# 1 Abstract

В нашей работе представлен генетический алгоритм для поиска центральных вершин в графе, который использует иной подход к способу представления решения и новый взгляд на процесс кроссинговера. Полученный алгоритм сравнивался с уже существующими точными и другими генетическими алгоритмами на различных моделях случайных графов. Из полученных результатов можно сделать вывод о том, что рассматриваемый подход может использоваться в прикладных задачах и конкурировать с существующими алгоритмами.

# 2 Introduction

В теории графов нахождение радиуса графа и центральных вершин одни из важных проблем, решаемых в теоретических и практических задачах. Рассмотрим невзвешенный нериентораванный граф $G = (V, E)$, где $|V| = n$ — количество вершин, $|E| = m$ — количество ребер. Так как граф невзвешенный, то длина пути между вершинами $u$ и $v$ равняется числу ребер. Для поиска центра графа используется понятие эксцентриситета вершины — расстояние от вершины до самой удаленной. На основе этого можно описать радиус графа, как минимальный из эксцентриситетов всех вершин. При этом вершины, на которых достигается этот минимум принято называть центральными. Данный вопрос наиболее часто возникает при решении оптимизационных задач в компьютерных сетях и задач транспортной маршрутизации.

Эта задача хорошо изучена с точки зрения точных алгоритмов. Зачастую она рассматривается вместе с проблемой вычисления all-pairs shortes path in graph. Для невзвешенных графов тривиальным алгоритмом может служить запуск обхода в ширину из каждой вершины, при этом алгоритм имеет временную оценку $O(nm)$, что при больших $m$ равно $O(n^3)$. Все существующие алгоритмы, дающие точное решение, созданы в попытке улучшить данную оценку. В целом можно выделить два подхода к решению данной задачи. Первый из них предложен в работе [1] и основывается на матричном умножении. На сегодняшний день существуют алгоритмы быстрого матричного умножения, которые дают теоретическую оценку $O(n^{2.376})$ или более применяемые на практике $O(n^{2.81})$.

Другой подход был разработан в работе [2]. В этом случае используется идея разделения вершин на множество вершин с высокой степенью и множество вершин с низкой степенью. С использованием такого подхода временную оценку удается улучшить и получить $O(m\sqrt{n})$. Данная работа дала толчок дальнейшим исследованиям с использованием аналогичного метода и существует ряд алгоритмов [3], [4], [5], улучшающих данную оценку, в которых используются особые структуры данных или же предполагается низкая плотность исследуемого графа.

Данные методы могут быть широко применены на графах небольших

размерностей, однако наибольший интерес для исследования представляют реально существующие графы, которые содержат в себе сотни тысяч вершин. Вследствие этого точные алгоритмы не могут применены для решения подобных задач из-за неприемлемых временных затрат. Наиболее подходящими в данной ситуации могут быть эвристические алгоритмы, которые дают лучшие временные результаты, но допускают существование некого процента ошибки.

В нашей работе мы представляем генетический алгоритм, позволяющий решать задачу поиска центральных вершин и радиуса графа. При этом в сравнении с некоторыми алгоритмами наш дает лучшие временные результаты.

# 3 Описание алгоритма

Генетические алгоритмы хорошо известный подход для решения оптимизационных задач. Основная идея генетического алгоритма была представлена Холландом [6]. Алгоритм использует процессы генетики, которые обеспечивают эволюционное развитие живых организмов. Согласно предложенному подходу решение представляется некоторым набором генов и основными этапами алгоритма являются процесс мутации кроссинговера и естественного отбора. В нашем алгоритме все эти процессы реализованы с учетом рассматриваемой задачи.

Основную идею построения алгоритма можно описать следующим образом. Мы можем абстрактно изобразить граф. При этом центральные вершины должны быть размещены в центре изображения а остальные вершины будут образовывать дискретные «сферы», находящиеся на расстоянии 1, 2, и т.д. от центра. Используя такое представление графа, легко заметить, что для нахождения центра графа можно создать набор вершин, которые будут представлять «сферу», внутри которой будет лежать центр. Тогда каждая итерация алгоритма должна уменьшать расстояние между элементами созданной «сфера» и стягивать её к центру. Наш алгоритм реализует это с помощью оператора кроссинговера.

## 3.1 Кодирование решения

Популяция для работы генетического алгоритма описывается одним множеством вершин, где каждая вершина представляет потенциальное решение. На языке ГА это означает, что каждая особь в популяции — вершина графа. В качестве начальной популяции генерируется набор случайных вершин.

## 3.2 Фитнесс-функция

Наиболее естественным способом оценки качества полученного решения в рамках рассматриваемой задачи является значение эксцентриситета вершины, который находится при помощи алгоритма обхода в ширину. При этом

вершины приоритет при естественном отборе отдается вершинам с меньшим эксцентриситетом.

### 3.3 Мутация

В качестве процесса мутации нами был выбран подход при котором для изменения существующей популяции с вероятностью заданной для оператора мутации вершина заменяется на случайную из множества ее соседей.

### 3.4 Кроссинговер

В качестве оператора кроссинговера нами использована следующая эвристика. Рассмотрим текущую популяцию, как уже описывалось ранее она может быть интерпретирована как окружность внутри которой лежит центр. В связи с этим можно рассмотреть пару вершин и найти между ними кротчайший путь с помощью поиска в ширину. После этого в качестве потомка от двух особей выбирается случайная вершина из найденного пути. Такой подход позволяет на каждой итерации приблизиться к некоторой вершине с оптимальным эксцентриситетом.

## 4 Результаты вычислительных экспериментов

Все алгоритмы выполнялись на компьютере AMD A8-7410 2.20 GHz and 6 GB RAM. Все алгоритмы были реализованы на языке прогарммирования C++.

Для определения точности предложенного алгоритма был реализован точный алгоритм [2], который позволяет сделать выводы о корректности найденного решения. Также для сравнения временных затрат был взят другой генетический алгоритм, описанный в работе [7], который имеет схожие подходы к решению рассматриваемой задачи, но отличается способом представления решения, оператором мутации и кроссинговера. Данный алгоритм использует оператор мутации, который был назван авторами N4N, поэтому далее будем так его и обозначать.

Сравнение алгоритмов проводилось на двух моделях случайных графов. Первая из них — это модель Барабаши-Альберта [8], вторая — геометрический случайный граф [9]. Для графа БА параметр $m = 2$, а в геометрическом случайном графе $r = 0.1$. Для обоих алгоритмов производились измерения времени и точность найденных параметров. Для более четкой картины оба алгоритма запускались 100 раз на каждом тесте, что позволило получить среднее время работы и процент ошибок. Так как наш алгоритм использует идею абстрактного представления графа в виде сферы будем называть его «сферическим». Результаты экспериментов приведены в таблице ?? и 1.

Из полученных результатов видно, что предложенный алгоритм работает в разы быстрее. При этом алгоритм дает значимый процент ошибки

| № | Размеры графа | | Время работы, сек. | Процент ошибки, % |
|---|---|---|---|---|
| | N | M | | |
| 1 | 500 | 996 | 0.07 | 16.0 |
| 2 | 1000 | 1996 | 0.18 | 12.0 |
| 3 | 1500 | 2996 | 0.37 | 4.0 |
| 4 | 2000 | 3996 | 0.55 | 1.0 |
| 5 | 2500 | 4996 | 0.69 | 0.0 |
| 6 | 5000 | 9996 | 1.84 | 0.0 |
| 7 | 10000 | 19996 | 3.90 | 0.0 |

Таблица 1: Время работы и процент ошибки алгоритмов на геометрических графах

| № | Graph size | | Time, sec. | | Eroor, % | |
|---|---|---|---|---|---|---|
| | N | M | Created alg. | N4N alg. | Created alg. | N4N alg. |
| 1 | 500 | 3572 | 0.11 | 0.39 | 38.0 | 40.0 |
| 2 | 1000 | 14202 | 0.31 | 0.77 | 21.0 | 50.0 |
| 3 | 1500 | 31861 | 0.71 | 1.44 | 13.0 | 62.0 |
| 4 | 2000 | 57438 | 1.20 | 1.92 | 8.0 | 48.0 |
| 5 | 2500 | 90268 | 1.76 | 2.68 | 4.0 | 30.0 |
| 6 | 5000 | 358553 | 4.48 | 8.61 | 0.0 | 0.0 |
| 7 | 10000 | 1439255 | 13.54 | 26.0 | 0.0 | 0.0 |

лишь на графах относительно малой размерности, где применять данный подход не целесообразно, так как временные затраты точных алгоритмов не значительны. Вместе с тем на с увеличением размерности графа процент неверных ответов стремится минимуму.

## 5 Заключение

Нами был создан и реализован генетический алгоритм для решения задачи поиска центральной вершины и радиуса графа. Мы протестировали созданный алгоритм на двух моделях случайных графов, которые описывают большинство графов, встречающихся в реальности. Получив эмпирические результаты можно говорить о том, что предложенный алгоритм может быть использован для решения практических задач на графах на графах большой размерности.

## Список литературы

[1] Seidel R 1995 *J. Comput. Syst. Sci.* **51** 400–403

| № | Размеры графа | | Время работы, сек. | Процент ошибки, % |
|---|---|---|---|---|
| | N | M | | |
| 1 | 500 | 3572 | 0.11 | 38.0 |
| 2 | 1000 | 14202 | 0.31 | 21.0 |
| 3 | 1500 | 31861 | 0.71 | 13.0 |
| 4 | 2000 | 57438 | 1.20 | 8.0 |
| 5 | 2500 | 90268 | 1.76 | 4.0 |
| 6 | 5000 | 358553 | 4.48 | 0.0 |
| 7 | 10000 | 1439255 | 13.54 | 0.0 |


[2] Aingworth D, Chekuri C, Indyk P and Motwani R 1999 *SIAM J. Comput.* **28** 1167–1181

[3] Berman P and Kasiviswanathan S P 2007 Faster approximation of distances in graphs *Algorithms and Data Structures* ed Dehne F, Sack J R and Zeh N (Berlin, Heidelberg: Springer Berlin Heidelberg) ISBN 978-3-540-73951-7

[4] Chan T M 2012 *ACM Trans. Algorithms* **8** 34:1–34:17 ISSN 1549-6325 URL `http://doi.acm.org/10.1145/2344422.2344424`

[5] Roditty L and Vassilevska Williams V 2013 Fast approximation algorithms for the diameter and radius of sparse graphs *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing* STOC '13 (New York, NY, USA: ACM) pp 515–524 ISBN 978-1-4503-2029-0 URL `http://doi.acm.org/10.1145/2488608.2488673`

[6] Holland J 1975 *Adaption in Natural and Artificial Systems Adaption in Natural and Artificial Systems* (University of Michigan Press)

[7] Alkhalifah Y and Wainwright R L 2004 A genetic algorithm applied to graph problems involving subsets of vertices *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)* vol 1 pp 303–308

[8] Albert R and Barabasi A L 2002 *Reviews of Modern Physics* **74** 47–97

[9] Gilbert E 1961 *Journal of the Society for Industrial and Applied Mathematics* **9** 533–543