Отчёт по алгоритмам поиска пути в графе

Содержание

Описание тестируемых алгоритмов

1) **Поиск в ширину**

Поиск в ширину – это алгоритм поиска пути в графе. Поиск в ширину последовательно обходит все уровни графа, начиная со стартовой вершины. Поиск пути завершается, если путь найден, либо были посещены всевозможные вершины, до которых можно дойти из стартовой вершины. Поиск в ширину позволяет найти самый короткий по количеству ребер путь в графе.

Худшие данные: нет пути, граф цепочка (в этих случаях мы вынуждены посетить все вершины)

Теоретическая асимптотика: O (n + m), где n — число вершин, m — число рёбер.

2) **Поиск в глубину**

Поиск в глубину – это алгоритм поиска пути в графе. Поиск в глубину посещает все исходящие ребра и затем рассматривает те вершины, которые еще не были рассмотрены, запуская поиск в глубину от них. Возврат происходит если в текущей вершине не осталось нерассмотренных ребер.

Худшие данные: нет пути

Теоретическая асимптотика: O (n + m), где n — число вершин, m — число рёбер.

3) **Алгоритм Дейкстры**

Алгоритм Дейкстры - алгоритм поиска пути в графе, который во взвешенном графе находит кратчайшие (по весу) пути от одной из вершин графа до всех остальных (до заданной). Алгоритм работает только для графов, где нет ребер отрицательного веса.

Худшие данные: нет пути

Теоретическая асимптотика: O (n^2 + m), где n — число вершин, m — число рёбер.

4) **Алгоритм Форда-Беллмана**

Алгоритм Форда-Беллмана - Пусть дан ориентированный взвешенный граф G с n вершинами и m рёбрами, и указана некоторая вершина v. Требуется найти длины кратчайших путей от вершины v до всех остальных вершин. В отличие от алгоритма Дейкстры, этот алгоритм применим также и к графам, содержащим рёбра отрицательного веса. Впрочем, если граф содержит отрицательный цикл, то, понятно, кратчайшего пути до некоторых вершин может не существовать (по причине того, что вес кратчайшего пути должен быть равен минус бесконечности); впрочем, этот алгоритм можно модифицировать, чтобы он сигнализировал о наличии цикла отрицательного веса, или даже выводил сам этот цикл.

Худшие данные: цикл отрицательного веса

Теоретическая асимптотика: O (nm), где n — число вершин, m — число рёбер.

5) **Алгоритм A*(A стар)**

Алгоритм A*(A стар) - алгоритм поиска, который находит во взвешенном графе маршрут наименьшей стоимости от начальной вершины до выбранной конечной. В процессе работы алгоритма для вершин рассчитывается функция f(v)=g(v)+h(v), где

- **g(v) — наименьшая стоимость пути в v из стартовой вершины,**
- **h(v) — эвристическое приближение стоимости пути от v до конечной цели.**

Фактически, функция f(v) — длина пути до цели, которая складывается из пройденного расстояния g(v) и оставшегося расстояния h(v). Исходя из этого, чем меньше значение f(v), тем раньше мы откроем вершину v, так как через неё мы предположительно достигнем расстояние до цели быстрее всего. Открытые алгоритмом вершины можно хранить в очереди с приоритетом по значению f(v)

Худшие данные: неудачно выбранная эвристика

Теоретическая асимптотика: при использовании оптимальной эвристики A* будет O(n) как по пространству, так и по времени, если не учитывать сложность самого эвристического расчета. n - длина пути решения.

Методика тестирования

Запуск тестирования осуществляется с помощью вызова из командной строки скрипта main.py со следующими параметрами:

py main.py -t

Результаты тестирования представлены таблицами и сохраняются в xls-файлы со следующими именами: random.xls, chain_graph.xls, random.graph.xls.

Тестирование позволит нам получить необходимую характеристику – время работы алгоритмов поиска пути в графе.

Параметры компьютера, на котором проводилось тестирование

1. Операционная система: Windows 10 Home
2. Процессор: Intel(R) Core (TM) i7-7700HQ 2.81 ГГц
3. Размер RAM: 8 Гб (доступно: 7,89 Гб)
4. Количество ядер: 4

При запуске тестирования было отключено большинство процессов (за исключением системных), способных негативно повлиять на результаты тестирования. Таким образом, все тестирование проходило в одинаковых условиях.

Анализ результатов

Рассмотрим графики, получившиеся при тестировании на следующих данных: три различных вида графов (случайные, полные, графы-цепочки), два различных диапазона вершин (100-190, 100-550). На данных графиках построены лучшие приближения к теоретическим асимптотикам, указана величина аппроксимации, а также показано уравнение получившихся приближений. На основании этих графиков и таблиц, расположенных в папке task/statement/ можно сделать выводы о соответствии реального времени работы и теоретической асимптотики, в зависимости от типов графов и количества вершин. Так же на графиках указан предел погрешности вычисления времени выполнения данного алгоритма.

Результаты тестирования

1. **Поиск в ширину**

I. **Тестирование по времени**

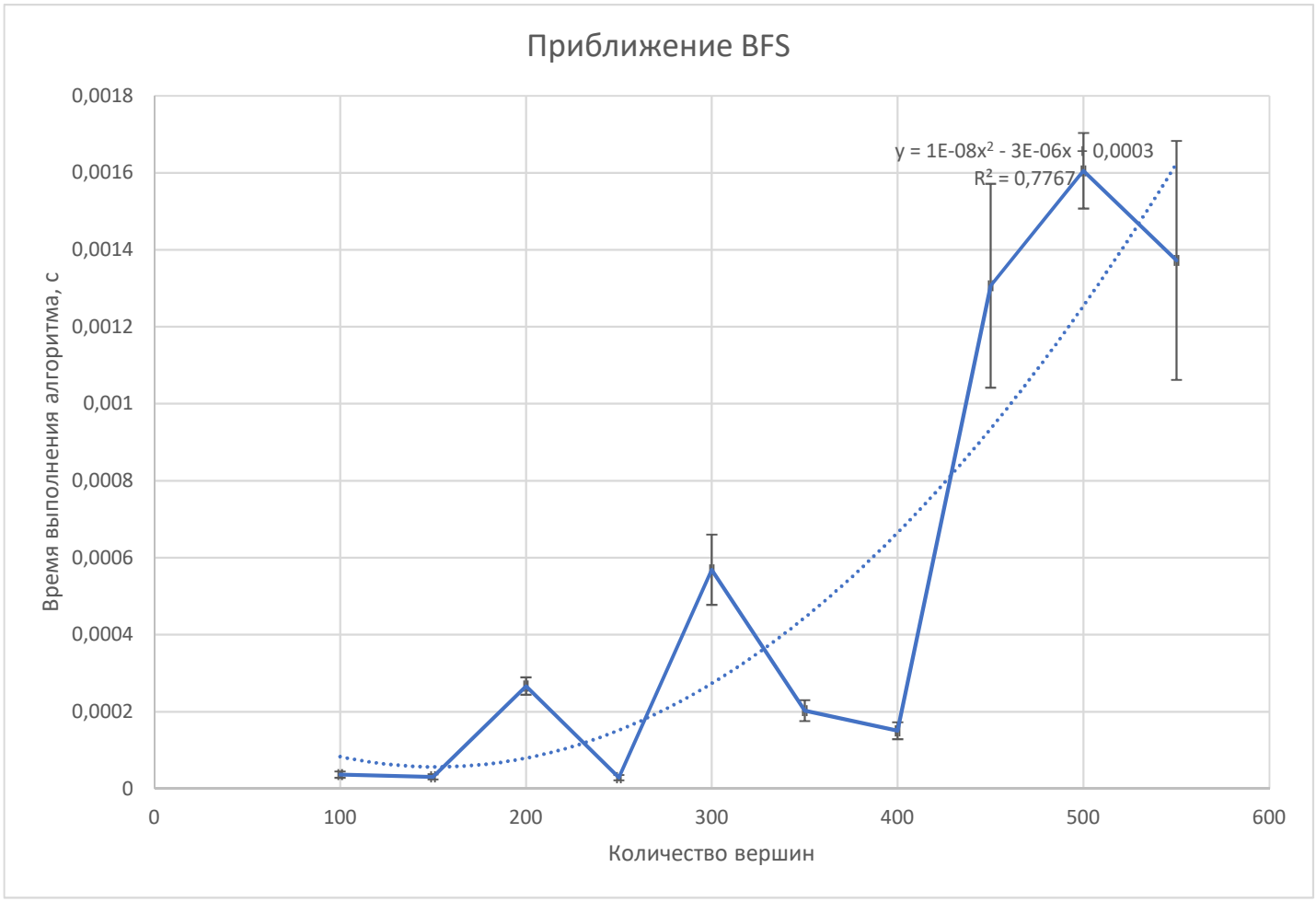

Рис. 1 График зависимости времени выполнения алгоритма «Поиск в ширину» от количества вершин для случайных графов с диапазоном вершин 100-550

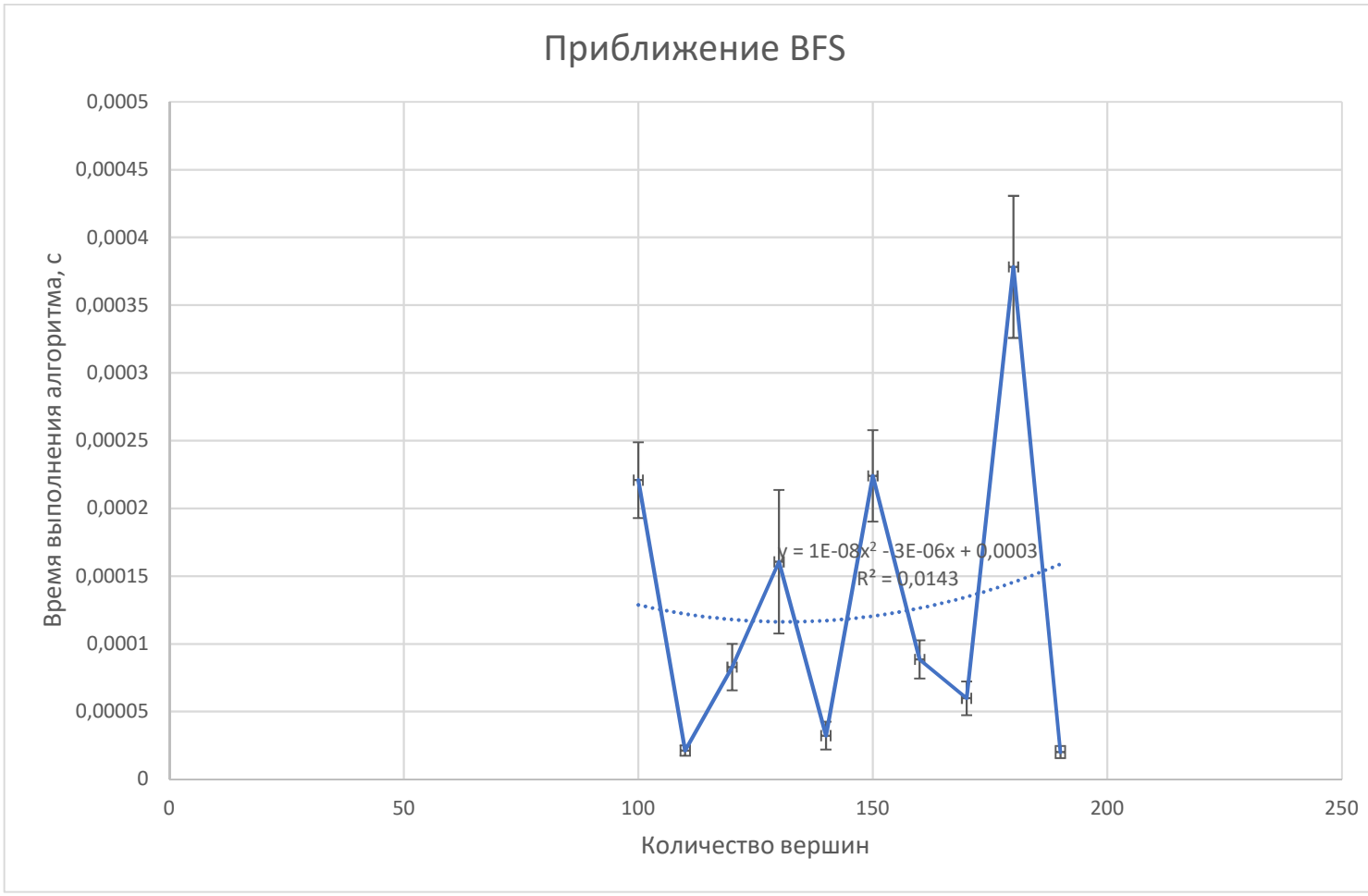

Рис. 2 График зависимости времени выполнения алгоритма «Поиск в ширину» от количества вершин для случайных графов с диапазоном вершин 100-190

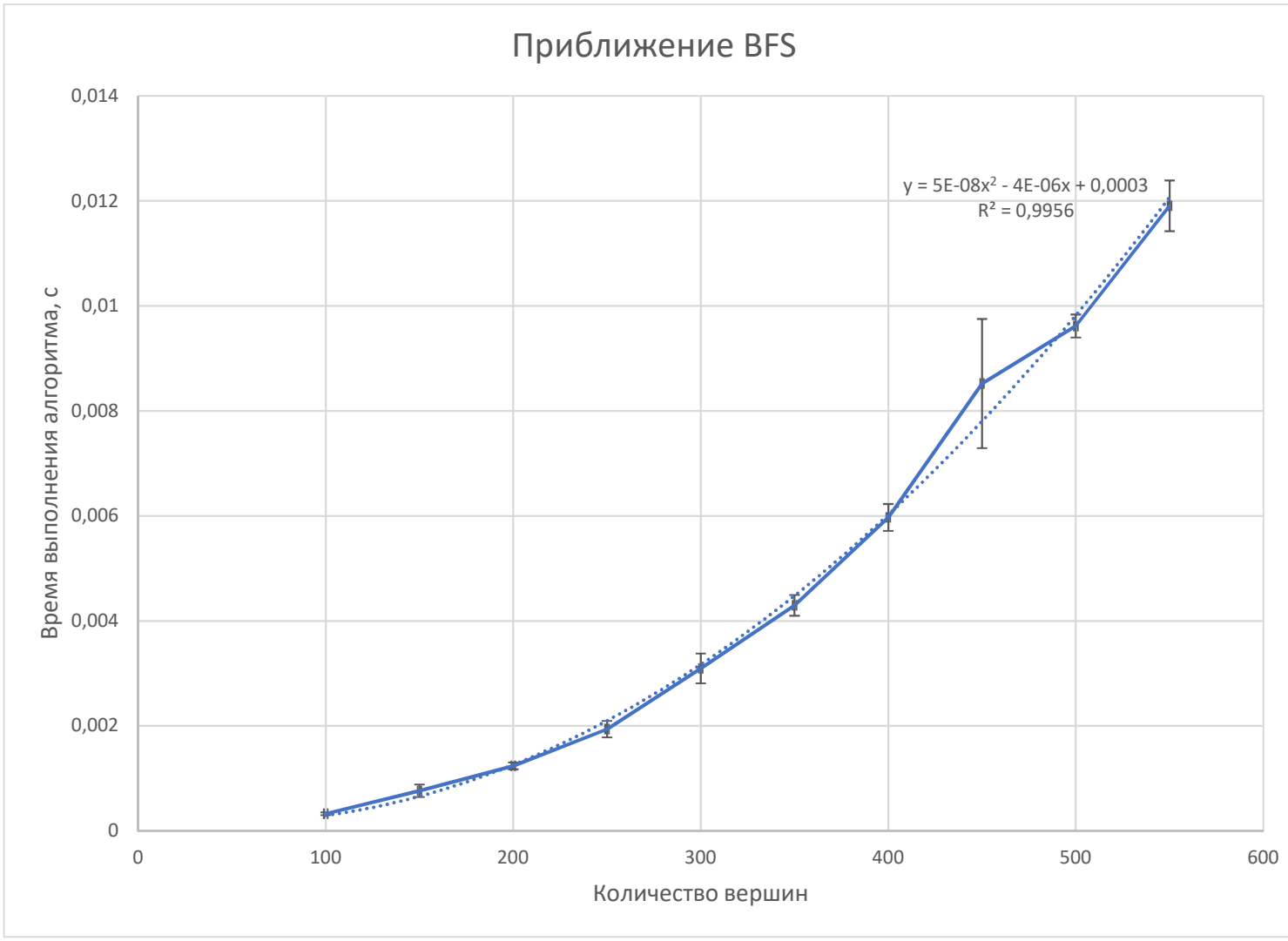

Рис. 3 График зависимости времени выполнения алгоритма «Поиск в ширину» от количества вершин для полных графов с диапазоном вершин 100-550

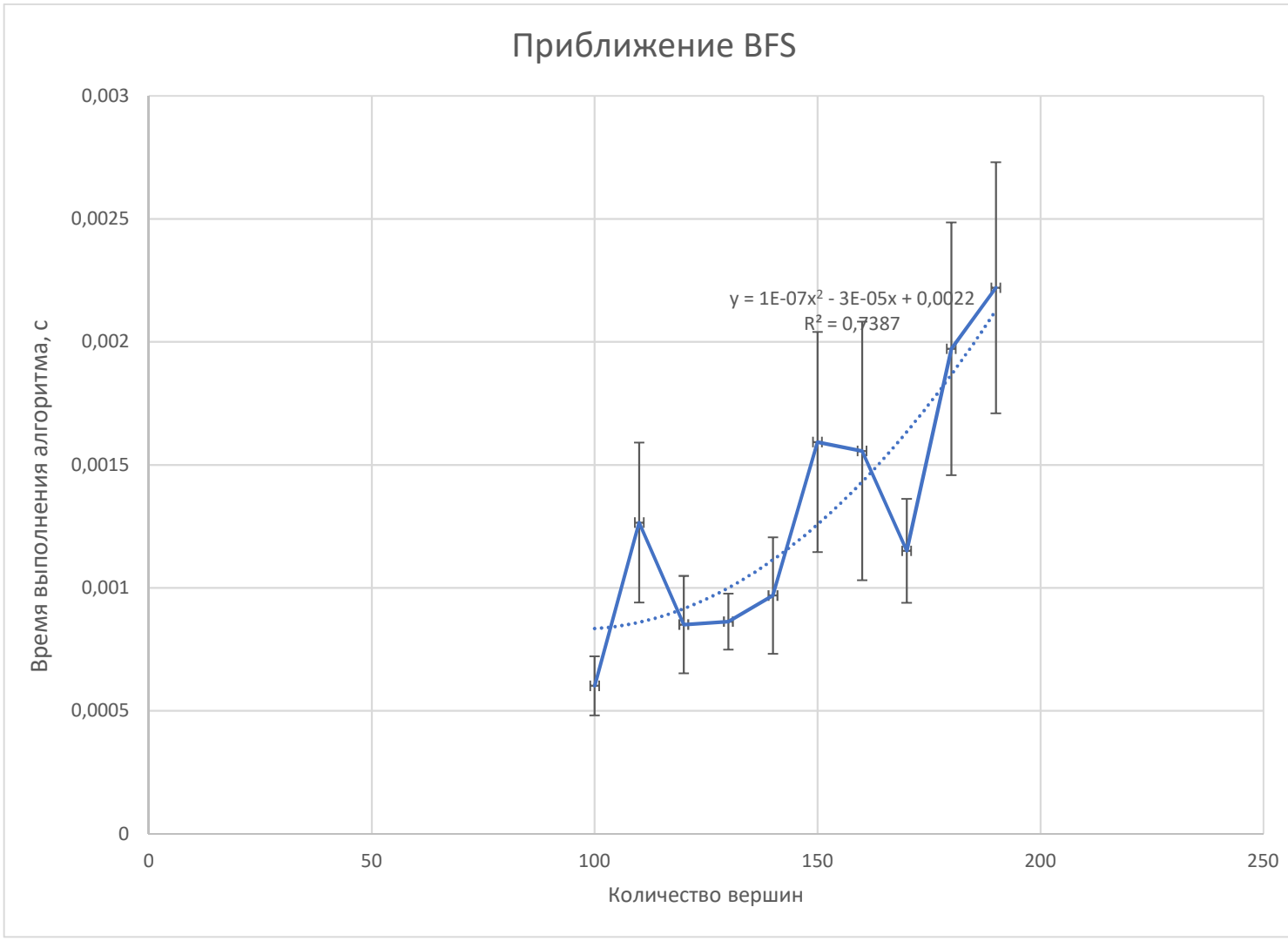

Рис. 4 График зависимости времени выполнения алгоритма «Поиск в ширину» от количества вершин для полных графов с диапазоном вершин 100-190

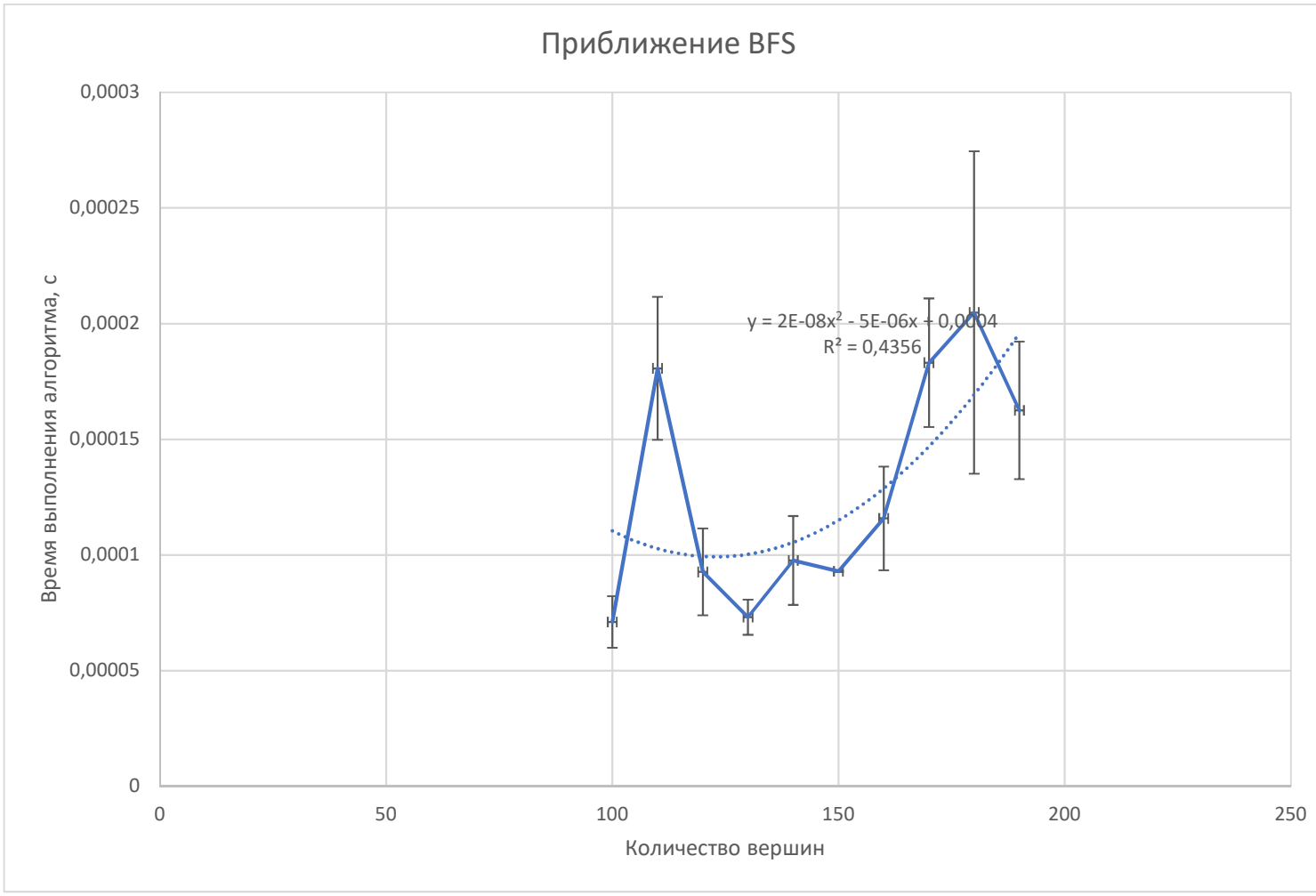

Рис. 5 График зависимости времени выполнения алгоритма «Поиск в ширину» от количества вершин для графов-цепочек с диапазоном вершин 100-190

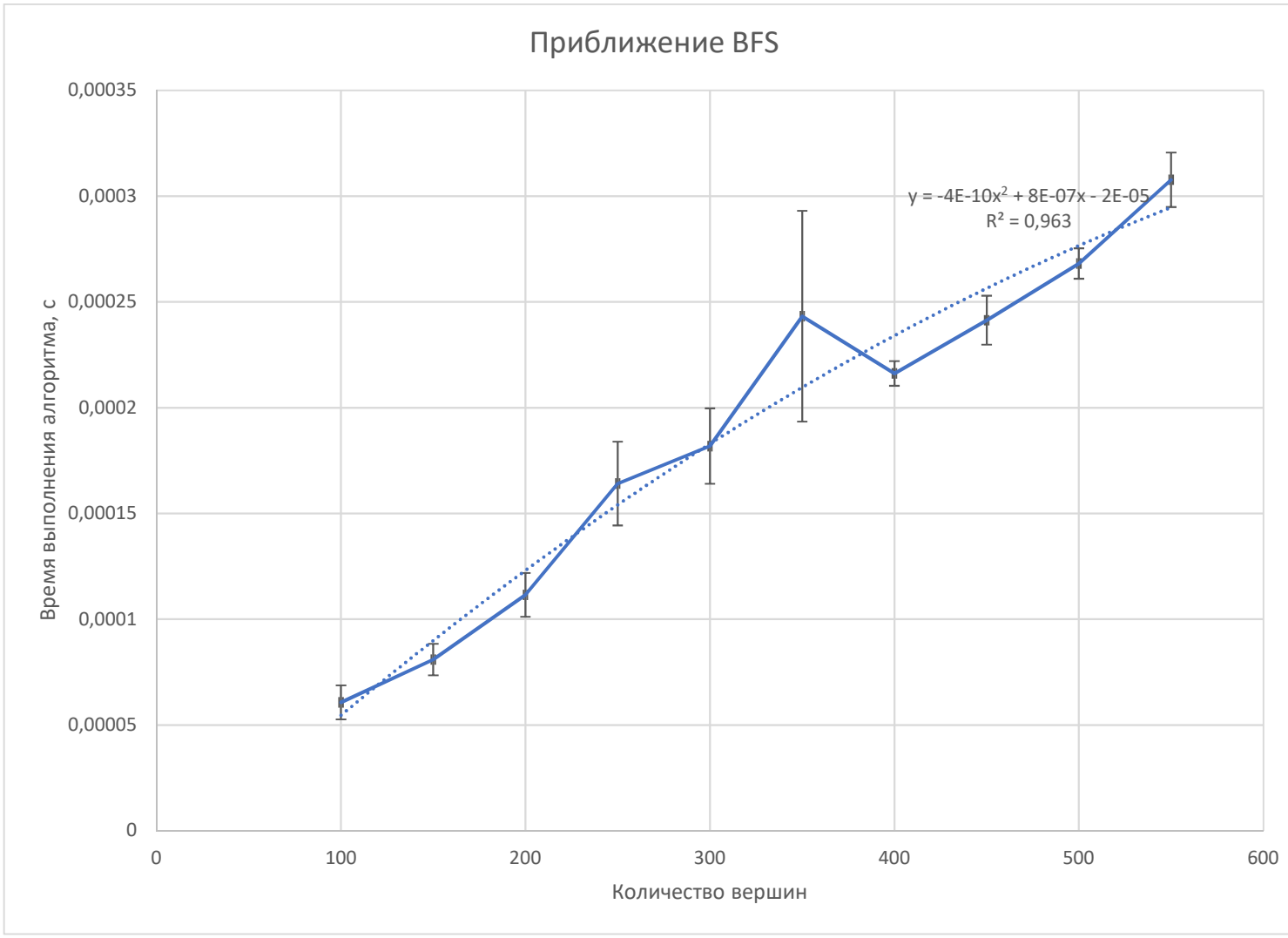

Рис. 6 График зависимости времени выполнения алгоритма «Поиск в ширину» от количества вершин для графов-цепочек с диапазоном вершин 100-550

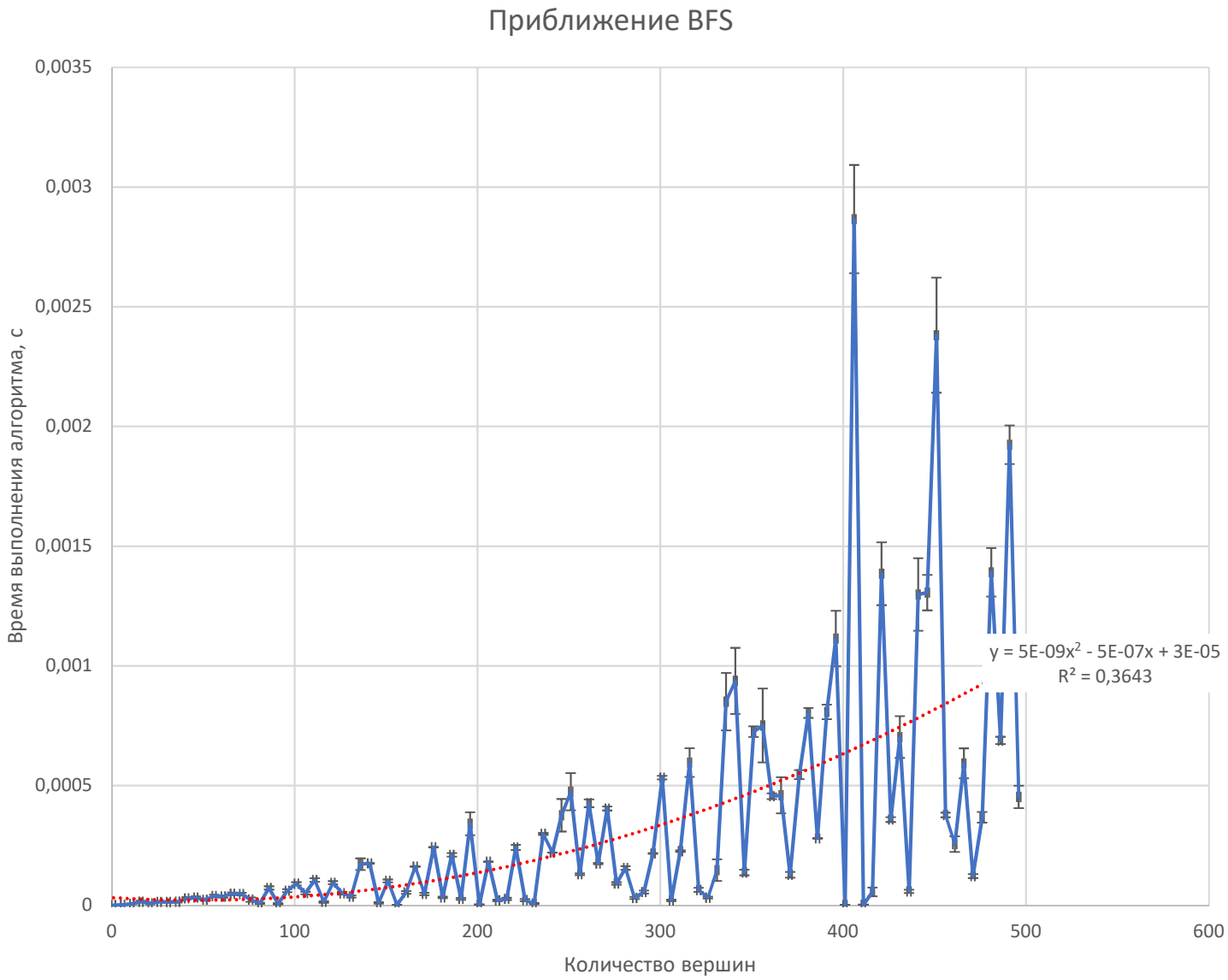

Рис. 7 График зависимости времени выполнения алгоритма «Поиск в ширину» от количества вершин для случайных графов с диапазоном вершин 1-496

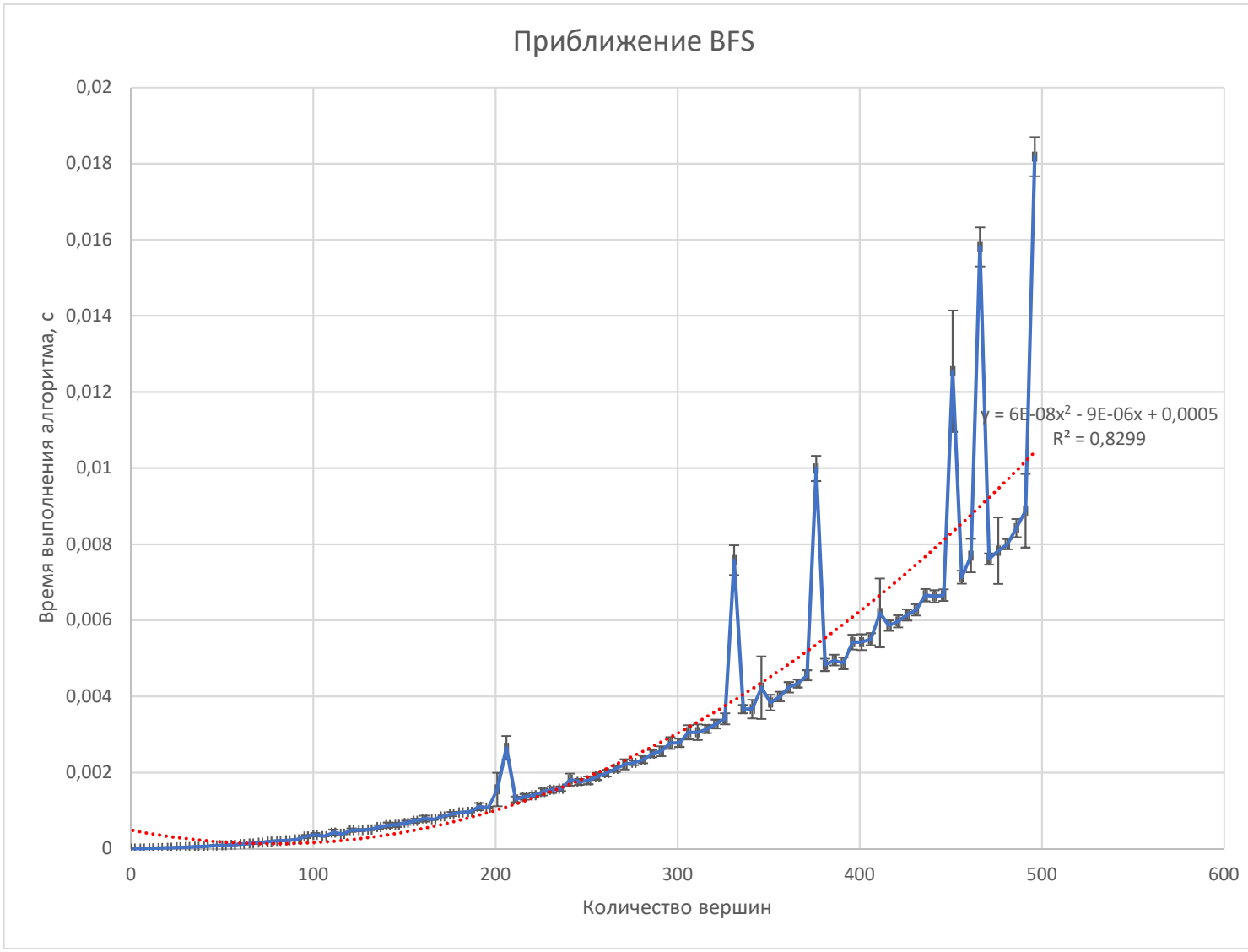

Рис. 8 График зависимости времени выполнения алгоритма «Поиск в ширину» от количества вершин для полных графов с диапазоном вершин 1-496

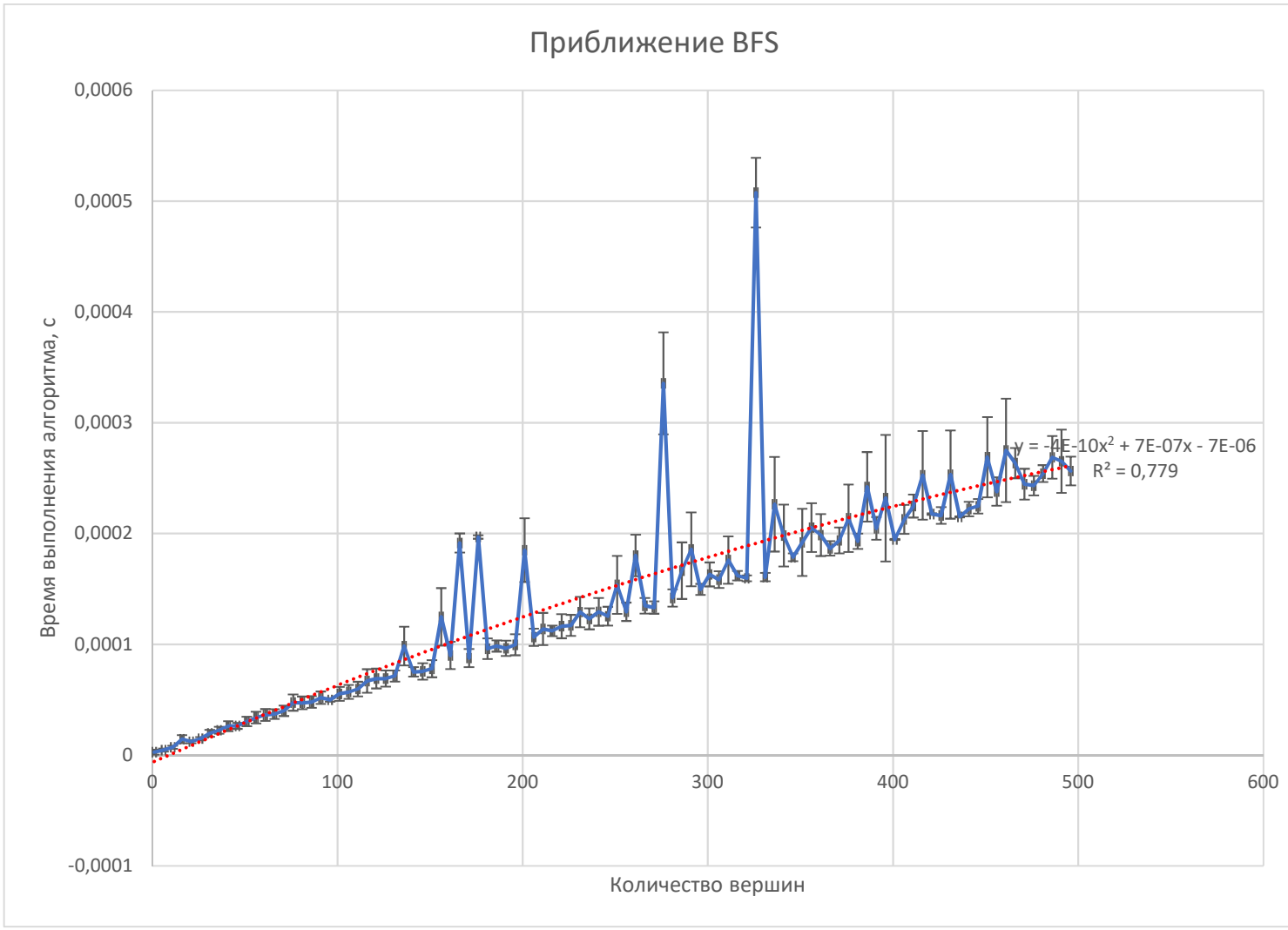

Рис. 9 График зависимости времени выполнения алгоритма «Поиск в ширину» от количества вершин для графов-цепочек с диапазоном вершин 1-496

Вывод

При анализе полученных графиков можно заметить, что на худших данных (полный граф) алгоритм BFS работает за время, которое хорошо приближается квадратичной функцией. На остальных данных, представленных на графиках, зависимость времени выполнения от количества вершин практически линейная. Из этого следует, что проведённые эксперименты коррелируют с теоретическими результатами, которые заключают, что время работы алгоритма $O(\boldsymbol{n}+\boldsymbol{m})$, где $\mathbf{n}$ – количество вершин в графе, $\mathbf{m}$ – количество рёбер. В графах количество рёбер не превосходит $\frac{\boldsymbol{n}\cdot(\boldsymbol{n}-\mathbf{1})}{\mathbf{2}}$ (это достигается в полных графах), а в графах-цепочках количество рёбер равно $\boldsymbol{n}-\mathbf{1}$, тогда теоретическая асимптотика становится линейной.

Некоторые неточности и скачки в измерениях обусловлены случайным выбором стартовой и конечной вершины в пути.

II. Тестирование по памяти

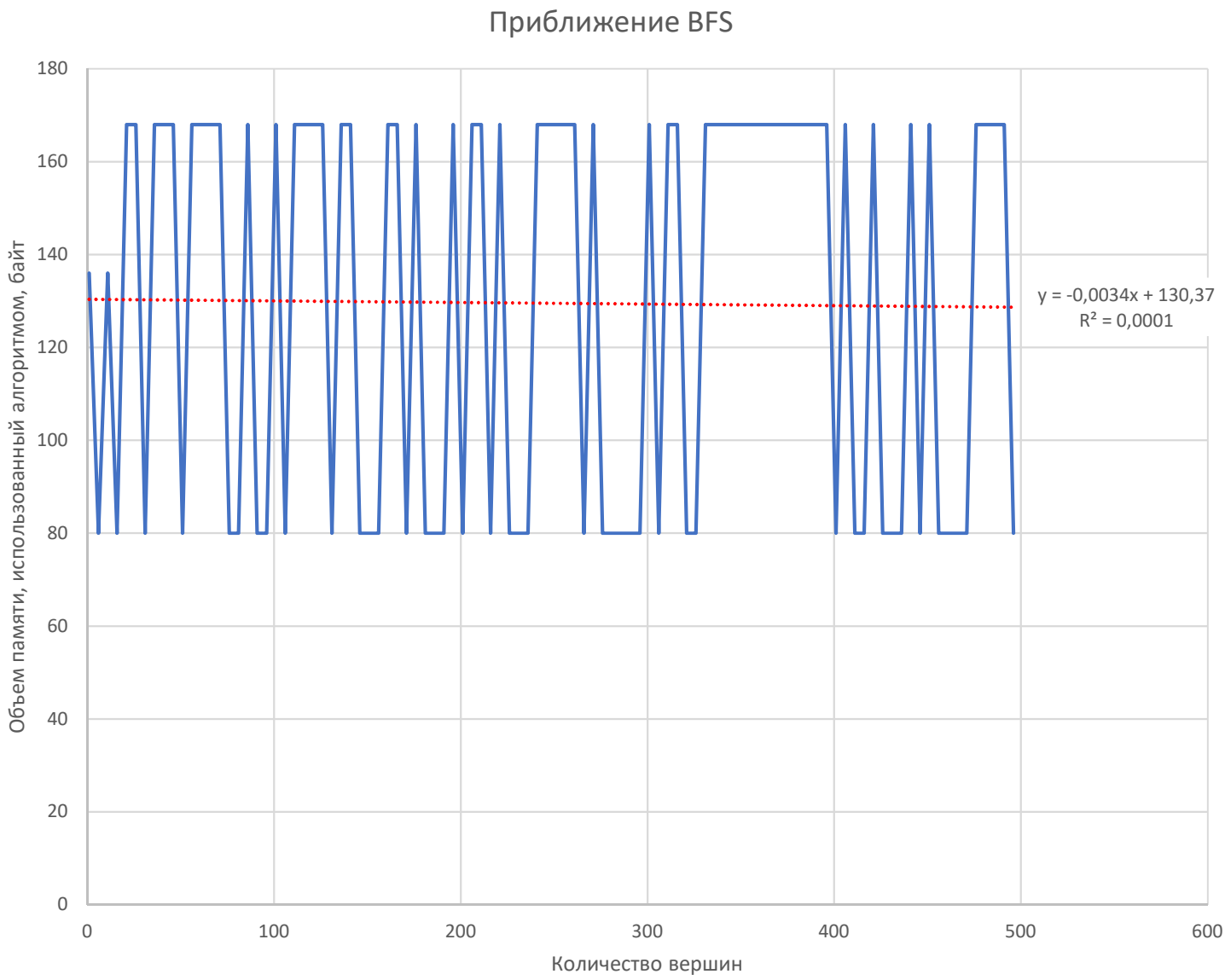

Рис. 1 График зависимости объёма памяти, использованной алгоритмом «Поиск в ширину» от количества вершин в случайном графе с диапазоном вершин 1-496

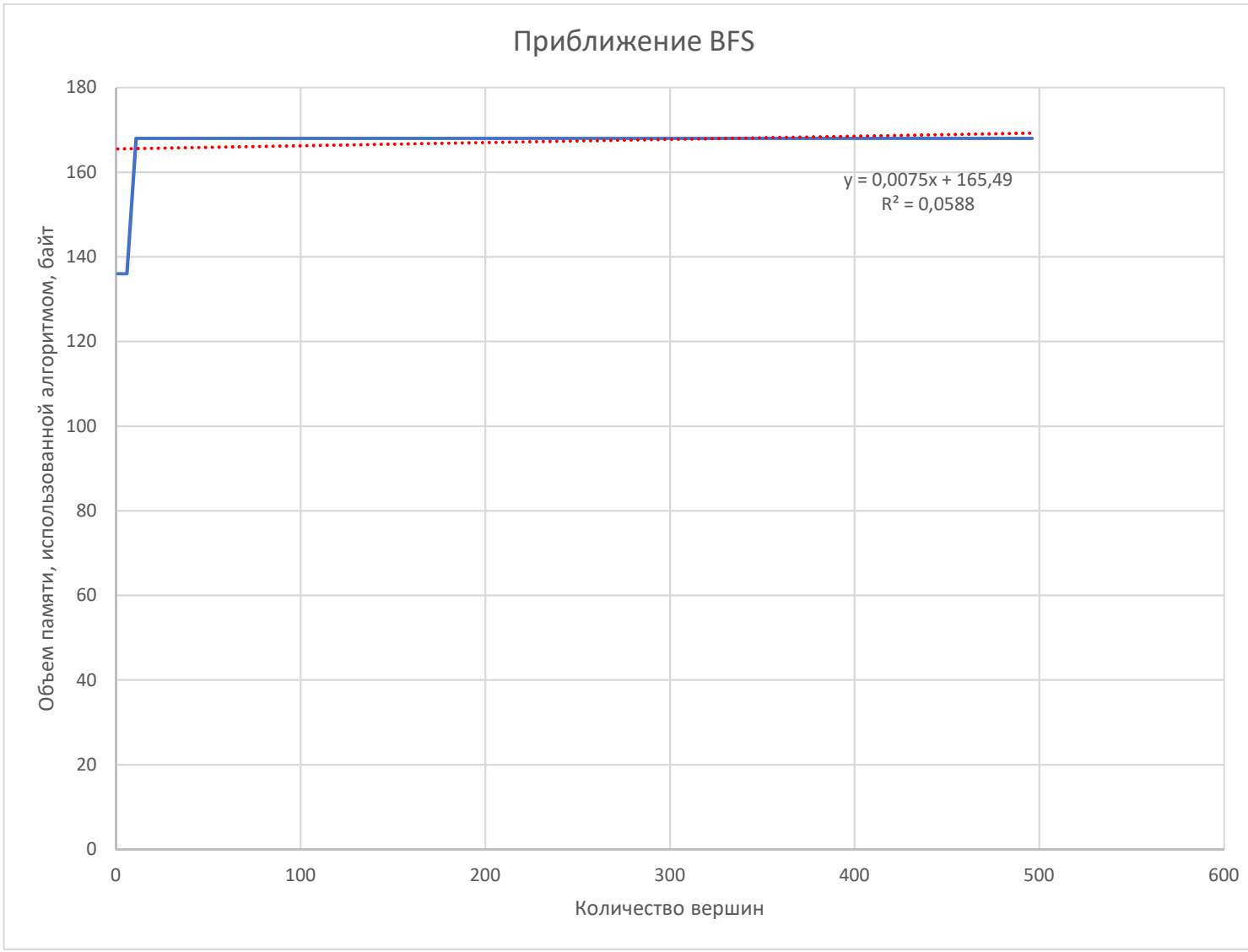

Рис. 2 График зависимости объёма памяти, использованной алгоритмом «Поиск в ширину» от количества вершин в полном графе с диапазоном вершин 1-496

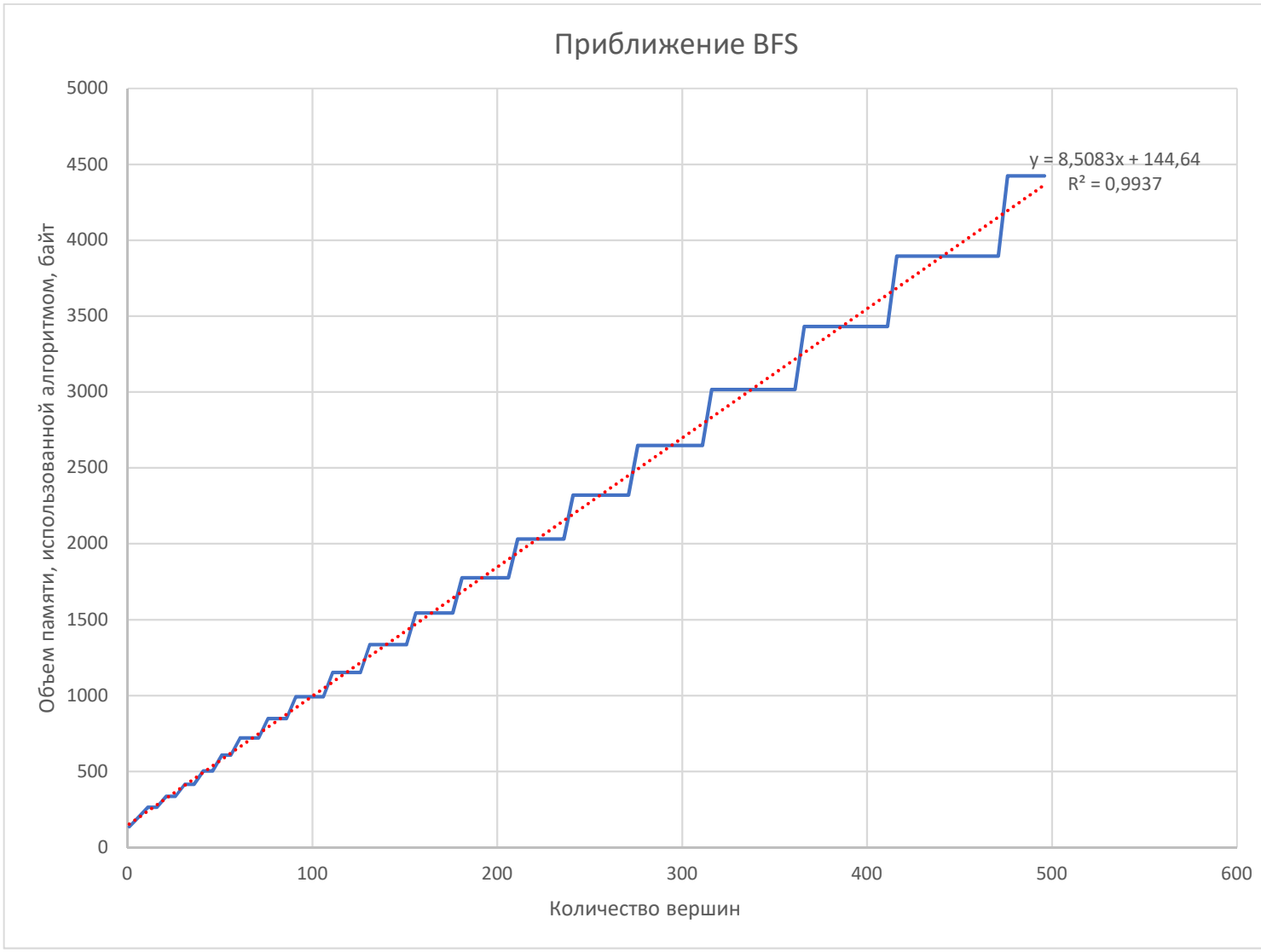

Рис. 3 График зависимости объёма памяти, использованной алгоритмом «Поиск в ширину» от количества вершин в графе-цепочке с диапазоном вершин 1-496

Вывод

Из полученных результатов очевидно, что при работе алгоритма «Поиск в ширину» количество используемой им памяти не зависит от числа вершин напрямую. Однако, количество используемой алгоритмом памяти зависит от размерности пространства решений (поскольку при тестировании данного алгоритма на графах-цепочках график имеет лестничнообразный вид, а в качестве точек, между которыми мы ищем путь берутся крайние вершины цепочки, а значит при увеличении размера графа увеличивается пространство решений). Так как график на рисунке 3 имеет лестничнообразный вид, то можно сделать вывод о том, что его лучше всего приближать линейной функцией. Таким образом мы получаем результат, что объем памяти, используемый алгоритмом, линейно зависит от размера пространства решений.

2. **Поиск в глубину**

I. **Тестирование по времени**

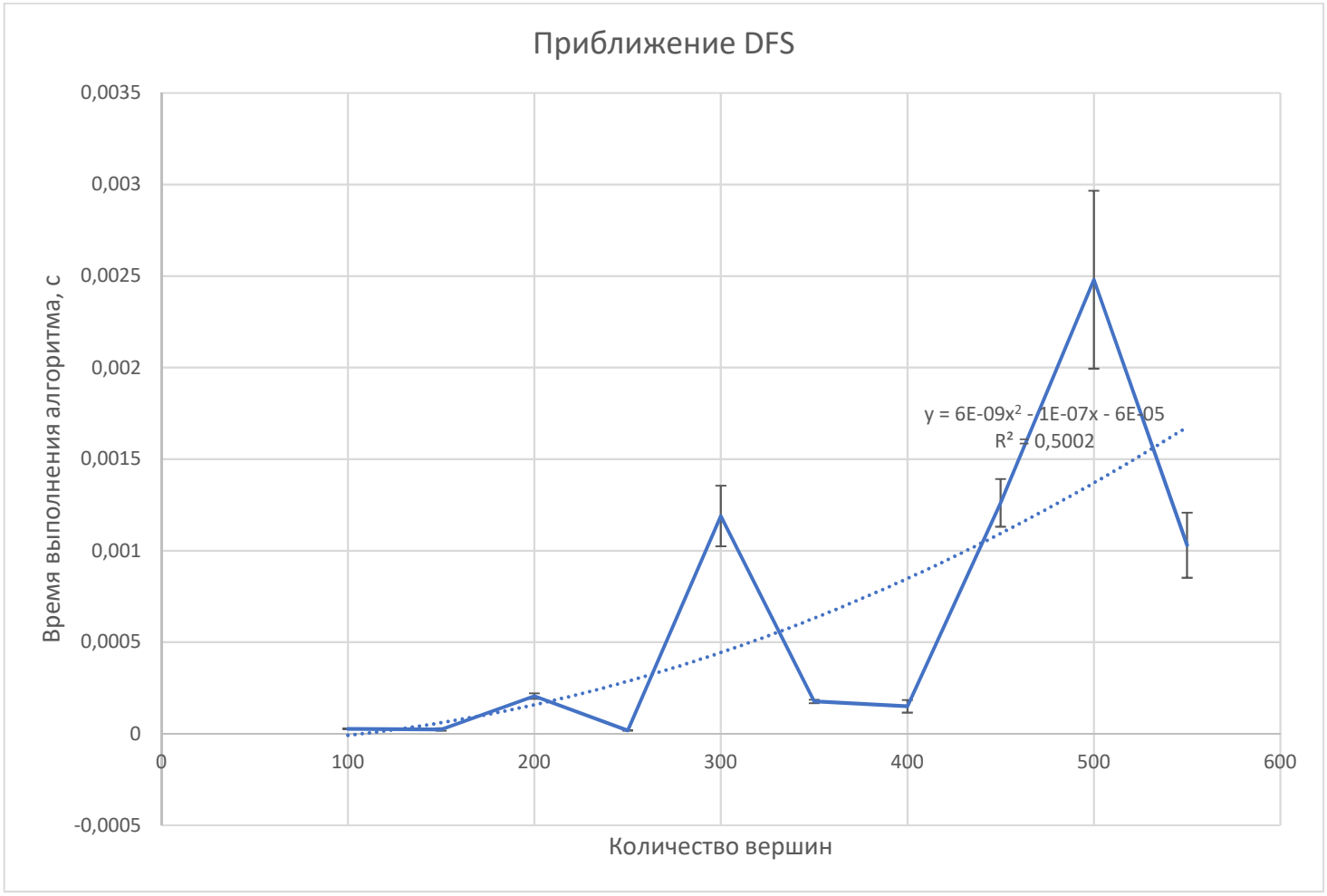

Рис. 1 График зависимости времени выполнения алгоритма «Поиск в глубину» от количества вершин для случайных графов с диапазоном вершин 100-550

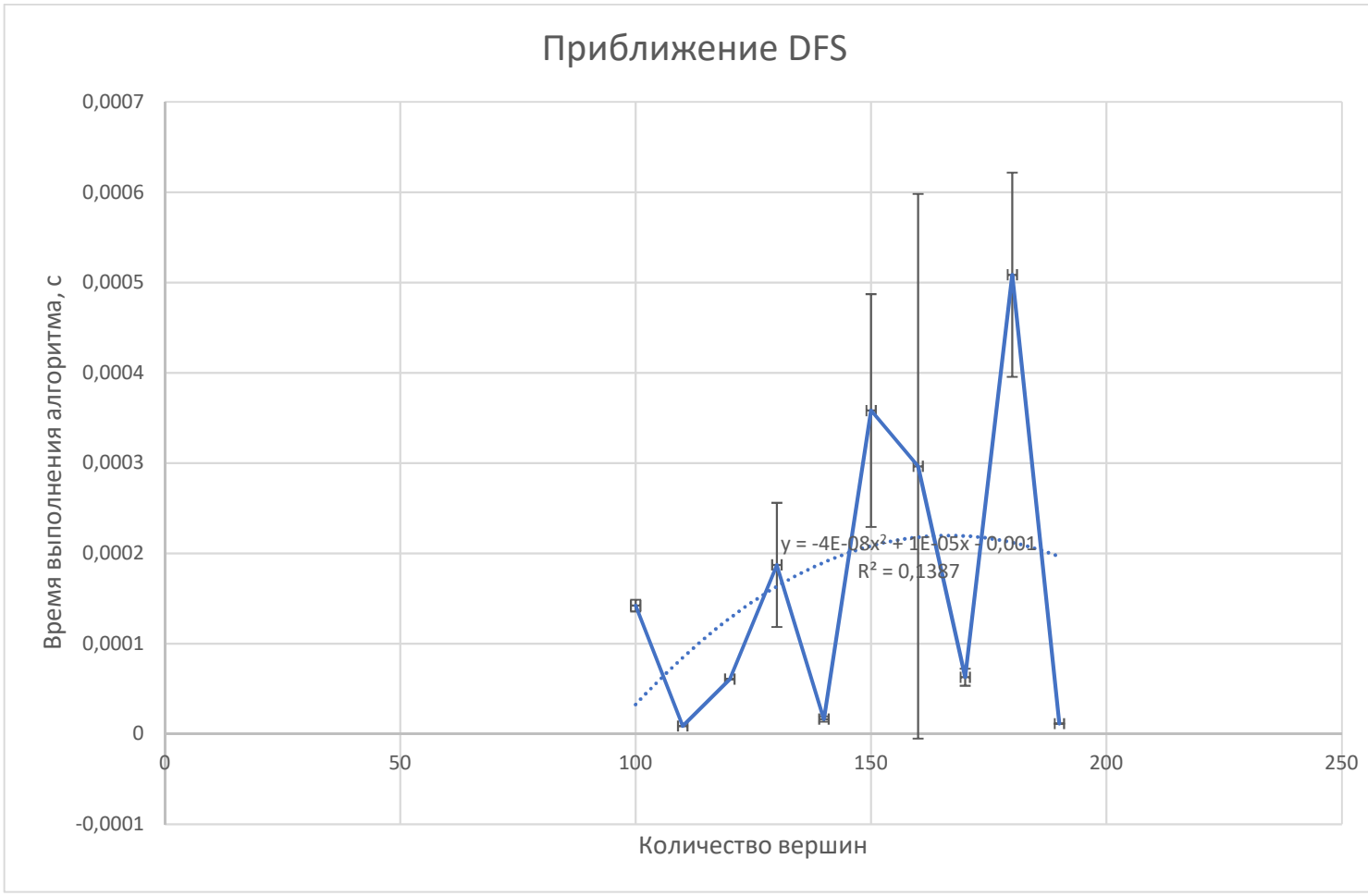

Рис. 2 График зависимости времени выполнения алгоритма «Поиск в глубину» от количества вершин для случайных графов с диапазоном вершин 100-190

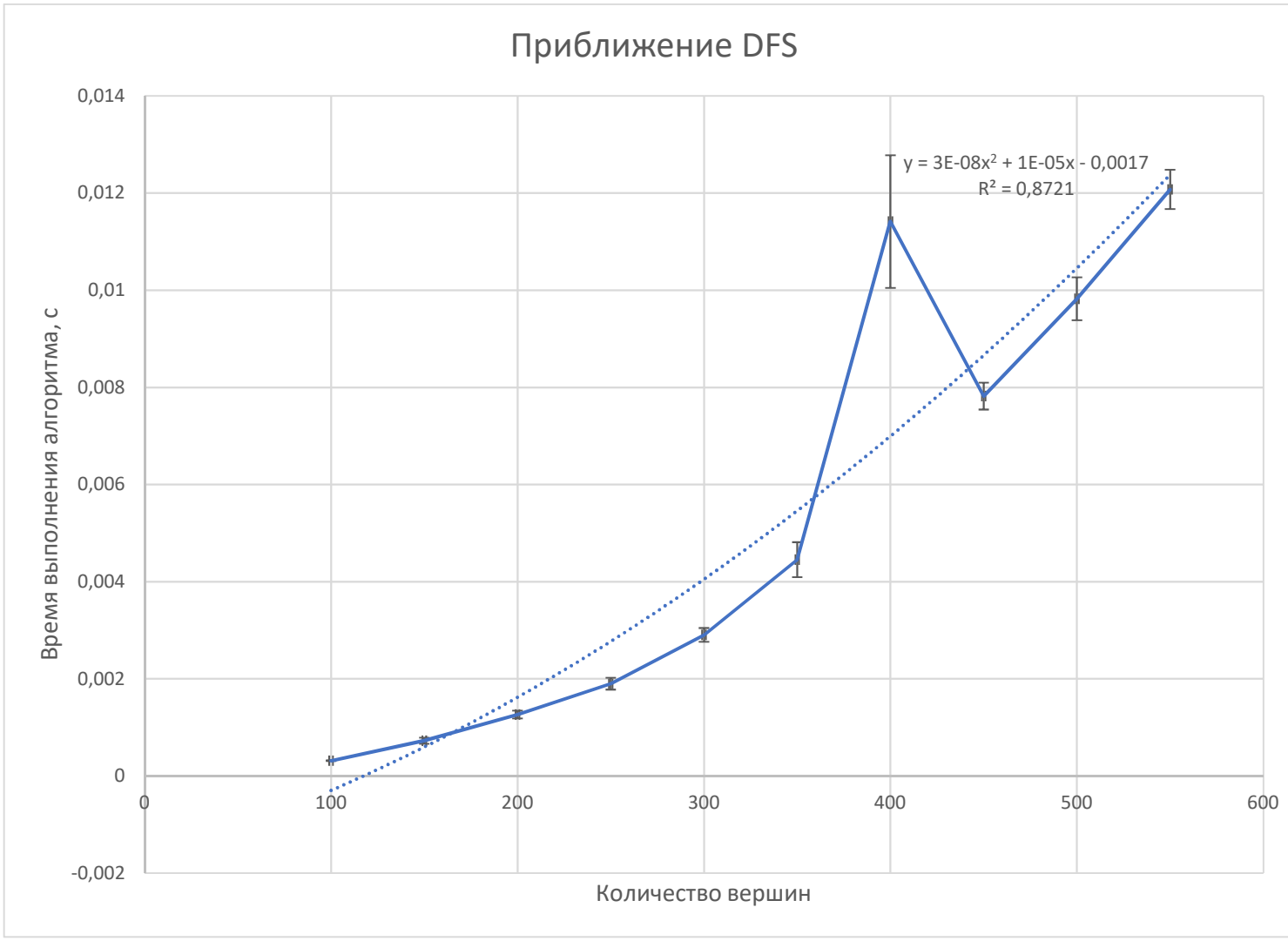

Рис. 3 График зависимости времени выполнения алгоритма «Поиск в глубину» от количества вершин для полных графов с диапазоном вершин 100-550

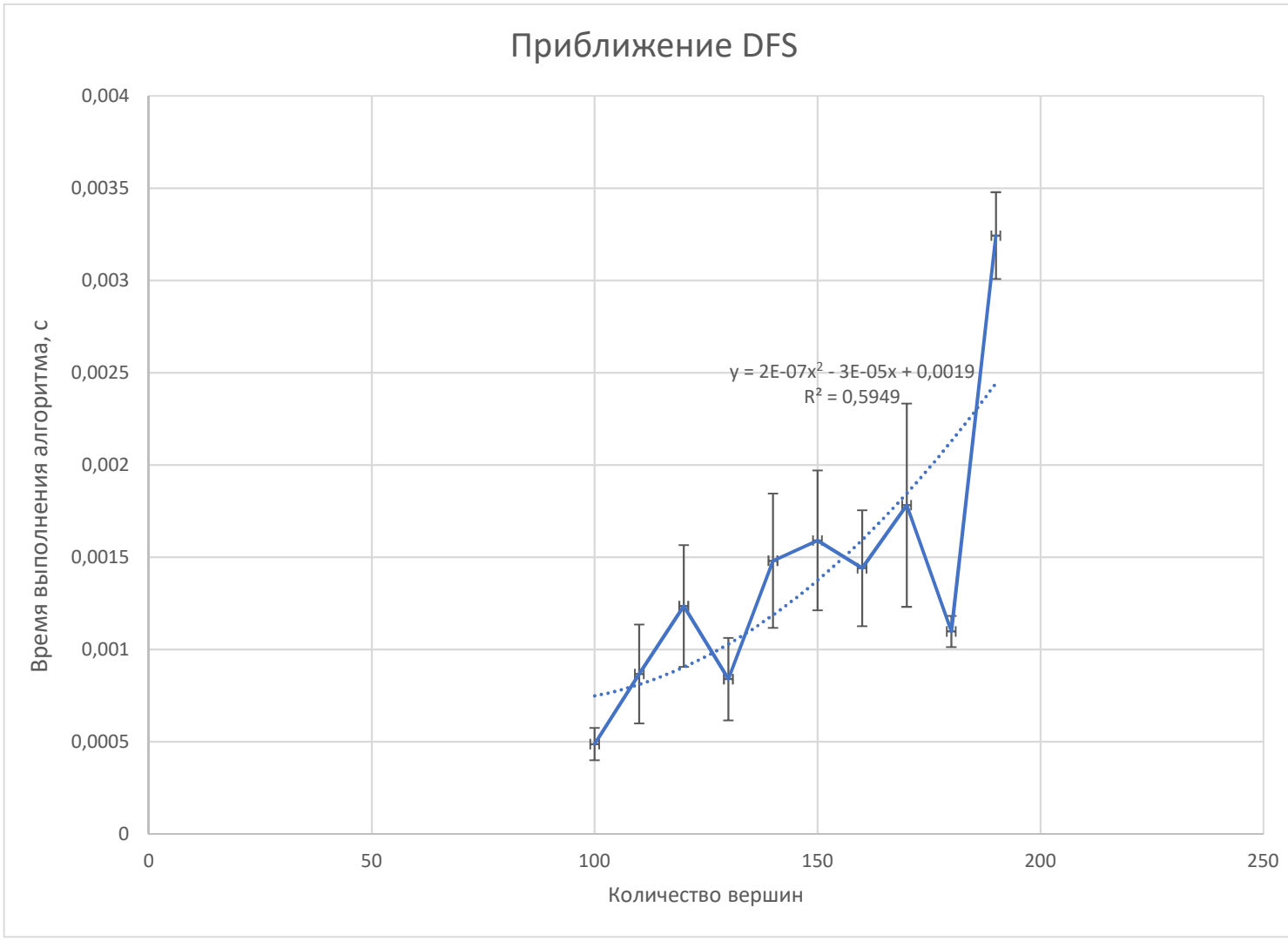

Рис. 4 График зависимости времени выполнения алгоритма «Поиск в глубину» от количества вершин для полных графов с диапазоном вершин 100-190

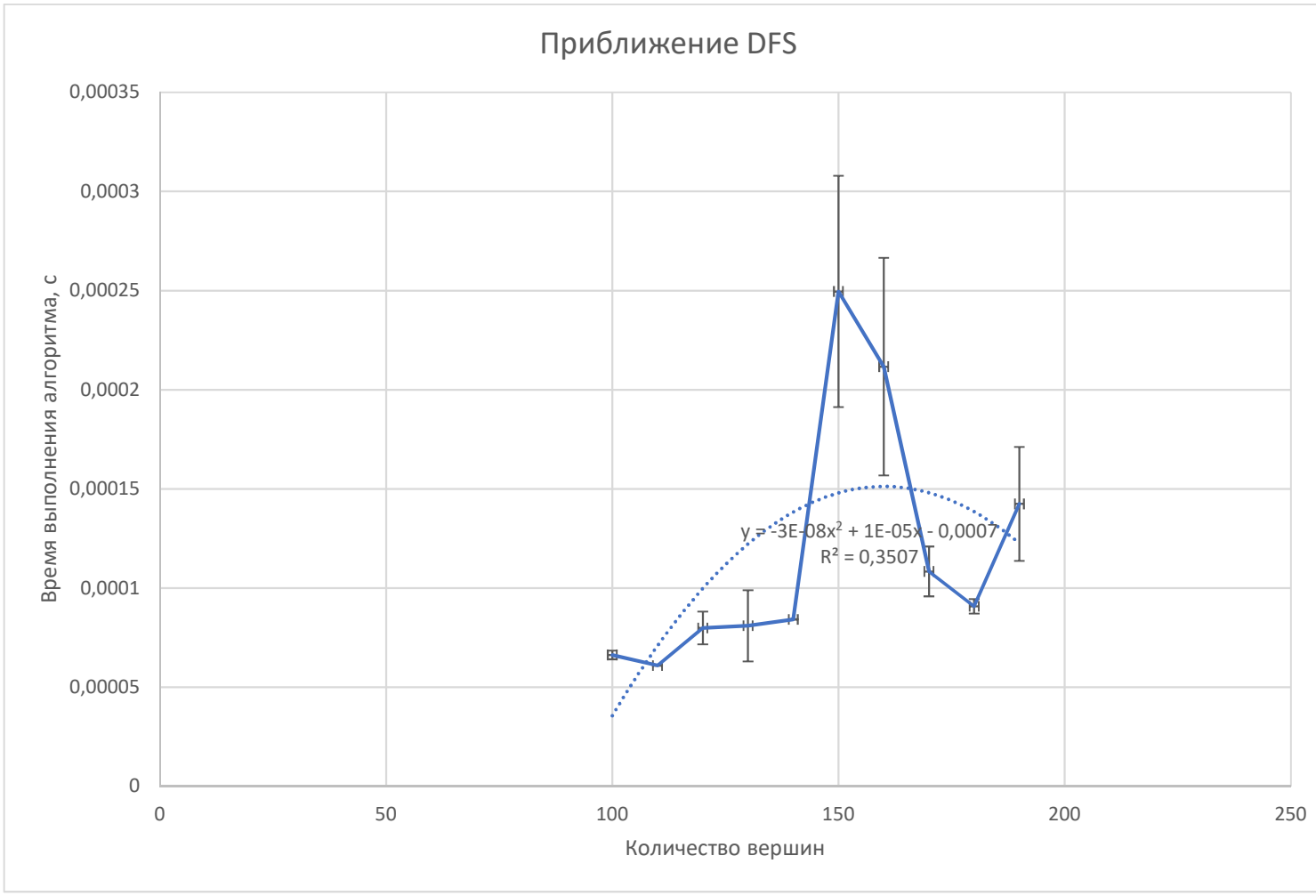

Рис. 5 График зависимости времени выполнения алгоритма «Поиск в глубину» от количества вершин для графов-цепочек с диапазоном вершин 100-190

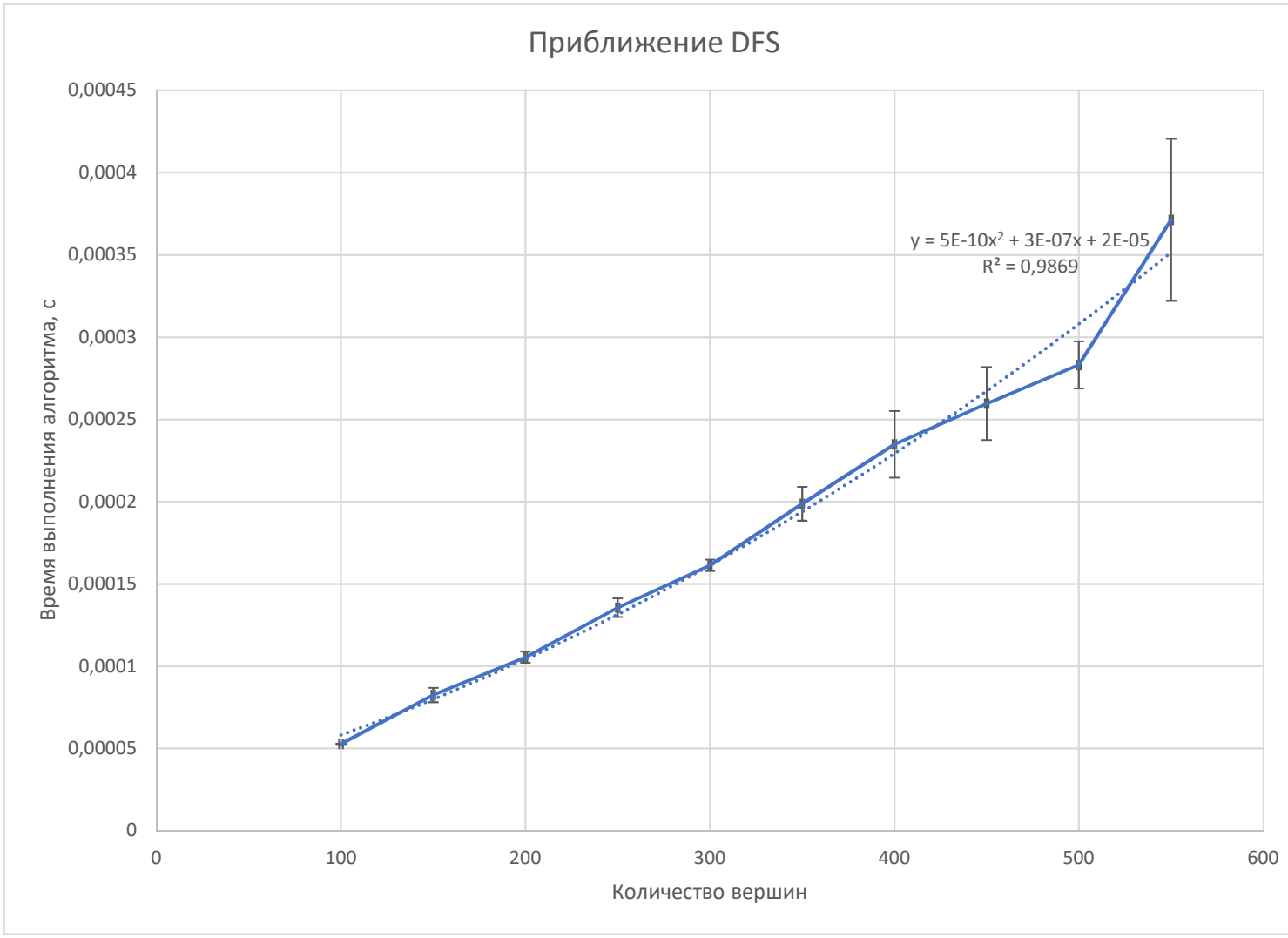

Рис. 6 График зависимости времени выполнения алгоритма «Поиск в глубину» от количества вершин для графов-цепочек с диапазоном вершин 100-550

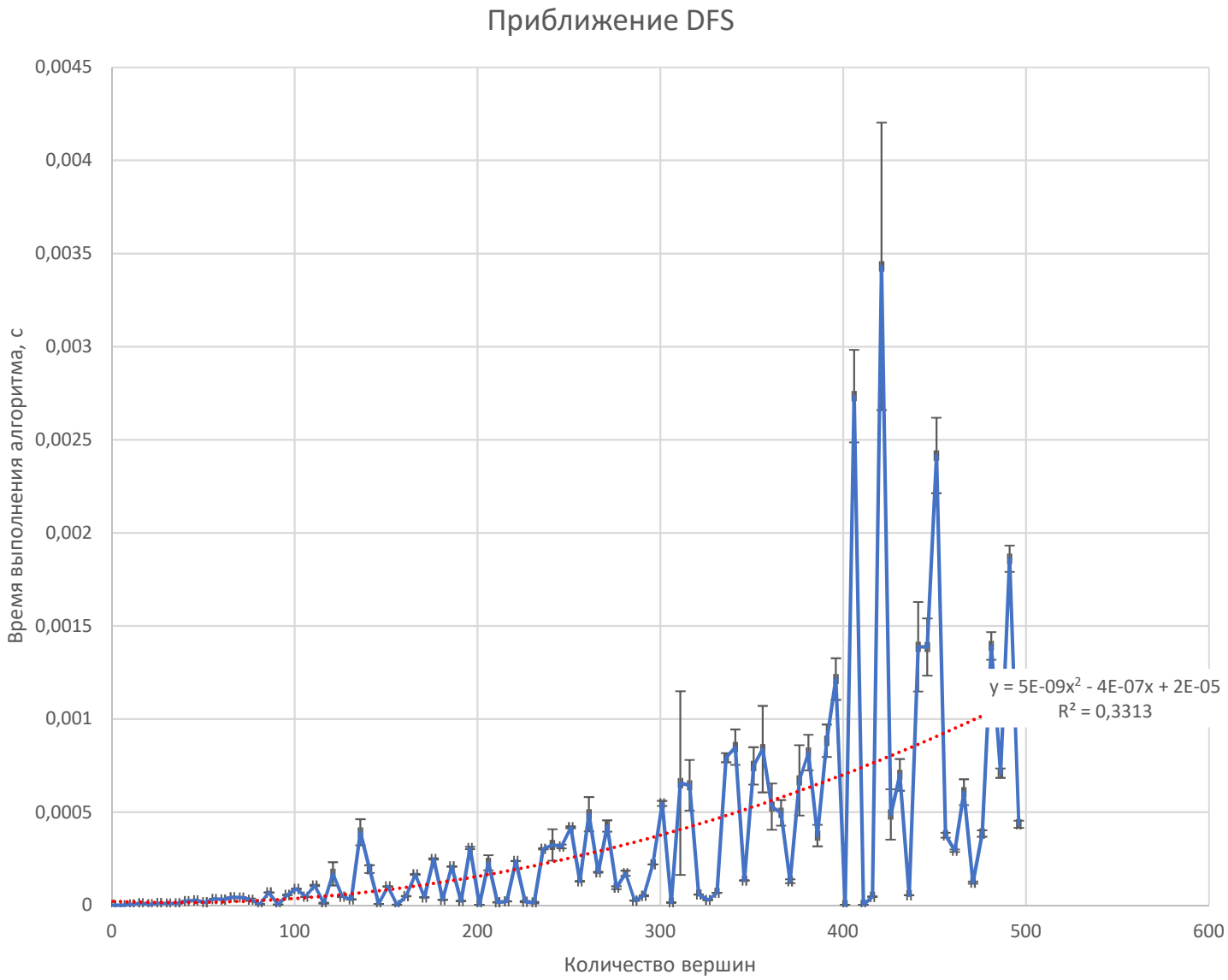

Рис. 7 График зависимости времени выполнения алгоритма «Поиск в глубину» от количества вершин для случайных графов с диапазоном вершин 1-496

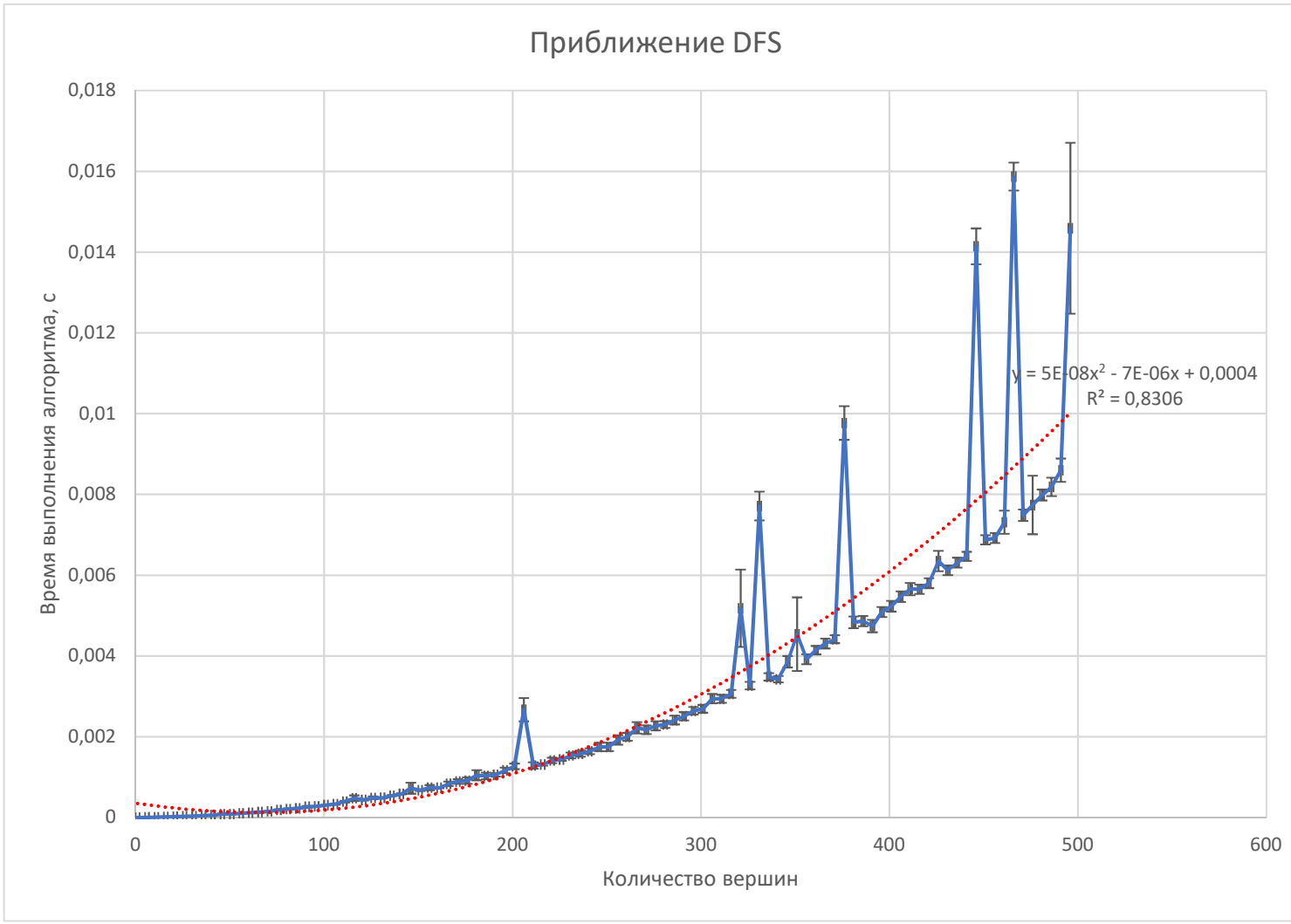

Рис. 8 График зависимости времени выполнения алгоритма «Поиск в глубину» от количества вершин для полных графов с диапазоном вершин 1-496

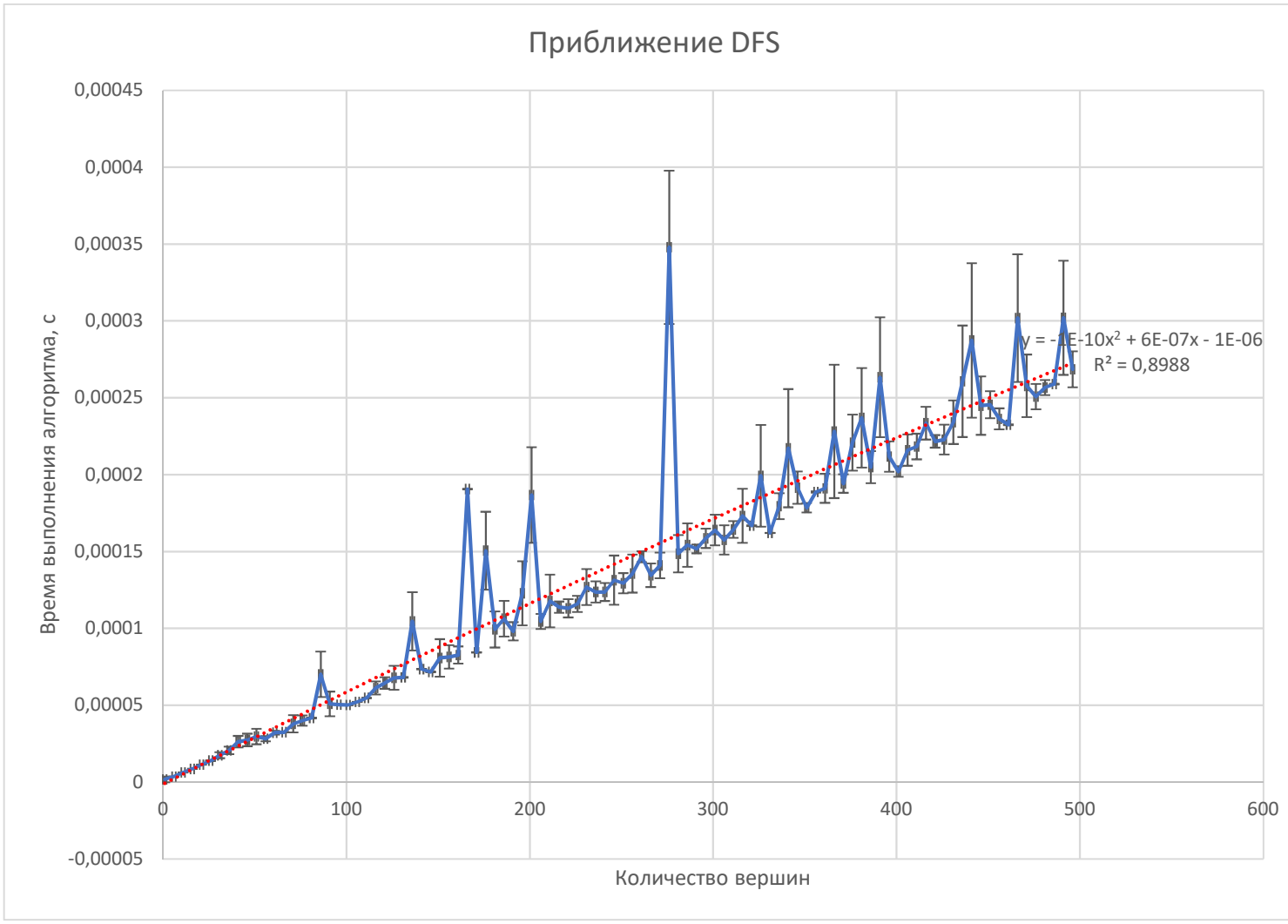

Рис. 9 График зависимости времени выполнения алгоритма «Поиск в глубину» от количества вершин для графов-цепочек с диапазоном вершин 1-496

Вывод

Стоит отметить, что алгоритмы BFS и DFS похожи как с теоретической точки зрения, так и с точки зрения реализации, вследствие чего, результаты, полученные при их тестировании, очень похожи. При анализе полученных графиков можно заметить, что на худших данных (полный граф) алгоритм DFS работает за время, которое хорошо приближается квадратичной функцией. На остальных данных, представленных на графиках, зависимость времени выполнения от количества вершин практически линейная. Из этого следует, что проведённые эксперименты коррелируют с теоретическими результатами, которые заключают, что время работы алгоритма $O(\boldsymbol{n}+\boldsymbol{m})$, где $\mathbf{n}$ – количество вершин в графе, $\mathbf{m}$ – количество рёбер. В графах количество рёбер не превосходит $\frac{\boldsymbol{n}\cdot(\boldsymbol{n}-\mathbf{1})}{\mathbf{2}}$ (это достигается в полных графах), а в графах-цепочках количество рёбер равно $\boldsymbol{n}-\mathbf{1}$, тогда теоретическая асимптотика становится линейной.

Некоторые неточности и скачки в измерениях обусловлены случайным выбором стартовой и конечной вершины в пути.

II. Тестирование по памяти

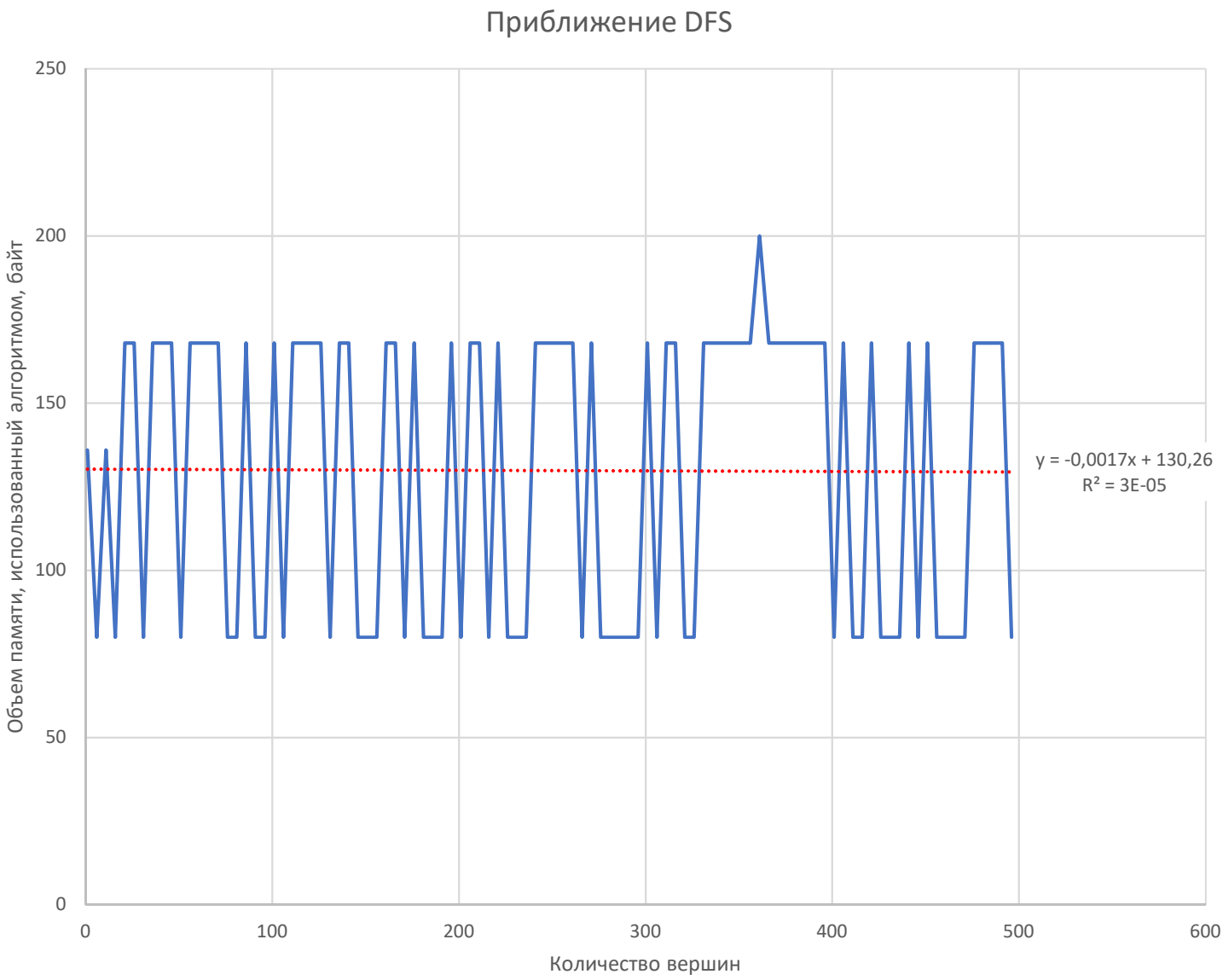

Рис. 1 График зависимости объёма памяти, использованной алгоритмом «Поиск в глубину» от количества вершин в случайном графе с диапазоном вершин 1-496

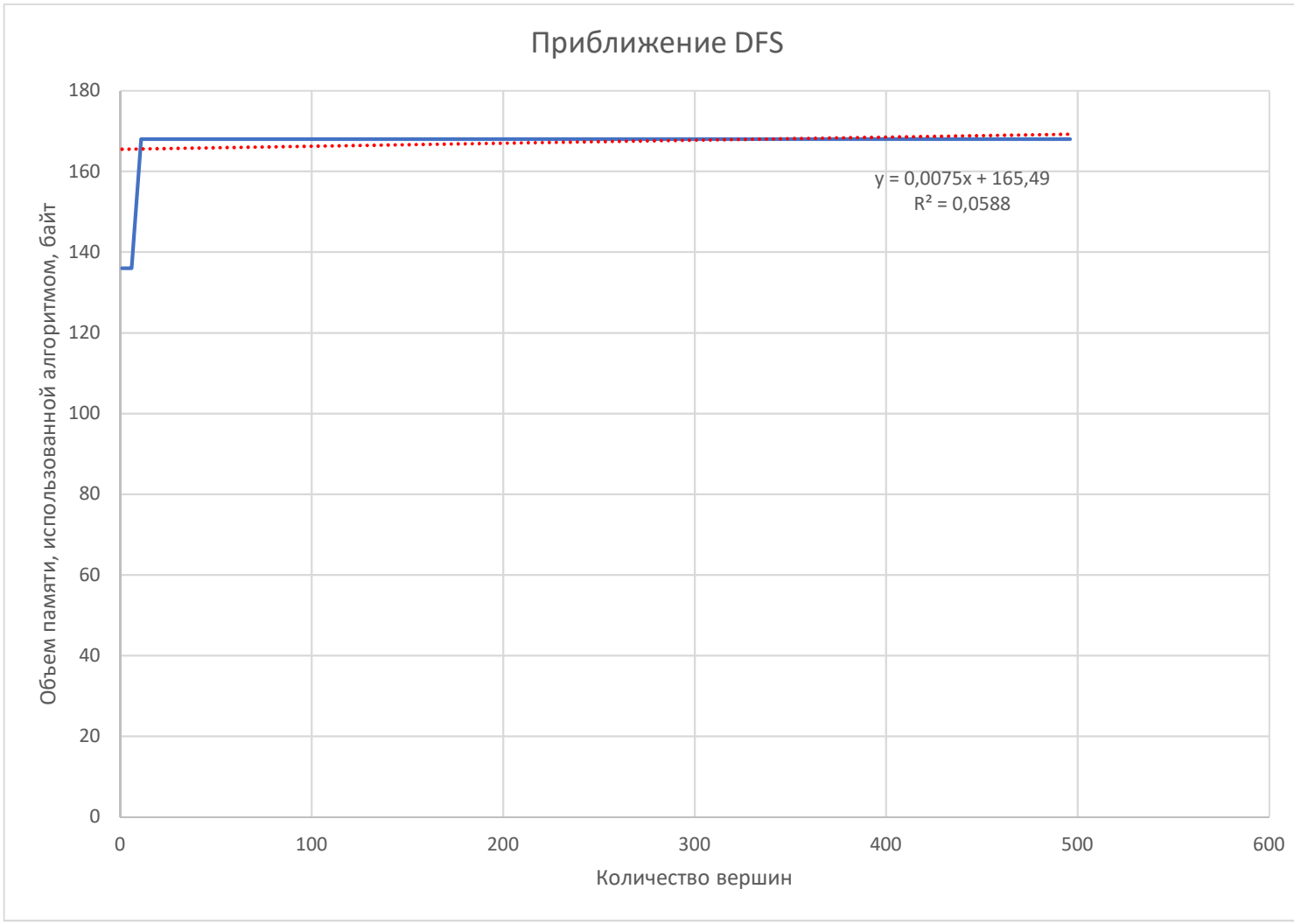

Рис. 2 График зависимости объёма памяти, использованной алгоритмом «Поиск в глубину» от количества вершин в полном графе с диапазоном вершин 1-496

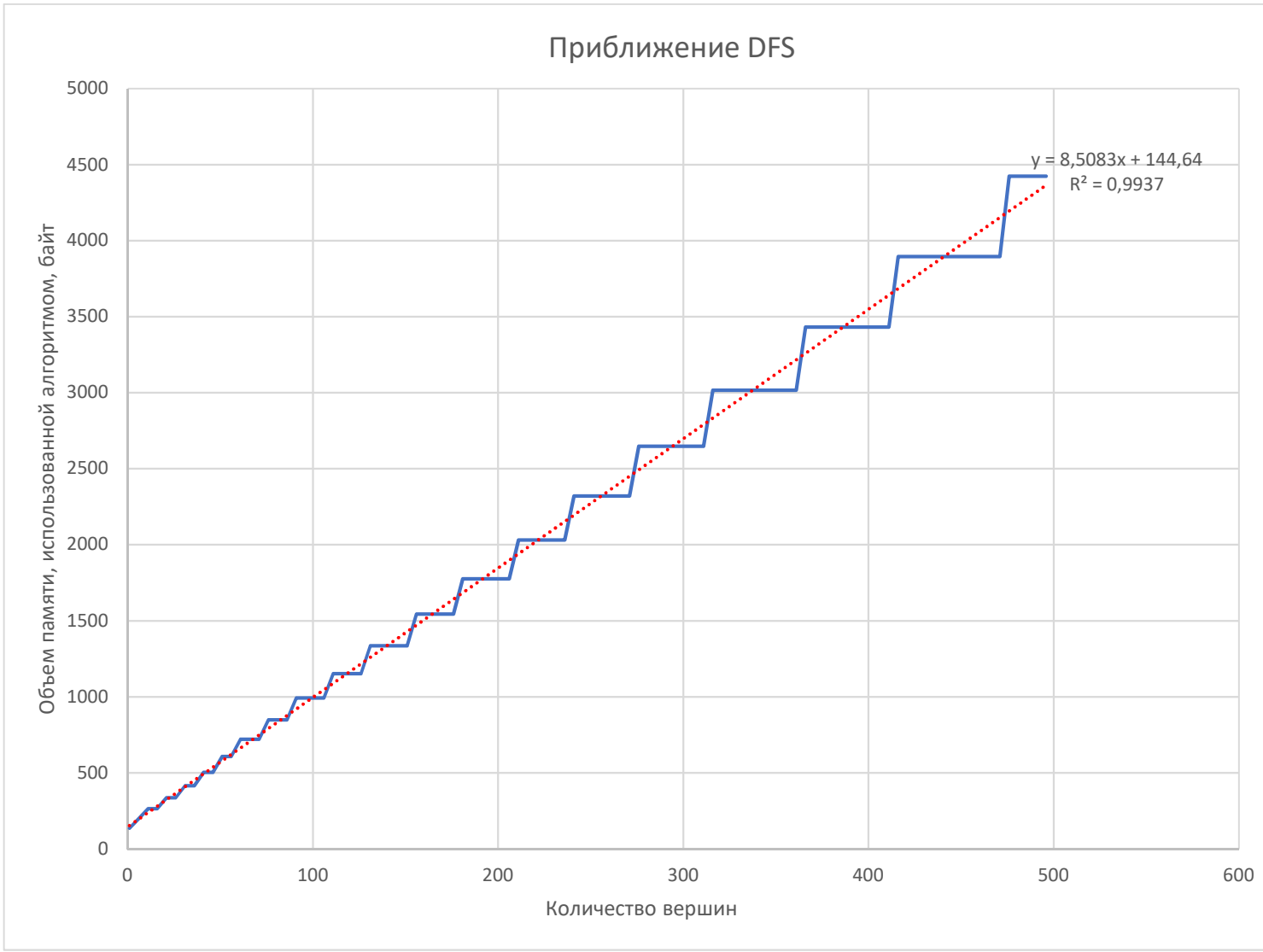

Рис. 3 График зависимости объёма памяти, использованной алгоритмом «Поиск в глубину» от количества вершин в графе-цепочке с диапазоном вершин 1-496

Вывод

Из полученных результатов очевидно, что при работе алгоритма «Поиск в глубину» количество используемой им памяти не зависит от числа вершин напрямую. Однако, количество используемой алгоритмом памяти зависит от размерности пространства решений (поскольку при тестировании данного алгоритма на графах-цепочках график имеет лестничнообразный вид, а в качестве точек, между которыми мы ищем путь берутся крайние вершины цепочки, а значит при увеличении размера графа увеличивается пространство решений). Так как график на рисунке 3 имеет лестничнообразный вид, то можно сделать вывод о том, что его лучше всего приближать линейной функцией. Таким образом мы получаем результат, что объем памяти, используемый алгоритмом, линейно зависит от размера пространства решений.

3. Алгоритм Дейкстры

I. Тестирование по времени

Рис. 1 График зависимости времени выполнения алгоритма Дейкстры от количества вершин для случайных графов с диапазоном вершин 100-550

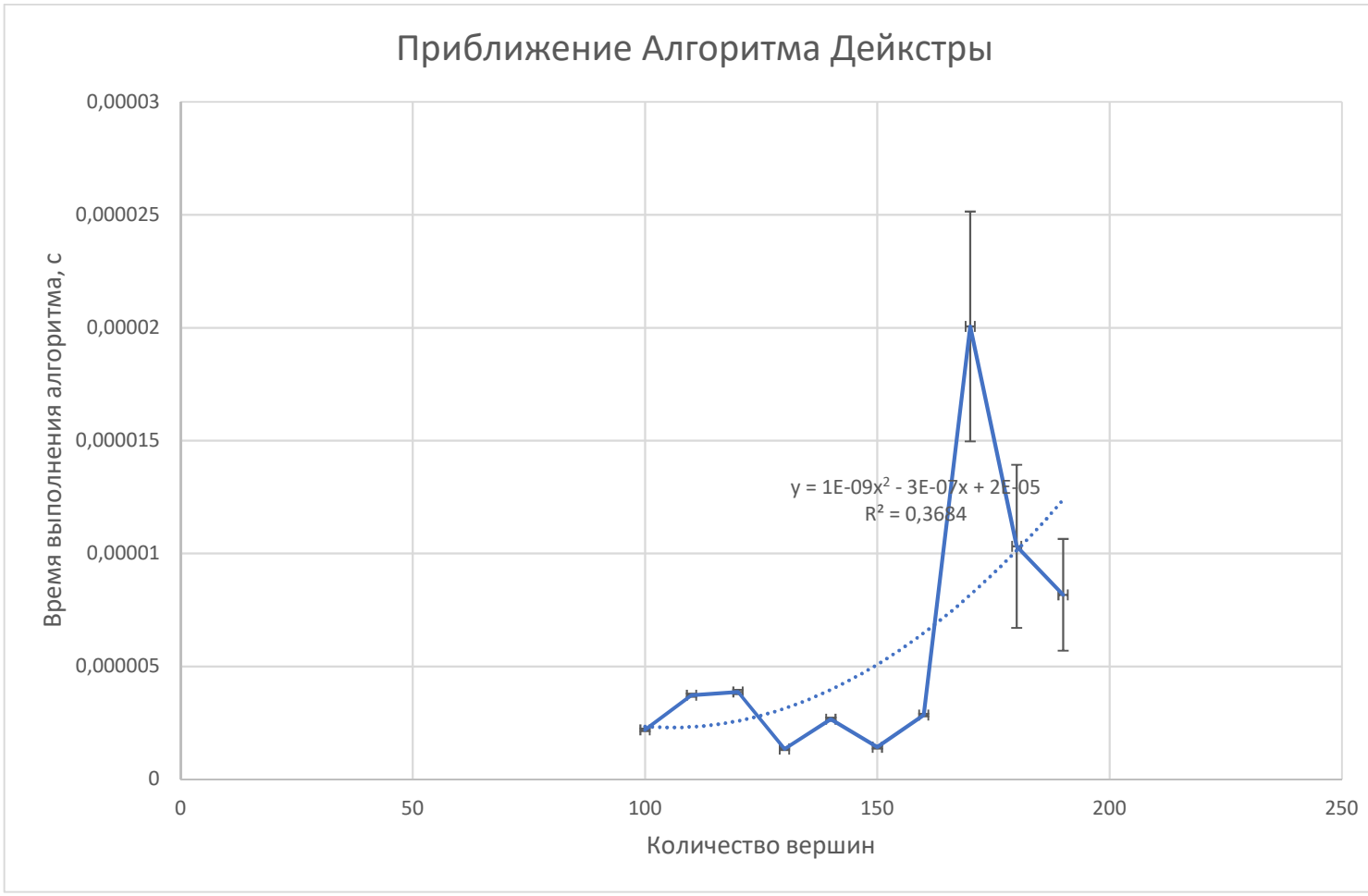

Рис. 2 График зависимости времени выполнения алгоритма Дейкстры от количества вершин для случайных графов с диапазоном вершин 100-190

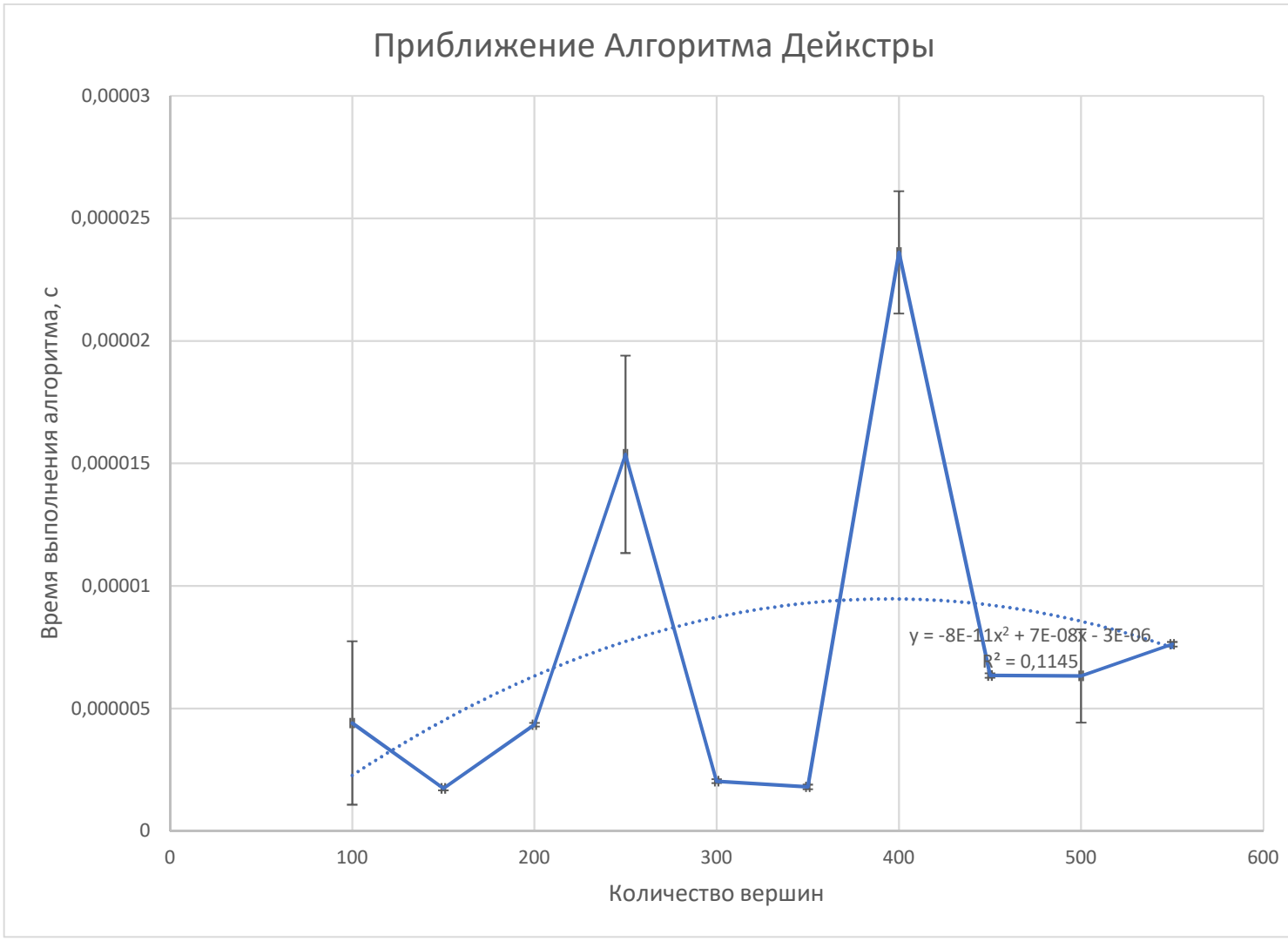

Рис. 3 График зависимости времени выполнения алгоритма Дейкстры от количества вершин для полных графов с диапазоном вершин 100-550

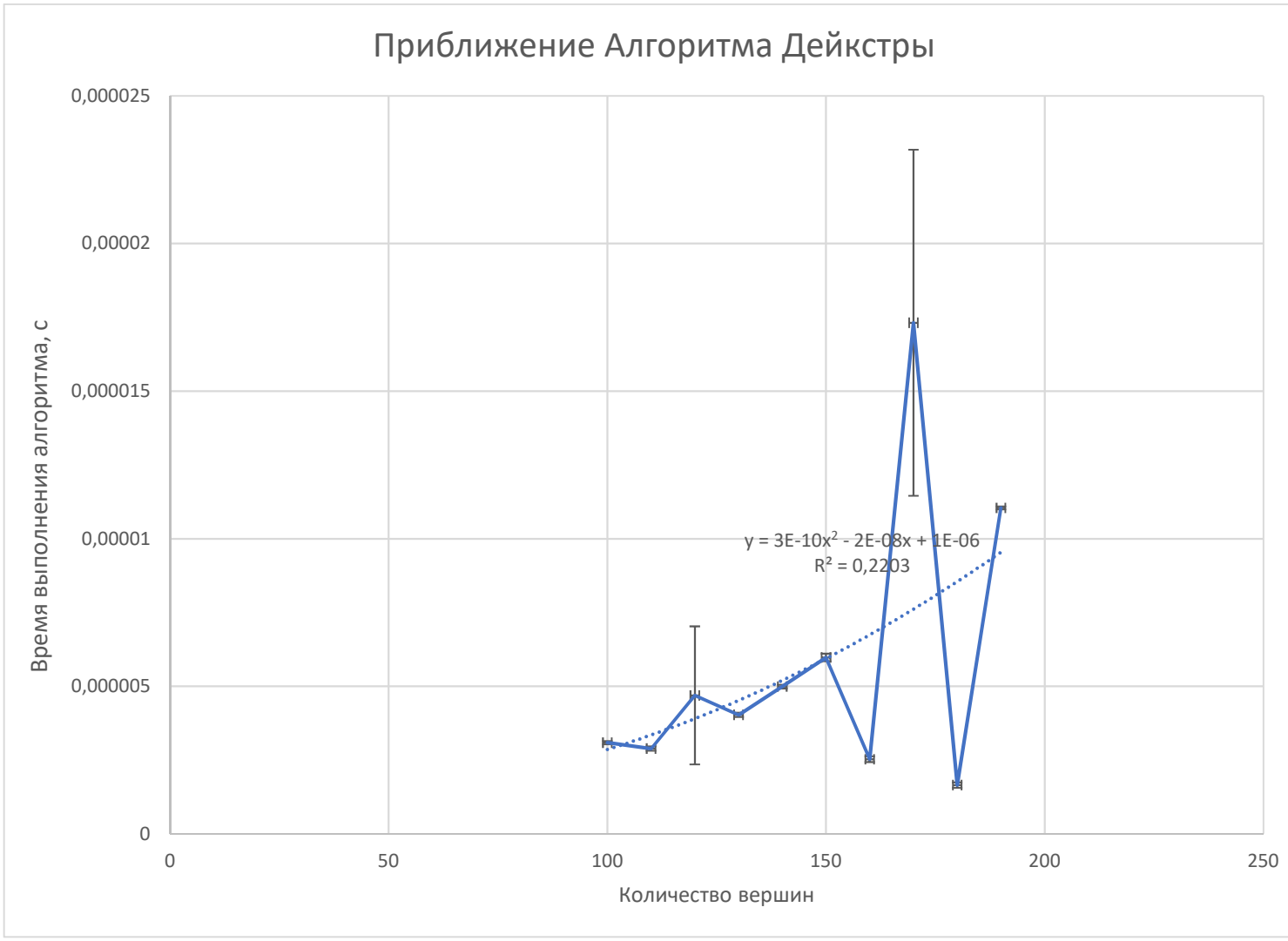

Рис. 4 График зависимости времени выполнения алгоритма Дейкстры от количества вершин для полных графов с диапазоном вершин 100-190

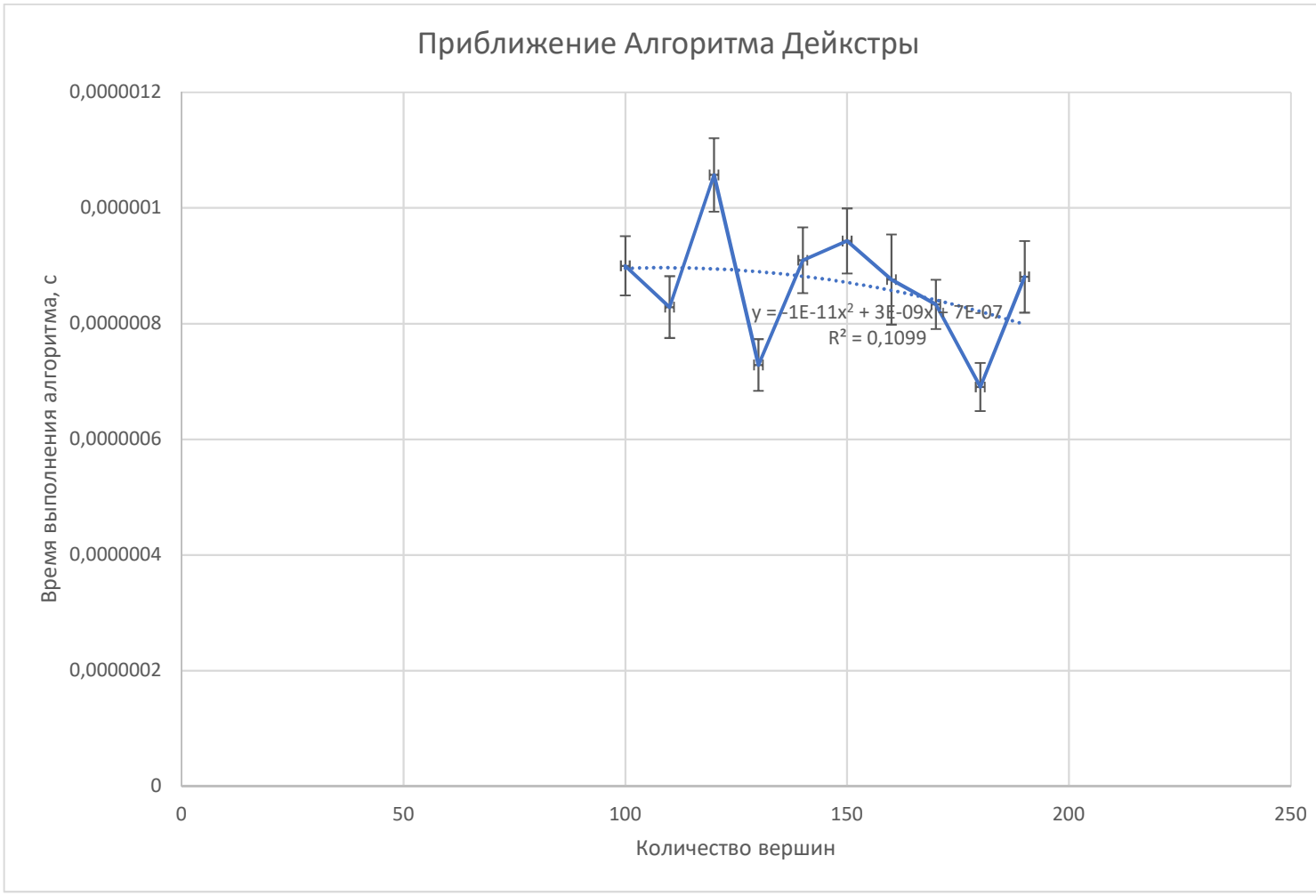

Рис. 5 График зависимости времени выполнения алгоритма «Поиск в глубину» от количества вершин для графов-цепочек с диапазоном вершин 100-190

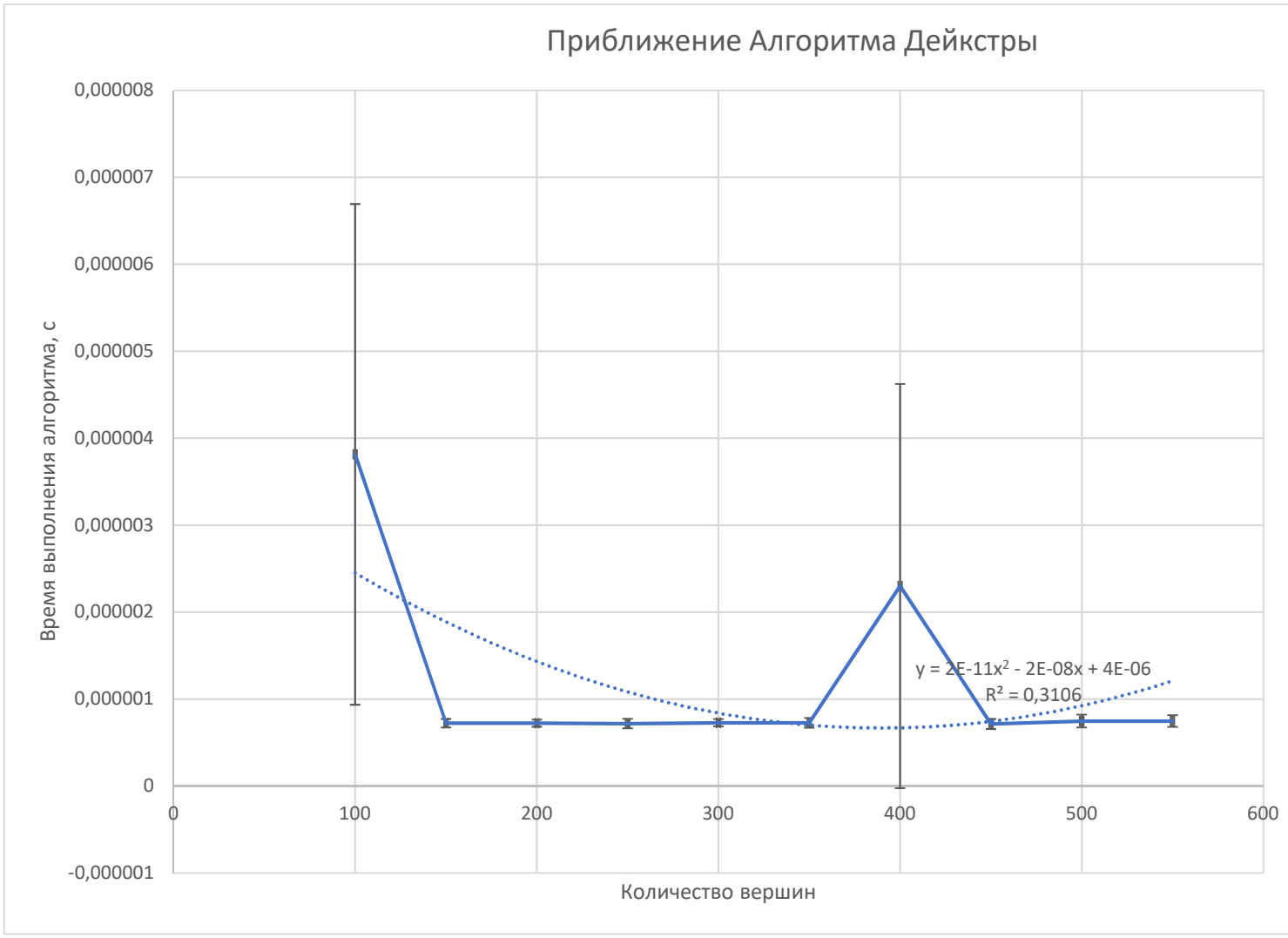

Рис. 6 График зависимости времени выполнения алгоритма Дейкстры от количества вершин для графов-цепочек с диапазоном вершин 100-550

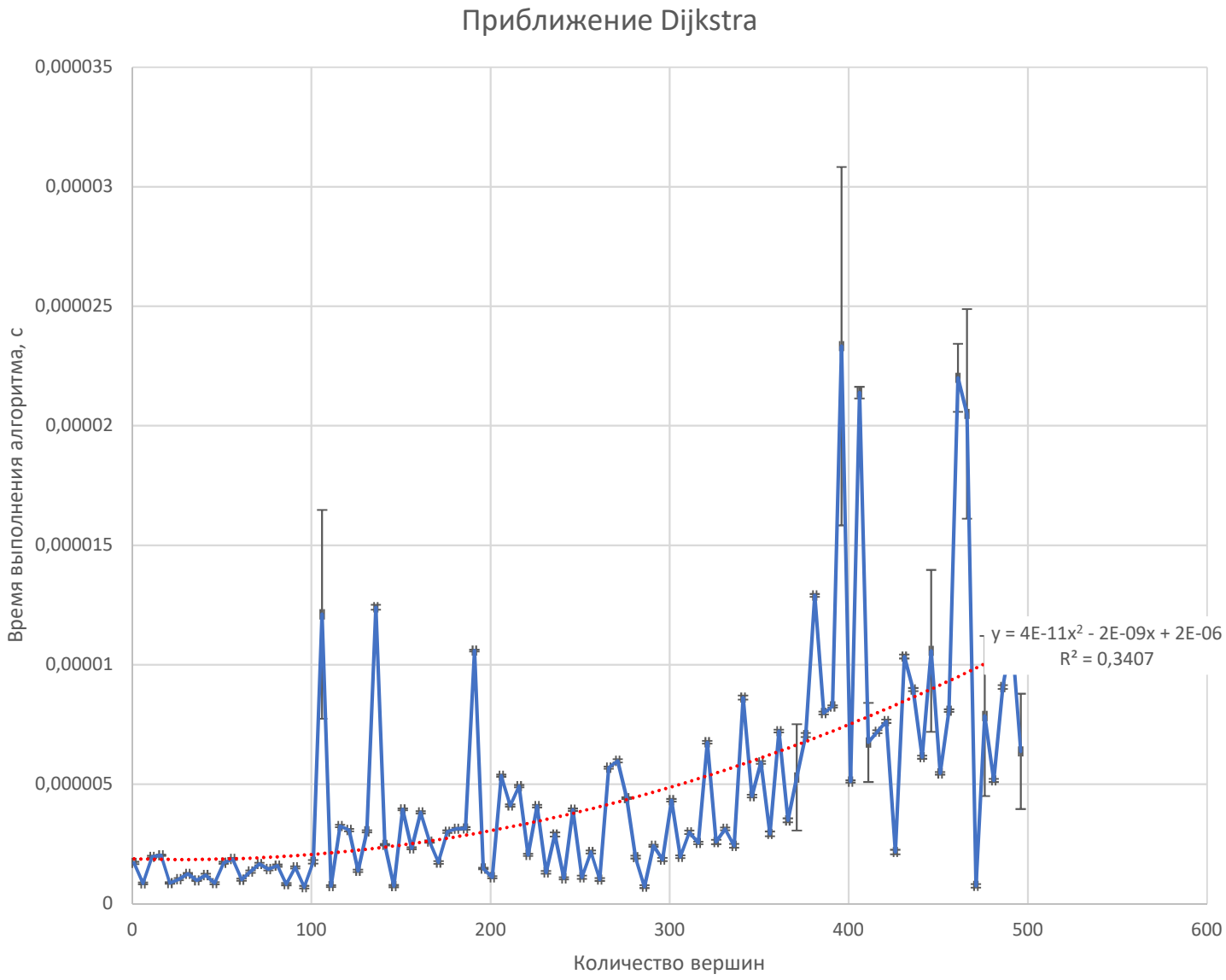

Рис. 7 График зависимости времени выполнения алгоритма Дейкстры от количества вершин для случайных графов с диапазоном вершин 1-496

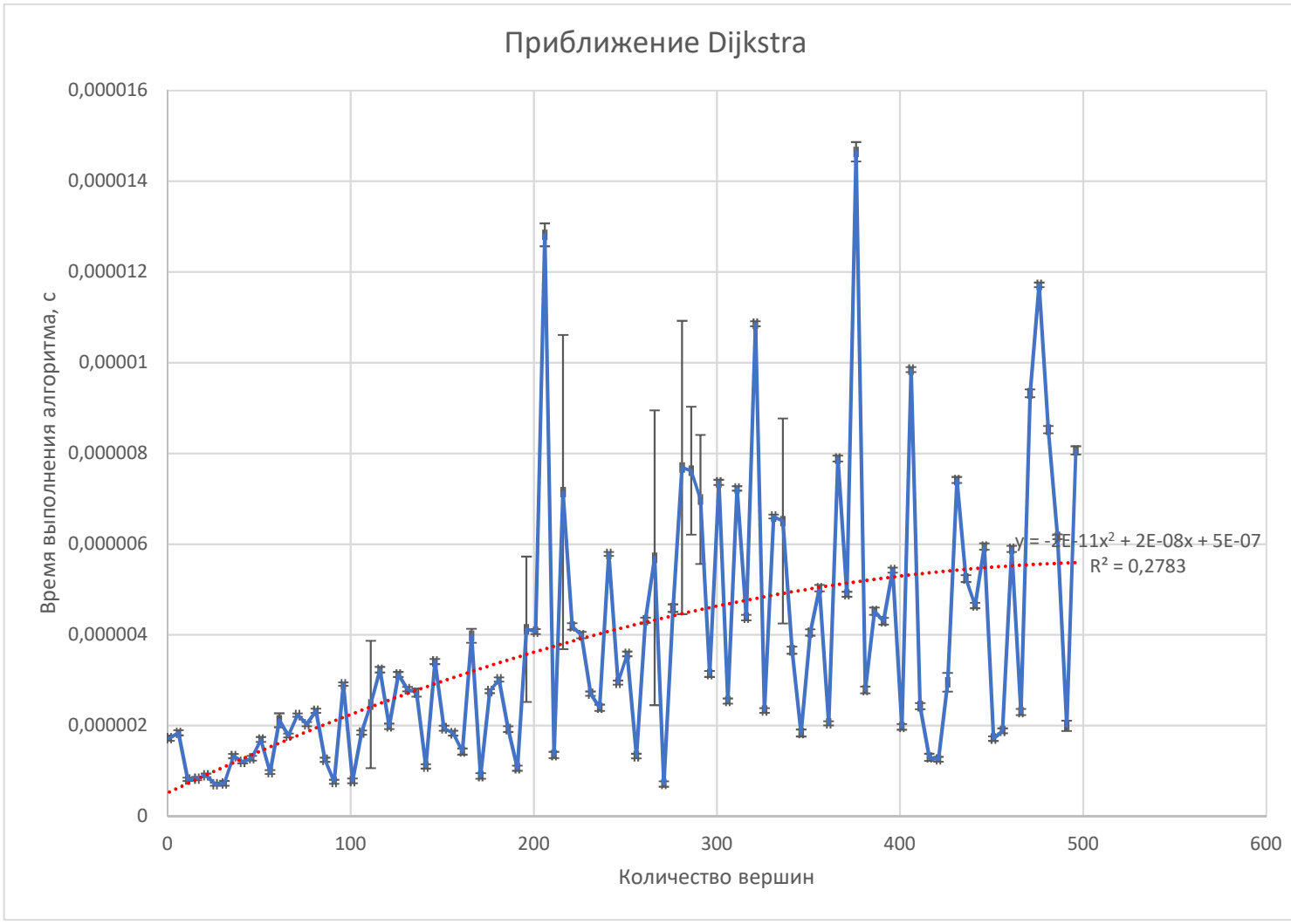

Рис. 8 График зависимости времени выполнения алгоритма Дейкстры от количества вершин для полных графов с диапазоном вершин 1-496

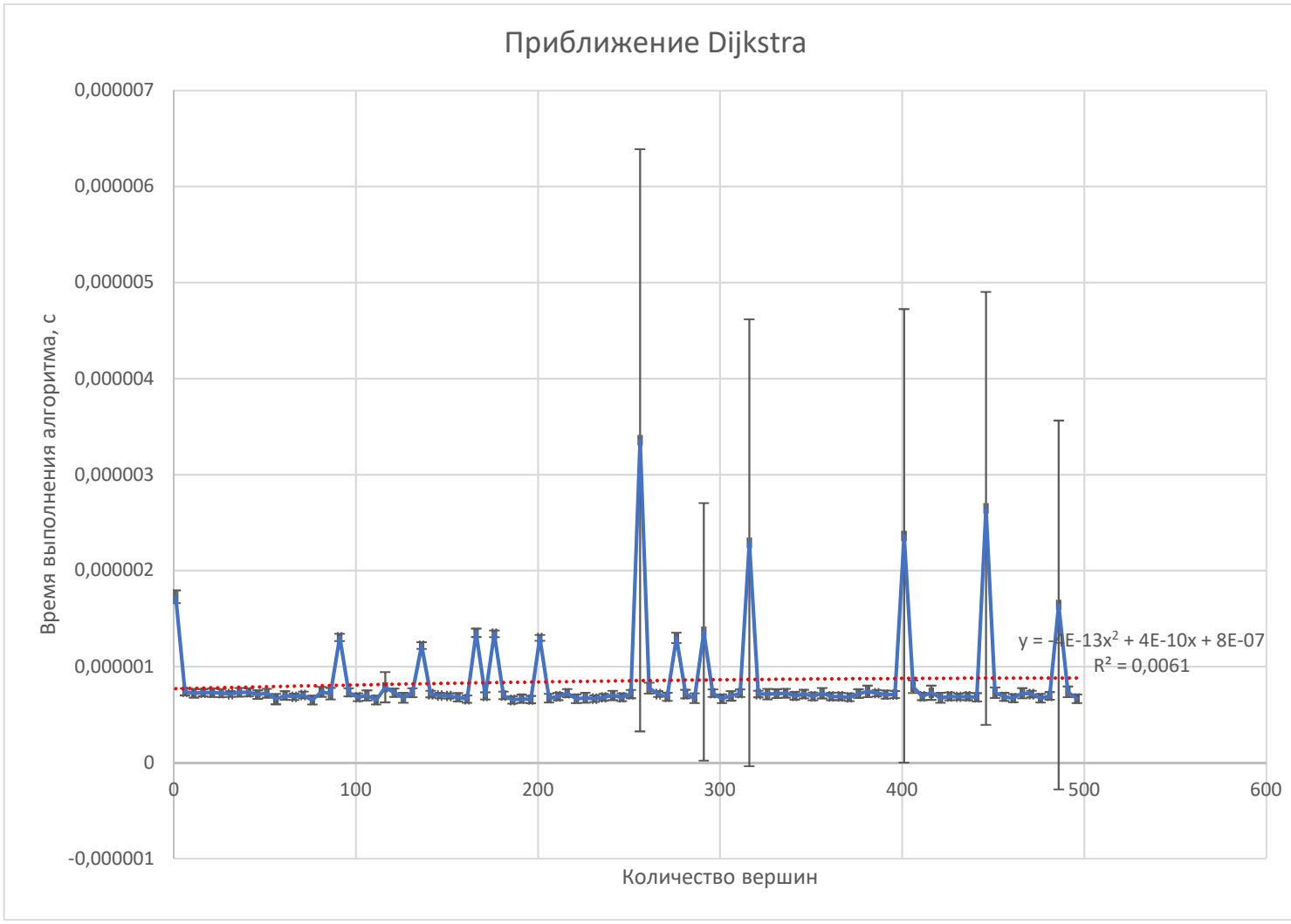

Рис. 9 График зависимости времени выполнения алгоритма Дейкстры от количества вершин для случайных графов с диапазоном вершин 1-496

Вывод

С точки зрения теории алгоритм Дейкстры имеет асимптотику $\boldsymbol{O(n^2 + m)}$, где $\mathbf{n}$ – количество вершин в графе, $\mathbf{m}$ – количество рёбер. В графах количество рёбер не превосходит $\frac{\boldsymbol{n \cdot (n-1)}}{\mathbf{2}}$ (это достигается в полных графах), таким образом, асимптотика времени работы этого алгоритма всегда квадратична. Имеет смысл приближать полученные при тестировании измерения квадратичной функцией. В отличие от BFS и DFS обход в алгоритме Дейкстры определяется стартовой вершиной, конечной вершиной и весом рёбер. Таким образом, измерения более восприимчивы к случайности выбора вершин. На графиках это видно за счёт более резких перепадов. Поэтому приближение здесь менее точно, однако, в некоторых случаях графики довольно хорошо приближаются квадратичной функцией, что соответствует теоретическому анализу. Таким образом результаты этого тестирования можно считать вполне успешными.

II. Тестирование по памяти

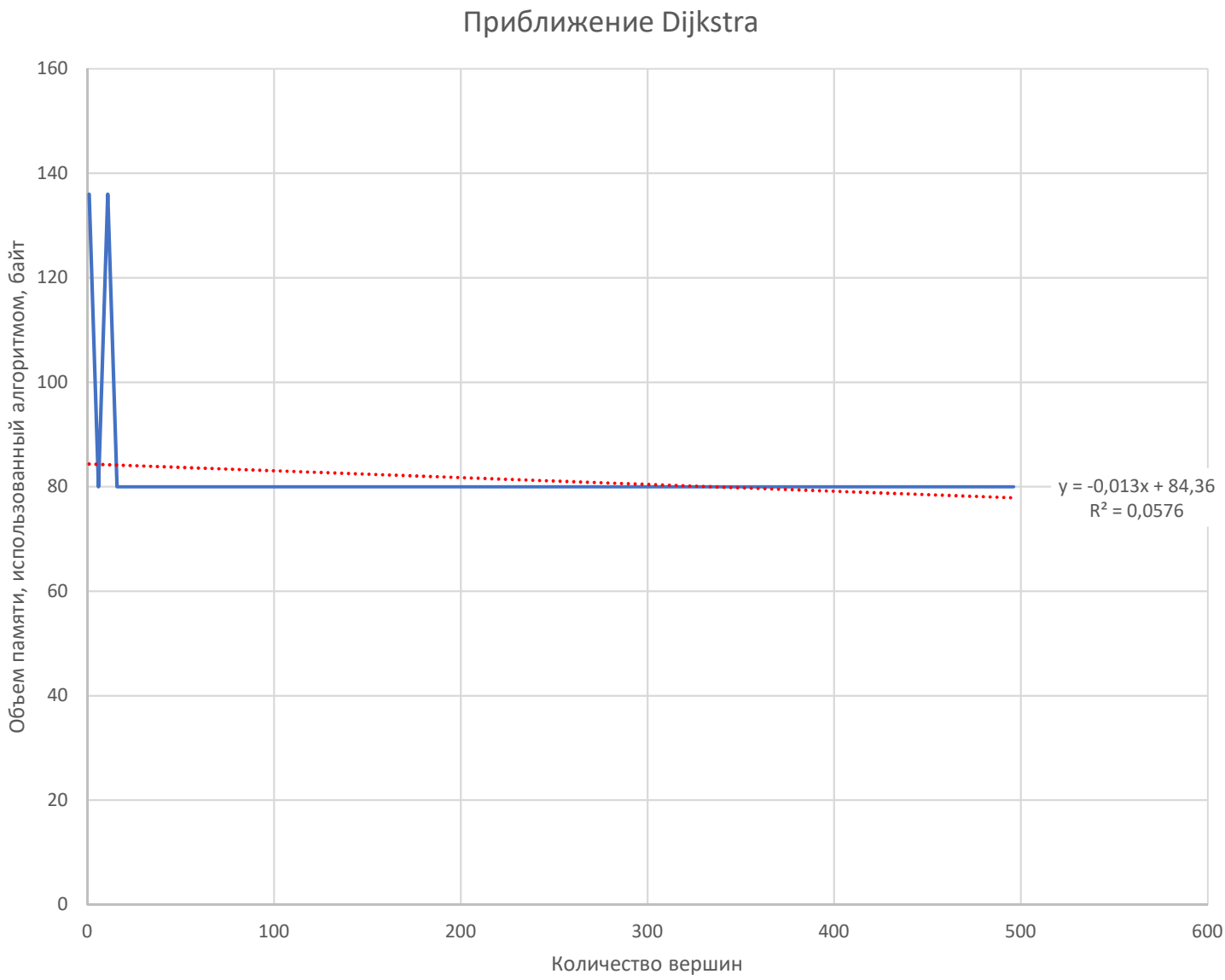

Рис. 1 График зависимости объёма памяти, использованной алгоритмом Дейкстры от количества вершин в случайном графе с диапазоном вершин 1-496

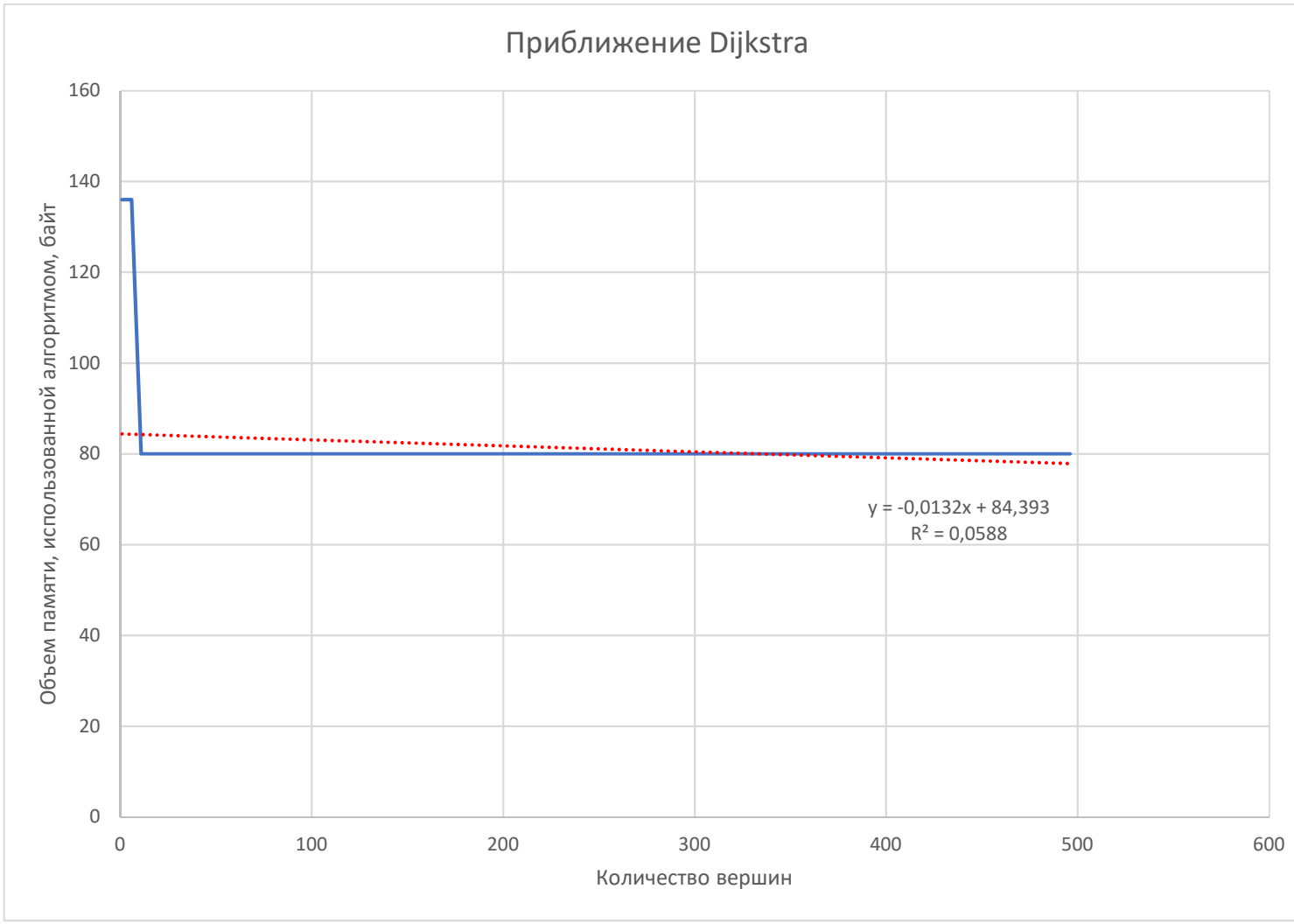

Рис. 2 График зависимости объёма памяти, использованной алгоритмом Дейкстры от количества вершин в полном графе с диапазоном вершин 1-496

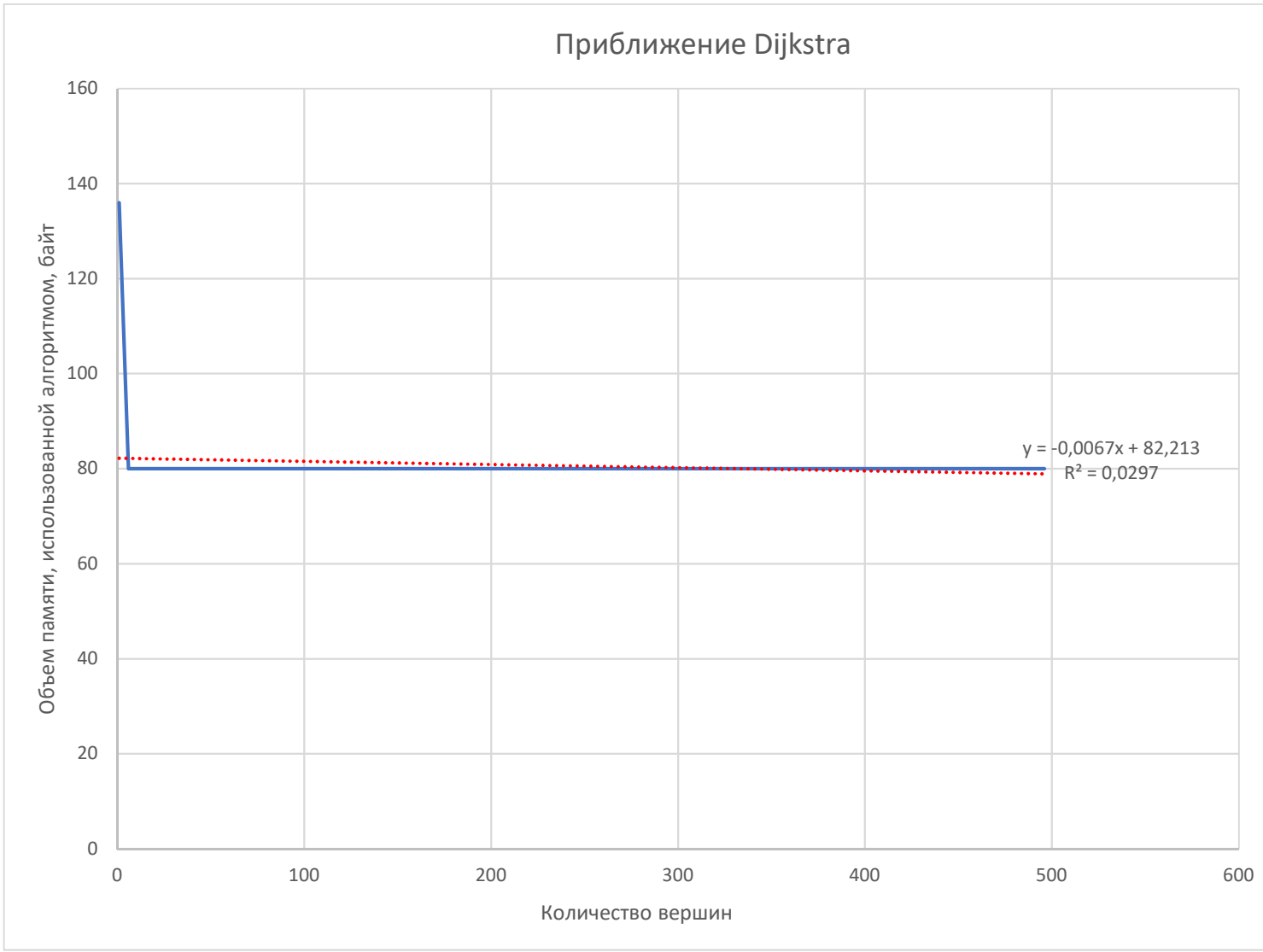

Рис. 3 График зависимости объёма памяти, использованной алгоритмом Дейкстры от количества вершин в графе-цепочке с диапазоном вершин 1-496

Вывод

Из полученных результатов очевидно, что при работе алгоритма Дейкстры количество используемой им памяти не зависит от числа вершин.

4. **Алгоритм Форда-Беллмана**

I. **Тестирование по времени**

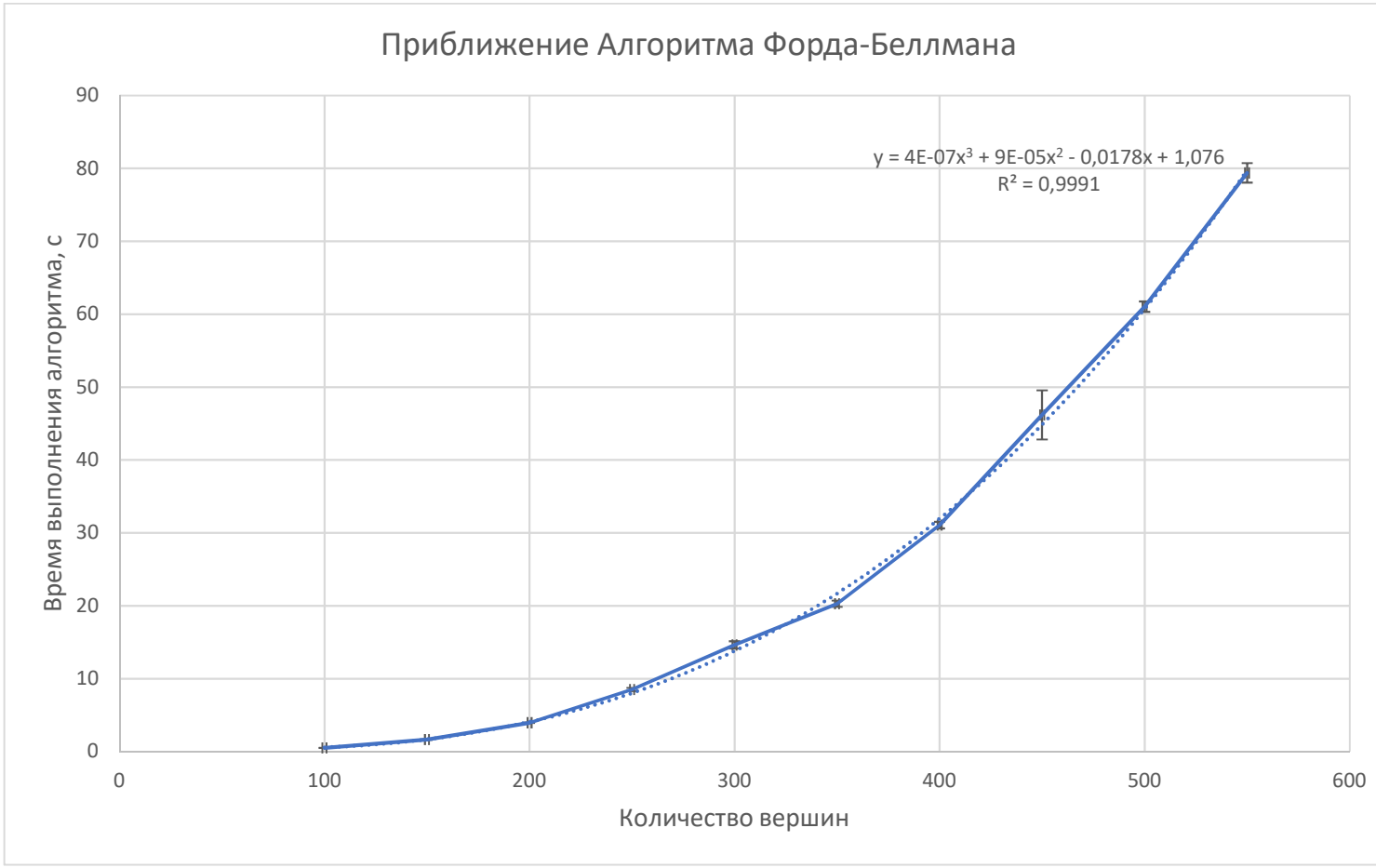

Рис. 1 График зависимости времени выполнения алгоритма Форда-Беллмана от количества вершин для случайных графов с диапазоном вершин 100-550

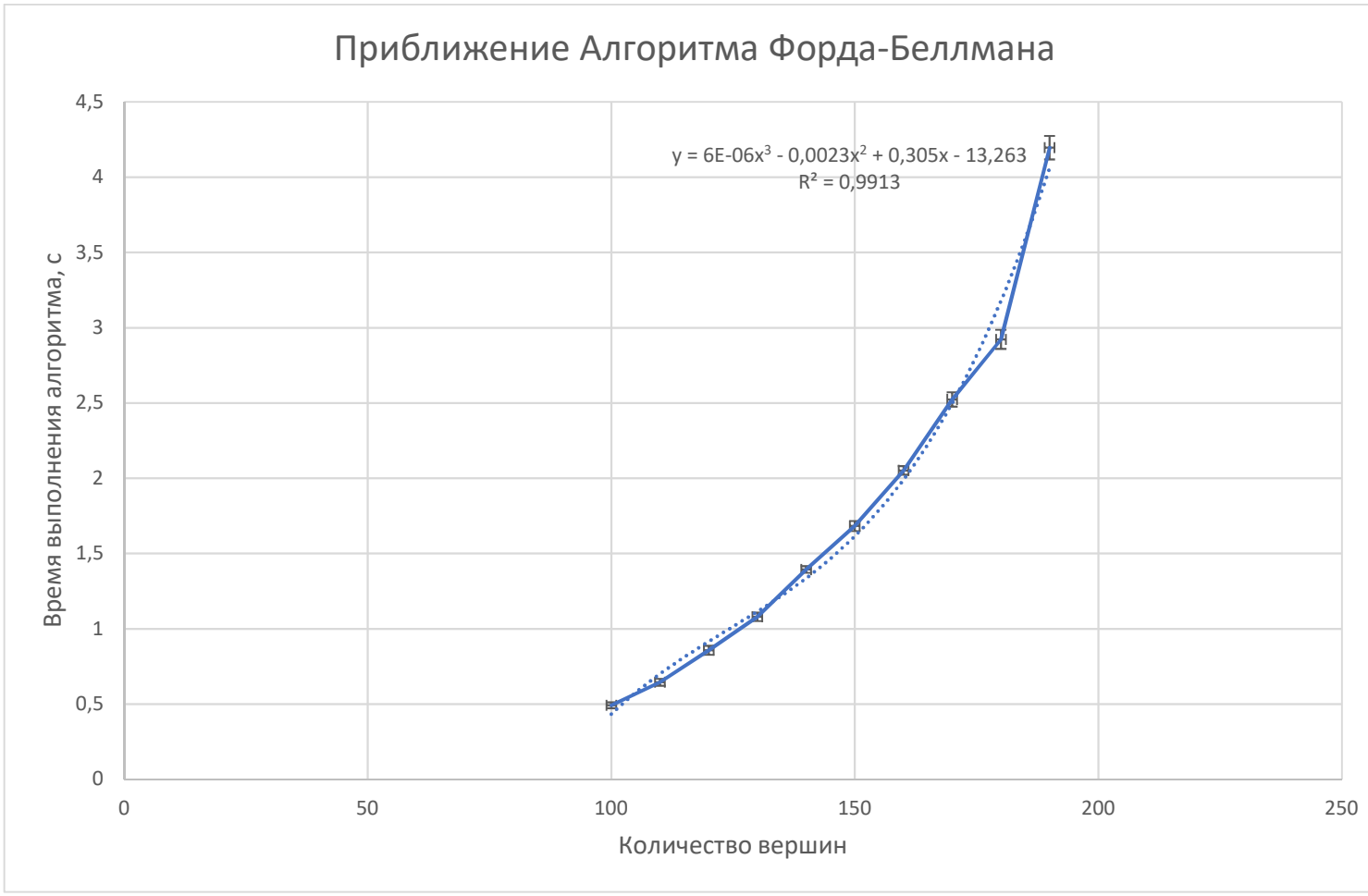

Рис. 2 График зависимости времени выполнения алгоритма Форда-Беллмана от количества вершин для случайных графов с диапазоном вершин 100-190

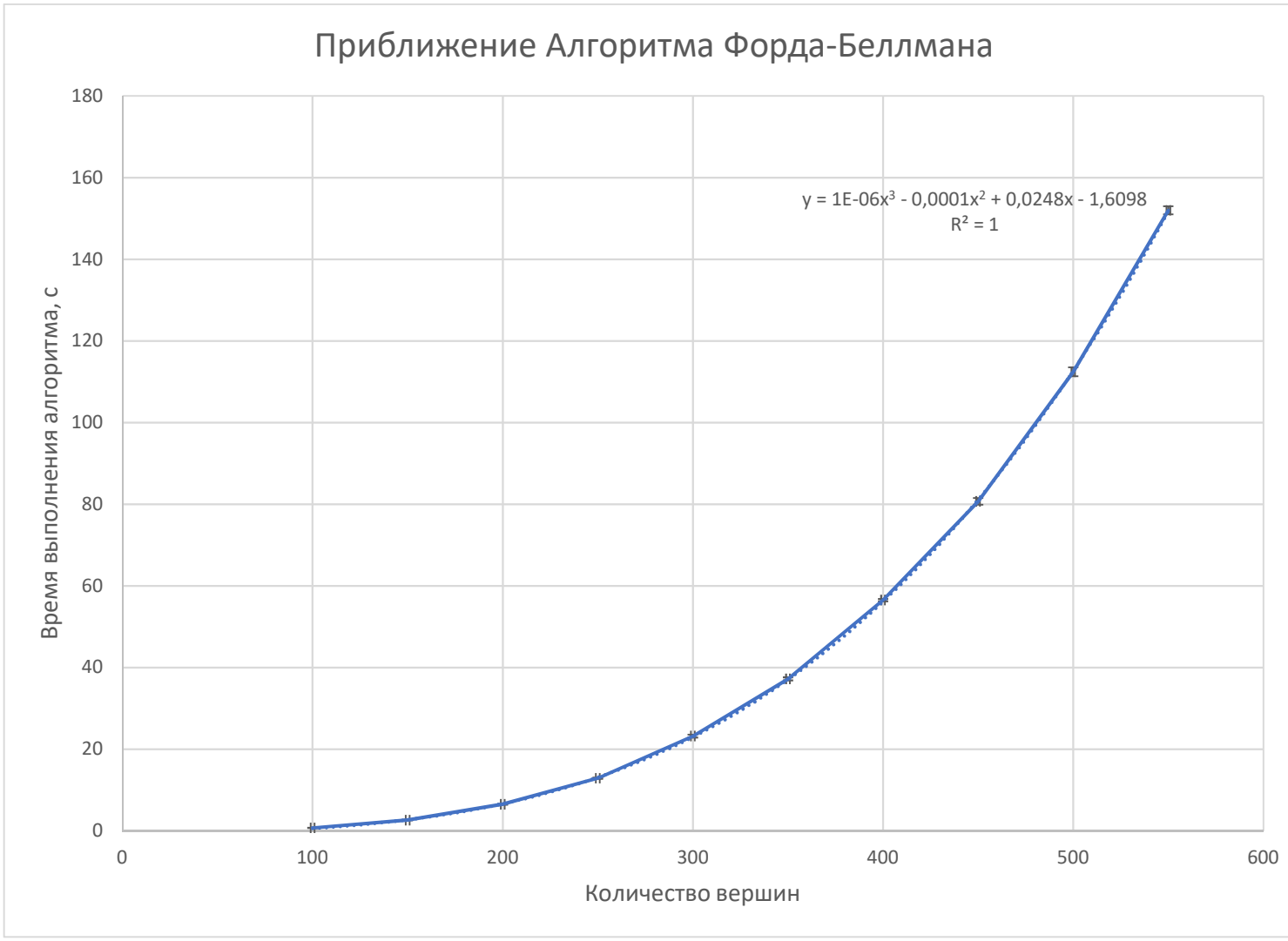

Рис. 3 График зависимости времени выполнения алгоритма Форда-Беллмана от количества вершин для полных графов с диапазоном вершин 100-550

Рис. 4 График зависимости времени выполнения алгоритма Форда-Беллмана от количества вершин для полных графов с диапазоном вершин 100-190

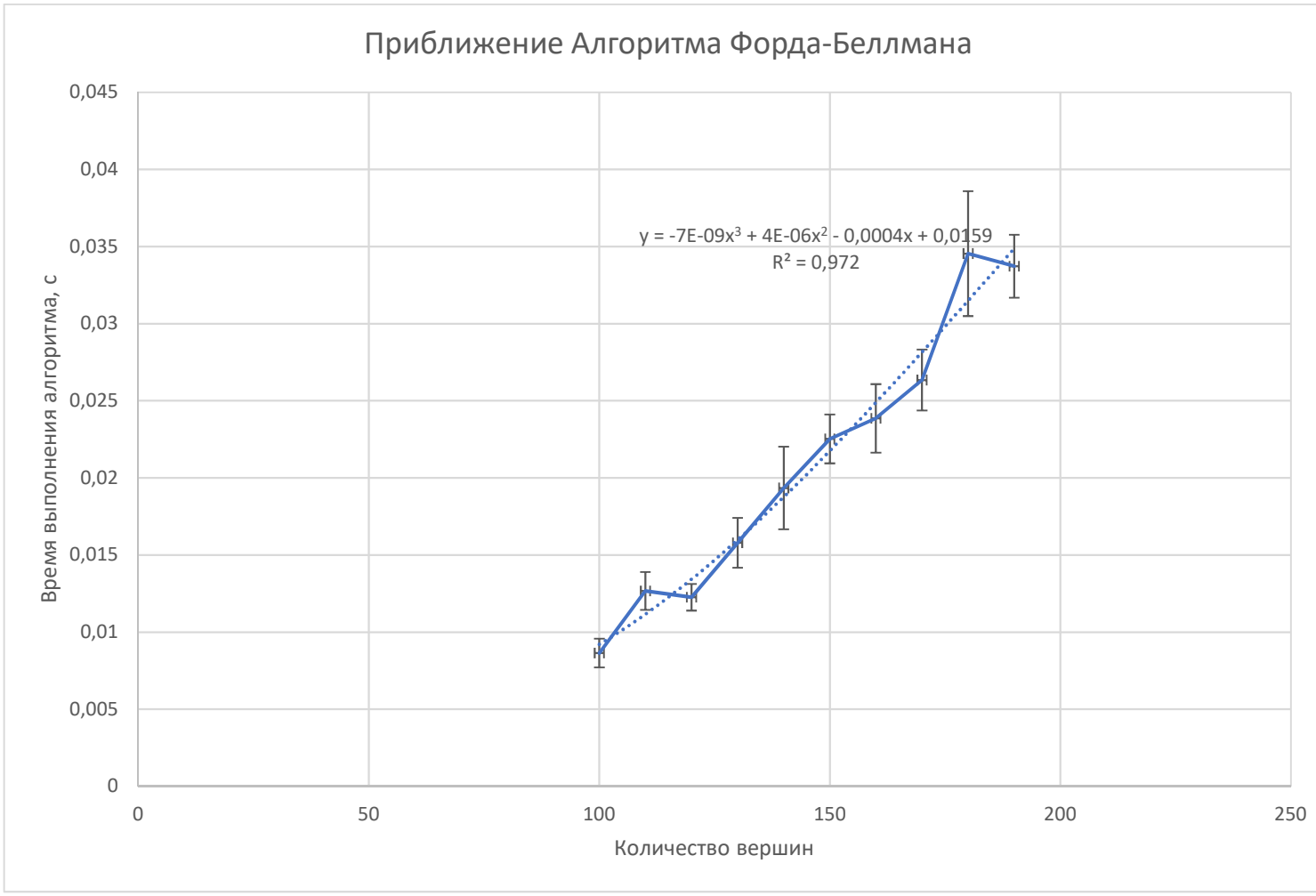

Рис. 5 График зависимости времени выполнения алгоритма Форда-Беллмана от количества вершин для графов-цепочек с диапазоном вершин 100-190

Рис. 6 График зависимости времени выполнения алгоритма Форда-Беллмана от количества вершин для графов-цепочек с диапазоном вершин 100-550

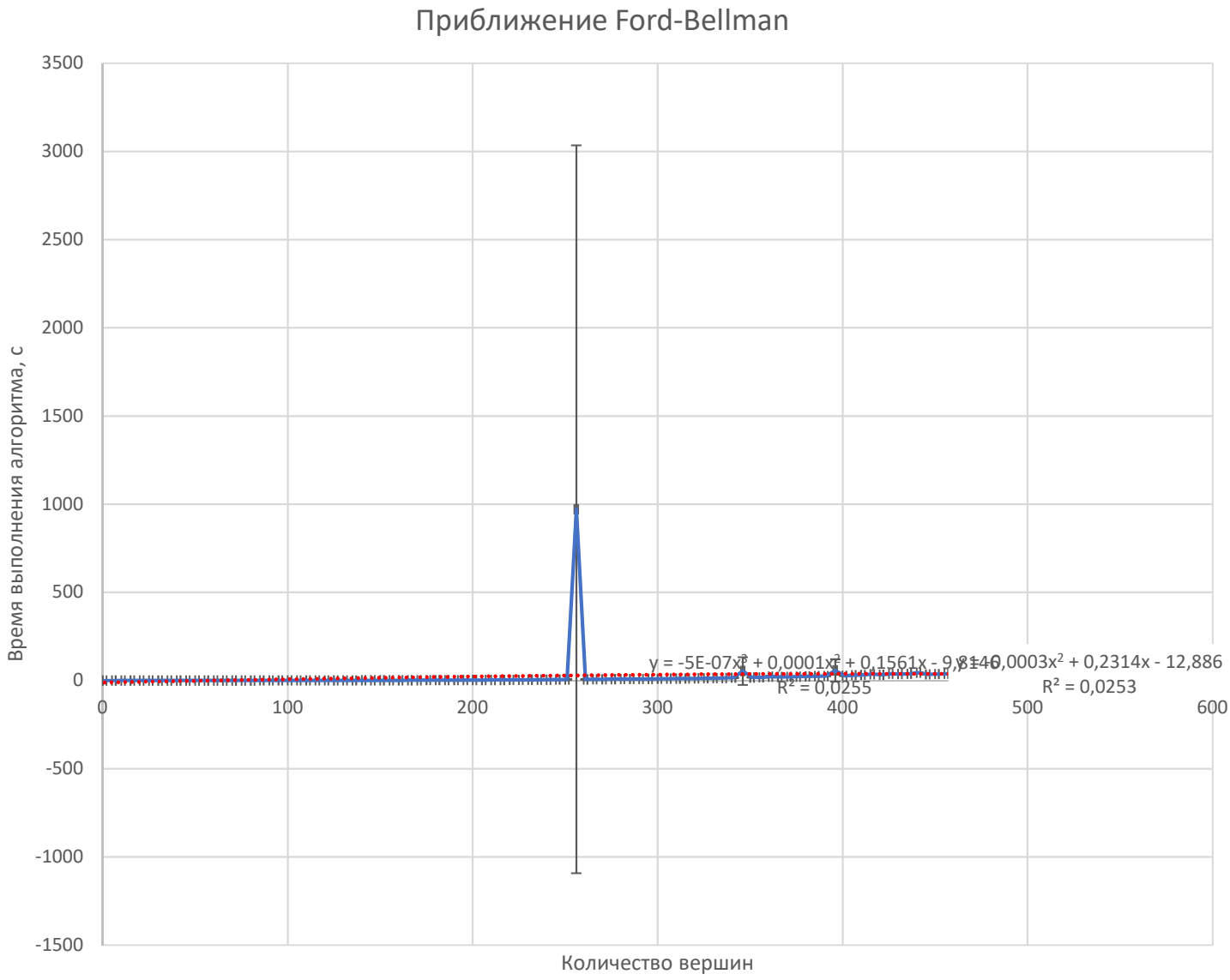

Рис. 7 График зависимости времени выполнения алгоритма Форда-Беллмана от количества вершин для случайных графов с диапазоном вершин 1-496

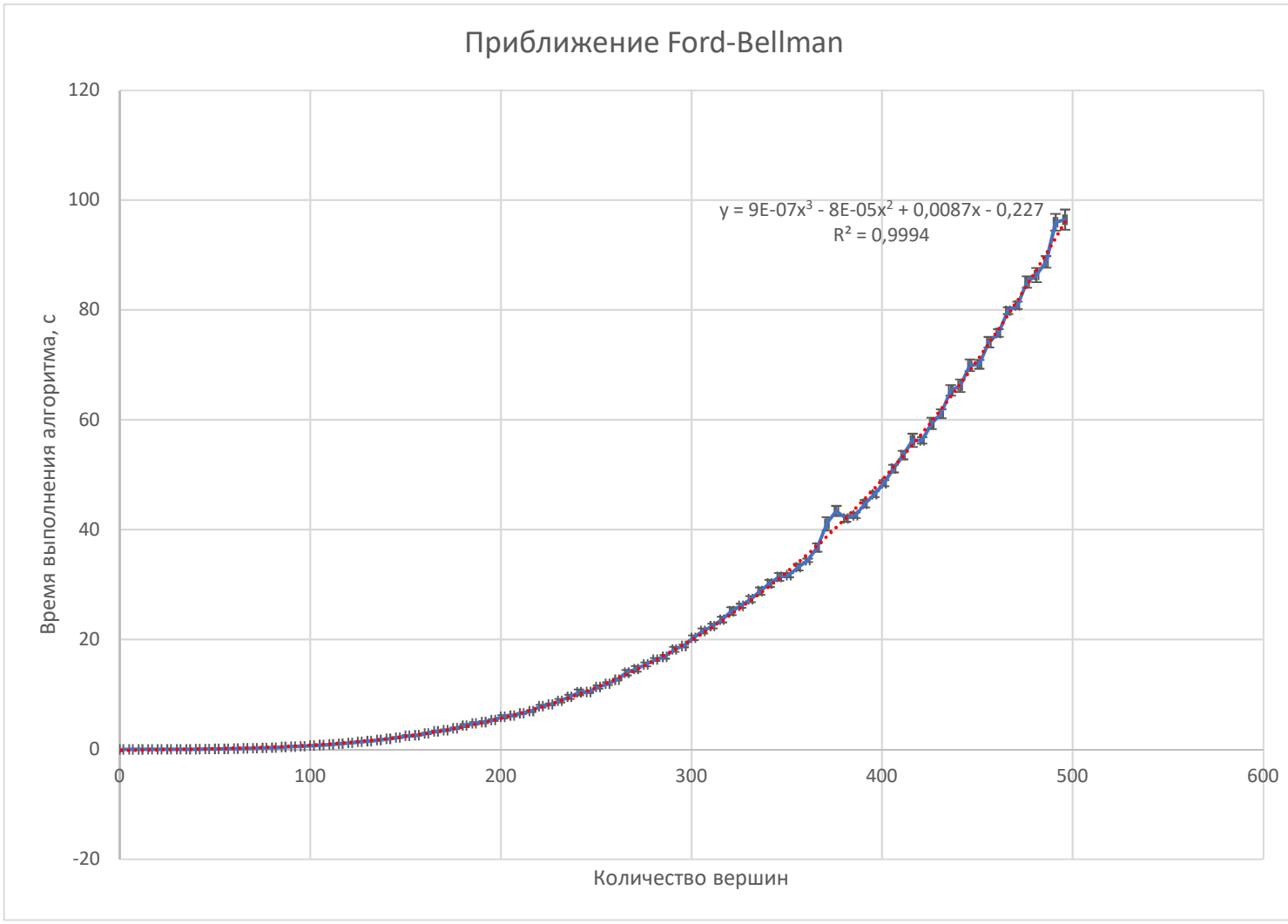

Рис. 8 График зависимости времени выполнения алгоритма Форда-Беллмана от количества вершин для полных графов с диапазоном вершин 1-496

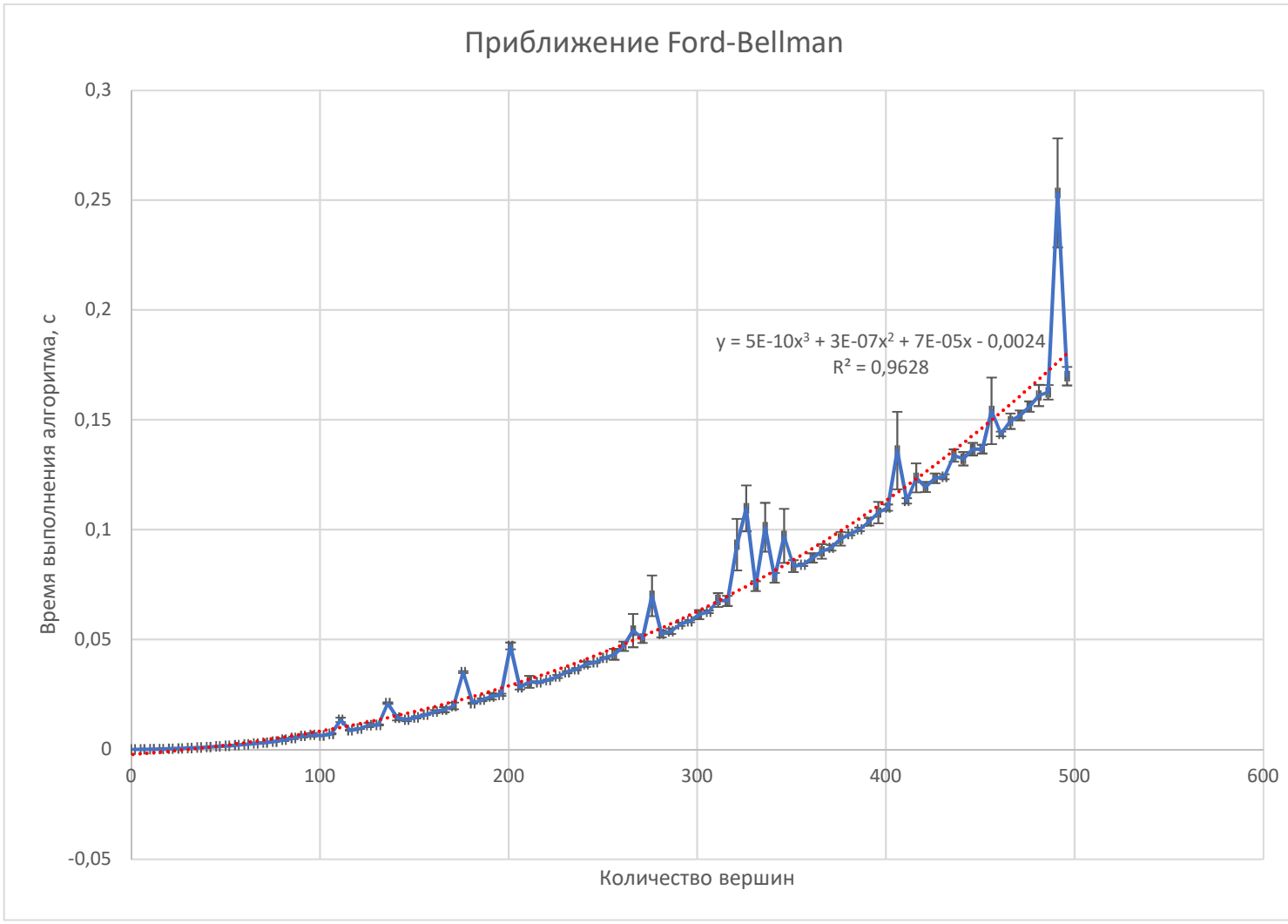

Рис. 9 График зависимости времени выполнения алгоритма Форда-Беллмана от количества вершин для графов-цепочек с диапазоном вершин 1-496

Вывод

Алгоритм Форда-Беллмана ищет кратчайший путь во взвешенных графах, вес ребер которых может быть отрицательным от стартовой до всех вершин в графе. Таким образом этот алгоритм никак не зависит от случайного выбора вершин для тестирования. Данный алгоритм имеет теоретическую асимптотику $\boldsymbol{O(nm)}$, где $\boldsymbol{n}$ – количество вершин в графе, $\boldsymbol{m}$ – количество рёбер. В графах количество рёбер не превосходит $\frac{\boldsymbol{n\cdot(n-1)}}{\boldsymbol{2}}$ (это достигается в полных графах), таким образом время выполнения этого алгоритма имеет кубическую зависимость от числа, что прекрасно видно на графиках. На некоторых из них приближение графика кубической функцией равно 1, то есть график совпадает с некоторой кубической функцией. На остальных этот коэффициент более, чем 0,9. Отсюда можно сделать вывод, что результаты, полученные в ходе эксперимента, отлично иллюстрируют теоретические выкладки.

II. Тестирование по памяти

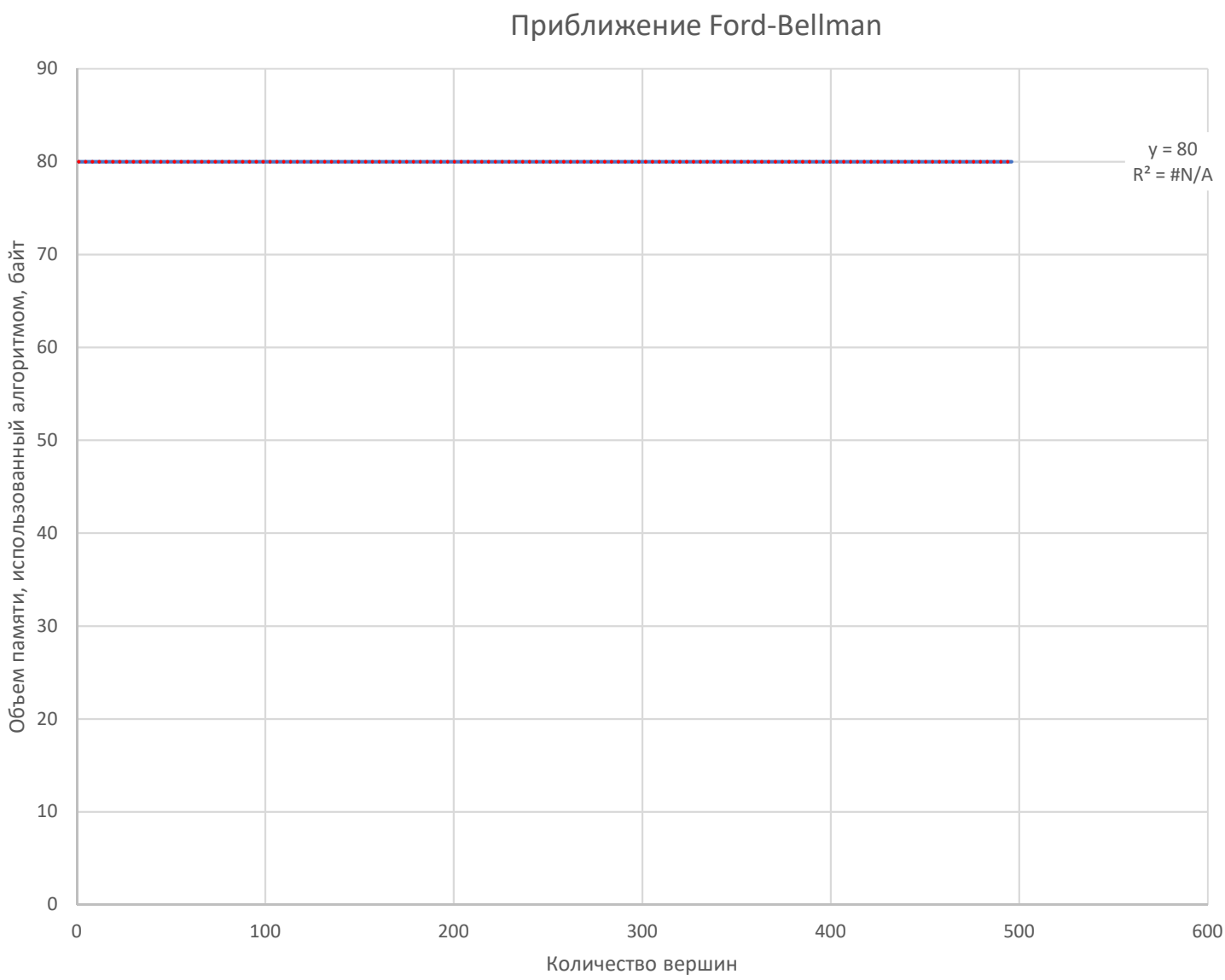

Рис. 1 График зависимости объёма памяти, использованной алгоритмом Форда-Беллмана от количества вершин в случайном графе с диапазоном вершин 1-496

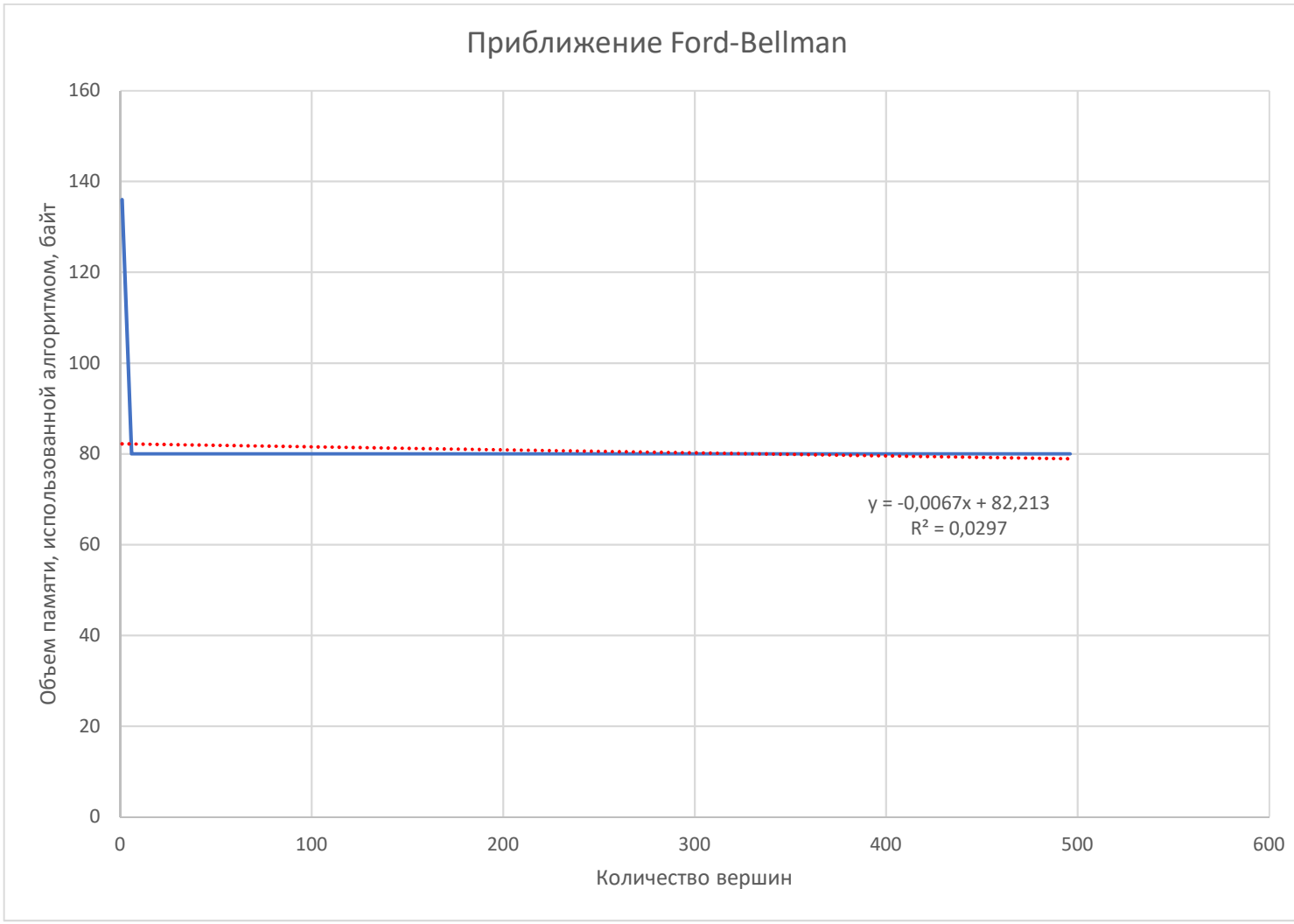

Рис. 2 График зависимости объёма памяти, использованной алгоритмом Форда-Беллмана от количества вершин в полном графе с диапазоном вершин 1-496

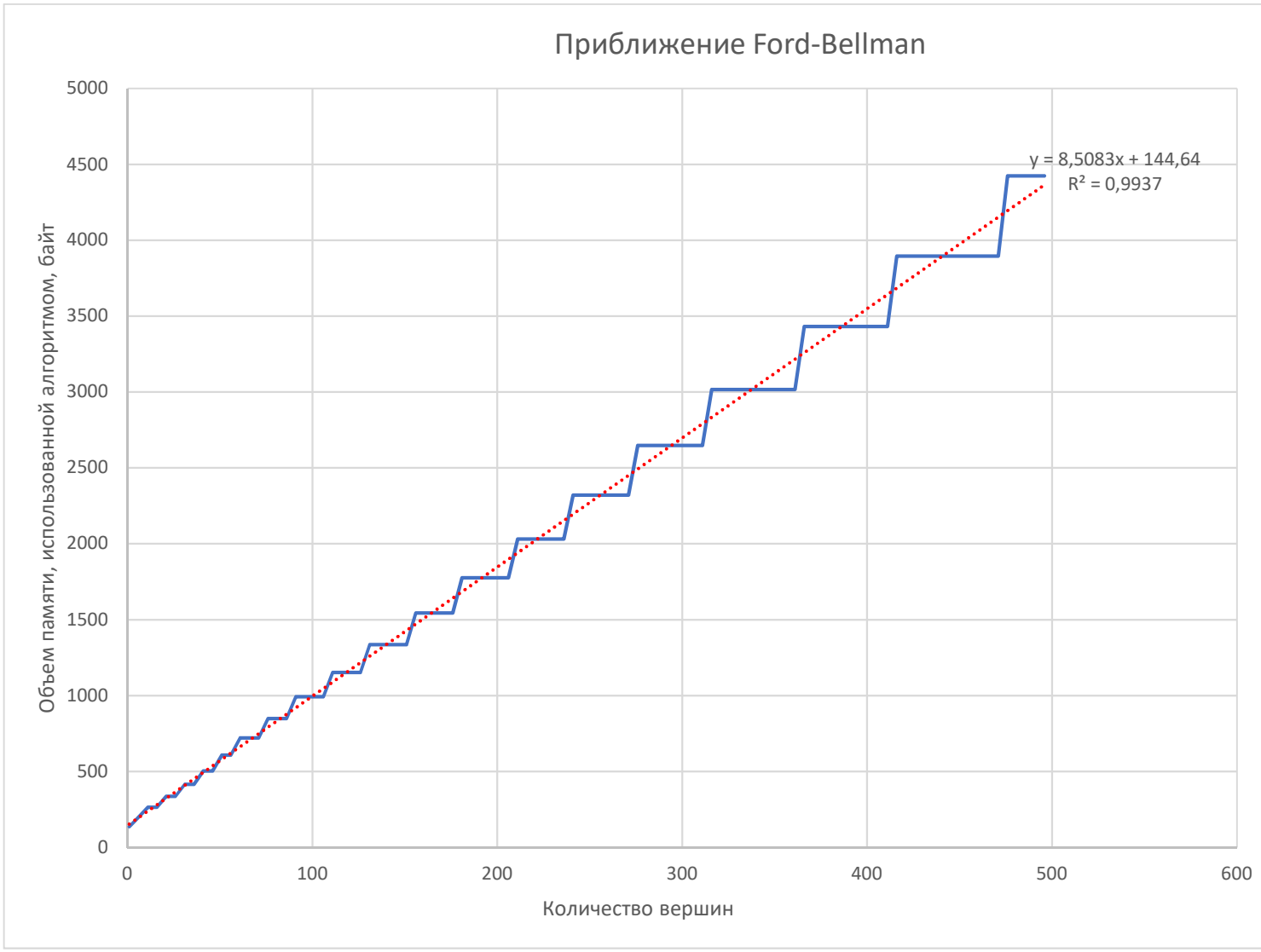

Рис. 3 График зависимости объёма памяти, использованной алгоритмом Форда-Беллмана от количества вершин в графе-цепочке с диапазоном вершин 1-496

Вывод

Из полученных результатов очевидно, что при работе алгоритма Форда-Беллмана количество используемой им памяти не зависит от числа вершин напрямую. Однако, количество используемой алгоритмом памяти зависит от размерности пространства решений (поскольку при тестировании данного алгоритма на графах-цепочках график имеет лестничнообразный вид, а в качестве точек, между которыми мы ищем путь берутся крайние вершины цепочки, а значит при увеличении размера графа увеличивается пространство решений). Так как график на рисунке 3 имеет лестничнообразный вид, то можно сделать вывод о том, что его лучше всего приближать линейной функцией. Таким образом мы получаем результат, что объем памяти, используемый алгоритмом, линейно зависит от размера пространства решений.

5. **Алгоритм A*(A стар)**

I. **Тестирование по времени**

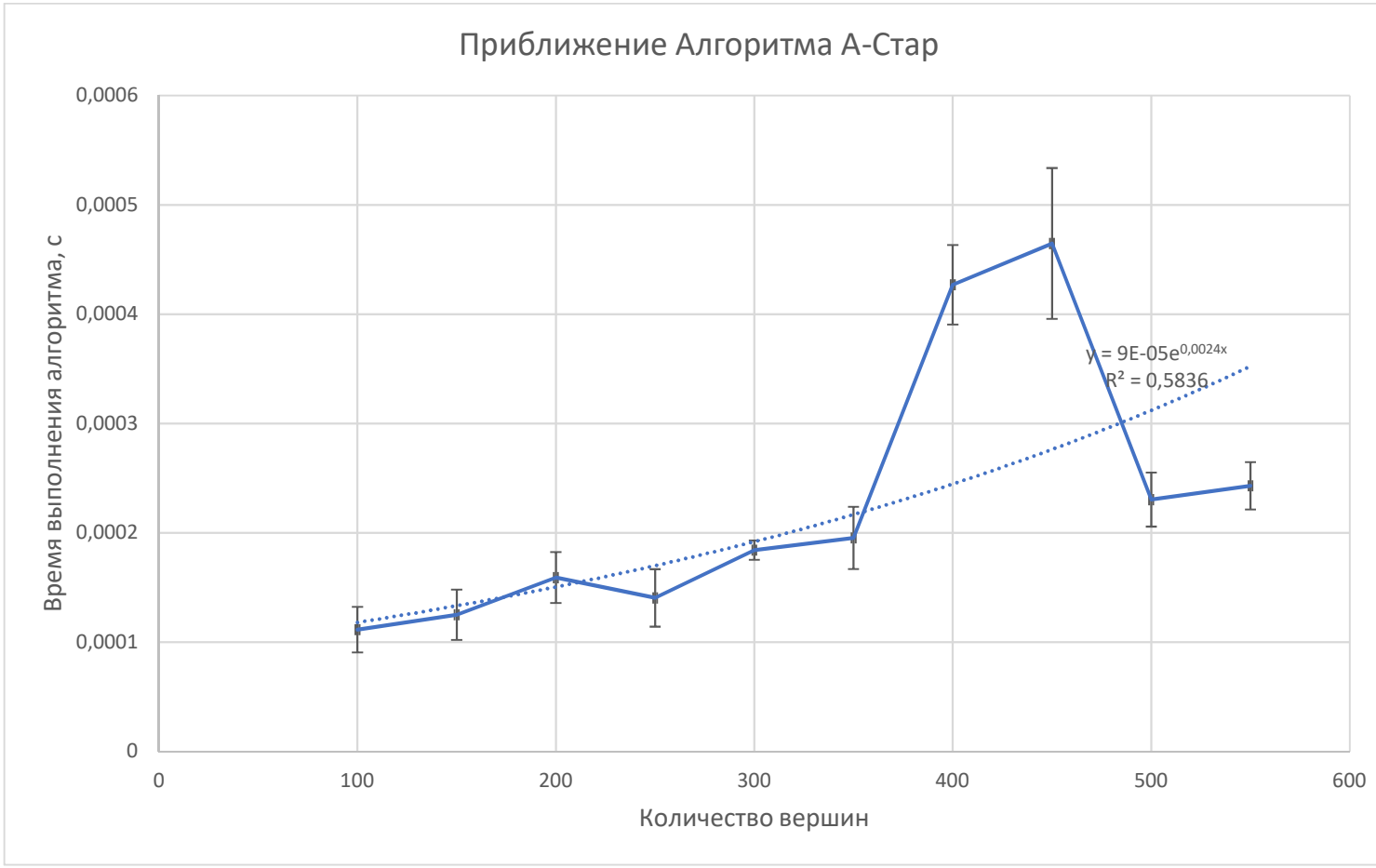

Рис. 1 График зависимости времени выполнения алгоритма A-Стар от количества вершин для деревьев с диапазоном вершин 100-550

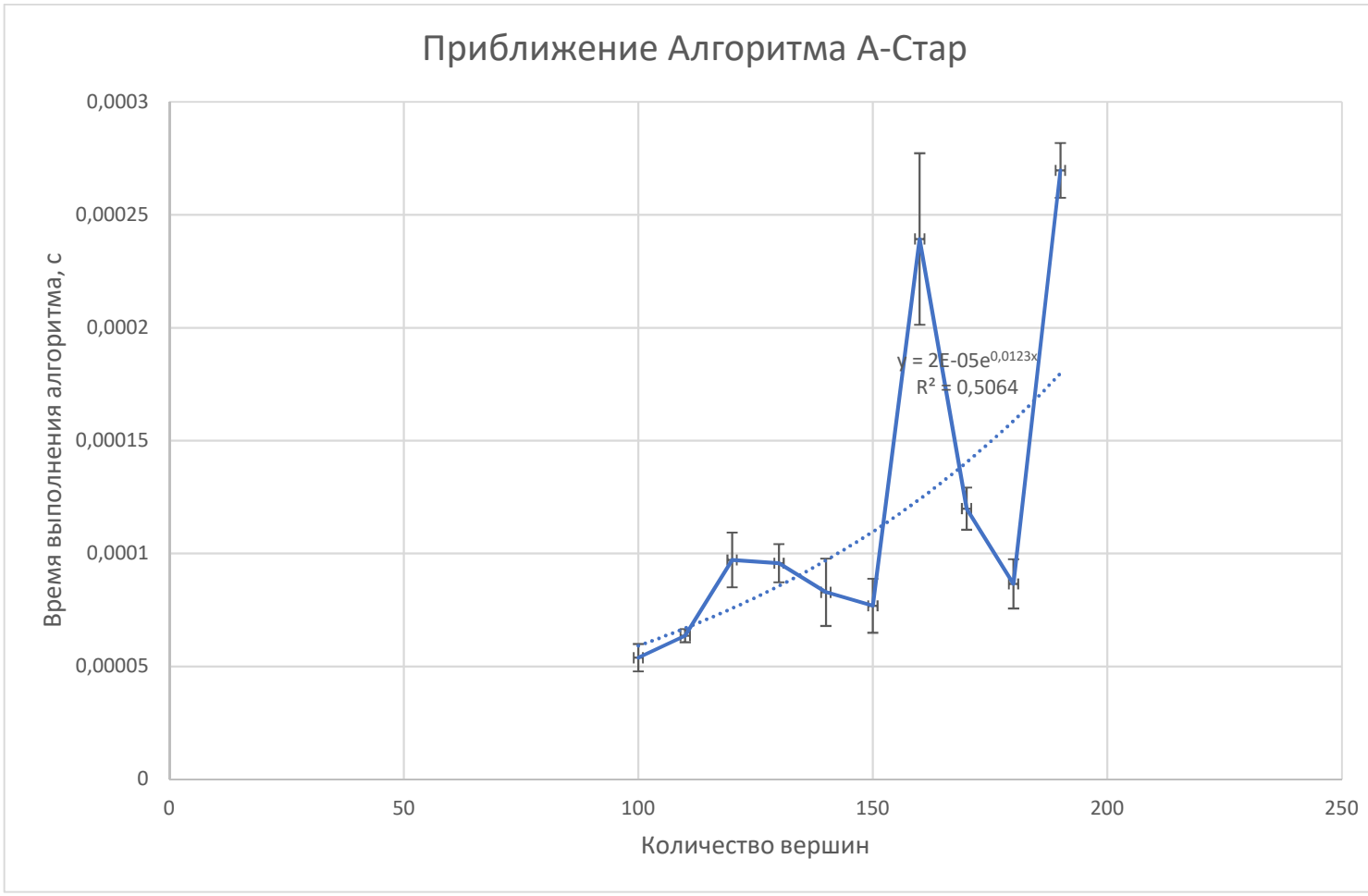

Рис. 2 График зависимости времени выполнения алгоритма A-Стар от количества вершин для деревьев с диапазоном вершин 100-190

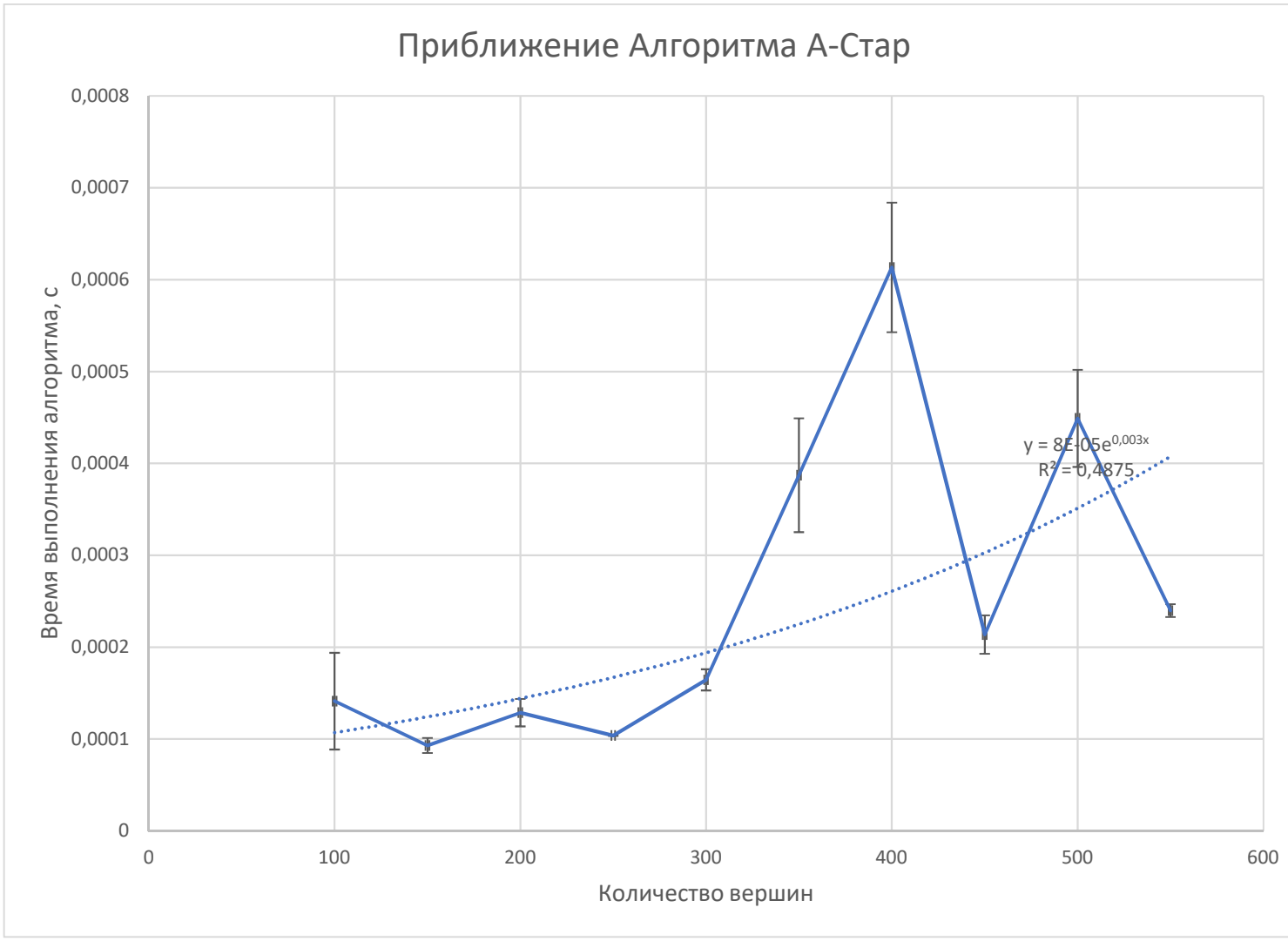

Рис. 3 График зависимости времени выполнения алгоритма A-Стар от количества вершин для деревьев с диапазоном вершин 100-550

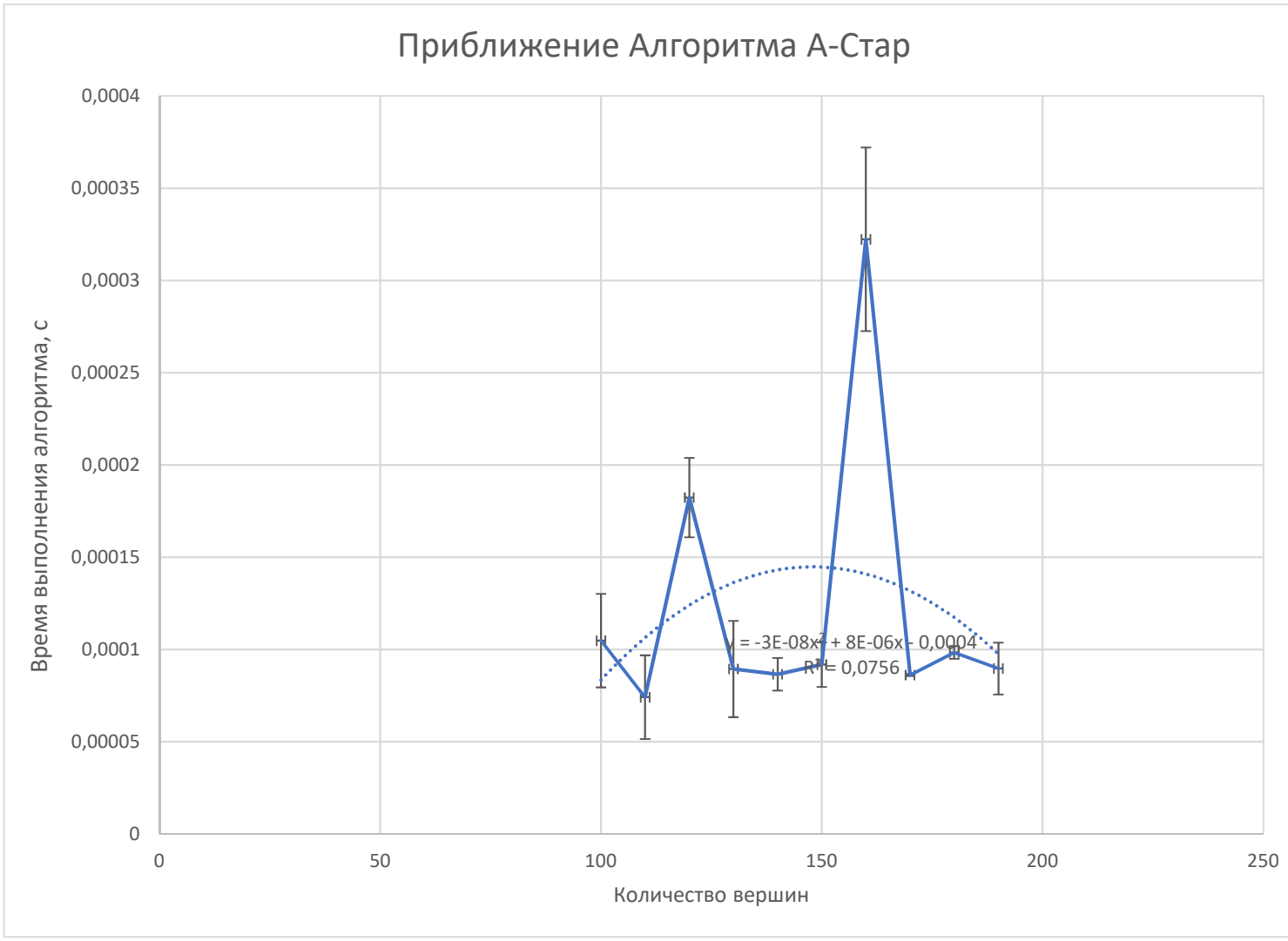

Рис. 4 График зависимости времени выполнения алгоритма А-Стар от количества вершин для деревьев с диапазоном вершин 100-190

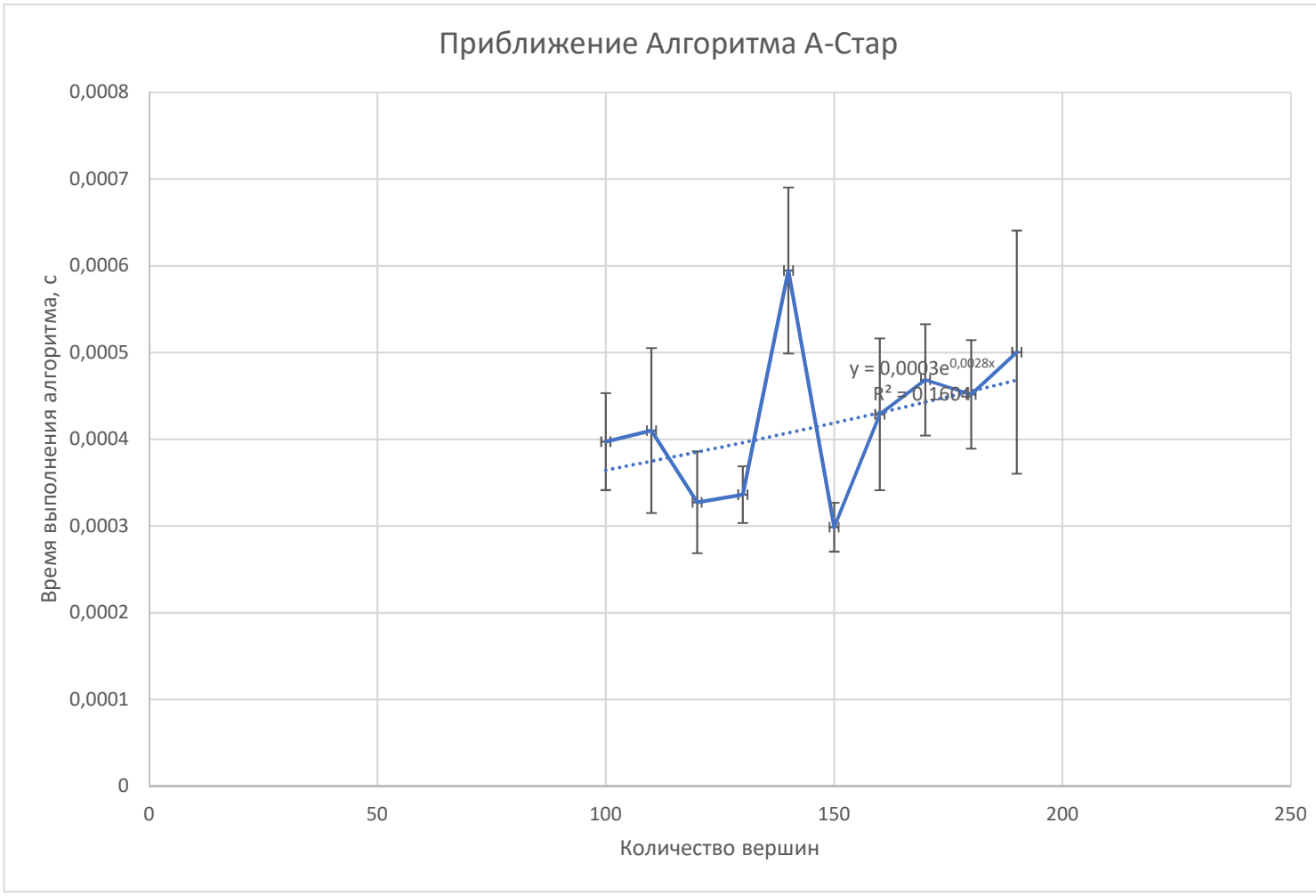

Рис. 5 График зависимости времени выполнения алгоритма А-Стар от количества вершин для графов-цепочек с диапазоном вершин 100-190

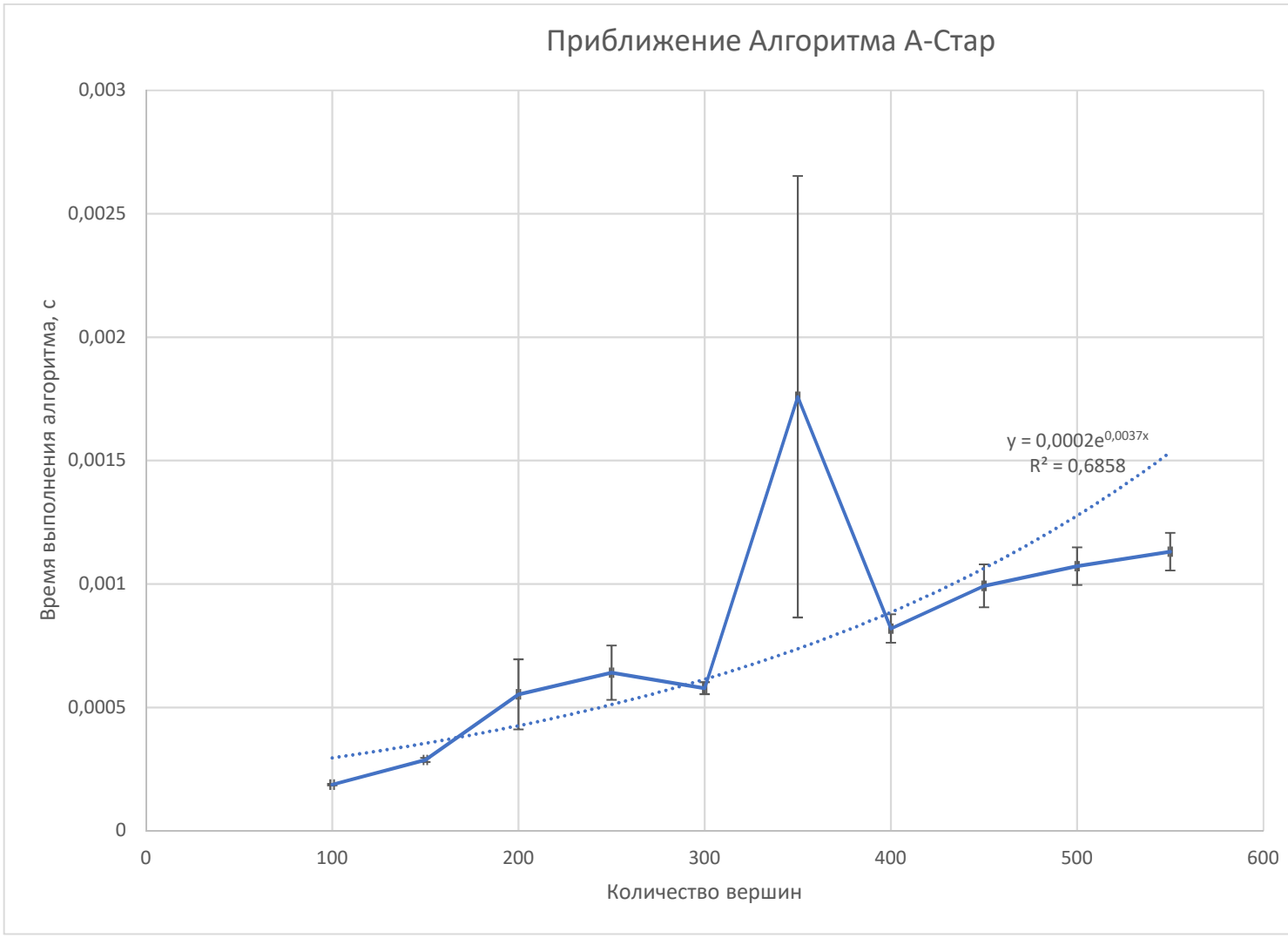

Рис. 6 График зависимости времени выполнения алгоритма A-Стар от количества вершин для графов-цепочек с диапазоном вершин 100-550

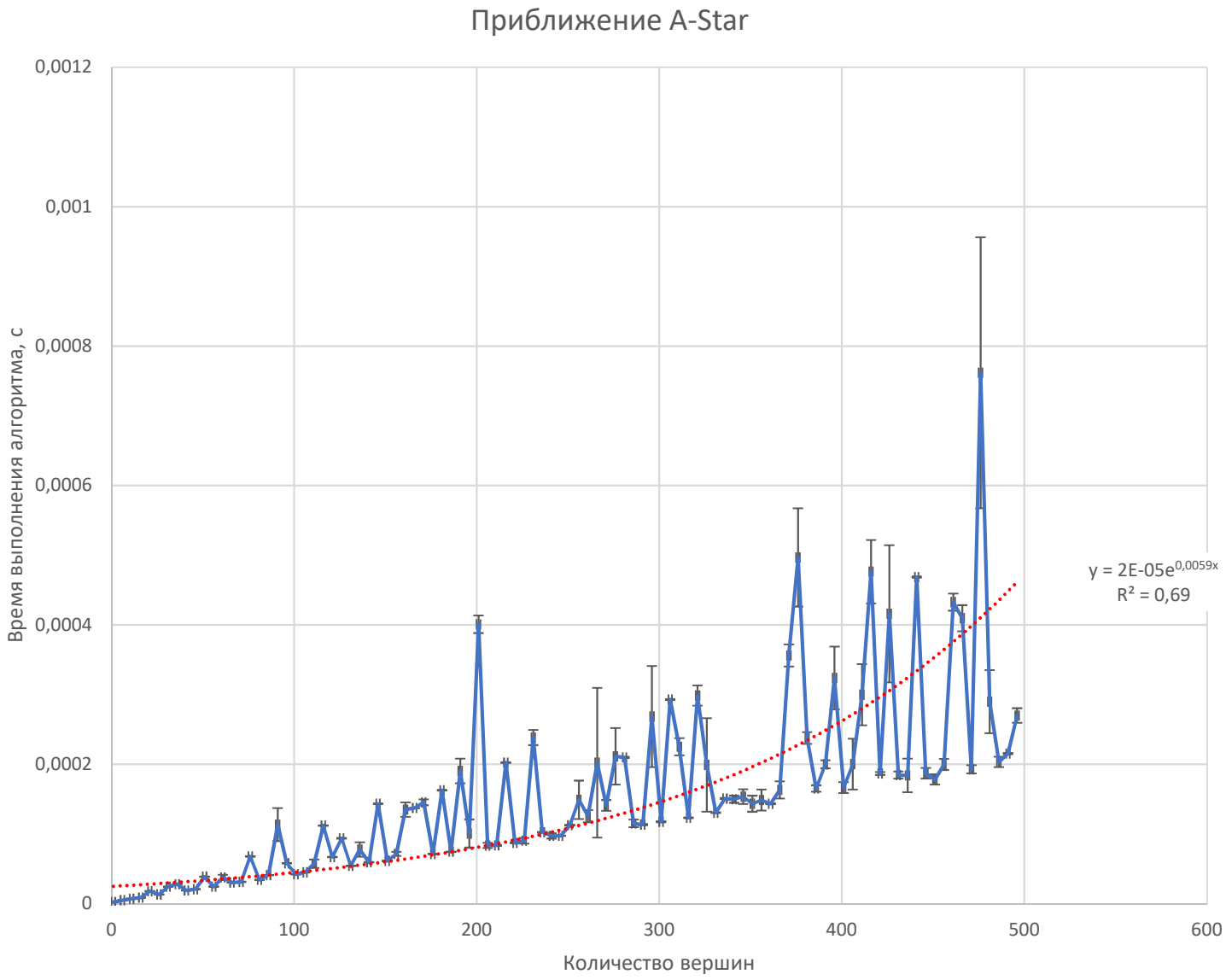

Рис. 7 График зависимости времени выполнения алгоритма А-Стар от количества вершин для деревьев с диапазоном вершин 1-496

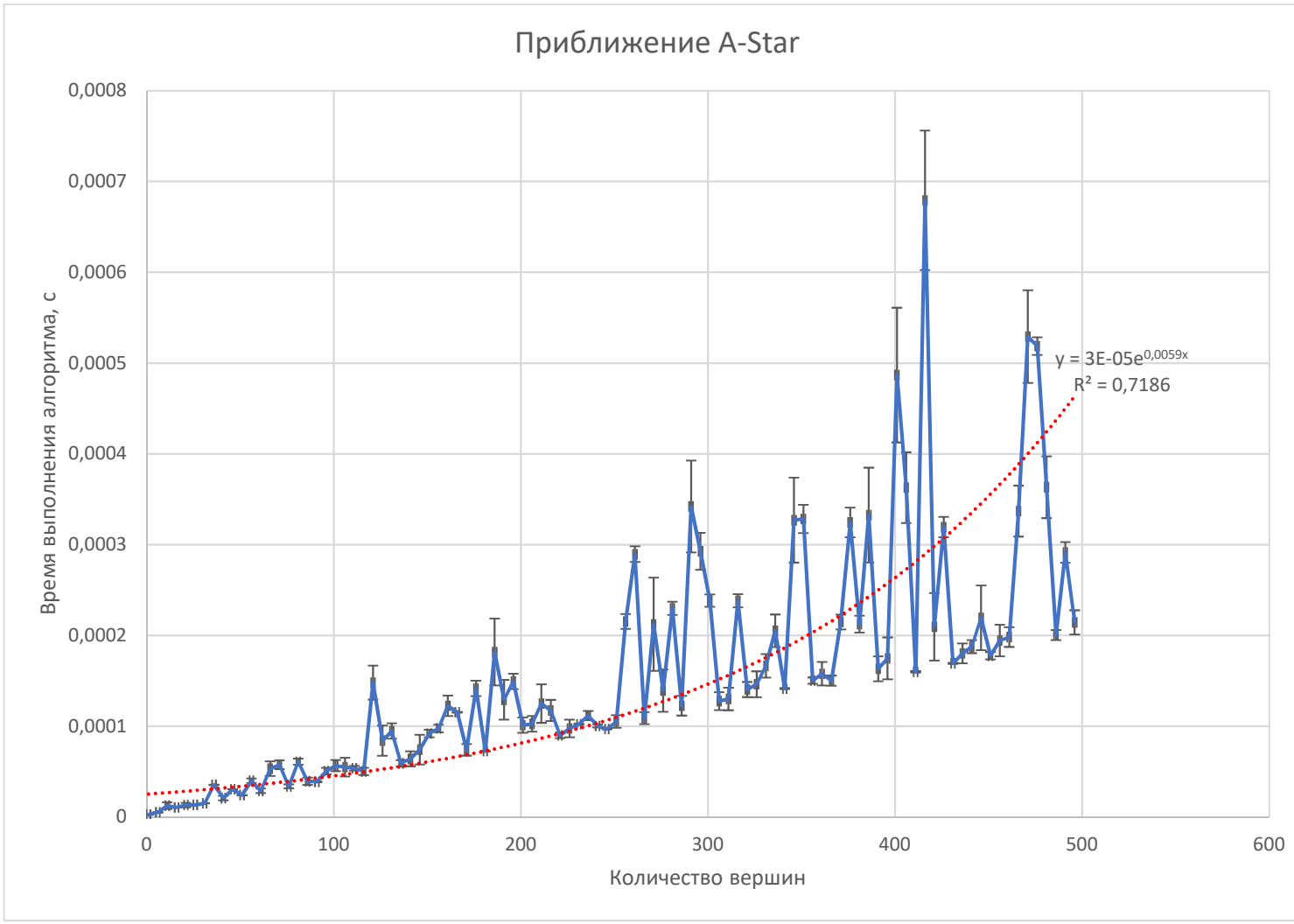

Рис. 8 График зависимости времени выполнения алгоритма A-Star от количества вершин для деревьев с диапазоном вершин 1-496

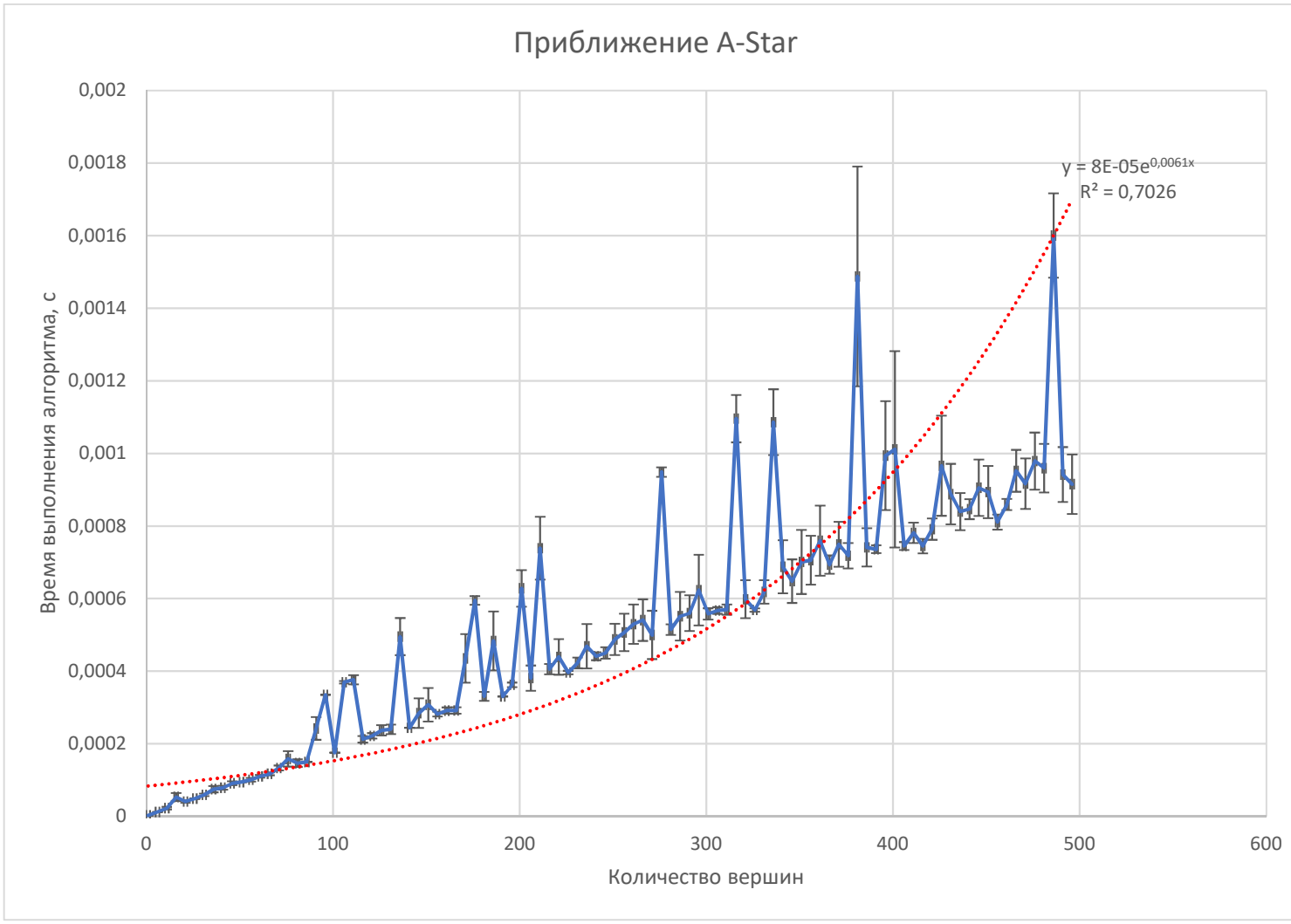

Рис. 9 График зависимости времени выполнения алгоритма A-Стар от количества вершин для графов-цепочек с диапазоном вершин 1-496

Вывод

Теоретическая сложность алгоритма A-Стар имеет экспоненциальную сложность. В зависимости от выбранной эвристики она может значительно улучшиться и даже стать линейной от количества вершин в пространстве решений. Мною была выбрана следующая эвристика: все вершины были занумерованы при обходе в глубину, сумма значения этой функции в текущей вершине и в конечной вершине и есть эвристика. Не было найдено теоретических данных, которые бы просчитывали асимптотику этого алгоритма для данной эвристики, поэтому было принято решение приближать полученные графики экспоненциальной функцией. Для некоторых типов графов это приближение было весьма успешно, что позволяет сделать вывод о том, что асимптотика этого алгоритма не превосходит экспоненциальную, а возможно и вовсе является полиномиальной, что соответствует теоретическим знаниям об этом алгоритме.

II. Тестирование по памяти

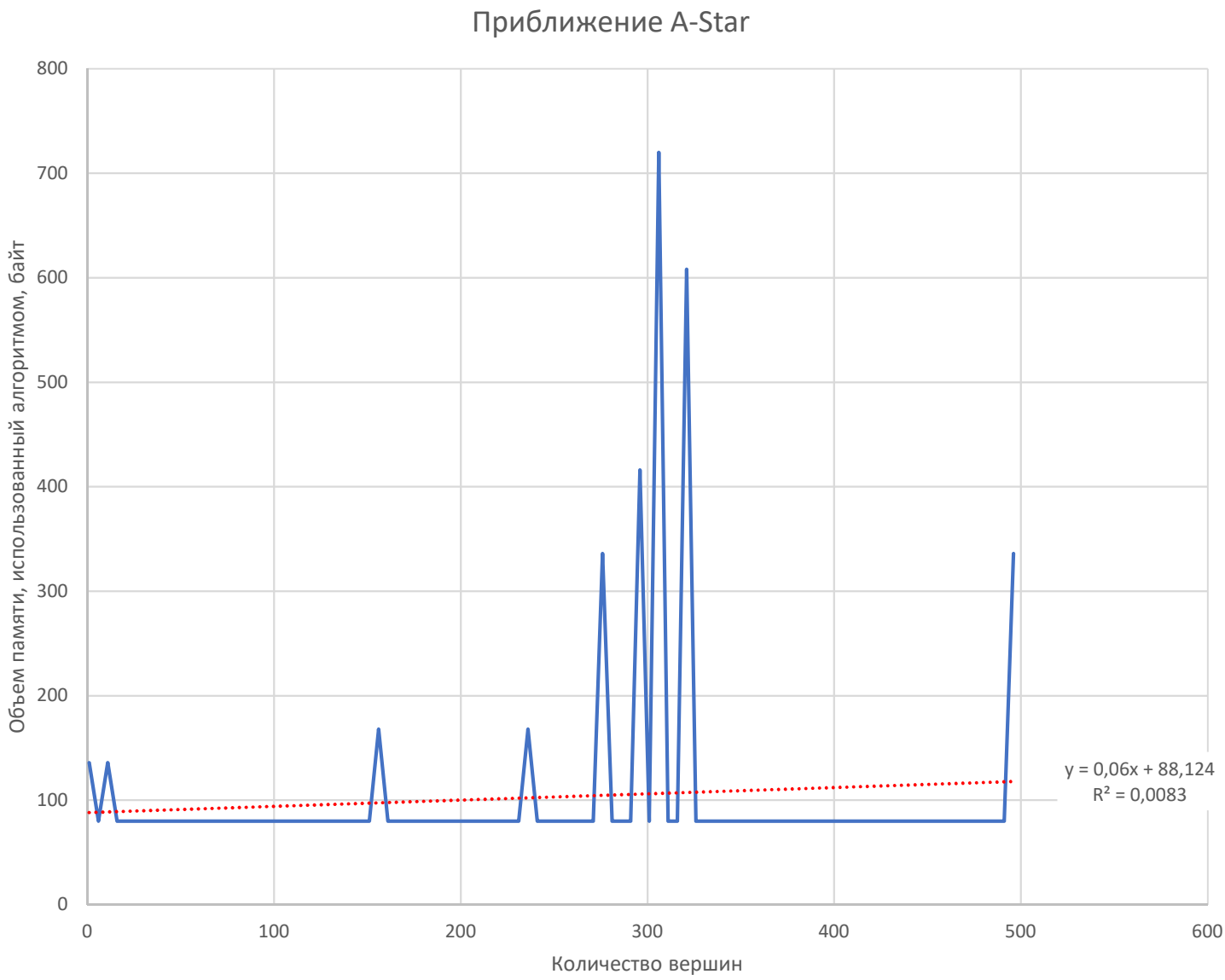

Рис. 3 График зависимости объёма памяти, использованной алгоритмом A-Стар от количества вершин в дереве с диапазоном вершин 1-496

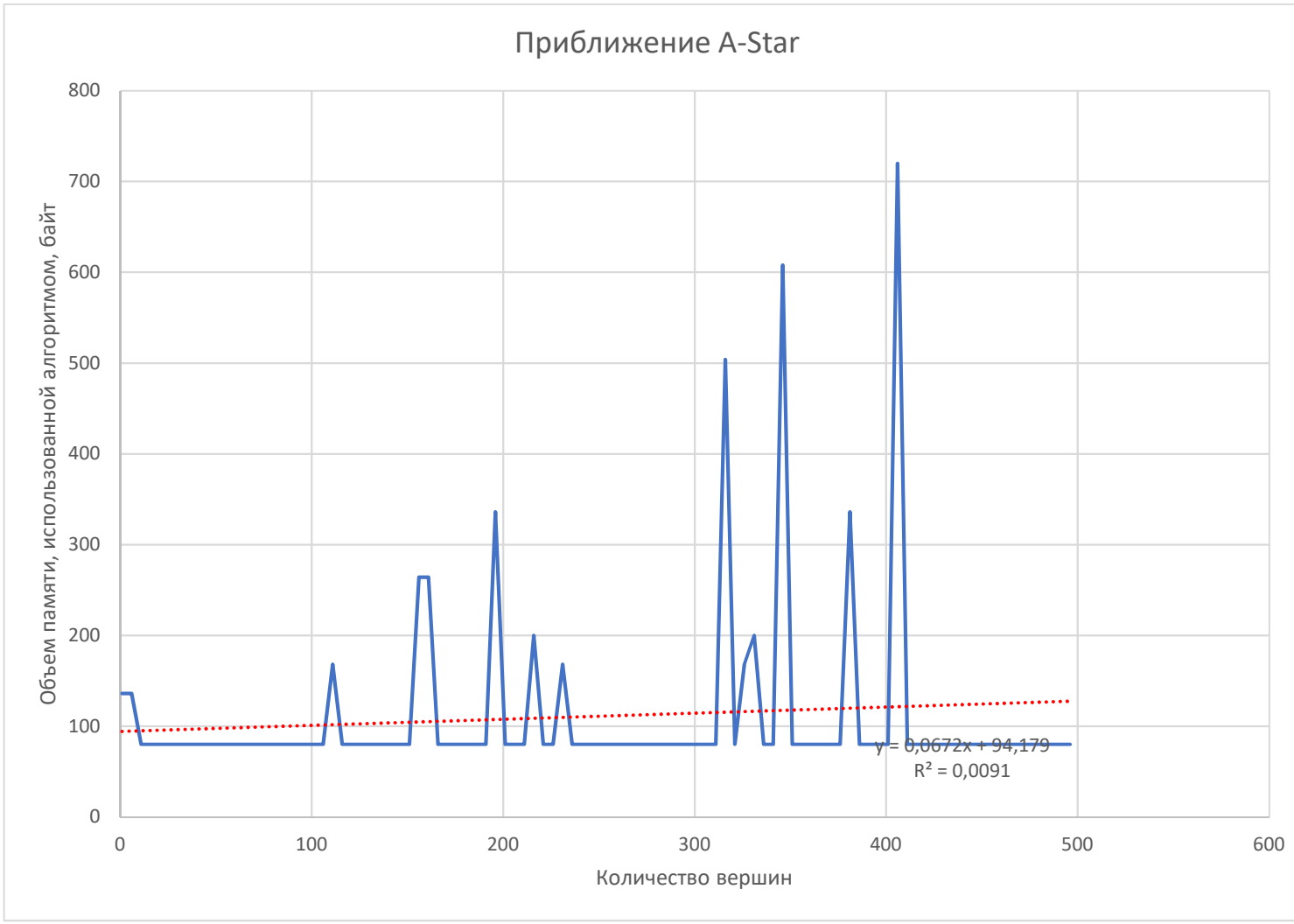

Рис. 2 График зависимости объёма памяти, использованной алгоритмом Дейкстры от количества вершин в дереве с диапазоном вершин 1-496

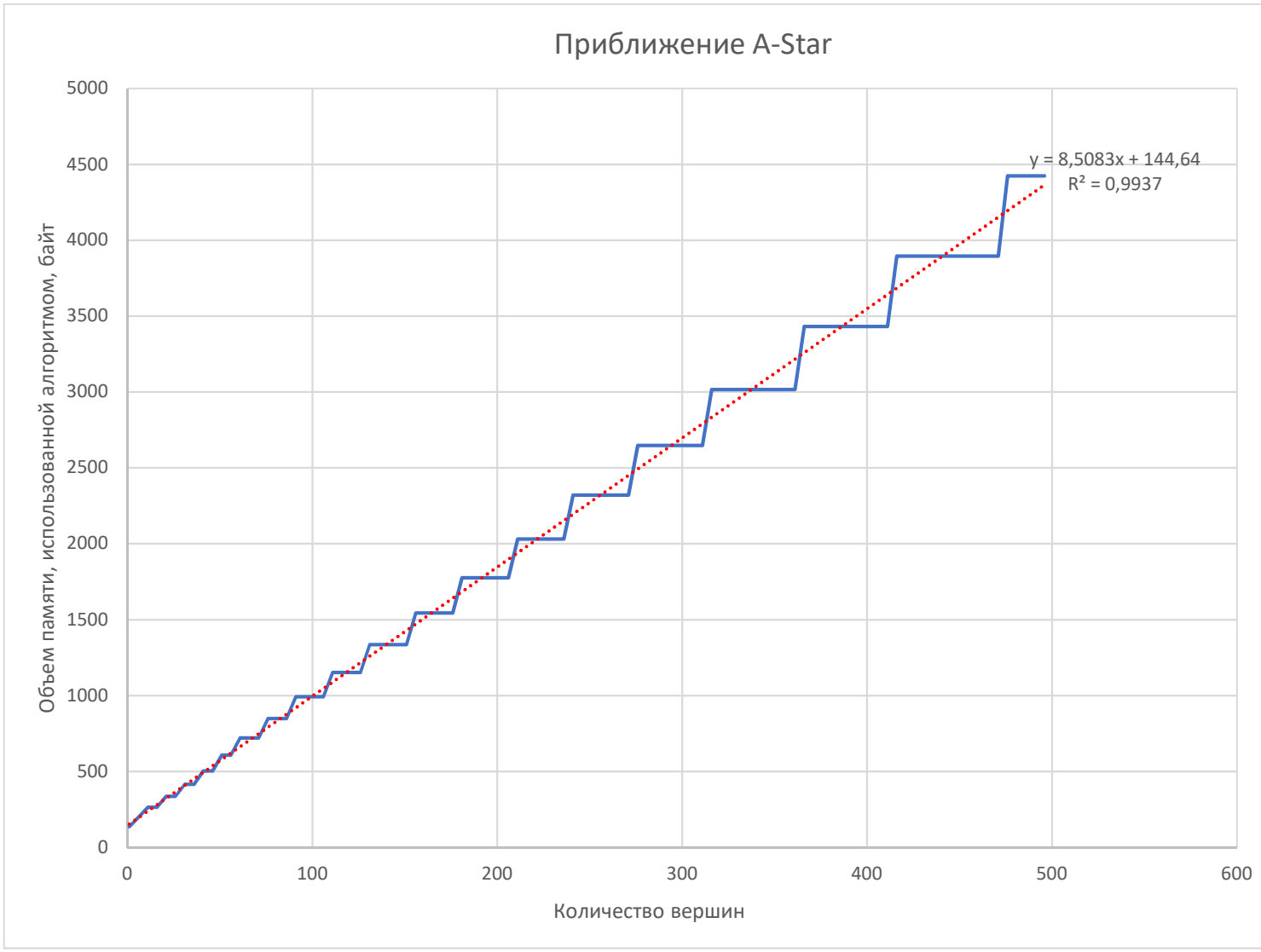

Рис. 3 График зависимости объёма памяти, использованной алгоритмом А-Стар от количества вершин в графе-цепочке с диапазоном вершин 1-496

Вывод

Из полученных результатов очевидно, что при работе алгоритма A-Стар количество используемой им памяти не зависит от числа вершин напрямую. Однако, количество используемой алгоритмом памяти зависит от размерности пространства решений (поскольку при тестировании данного алгоритма на графах-цепочках график имеет лестничнообразный вид, а в качестве точек, между которыми мы ищем путь берутся крайние вершины цепочки, а значит при увеличении размера графа увеличивается пространство решений). Так как график на рисунке 3 имеет лестничнообразный вид, то можно сделать вывод о том, что его лучше всего приближать линейной функцией. Таким образом мы получаем результат, что объем памяти, используемый алгоритмом, линейно зависит от размера пространства решений.

Заключение

Для выполнения экспериментов была написана библиотека на языке Python 3.6, которая реализует пять алгоритмов поиска пути на графе. После проведения большого количества тестов для разных типов графов и диапазонов вершин в них были получены разнообразные графики, которые позволили сравнить результаты, полученные в ходе данного эксперимента с теоретическими результатами. Оказалось, что все результаты тестирования по времени не противоречат теоретическим данным. Некоторые из них более, некоторые менее успешно им соответствуют.

Результаты тестирования по памяти оказались более единообразными. Память используемая алгоритмом Дейкстры не зависит от числа вершин в графе, в котором осуществляется поиск. Во всех остальных случаях память не зависит от количества вершин напрямую, однако имеет линейную зависимость от размера пространства решений, что не противоречит теоретическим данным