Д. Е. Иванов, А. С. Семенов, Алгоритмы выявления аномалий типа «черная дыра» в ориентированном графе при помощи топологического подхода, *Comp. nanotechnol.*, 2020, выпуск 2, 49–57

DOI: https://doi.org/10.33693/2313-223X-2020-7-2-49-57

05.13.11 МАТЕМАТИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ВЫЧИСЛИТЕЛЬНЫХ МАШИН, КОМПЛЕКСОВ КОМПЬЮТЕРНЫХ СЕТЕЙ

MATHEMATICAL AND SOFTWARE OF COMPUTERS AND COMPUTER NETWORKS

05.13.15 ВЫЧИСЛИТЕЛЬНЫЕ МАШИНЫ, КОМПЛЕКСЫ И КОМПЬЮТЕРНЫЕ СЕТИ

COMPUTERS, COMPLEXES AND COMPUTER NETWORKS

DOI: 10.33693/2313-223X-2020-7-2-49-57

Алгоритмы выявления аномалий типа «черная дыра» в ориентированном графе при помощи топологического подхода

Д.Е. Иванов[1, a] ©, А.С. Семенов[1, 2, 3, b] ©

[1] Московский Государственный Университет имени М.В. Ломоносова, г. Москва, Российская Федерация

[2] Научно-исследовательский центр электронной вычислительной техники (АО «НИЦЭВТ»), г. Москва, Российская Федерация

[3] Высшая школа экономики, г. Москва, Российская Федерация

[a] E-mail: mr.salixnew@gmail.com

[b] E-mail: semenov@nicevt.ru

Аннотация. В данной работе *рассмотрена задача поиска подграфа типа «черная дыра» в ориентированном графе без весов*. Постановка задачи рассматривается в той формулировке, которая дана в работе коллектива авторов из университета Нью-Джерси в 2010 г. Данная работа вносит вклад в область выявления в графе подграфов определенного вида, результаты работы могут применяться для выявления аномалий в финансовой области и природных катаклизмов, анализе городских ситуаций. Цель работы – предложить новый алгоритм для поиска «черных дыр», учитывающий структуру данного паттерна и использующий ее для более эффективного перебора возможных вариантов, рассмотреть уже существующие решения на графах гораздо большего размера по сравнению с рассмотренными предыдущими исследователями. В работе рассмотрен предложенный ранее алгоритм и его модификация iBlackholeDC на основе принципа Divide and Conquer, выделены его недостатки. Предложено использовать конденсацию графа для сокращения размера задачи. В ходе работы доказаны теоремы о строении «черных дыр» на графах. Предложен подход к перебору множеств-кандидатов, основанный на доказанных теоремах. Введены правила для сокращения такого перебора. Для сокращения перебора используется топологическая сортировка графа, а также введенное нами понятие «особая вершина». Дано определение особой вершины, доказаны ее свойства. Предложен новый алгоритм TopSortBH, задействующий конденсацию, новый подход к перебору кандидатов и сокращение перебора на основе топологии. Для TopSortBH разработана модификация FastSkip, позволяющая эффективно пропускать большие группы неподходящих кандидатов. Все рассмотренные алгоритмы реализованы, проведено экспериментальное сравнение на системе Polus. Показана эффективность конденсации в качестве предобработки графа для данной задачи. Продемонстрировано преимущество алгоритма TopSortBH и его модификации FastSkip на графах RMAT, SSCA2, Uniform Random с числом вершин до 2^{18} по сравнению с алгоритмом iBlackholeDC.

Ключевые слова: ориентированный граф, поиска подграфов в графе, подграф «черная дыра»

ССЫЛКА НА СТАТЬЮ: Иванов Д.Е., Семенов А.С. Алгоритмы выявления аномалий типа «черная дыра» в ориентированном графе при помощи топологического подхода // Computational nanotechnology. 2020. Т. 7. № 2. С. 49–57. DOI: 10.33693/2313-223X-2020-7-2-49-57

DOI: 10.33693/2313-223X-2020-7-2-49-57

Topological approach to blackholes anomaly detection in directed networks

D.E. Ivanov[1, a] ©, A.S. Semenov[1, 2, 3, b] ©

[1] **Lomonosov Moscow State University, Moscow, Russian Federation**

[2] **JSC NICEVT, Moscow, Russian Federation**

[3] **Higher School of Economics, Moscow, Russian Federation**

[a] **E-mail: mr.salixnew@gmail.com**

[b] **E-mail: semenov@nicevt.ru**

Abstract. In this paper we consider the problem of finding a blackhole pattern in directed unweighted graphs. The problem statement is the same as in an original paper by scientists from University of New-Jersey published in 2010. The paper contributes to the special graph pattern matching, the work results can be used for anomaly detection in finances, natural disasters, urban analysis. This paper aims to propose a novel algorithm for blackhole mining, which would take into account inner structure of the "blackhole" pattern and utilize this knowledge for more efficient mining. This paper reviews previously published solutions and tests them on larger graphs containing up to 1 million of nodes. In particular, an iBlackhole algorithm and its Divide and Conquer modification iBlackholeDC are considered, their weak spots are highlighted and reviewed upon. Graph condensation is introduced as an efficient preprocessing for the problem. This paper provides theorems and definitions describing inner structure of the blackhole pattern. Based on the new theorems, a new approach to enumeration of candidates is introduced as well as rules and heuristics aiming for faster filtration of candidates: they utilize topological sorting of a graph and definition of a "special" node, which is also introduced in this paper. Special nodes properties are described. We propose a novel TopSortBH algorithm. It consists of the graph condensation, candidates enumeration and heuristics for candidates filtration. The algorithm is provided with modification called FastSkip, which allows for more aggressive filtering strategy in time-sensitive cases. All mentioned algorithms are implemented and tested on the IBM Power8 based system. Experimental results show efficiency of the condensation as graph preprocessing for the problem. Strong advantage in found blackholes count is demonstrated for TopSortBH in comparison to iBlackholeDC on RMAT, SSCA2 and UR graphs containing up to 1 million nodes.

Keywords: directed networks, subgraph mining, blackhole pattern

FOR CITATION: Ivanov D.E., Semenov A.S. Topological approach to blackholes anomaly detection in directed networks. *Computational nanotechnology*. 2020. Vol. 7. No. 2. Pp. 49–57. (In Russ.) DOI: 10.33693/2313-223X-2020-7-2-49-57

1. ВВЕДЕНИЕ

Одним из важных свойств графов, которые описывают объекты реального мира, является образование в них кластеров или сообществ. Задача поиска этих сообществ имеет важное значение в социологии, биологии и компьютерных науках, поскольку объекты, изучаемые в данных прикладных областях, часто естественно моделируются при помощи графов [Fortunato, 2010].

В работе [Li et al., 2010] впервые сформулирована задача поиска подграфов (паттернов) типа «черная дыра» и «вулкан» в больших ориентированных графах. В соответствии с этой работой, черная дыра – множество вершин графа, у которого отсутствуют исходящие ребра. Вулкан – это, наоборот, множество вершин графа, у которого имеются только исходящие ребра.

Паттерны черная дыра и вулкан встречаются в прикладных задачах. Например, в торговой сети, паттерн черная дыра может представлять собой группу торговцев, которые манипулируют рынком. Также черная дыра и вулкан могут описывать схемы отмывания денег [Semenov et al., 2017]. Поиск черных дыр и вулканов в реальном времени может своевременно обнаружить пагубные явления, такие как природные катаклизмы, катастрофы, происшествия. Разработка алгоритмов поиска таких паттернов позволяет сохранять общественную безопасность [Hong et al, 2015].

Основной вклад в исследование задачи поиска паттерна черная дыра внесла группа авторов из университета Нью-Джерси. Они сформулировали задачу и предложили алгоритм поиска черных дыр в случае ориентированного графа без весов [Li et al., 2010]. Два года спустя эти же авторы опубликовали алгоритм для приближенного поиска черных дыр в случае взвешенного графа [Li et al., 2012, 2014]. Hong [Hong et al, 2015] рассматривает задачу поиска черных дыр в реальном времени для нужд современного города. В этой работе предлагается оригинальный алгоритм, приближенно решающий задачу для динамических графов.

Иванов Д.Е., Семенов А.С.

В данной статье рассматривается упрощенная постановка задачи поиска черных дыр для случая ориентированного графа без весов. Цель работы – разработать новый алгоритм для поиска черных дыр, устраняющий недостатки разработанных ранее алгоритмов, учитывающий структуру паттерна черных дыр и использующий ее для более эффективного перебора возможных вариантов. Задачей работы является исследование результатов работы алгоритмов на графах гораздо большего размера по сравнению с рассмотренными предыдущими исследователями.

2. ПОСТАНОВКА ЗАДАЧИ И ОПРЕДЕЛЕНИЯ

В этом разделе приводятся основные обозначения и определения, которые будут использованы далее в тексте.

2.1. ПОСТАНОВКА ЗАДАЧИ

Рассмотрим граф $G = G(V, E)$, где V – множество вершин, а E – множество ребер.

Определение 1. Подмножество вершин $B \subseteq V$ называется черной дырой, если одновременно выполняются следующие условия:

1) подграф $G' = (B, E')$ слабо связан;
2) нет такой пары вершин (u, v), что $u \in B$, $v \in V/B$; $(u, v) \in E$.

Задача состоит в том, чтобы в ориентированном графе без весов найти как можно больше черных дыр за ограниченное время.

2.2. ОПРЕДЕЛЕНИЯ

Введем несколько дополнительных определений

Определение 2. Пусть дан ориентированный граф $G(V, E)$. Последовательность $v_0v_1v_2 \ldots v_k$, состоящая из вершин $v_i \in V$; $0 \le i \le k$ образует ориентированный путь из v_0 в v_k, если существуют ребра $(v_{i-1}, v_i) \in E$; $1 \le i \le k$ и $v_i \ne v_j$ для всех $0 \le i, j \le k$, $i \ne j$. Длина такого ориентированного пути равна k.

Определение 3. Пусть дан ориентированный граф $G = (V, E)$. Вершина $v \in V$ достижима из $u \in V$, если существует ориентированный путь, который начинается в u и заканчивается в v.

Определение 4. Пусть дан ориентированный граф $G = (V, E)$. Если вершина $v \in V$ достижима из $u \in V$, тогда u является ориентированным предком v, а v является ориентированным потомком u. Если существует ребро из u в v, то u является непосредственным предком v, а v является непосредственным потомком u.

Определение 5. Подмножество вершин $B \subseteq G$ называется компонентой сильной связности (КСС), если все вершины B попарно взаимно достижимы.

Определение 6. Пусть дан ориентированный граф $G = (V, E)$. Рассмотрим вершину $v \in V$. Замыкание вершины Closure(v) = {u|u достижима из v} $\cup$ v. Другими словами, замыкание вершины v – это множество всех вершин, достижимых из v, включая ее саму.

Следующие утверждения были ранее доказаны в статье [Li et al., 2010].

Лемма 1. Если вершина $v \in B$, причем $B \subseteq V$ – черная дыра, то все потомки v лежат в B.

Лемма 2. Если вершина $v \in B$, причем $B \subseteq V$ – черная дыра, то замыкание v полностью содержится в B.

Лемма 3. Замыкание любой вершины образует черную дыру.

2.3. ИЗВЕСТНЫЕ ПРОБЛЕМЫ

В данном разделе рассматривается применимость алгоритма iBlackhole (алгоритм 1) с использованием модификации Divide and Conquer iBlackholeDC (алгоритм 2), предложенного в статье [Li et al., 2010]. Алгоритм iBlackholeDC создан для поиска черных дыр фиксированного размера. Он имеет значительный ресурс параллелизма, однако в работе [Там же] не рассматривается случай больших графов: практические исследования проводятся на графах, размер которых не превышает 1500 вершин.

Далее мы вводим дополнительное утверждение, чтобы описать проблемы существующего подхода.

Лемма 4. Пусть даны ориентированный граф $G = (V, E)$, КСС $S \subseteq V$, черная дыра $B \subseteq V$, тогда если $\exists v \in S : v \in B$, то $\forall s \in S$ будет верно, что $s \in B$. Другими словами, если любая вершина КСС принадлежит какой-то черной дыре, тогда все вершины данной КСС принадлежат той же черной дыре.

Доказательство

В КСС все вершины попарно взаимно достижимы. Это значит, что замыкание произвольной вершины КСС $v \in S$ содержит всю КСС как подмножество. Опираясь на лемму 3, получаем, что КСС принадлежит той же черной дыре, что и вершина v.

Существуют графы, которые имеют сравнительно небольшое количество черных дыр относительно общего количества вершин. Они состоят из нескольких больших КСС и относительно малого числа независимых листьев или корней. В связи с этим возникает два вопроса по поводу применимости алгоритма iBlackholeDC.

- Как узнать, что в графе содержатся черные дыры заданного размера? В общем случае необходимо проверить все возможные размеры, что займет много времени даже в параллельном режиме.
- Как можно сократить область поиска на стадии полного перебора и избежать повторных проверок комбинаций вершин?

Рассмотрим следующий пример. Граф состоит из 10^6 вершин, которые образуют единую КСС. Это значит, что в данном графе существует только одна черная дыра – это весь граф. Пока мы не знаем ничего о структуре графа и не обладаем достаточно большими вычислительными ресурсами, необходимо будет перебирать слишком большое число возможных размеров черных дыр. В худшем случае нам придется перебрать все возможные размеры графа.

Другой пример – это графы малого мира [Watts, 1999]. Большая часть вершин такого графа содержится в единственной большой КСС. Также имеется несколько вершин и корней. Мы утверждаем, что при просмотре от меньших размеров черной дыры к большим, алгоритм iBlackholeDC быстро обнаружит все небольшие черные дыры. Далее он будет очень долго пытаться найти черные дыры внутри крупных КСС, где их попросту нет, пока не доберется до размеров больше размера КСС.

3. РАЗРАБОТКА ТОПОЛОГИЧЕСКОГО АЛГОРИТМА

Мы предлагаем рассмотреть решение задачи поиска черных дыр при помощи комбинации двух этапов. Первый этап состоит в предобработке графа, задача которого – упростить структуру графа настолько, насколько это возможно. Уменьшение размера графа значительно сокращает вычисления, которые в худшем случае являют собой полный

перебор вариантов. Второй этап – это непосредственно поиск черных дыр. Целью данного этапа является обнаружение как можно большего числа черных дыр при заданном ограничении по времени.

3.1. ПРЕДОБРАБОТКА ГРАФА

Как было сказано выше, большие КСС требуют значительных вычислительных ресурсов для обработки и при этом не образуют дополнительных черных дыр. Это может быть критично при обработке графов малого мира, потому что одна КСС может включать до 100% вершин некоторого графа. В таком случае будет удобно рассмотреть КСС как единую вершину, которая объединяет в себе все входящие и исходящие связи периферических вершин КСС. На стадии предобработки происходит сжатие каждой КСС до единственной вершины. Процесс такого сжатия известен как конденсация графа.

Определение 7. Если каждую компоненту сильной связности заменить единственной вершиной, то получится ориентированный ациклический граф, который называется конденсацией исходного графа.

В данной работе мы используем алгоритм Шарира для поиска компонент связности графа [Sharir, 1999].

3.2. ПОИСК ЧЕРНЫХ ДЫР

После предобработки графа происходит переход к этапу поиска черных дыр. Задача имеет комбинаторную природу, поэтому на данном этапе основная цель – это уменьшить число потенциальных кандидатов. Опишем сначала подход, основанный на полном переборе вершин.

Определение 8. Корень черной дыры $B \subseteq V$ – это такая вершина $v \in B$, что не найдется ребра $(u, v) \in E$, где $u \in B$, $u \neq v$.

Определение 9. Множество всех корней черной дыры называется базисом черной дыры.

Отметим, что базис черной дыры может состоять из произвольного числа вершин, отличного от нуля.

Во время полного перебора мы просматриваем все возможные неупорядоченные подмножества вершин графа. Далее для каждой вершины мы получаем ее замыкание. Объединение всех таких замыканий является кандидатом, потенциальной черной дырой. Если кандидат является слабо связным подграфом, тогда это черная дыра.

Такой подход не застрахован от повторного обнаружения уже известной черной дыры. Для уменьшения количества повторного обнаружения уже известных дыр мы вводим эвристику.

Согласно определениям, данным выше, если какая-то вершина не является корнем черной дыры, то она может быть опущена, так как набор корней описывает черную дыру единственным образом. Каждая не корневая вершина в черной дыре достижима из какого-то корня этой дыры. Значит, если мы располагаем матрицей достижимости для данного графа, то мы можем эффективно определять лишние, не являющиеся базисами, комбинации вершин. Конечно же, мы могли бы прямо вычислить такую матрицу достижимости, но это вычислительно сложная задача $O(V^3)$, что не позволяет применить такое правило для больших графов. Мы можем столкнуться с нехваткой как памяти, так и времени на предобработку. Надо отметить, что неплохо бы получить первые черные дыры как можно скорее после старта, что ограничивает нас в использовании особенно дорогих вычислений. Поэтому мы воспользуемся эвристикой, которая является частным случаем описанного правила, чтобы частично сократить поле перебора, не жертвуя временем первого ответа. Те дубликаты, которые не могут быть отфильтрованы эвристикой, будут проверены «в лоб».

Далее мы используем идею топологической сортировки графа.

Определение 10. Топологическая сортировка или топологическое упорядочивание ациклического орграфа – такой массив вершин, что для любого ребра $(u, v) \in E$ вершина v будет иметь индекс меньше, чем u.

Определение 11. Пусть дан ориентированный граф $G(V, E)$, корень $r \in Roots(G)$, $topSortOrder_r$ – топологическая сортировка замыкания $Closure(r)$ и $|Closure(v)|$ – размеры замыканий для всех вершин $v \in Closure(r)$, то есть доступных из r. Вершина v называется особой вершиной под корнем r, если размер ее замыкания $|Closure(v)|$ равен ее индексу в массиве $topSortOrder_r$ плюс один. Заметим, что множество $Special_r$ особых вершин под корнем r полностью содержится в замыкании данного корня, то есть

$$Special_r \subseteq Closure(r).$$

Замечание 1. Решение о том, является ли вершина особой, принимается независимо в поддереве каждого корня. Одна и та же вершина может быть особой в поддереве одного корня и не являться таковой в поддереве другого корня. Пример такой ситуации отображен на рис. 1.

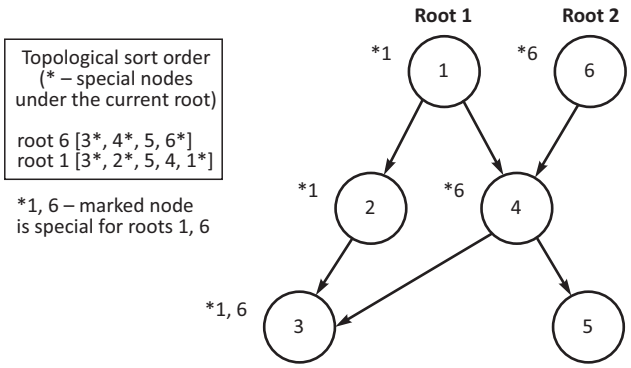

Рис. 1. Special nodes in a graph with 2 roots.
* *x, y* – marked node is special for roots *x, y*

Определение 12. Пусть дан ориентированный граф $G(V, E)$, один из его корней $r \in Roots(G)$, $topSortOrder_r$ – топологическая сортировка замыкания $Closure(r)$, $v \in Special_r$ и $u \in Closure(r)$. Вершина v доминирует над вершиной u под корнем r, если v имеет больший индекс, чем u в массиве $topSortOrder_r$.

Замечание 2. Если дано две особых вершины под одним корнем, то одна из них всегда будет доминировать над другой.

Теорема 1. Пусть дан ориентированный граф $G(V, E)$, один из его корней $r \in Roots(G)$, v , $u \in Closure(r)$. Если v доминирует над u под корнем r, значит u достижима из v.

Доказательство. По определению особой вершины размер замыкания v равен ее индексу плюс один в $topSortOrder_r$. Рассмотрим все вершины, отличные от v в замыкании $Closure(v)$. Эти вершины в массиве $topSortOrder_r$ имеют индексы меньше, чем вершина v. Отсюда $\forall u \in Closure(v)$ вершина u будет достижима из v.

Теорема 2. Пусть дан ориентированный граф $G(V, E)$, один из его корней $r \in Roots(G)$, $v \in Special_r$, $u \in Closure(r)$. Если v доминирует над u под корнем r, значит

$$Closure(v) \cup Closure(u) = Closure(v).$$

Доказательство. Согласно предыдущей теореме, если v доминирует над u под корнем r, то вершина u будет достижима из v. Тогда $u \in Closure(v)$, откуда следует

$$Closure(u) \cup Closure(v).$$

Определение 13. Те вершины графа $G(V, E)$, которые не имеют входящих ребер, называются глобальными корнями.

Рассмотрим две вершины в замыкании глобального корня. Одна из них особая, а другая обычная. Чтобы эти две вершины образовали базис, особая вершина не должна доминировать над обычной. В противном случае обычная вершина могла бы быть опущена как не значимая. В общем случае, если дан набор вершин, мы можем опустить все доминируемые вершины и получить корректный базис.

Алгоритм IsBasis (алгоритм 3) проверяет, является ли набор вершин базисом. Этот алгоритм принимает набор вершин и возвращает True, если данный набор образует базис и False в противном случае. Этот алгоритм используется как составная часть алгоритма TopSortBH (алгоритм 4), который перебирает наборы-кандидаты и отфильтровывает некорректные наборы. Если IsBasis возвращает False для кандидата, то этот набор может быть пропущен.

IsBasis действует следующим образом.

В первую очередь нам нужно знать, какие вершины являются особыми. Этот шаг может быть посчитан заранее. Чтобы определить особые вершины, мы строим топологическую сортировку замыкания для каждого глобального корня. Далее по определению отмечаем особые вершины.

Вторым шагом, если мы находим, что какие-то вершины доминируемы, то мы опускаем данный набор вершин.

На третьем шаге закончились дешевые методы принятия решения. Проверяем попарную достижимость вершин в наборе. Если найдется хоть одна такая пара, опускаем данный набор вершин.

Если ни одна из предыдущих проверок не ответила *False* (опустить), то IsBasis возвращает *True*. Это значение возвращается в алгоритм TopSortBH и означает, что мы имеем дело с корректным базисом черной дыры. Для получения черной дыры мы должны, согласно определению, объединить все замыкания базисных вершин. Последним шагом убедимся, что полученное множество является слабо связным, это будет означать, что мы получили черную дыру.

Очевидно, что процесс не оптимален, но исследования демонстрируют его эффективность. Мы успешно избегаем дорогой проверки на слабую связность в большом количестве случаев.

Также мы вводим модификацию SkipFast для нашего алгоритма. Она позволяет быстрее находить черные дыры в ситуациях, когда время ограничено, но допускает ложно отрицательные результаты.

Пусть дан массив *pos*, описывающий набор кандидатов в порядке топологической сортировки и содержащий в себе особую вершину на некоторой позиции $skip \neq 0$. Сформируем массив *pos'* такой, что единственная особая вершина, если такая найдется, будет находиться по индексу равному 0. Мы знаем, что первая особая вершина находится в позиции *skip* и имеет номер $skipVal = pos[skip]$. Перейдем от массива [..., *skipVal*, ...] к массиву

$$[skipVal, skipVal + 1, \dots, skipVal + pos.size() - skip].$$

Такой набор по построению будет иметь особую вершину на первой позиции, но не обязательно будет являться корректным базисом. При таком переходе мы пропустим большое число вариантов с одинаковым некорректным префиксом.

4. РЕЗУЛЬТАТЫ ЭКСПЕРИМЕНТА

Чтобы показать эффективность нашего подхода, мы провели серию экспериментов, состоящую из сравнения производительности алгоритмов iBlackholeDC и TopSortBH. Для данных экспериментов мы выбрали равномерно случайные (Uniform Random, UR) [Erdos et al. 1959], RMAT [Chakrabarti et al. 2004] SSCA2 [Bader et al. 2005] типы графов. Они были сгенерированы для масштабов от 4 до 18. Масштаб *i* означает, что граф имеет 2^i вершин и приблизительно $32 * |V|$ ребер.

Эксперименты были проведены на одном узле вычислительной системы Polus, установленной на факультет ВМК МГУ им. М.В. Ломоносова, в котором установлено 2 десятиядерных процессора IBM POWER 8, оперативная память узла – 256 ГБ. Запуски проводились в однопоточном режиме. Ограничение по времени работы для всех запусков составило 30 минут.

На рис. 2, 3 показано сравнение производительности на графах RMAT. По рис. 2 видим, что на графах масштаба до 11 включительно TopSortBH и iBlackholeDC с использованием конденсации успевают найти все черные дыры и завершиться. В то же время iBlackholeDC без конденсации работает значительно дольше на всех размерах графов и не успевает завершиться в случае *scale* = 11.

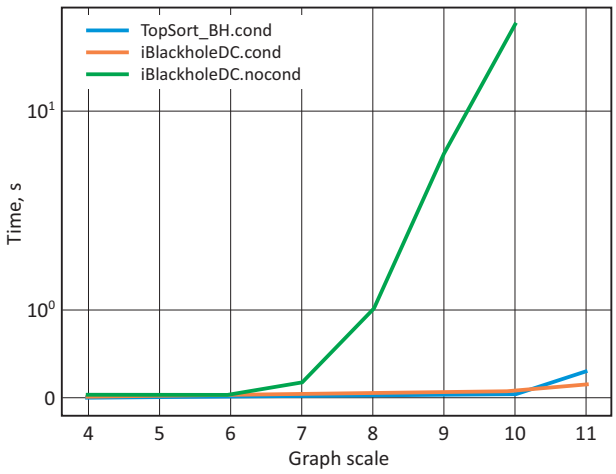

Рис. 2. Execution time on RMAT graphs for scales up to 11. Timeout: 30 minutes

Для графов больших размеров (см. рис. 3), ни один из алгоритмов не успевает отработать полностью, поэтому производится сравнение количества черных дыр, найденных за полчаса работы. TopSortBH без модификации SkipFast показывает превосходящий результат, тем не менее, с увеличением размера графа его преимущество постепенно уменьшается. С использованием модификации SkipFast наблюдаем увеличение числа найденных черных дыр на несколько порядков по сравнению с другими алгоритмами.

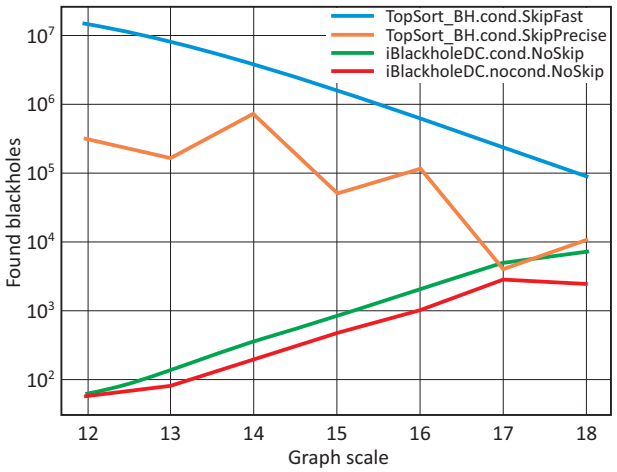

Рис. 3. Found blackholes count on RMAT graphs for scales from 12 to 18. Timeout: 30 minutes

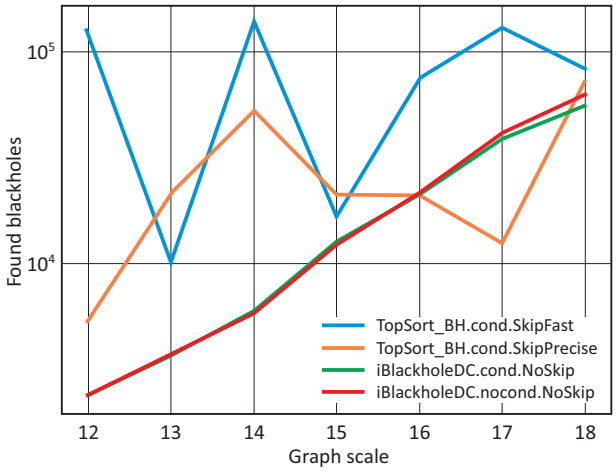

Рис. 5. Found blackholes count on SSCA2 graphs for scales from 12 to 18. Timeout: 30 minutes

Для графов SSCA2 все алгоритмы успевают завершиться для масштабов графа не превышающих 6. Поэтому на рис. 4–5 показано только количество черных дыр, которые были обнаружены за 30 минут работы. Видимо, что алгоритм TopSortBH демонстрирует преимущество в большинстве случаев, которое как и для RMAT графов, может быть увеличено при использовании модификации SkipFast. В то же время, на SSCA2 графах это преимущество не настолько значительное, а иногда и вовсе отсутствует. Скачки в количестве найденных черных дыр возникают из-за различных размеров черных дыр и неравномерного их распределения в области перебора.

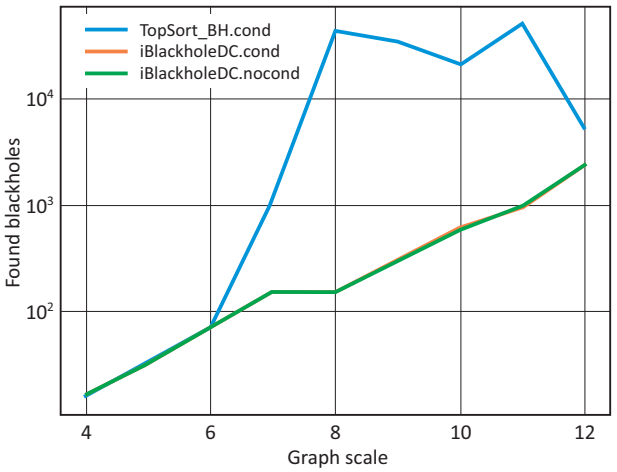

Рис. 4. Found blackholes count on SSCA2 graphs for scales up to 11. Timeout: 30 minutes

Uniform Random графы всех размеров в нашем случае состояли из единственной большой КСС, поэтому для всех алгоритмов, использующих конденсацию графа, время работы отличается незначительно, а алгоритм iBlackholeDC без использования конденсации заметно проигрывает по времени работы даже на графах небольших размеров. Для графов данного типа все алгоритмы успевали завершиться для всех размеров графов. Время выполнения приводится на рис. 6–7.

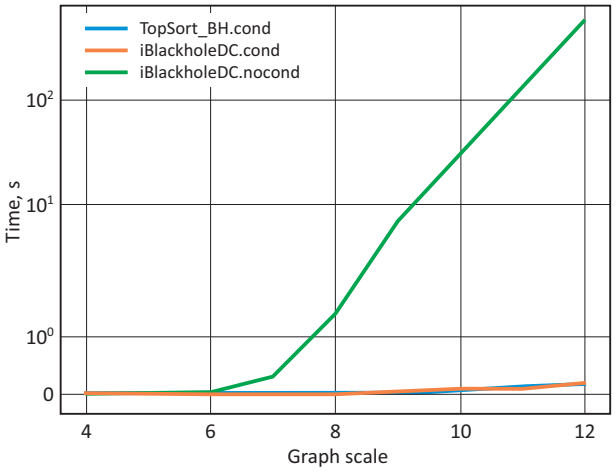

Рис. 6. Execution time on UR graphs for scales up to 11. Timeout: 30 minutes

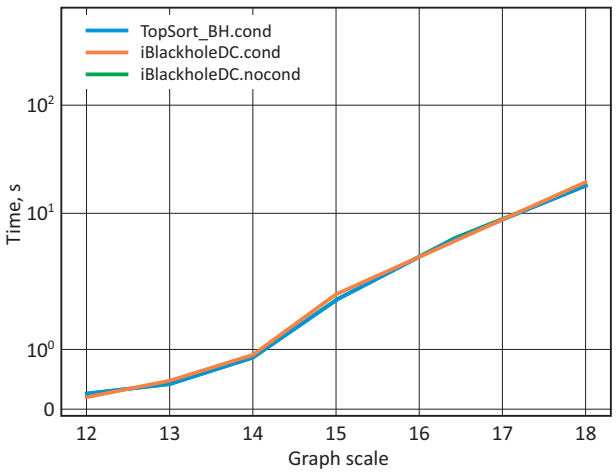

Рис. 7. Execution time on UR graphs for scales from 12 to 18. Timeout: 30 minutes

5. ЗАКЛЮЧЕНИЕ

В данной статье мы рассмотрели проблемы, связанные с использованием уже известного алгоритма iBlackholeDC для решения задачи поиска черных дыр в графе. Мы предложили и протестировали новый алгоритм TopSortBH. В рамках работы над ним мы предложили предобработку графа, которая раньше не применялась для решения рассматриваемой задачи, оригинальный подход к перебору черных дыр, а также способы сокращения этого перебора. Мы проверили наш подход на трех различных видах графов в различных масштабах. Мы впервые описываем поиск черных дыр для графов, состоящих из более миллиона вершин.

Для графов, содержащих большие компоненты сильной связности, мы демонстрируем уверенное преимущество в скорости работы перед алгоритмом iBlackholeDC. Кроме того, мы умеем относительно быстро определять ситуацию полного отсутствия черных дыр. Однако в ряде случаев, довольно дорогая предобработка нивелирует выигрыш по времени поиска, ухудшая производительность нашего алгоритма.

Литература / References

1. *Bader D.A., Madduri K*. Design and implementation of the hpcs graph analysis benchmark on symmetric multiprocessors. *International Conference on High Performance Computing*. Springer, 2005. Pp. 465–476.
2. *Chakrabarti D., Zhan Y., Faloutsos C.* R-mat: A recursive model for graph mining. *Proceedings of the 2004 SIAM International Conference on Data Mining*. SIAM, 2004. Pp. 442–446.
3. *Erdoos P., Reyni A.* On random graphs. *Publicationes Mathematicae.* 1959. No. 6. Pp. 290–297.
4. *Fortunato S*. Community detection in graphs. *Physics Reports*. 2010. No. 486 (3-5). Pp. 75–174.
5. *Hong L., Zheng Y., Yung D.* et al. Detecting urban black holes based on human mobility data. *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2015. P. 35.
6. *Li Z., Xiong H*. Mining blackhole and volcano patterns for fraud detection. In: Encyclopedia of Social Network Analysis and Mining. 2014. Pp. 904–915.
7. *Li Z., Xiong H., Liu Y*. Mining blackhole and volcano patterns in directed graphs: A general approach. *Data Mining and Knowledge Discovery*. 2012. No. 25 (3). Pp. 577–602.
8. *Li Z., Xiong H., Liu Y., Zhou A*. Detecting blackhole and volcano patterns in directed networks. *2010 IEEE International Conference on Data Mining*. 2010. Pp. 294–303.
9. *Semenov A., Mazeev A., Doropheev D.* et al. Survey of common design approaches in aml software development. *GraphHPC 2017 Conference (GraphHPC). CEUR Workshop Proceedings*. 2017. Vol. 1981. Pp. 1–9.
10. *Sharir M*. A strong-connectivity algorithm and its applications in data flow analysis. *Computers & Mathematics with Applications*. 1981. No. 7 (1). Pp. 67–72.
11. *Watts D.J*. Networks, dynamics, and the small-world phenomenon. *American Journal of sociology.* 1999. No. 105 (2). Pp. 493–527.

ПРИЛОЖЕНИЕ

Algorithm 1: iBlackhole

Input: $G(V, E)$ – ориентированный граф, V – множество вершин, E – множество ребер, n – максимальное число вершин, которое может содержать черная дыра
Output: *Blackholes* – множество всех черных дыр графа размером от 1 до n
Blackholes = ∅, C_0 = ∅.

for i **in** 1 to n **do**
  $P_i = \{v \mid d_{out}(v) < i\}$ // $d_{out}(v)$ – количество исходящих из v ребер
  foreach $v \in P_i$ **do**
    if $v \in C_{i-1}$ **then** v
      if как минимум один из непосредственных потомков v не принадлежит P_i **then**
        удалить v из P_i
        удалить всех предков v из P_i
      end
    end
  end
  $C_i = P_i$ // C_i – список кандидатов
  foreach $v \in C_i$ **do**
    if $|Closure(v)| == i$ **then**
      $Blackholes = Blackholes \cup Closure(v)$
    end
    if $|Closure(v)| >= i$ **then**
      удалить v из C_i
      удалить всех предков v из C_i
    end
  end
  /* Применить алгоритм полного перебора к $C_i(i)$ – множеству всех подмножеств, содержащих i вершин */
  foreach $B \subset C_i(i)$ **do**
    if $G(B)$ слабо связан **then**
      if $d_{out}(B) == 0$ **then**
        $Blackholes = Blackholes \cup B$
      end
    end
  end
end
return *Blackholes*

Algorithm 2: iBlackholeDC

Input: $G(V, E)$ – ориентированный граф, V – множество вершин, E – множество ребер
n – максимальное число вершин, которое может содержать черная дыра
Output: *Blackholes* – множество всех черных дыр размером от 1 до n
$Blackholes = \emptyset$, $C_0 = \emptyset$
for i **in** 1 to n **do**
$P_i = \{v \mid d_{out}(v) < i\}$ // $d_{out}(v)$ – количество исходящих из v ребер
убрать лишние вершины из P_i и сформировать список кандидатов C_i
убрать лишние вершины из C_i и сформировать список кандидатов F_i
foreach $WCC \in G(F_i)$ **do**
/* Применить алгоритм полного перебора к $WCC_i(i)$ – множеству всех подмножеств, содержащих i вершин */
foreach $B \subset WCC_i(i)$ **do**
if $G(B)$ слабо связан **then**
if $d_{out}(B) = = 0$ **then**
$Blackholes = Blackholes \cup B$
end
end
end
end
end
return *Blackholes*

Алгоритм 3: IsBasis

Input: $G(V, E)$ – орграф без весов, не содержащий циклов
$Cand \subset V$ – набор вершин, потенциальный базис черной дыры
Output: *True*, если *Cand* – корректный базис, иначе – *False*.
/* (1) найти особые вершины для каждого корня */
foreach $r \in Roots(G)$ **do**
$Special_r = \emptyset$
построить $topSortOrder_r$
for $i = 0$; $i < |topSortOrder_r|$; $i = i + 1$ **do**
$v = topSortOrder_r[i]$
$C_v = Closure(v)$
if $|C_v| = = i$ **then**
$Special_r = Special_r \cup v$
$maxSpecialIndex_r = max(SpecialIndexr, i)$
end
end
end
/* (2) проверить наличие доминируемых вершин */
foreach $r \in Roots(G)$ **do**
$maxSpecialIndex_r = -1$
end
foreach $v \in Cand$ **do**
foreach $r \in Roots(G)$ **do**
if $v \in Special_r$ **then**
$vTopSortIndx_r = \arg_v(topSortOrder_r)$
$maxSpecialIndex_r = max(maxSpecialIndex_r, vTopSortIndex_r)$
end
end
end
foreach $v \in Cand$ **do**
foreach $r \in Roots(G)$ **do**
if $v \in Closure(r)$ **then**
$vTopSortIndex_r = arg_v(TopSortIndex_r)$
if $vTopSortIndex_r < maxSpecialIndex_r$ **then**
return *False*
end
end
end
end

```
/* (3) Проверка попарной достижимости */
foreach (v, u) ∈ Cand(G)xCand(G) do
        if v достижима из u then
                return False
        end
end
return True
```

Алгоритм 4: TopSortBH

Input: $G(V, E)$ – орграф без весов
Output: *Blackholes* – множество всех черных дыр размером от 1 до n
$Blackholes = \emptyset$

```
Построить конденсацию графа Cond(V', E')
for i = 1 to |V'| do
        foreach B_i ∈ V'(i) do
                if IsBasis(Cond(V', E'), B_i) then
                        if B_i – слабо связанный подграф then
                                Blackholes = Blackholes ∪ B_i
                        end
                end
        end
end
return Blackholes
```

Статья поступила в редакцию 07.05.2020, принята к публикации 18.06.2020
The article was received on 07.05.2020, accepted for publication 18.06.2020

СВЕДЕНИЯ ОБ АВТОРАХ

Иванов Денис Евгеньевич, магистр Московского государственного университета имени М.В. Ломоносова. Москва, Российская Федерация. E-mail: mr.salixnew@gmail.com

Семенов Александр Сергеевич, кандидат технических наук; начальник отдела АО «Научно-исследовательский центр электронной вычислительной техники» (АО “НИЦЭВТ”); доцент Московского государственного университета имени М.В. Ломоносова; ведущий научный сотрудник, Высшая школа экономики. Москва, Российская Федерация. ORCID: 0000-0003-4878-6287. ResearcherID: V-4265-2018. Scopus, AuthorID: 57061601800. E-mail: semenov@nicevt.ru

ABOUT THE AUTHORS

Denis E. Ivanov, graduate student at the Lomonosov Moscow State University. Moscow, Russian Federation. E-mail: mr.salixnew@gmail.com

Alexander S. Semenov, Cand. Sci. (Eng.); Head of a Department at the JSC NICEVT; associate professor at the Moscow State University; leading researcher at the Higher School of Economics. Moscow, Russian Federation. ORCID:0000-0003-4878-6287. ResearcherID:V-4265-2018. Scopus, AuthorID: 57061601800. E-mail: semenov@nicevt.ru