CUDA позволяет сократить время вычислений в среднем в 5 раз.

Время работы алгоритмов зависит от выбранного бюджета поездки, т.е. количества временных шагов алгоритма. График зависимости времени работы от бюджета поездки показан на рисунке 2.

При увеличении бюджета поездки выигрыш параллельного алгоритма по времени работы увеличивается.

Подробный анализ времени работы параллельного алгоритма позволяет сделать вывод, что большую часть времени работы алгоритма занимает обмен данными хост-компьютера с устройством.

5. Заключение

В работе рассматривалась задача нахождения надежного кратчайшего пути в стохастической сети, максимизирующего вероятность прибытия в пункт назначения в течение заранее определенного промежутка времени. Была разработана стратегия параллелизации алгорит-

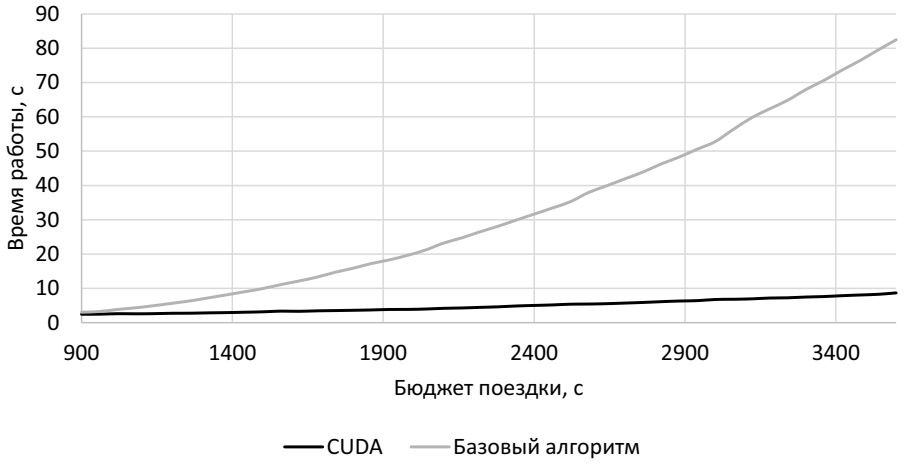

Рисунок 2. Зависимость времени работы алгоритма от бюджета поездки.

ма с использованием графического акселератора и реализован параллельный алгоритм на программно-аппаратной архитектуре CUDA.

Экспериментальные исследования, проведенные на крупномасштабной улично-дорожной сети г. Самары, показали, что параллельная реализация алгоритма позволяет сократить время вычислений в среднем в 5 раз.

Дальнейшие исследования могут быть направлены на разработку алгоритма с меньшим числом пересылок данных между хост-компьютером и графическим акселератором.

6. Благодарности

Работа выполнена при финансовой поддержке Министерства науки и высшего образования РФ (уникальный идентификатор проекта RFMEFI57518X0177).

7. Литература

[1] Агафонов, А.А. Анализ больших данных в геоинформационной задаче краткосрочного прогнозирования параметров транспортного потока на базе метода k ближайших соседей / А.А. Агафонов, А.С. Юмаганов, В.В. Мясников // Компьютерная оптика. – 2018. – Vol. 42(6). – P. 1101-1111. DOI: 10.18287/2412-6179-2018-42-6-1101-1111.
[2] Chen, P. The alpha-reliable path problem in stochastic road networks with link correlations: A moment-matching-based path finding algorithm / P. Chen // Expert Systems with Applications. – 2018. – Vol. 110. – P. 20-32.
[3] Samaranayake, S. A tractable class of algorithms for reliable routing in stochastic networks / S. Samaranayake, S. Blandin, A. Bayen // Transportation Research Part C: Emerging Technologies. – 2012. – Vol. 20(1). – P. 199-217.
[4] Nie, Y. Shortest path problem considering on-time arrival probability / Y. Nie, X. Wu // Transportation Research Part B: Methodological. – 2009. – Vol. 43(6). – P. 597-613.
[5] Fu, L. Expected shortest paths in dynamic and stochastic traffic networks / L. Fu, L.R. Rilett // Transportation Research Part B: Methodological. – 1998. – Vol. 32(7). – P. 499-516.
[6] Gao, S. Optimal routing policy problems in stochastic time-dependent networks / S. Gao, I. Chabini // Transportation Research Part B: Methodological. – 2006. – Vol. 40(2). – P. 93-122.
[7] Chen, A. Path finding under uncertainty / A. Chen, Z. Ji // Journal of Advanced Transportation. – 2005. – Vol. 39(1). – P. 19-37.
[8] Chen, B.Y. Finding Reliable Shortest Paths in Road Networks Under Uncertainty / B.Y. Chen // Networks and Spatial Economics. – 2013. – Vol. 13(2). – P. 123-148.

[9] Fan, Y. Optimal routing for maximizing the travel time reliability / Y. Fan, Y. Nie // Networks and Spatial Economics. – 2006. – Vol. 6(3-4). – P. 333-344.
[10] Агафонов, А.А. Метод определения надёжного кратчайшего пути в зависящей от времени стохастической сети и его применение в геоинформационных задачах управления транспортом / А.А. Агафонов, В.В. Мясников // Компьютерная Оптика. – 2016. – Vol. 40(2). – P. 275-283. DOI: 10.18287/2412-6179-2016-40-2-275-283.
[11] Samaranayake, S. Speedup techniques for the stochastic on-time arrival problem / S. Samaranayake, S. Blandin, A. Bayen // Open Access Series in Informatics. – 2012. – Vol. 25. – P. 83-96.
[12] Niknami, M. Tractable pathfinding for the stochastic on-time arrival problem / M. Niknami, S. Samaranayake // Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). – 2016. – Vol. 9685. – P. 231-245.
[13] Sabran, G. Precomputation techniques for the stochastic on-time arrival problem / G. Sabran, S. Samaranayake, A. Bayen // Proceedings of the Workshop on Algorithm Engineering and Experiments. – 2014. – P. 138-146.
[14] Abeydeera, M. GPU parallelization of the stochastic on-time arrival problem / M. Abeydeera, S. Samaranayake // 21st International Conference on High Performance Computing, HiPC. – 2014. – Vol. 22.
[15] гафонов, А.А. Метод определения надежного кратчайшего пути в стохастической сети с использованием параметрически заданных устойчивых распределений вероятностей / А.А. Агафонов, В.В. Мясников // Труды СПИИРАН. – 2019. – Vol. 18(3). – P. 558-582.

Performance comparison of GPU parallelization algorithms for the reliable shortest path problem

A.A. Agafonov[1], **A.I. Maksimov**[1], **A.A. Borodinov**[1]

[1]Samara National Research University, Moskovskoe Shosse 34A, Samara, Russia, 443086

Abstract. The paper considers the reliable shortest path problem in a stochastic transportation network that maximizes the probability of arriving at a destination within a predetermined period of time (time budget). Existing algorithms solves this problem with good results in terms of the quality (travel time) of the found route, however, but have high computational complexity, which does not allow using them in practical navigational applications in real time. The paper presents a parallelization strategy for finding a reliable path using the CUDA GPU. Experimental studies conducted on the transport network of the Samara city show that the proposed approach can reduce the computation time by an average of 5 times.