ПРИКЛАДНАЯ ТЕОРИЯ ГРАФОВ

УДК 519.174.2 DOI 10.17223/20710410/68/5

ОБХОДЫ ГРАФОВ, РЕАЛИЗУЕМЫЕ ИТЕРАЦИОННЫМИ МЕТОДАМИ РЕШЕНИЯ СИСТЕМ ЛИНЕЙНЫХ УРАВНЕНИЙ

А. В. Пролубников

Новосибирский государственный университет, г. Новосибирск, Россия

E-mail: a.v.prolubnikov@mail.ru

Обходы графов используются для решения многих задач. Обычные варианты обхода графа — это поиск в глубину и в ширину. При обходе связного графа последовательно достигаются все его вершины в результате переходов по рёбрам. Поиск в ширину — обычный выбор при построении эффективных алгоритмов нахождения компонент связности графа. Методы простой итерации для решения систем линейных алгебраических уравнений с модифицированными матрицами смежности графов и заданной правой частью могут быть рассмотрены как алгоритмы обхода графа. Эти алгоритмы дают обходы, вообще говоря, отличные от обходов графа в глубину и в ширину. Примером такого алгоритма является алгоритм обхода графа, который даёт метод Гаусса — Зейделя. Для произвольного связного графа этому алгоритму требуется количество итераций не большее, чем для обхода в ширину. Для большого количества индивидуальных задач достаточно меньшего числа итераций.

Ключевые слова: *обходы графов, задачи о связности на графах.*

GRAPH TRAVERSALS IMPLEMENTED BY ITERATIVE METHODS FOR SOLVING SYSTEMS OF LINEAR EQUATIONS

A. V. Prolubnikov

Novosibirsk State University, Novosibirsk, Russia

Graph traversals, such as depth-first search and breadth-first search, are commonly used to solve many problems on graphs. By implementing a graph traversal, we consequently reach all graph vertices that belong to a connected component. The breadth-first search is the usual choice when constructing efficient algorithms for finding connected components of a graph. Methods of simple iteration for solving systems of linear equations with modified graph adjacency matrices can be considered as graph traversal algorithms if we use a properly specified right-hand side. These traversal algorithms, generally speaking, turn out to be neither equivalent to depth-first search nor to breadth-first search. An example of such a traversal algorithm is the one associated with the Gauss — Seidel method. For an arbitrary connected graph, the algorithm requires no more iterations to visit all its vertices than it takes for breadth-first search. For a large number of instances of the problem, fewer iterations will be required.

Keywords: *graph traversals, connectivity problems on graphs.*

Введение

Многие задачи, связанные с надёжностью транспортных сетей, сетей передачи данных, больших интегральных схем и др. формулируются как задачи о связности на графах. Примеры таких задач — задачи нахождения компонент связности, точек сочленения, мостов и др.

Пусть G — обыкновенный граф, то есть неориентированный невзвешенный граф без кратных рёбер и петель; V — множество его вершин, E — множество рёбер, имеющие соответственно мощности n и m. Вершины графа помечены (пронумерованы) в некотором произвольном порядке от 1 до n. *Компонента связности* графа — это максимальный по включению его связный подграф. Компонента связности определяется множеством вершин такого подграфа.

Обход графа представляет собой итеративный процесс, в ходе которого, начиная с некоторой *стартовой вершины*, производятся переходы по рёбрам графа. Обход графа завершается, когда посещены все его вершины. Под реализацией обхода понимается процесс получения последовательности вершин, которые посещаются в результате переходов по рёбрам.

Обычными вариантами реализации обхода графов для решения задач о связности являются поиск в глубину и поиск в ширину. *Поиск в глубину* (Depth-First Search, DFS) [1] представляет собой рекурсивную процедуру, в ходе которой производятся переходы через все вершины, смежные текущей достигнутой вершине. Если через ребро достижима ещё не посещённая вершина, то через него совершается переход в эту вершину, из которой рекурсивно запускается алгоритм обхода. Возврат из рекурсии происходит, если для текущей вершины среди смежных ей нет ещё не посещённых.

Для вычислительно эффективного нахождения компонент связности больших графов обычно используется *поиск в ширину* (Breadth-First Search, BFS) [1]. При проведении такого обхода, начиная со стартовой вершины, принадлежащей компоненте связности графа, производятся переходы в ещё не посещённые вершины, смежные вершинам, посещённым на предыдущей итерации. В ходе таких итераций строится *дерево достижимости* графа; $(k+1)$-му уровню этого дерева принадлежат вершины, достигаемые в результате последовательных переходов по k рёбрам некоторой простой цепи в графе. Построение дерева достижимости для компоненты связности производится за $\ell_{\max}$ итераций, где $\ell_{\max}$ — длина кратчайшей простой цепи, соединяющей стартовую вершину с наиболее удалённой от неё вершиной.

BFS был впервые использован К. Цузе в 1945 г., но не был опубликован с указанием автора [2] до 1972 г. Впервые BFS опубликован Э. Муром в 1959 г. в контексте поиска пути в лабиринте [3]. Позже его независимо от Мура предложил Ч. Ли в контексте разводки проводников на печатных платах [4].

BFS и его обобщения для взвешенных графов лежат в основе многих алгоритмов решения задач дискретного анализа и дискретной оптимизации. Обходы используются для нахождения дерева кратчайших путей в графе, проверки графа на двудольность, нахождения максимального потока в сети и др.

Пусть $s \in V$ — стартовая вершина. *Алгебраический BFS* представляет собой реализацию BFS через последовательное умножение вектора на матрицу смежности A графа:

$$x^{(k+1)} = Ax^{(k)}. \tag{1}$$

Здесь $x^{(0)} = e_s$; e_s — s-й единичный вектор стандартного базиса в $\mathbb{R}^n$, то есть все его компоненты нулевые, за исключением s-й. На каждой итерации некоторые компонен-

ты этого вектора становятся ненулевыми. Индексы таких компонент соответствуют посещённым на этой итерации вершинам.

Алгебраический BFS реализован в библиотеках программ, наиболее эффективно использующих возможности современных компьютерных архитектур для параллельных вычислений и оптимизации работы с памятью. Они предназначены для работы с разреженными графами, то есть с теми, для которых $m = O(n)$ и которые наиболее часто встречаются в приложениях. Такие низкоуровневые реализации алгебраического BFS при решении практических задач требуют меньше времени для обхода графа, чем реализации теоретически оптимального комбинаторного BFS [5–9]. Поскольку вычислительная сложность BFS составляет $O(m + n)$, для разреженных графов алгебраический BFS позволяет получать реализации с наименьшей возможной для задачи линейной вычислительной сложностью [10].

Однако несмотря на то, что вычисления (1) допускают эффективные параллельные реализации при нахождении текущего уровня дерева достижимости, не существует реализаций BFS с вычислительной сложностью меньше линейной, то есть имеющих сложность $O(n^c)$, где $0 < c < 1$.

Таким образом, имеется два принципиально отличных друг от друга подхода к численной реализации обхода графа — комбинаторный и алгебраический (линейно-алгебраический).

При реализации *комбинаторного* подхода, начиная с некоторой стартовой вершины, последовательно рассматриваются возможные варианты переходов по рёбрам, инцидентным текущей достигнутой вершине. После этого совершаются переходы в ещё не посещённые вершины по некоторым выбранным рёбрам.

При реализации *алгебраического* подхода на каждой итерации производится линейное преобразование *вектора состояния* $x^{(k)}$. Анализ компонент этого вектора позволяет определить вершины, посещённые в ходе итерации. Посещение вершины $i \in V$ регистрируется в случае, если значение i-й компоненты вектора состояния перестаёт быть нулевым. Примером реализации алгебраического подхода является алгебраический BFS, который в качестве линейного преобразования использует умножение вектора на матрицу смежности графа.

Рассматриваемые нами алгоритмы обхода графа используют другие линейные преобразования. Эти преобразования реализуются методами простой итерации решения систем линейных алгебраических уравнений (СЛАУ). Варианты метода простой итерации решения СЛАУ с модифицированными матрицами смежности графов и заданной правой частью могут быть рассмотрены как обходы графов. Эти методы дают два варианта обхода графа, первый из которых реализуется методом Якоби и эквивалентен BFS, второй — методом Гаусса — Зейделя, и он не эквивалентен BFS.

Назовём простую цепь в графе *правильной*, если номера её вершин образуют возрастающую последовательность. В отличие от DFS и BFS, при получении обхода графа с помощью итераций метода Гаусса — Зейделя учитывется нумерация вершин в графе, которая является обязательным параметром задачи. Такой алгоритм обхода не эквивалентен ни BFS, ни DFS. На каждой его итерации сначала производится итерация BFS и если при этом были достигнуты вершины, из которых начинаются правильные цепи, то производятся переходы по всем рёбрам этих правильных цепей.

Для обхода произвольного связного графа с помощью такого алгоритма нужно произвести не больше итераций, чем требуется для BFS. При этом для большого количества графов понадобится меньше итераций. Если в графе имеется достаточно много

правильных цепей, то даже последовательная реализация такого алгоритма может давать обход графа быстрее, чем параллельная версия BFS.

1. Итеративные численные методы решения СЛАУ и обходы графов, ассоциированные с ними

Рассмотрим итерации методов Якоби и Гаусса — Зейделя для решения СЛАУ

$$Ax = b. \tag{2}$$

Оба метода являются вариантами метода простой итерации. Итерации метода Якоби имеют следующий вид:

$$x^{(k+1)} = b - D^{-1}Ax^{(k)},$$

где D — диагональная матрица с диагональными элементами матрицы A. То есть для i-й компоненты приближённого решения, получаемого после $(k+1)$-й итерации, имеем

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i} a_{ij} x_j^{(k)} \right). \tag{3}$$

Итерации метода Гаусса — Зейделя имеют следующий вид:

$$(L + D)x^{(k+1)} = -Ux^{(k)} + b,$$

где L и U — матрицы с элементами матрицы A, находящимися соответственно под её диагональю и над ней, то есть

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right). \tag{4}$$

Для того чтобы СЛАУ с матрицами графов имели решение, а используемые численные методы их решения сходились, мы можем модифицировать матрицы смежности, заменяя нулевые элементы на их диагонали на значения $d > 0$, как это делается в [11, 12]. Если матрица A имеет диагональное преобладание, то есть для $i = 1, \ldots, n$ имеем

$$|a_{ii}| = d \geqslant \sum_{i \neq j} |a_{ij}|,$$

где суммирование ведётся по j от 1 до n, и если хотя бы одно из этих неравенств строгое, то приближённые решения, получаемые на итерациях (3) и (4), сходятся к точному решению, которое существует и единственно для любой правой части b.

Рассмотрим СЛАУ (2), где A — модифицированная матрица смежности с диагональным преобладанием и $b = e_s$. Алгоритмы обхода графа, рассматриваемые далее, производят итерации (3) и (4). После проведения не более чем n итераций (3) или (4) с начальным вектором $x^{(0)} = d \cdot e_s$, возможно, без достижения сходимости к точному решению СЛАУ (2), мы получаем последовательность приближённых решений $x^{(k)}$. Далее будем рассматривать их как текущее значение вектора состояния $x^{(k)}$ на k-й итерации, нежели как приближённые решения СЛАУ. Для проведения обходов графов с помощью методов решения СЛАУ диагональное преобладание не обязательно. Достаточно того, чтобы на диагонали модифицированной матрицы смежности графа были ненулевые значения, равные некоторому заданному d.

Переходы между вершинами графа в ходе этих итераций регистрируются следующим образом. Переход из вершины, достигнутой на предыдущей итерации, в вершину $i \in V$ происходит на $(k+1)$-й итерации, если выполнено следующее условие:

$$(x_i^{(k)} = 0) \quad \text{и} \quad (x_i^{(k+1)} \neq 0). \tag{5}$$

То есть вершина i не была достигнута на итерациях, предшествующих $(k+1)$-й итерации, и была достигнута после её выполнения. Вершина j, из которой был совершён переход в вершину i по ребру графа, может быть определена в ходе выполнения итерации (3) или (4). *Фронтир* $\mathcal{F}^{(k+1)}$ для $(k+1)$-й итерации — это множество вершин, которые были достигнуты в результате её выполнения; $\mathcal{F}^{(0)} = \{s\}$.

Производя итерации метода Якоби или метода Гаусса — Зейделя с заданными начальным значением вектора состояния и правой частью СЛАУ (2), последовательно получаем фронтиры для соответствующих им обходов графа, в результате посещая все вершины связного графа.

Методы простой итерации решения СЛАУ дают два отличных друг от друга алгоритма обхода графа. Один из них может быть получен проведением итераций (3) и, как мы покажем далее, он эквивалентен BFS, тогда как другой может быть получен проведением итераций (4), и он не эквивалентен BFS.

2. Использование операции умножения вместо операции деления в итерациях методов Якоби и Гаусса — Зейделя

Поскольку для проведения обхода графа с помощью итераций (3) и (4) нет необходимости добиваться сходимости к точному решению СЛАУ (2), вместо операции деления в (3) и (4) может быть использовано умножение. Не влияя на обходы, производимые при выполнении итераций, и на доказательства, которые проводятся далее, это позволяет сделать текст более компактным. После такой модификации значения $x_i^{(k+1)}$ становятся полиномами от d, а не от $1/d$, что имеет место при использовании деления.

Более того, такая модификация имеет и практический смысл, поскольку реализация операции деления машинных чисел с плавающей точкой в несколько раз медленней, чем реализация операции умножения. В результате уменьшается время, требуемое для проведения обхода.

Таким образом, далее, сопоставляя графу матрицу смежности, в которой нулевые диагональные элементы заменены некоторым $d > 0$, мы рассматриваем итерации двух методов как соответственно

$$x_i^{(k+1)} = \left(-b_i + \sum_{j \neq i} a_{ij} x_j^{(k)} \right) (-d); \tag{6}$$

$$x_i^{(k+1)} = \left(-b_i + \sum_{l=1}^{i-1} a_{ij} x_j^{(k+1)} + \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right) (-d). \tag{7}$$

Здесь d может быть произвольным положительным значением.

3. Комбинаторные варианты алгоритмов обхода графа

Цепью в графе называется конечная последовательность вершин, в которой каждая вершина соединена с последующей ребром. Цепь может пониматься и как набор этих рёбер. *Простые* цепи — это цепи без повторяющихся вершин. *Длина* цепи c, которую мы обозначаем как $\ell(c)$, — количество рёбер в ней. Цепь с совпадающими начальной и последней вершиной называется *циклом*. Мы называем цепь *правильной*, если

номера вершин в ней образуют возрастающую последовательность, то есть это цепь $c = \{i_0, i_1, \ldots, i_k\}$, $i_j \in V$, такая, что $(i_{j-1}, i_j) \in E$, $i_{j-1} < i_j$, $j = 1, \ldots, k$. Правильные цепи — это простые цепи. *Маршрут* — это такая цепь, в которой могут повторяться как вершины, так и рёбра.

Назовём *поиском правильных цепей* (Correct Chain Search, CCS) алгоритм обхода графа, получаемый в ходе итераций (7). Мы рассматриваем и BFS, и CCS как алгоритмы двух типов: комбинаторные (описание приведено в алгоритмах 1 и 2) и алгебраические (общая схема даётся в п. 4 алгоритмом 3).

Сходство BFS и CCS состоит в том, что, производя итерации (7), как и при выполнении итерации BFS, мы производим переходы по рёбрам, инцидентным вершинам, достигнутым на предыдущей итерации. Разница между ними в том, что если после этих переходов мы оказываемся в вершинах, из которых исходят правильные цепи, то в случае CCS на той же итерации будут произведены переходы по всем рёбрам, принадлежащим этим цепям. Это происходит потому, что, производя итерации (7), мы производим вычисление компонент текущего вектора состояния, используя значения компонент, которые уже рассчитаны ранее на текущей итерации. Так, если вершина j смежна вершине i, $j < i$, и j-я компонента вектора состояния перестала быть нулевой на текущей итерации, то на этой же итерации её значение будет использовано при вычислении i-й компоненты вектора состояния и она также перестанет быть нулевой, если она была таковой до этого.

При выполнении итераций BFS вектор состояния обновляется только исходя из результатов отдельных вычислений для каждой его компоненты. Это позволяет эффективно распараллеливать вычисления при умножении вектора на матрицу, но при этом в новый фронтир попадают только вершины, смежные вершинам фронтира, полученного на предыдущей итерации.

Пусть $\mathcal{N}$ обозначает множество вершин, смежных вершинам из $\mathcal{F}^{(k)}$ и не посещённых после k итераций алгоритма; $\mathcal{C}^{(k)}$ — множество вершин из V, посещённых после k итераций.

Алгоритм 1. Комбинаторный BFS

Вход: граф G, стартовая вершина $s \in V$.
Выход: компонента связности $\mathcal{C}$.

1: $k := 0$, $x^{(0)} := e_s$, $\mathcal{F}^{(0)} := \{s\}$, $\mathcal{C}^{(0)} := \{s\}$, $\mathcal{C}^{(1)} := \varnothing$.
2: **Пока** $\mathcal{C}^{(k)} \neq \mathcal{C}^{(k+1)}$:
3: $\quad \mathcal{N} := \big\{i \in V \setminus \mathcal{C}^{(k)} : \exists j \in \mathcal{F}^{(k)}\big((i,j) \in E\big)\big\}$;
4: $\quad \mathcal{F}^{(k+1)} := \mathcal{N}$;
5: $\quad \mathcal{C}^{(k+1)} := \mathcal{C}^{(k)} \cup \mathcal{F}^{(k+1)}$;
6: $\quad k := k + 1$.
7: $\mathcal{C} := \mathcal{C}^{(k+1)}$.

На итерации CCS в дополнение к переходам по рёбрам, инцидентным вершинам из $\mathcal{F}^{(k)}$, в вершины $\mathcal{F}^{(k+1)}$ производятся переходы по рёбрам правильных цепей, исходящим из уже достигнутых на этой итерации вершин $\mathcal{F}^{(k+1)}$. Пусть $C(\tilde{s}, i)$ — множество правильных цепей, исходящих из вершин $\tilde{s} \in \mathcal{N}$ и заканчивающихся в вершине i.

Таким образом, на шаге 3 обоих алгоритмов производятся переходы по рёбрам, инцидентным вершинам фронтира $\mathcal{F}^{(k)}$, полученного на предыдущей итерации. Дополнительные переходы по рёбрам правильных цепей из $C(\tilde{s}, i)$ выполняются на шаге 4 алгоритма 2.

Алгоритм 2. Комбинаторный CCS

Вход: граф G, стартовая вершина $s \in V$.
Выход: компонента связности $\mathcal{C}$.
1: $k := 0$, $x^{(0)} := e_s$, $\mathcal{F}^{(0)} := \{s\}$, $\mathcal{C}^{(0)} := \{s\}$, $\mathcal{C}^{(1)} := \varnothing$.
2: **Пока** $\mathcal{C}^{(k)} \neq \mathcal{C}^{(k+1)}$:
3: $\quad \mathcal{N} := \big\{i \in V \setminus \mathcal{C}^{(k)} : \exists j \in \mathcal{F}^{(k)}\big((i,j) \in E\big)\big\}$;
4: $\quad \mathcal{F}^{(k+1)} := \mathcal{N} \cup \big\{i \in V \setminus (\mathcal{C}^{(k)} \cup \mathcal{N}) : \exists \tilde{s} \in \mathcal{N}\big(C(\tilde{s}, i) \neq \varnothing\big)\big\}$;
5: $\quad \mathcal{C}^{(k+1)} := \mathcal{C}^{(k)} \cup \mathcal{F}^{(k+1)}$;
6: $\quad k := k + 1$.
7: $\mathcal{C} := \mathcal{C}^{(k+1)}$.

4. Алгебраические версии комбинаторных BFS и CCS, реализуемые как итерации методов Якоби и Гаусса — Зейделя

Обозначим через $\mathbf{F}\big(x^{(k)}\big)$ преобразование вида (6) или (7). Для заданной стартовой вершины s алгоритм 3 даёт компоненту связности $\mathcal{C}$, которой принадлежит s, производя обход этой компоненты связности. В случае, когда $\mathbf{F}$ — преобразование вида (6), этот алгоритм, как показано далее, является вариантом алгебраического BFS; при использовании $\mathbf{F}$ вида (7) получаем алгебраический CCS.

Алгоритм 3. Обход компоненты связности

Вход: граф G, стартовая вершина $s \in V$.
Выход: компонента связности $\mathcal{C}$.
1: $k := 0$, $x^{(0)} := e_s$, $\mathcal{F}^{(0)} := \{s\}$, $\mathcal{C}^{(0)} := \{s\}$, $\mathcal{C}^{(1)} := \varnothing$.
2: **Пока** $\mathcal{C}^{(k)} \neq \mathcal{C}^{(k+1)}$:
3: $\quad x^{(k+1)} := \mathbf{F}\big(x^{(k)}\big)$;
4: $\quad \mathcal{F}^{(k+1)} := \big\{i \in V(G) : (x_i^{(k)} = 0) \wedge (x_i^{(k+1)} \neq 0)\big\}$;
5: $\quad \mathcal{C}^{(k+1)} := \mathcal{C}^{(k)} \cup \mathcal{F}^{(k+1)}$;
6: $\quad k := k + 1$.
7: $\mathcal{C} := \mathcal{C}^{(k)}$.

Алгоритм 4 находит все компонент связности графа G. Компоненты связности обозначаются $\mathcal{C}_i$, $i = 1, \ldots, K$, где K — их количество.

Алгоритм 4. Нахождение всех компонент связности

Вход: граф G.
Выход: все компоненты связности $\{\mathcal{C}_1, \ldots, \mathcal{C}_K\}$.
1: $V' := \varnothing$, $K := 1$.
2: **Пока** $V' \neq V$:
3: $\quad$ выбрать $s \in V \setminus V'$;
4: $\quad \mathcal{C}_K :=$ Обход компоненты связности (G, s);
5: $\quad V' := V' \cup \mathcal{C}_K$;
6: $\quad K := K + 1$.

Последовательно выбирая стартовые вершины для обходов на шаге 3 алгоритма 4, мы находим все компоненты связности графа.

Покажем, что итерации (6) дают алгоритм обхода графа, эквивалентный комбинаторному BFS, тогда как итерации (7) — комбинаторному CCS.

5. Итерации метода Якоби как реализация комбинаторного BFS

Пусть $C(s,i)$ обозначает множество простых цепей, соединяющих стартовую вершину s и вершину $i \in \mathcal{F}^{(k+1)}$. Если $C(s,i) \neq \varnothing$, то все они имеют длину k: $\ell(c) = k$ для всех $c \in C(s,i)$.

Лемма 1. Для итераций (6)

$$x_i^{(k+1)} = \sum_{c \in C(s,i)} (-1)^{\ell(c)+1} d^{\ell(c)+2},$$

в соответствии с чем для i-й компоненты вектора состояния имеем

$$\left(x_i^{(t)} = 0,\ t = 1, \ldots, k\right) \ \text{ и } \ \left(x_i^{(k+1)} \neq 0\right)$$

тогда и только тогда, когда $i \in \mathcal{F}^{(k+1)}$ для комбинаторного BFS.

Доказательство. Индукция по количеству k выполненных итераций.

Пусть $k = 1$. Чтобы получить значения $x_i^{(1)}$ из уравнений (6) для $(s,i) \in E$, в них подставляется значение $x_s^{(0)} = d$, в результате получаем $x_i^{(1)} = -d^2 \neq 0$. Если i-е уравнение в (6) не содержит $x_s^{(k)}$ в правой части, то есть если $a_{is} = 0$ и $(s,i) \notin E$, то $x_i^{(1)} = 0$. Таким образом, для первой итерации лемма 1 верна.

Предположим, что лемма 1 верна для k-й итерации, т. е. для $t < k$ и $i \in \mathcal{F}^{(k)}$ выполняется $x_i^{(t)} = 0$ и

$$x_i^{(k)} = \sum_{c \in C(s,i)} (-1)^{\ell(c)} d^{\ell(c)+1} \neq 0.$$

Покажем, что лемма 1 верна и для $(k+1)$-й итерации. Для $i \neq s$ имеем

$$x_i^{(k+1)} = \Big(\sum_{j=1}^{n} a_{ij} x_j^{(k)}\Big)(-d) = \left(\sum_{(i,j)\in E} \Big(\sum_{c \in C(s,j)} (-1)^{\ell(c)} d^{\ell(c)+1}\Big)\right)(-d) =$$

$$= \left(\sum_{(i,j)\in E} \Big(\sum_{c \in C(s,j)} (-1)^{\ell(c)+1} d^{\ell(c)+2}\Big)\right) = \sum_{c \in C(s,i)} (-1)^{\ell(c)+1} d^{\ell(c)+2} \neq 0.$$

Таким образом, условие (5) выполнено для $i \in \mathcal{F}^{(k+1)}$. Покажем, что $x_i^{(k+1)} = 0$, если $i \notin \mathcal{F}^{(k+1)}$ и вершина i не была посещена на предыдущих k итерациях комбинаторного BFS. Это значит, что среди вершин j, смежных i, нет таких, для которых в ходе итераций (6) были пройдены простые цепи длины k, соединяющие s и j в ходе предыдущих k итераций. Поскольку лемма 1 верна для k, для всех таких вершин j имеем $x_j^{(k)} = 0$. Поэтому

$$x_i^{(k+1)} = \Big(\sum_{i=1}^{n} a_{ij} x_j^{(k)}\Big)(-d) = \Big(\sum_{(i,j)\in E}^{n} x_j^{(k)}\Big)(-d) = 0 \cdot (-d) = 0.$$

Лемма 1 доказана. ■

Таким образом, по (5) и лемме 1, итерации (6) дают те же самые фронтиры $\mathcal{F}^{(k)}$, $k = 1, 2, \ldots$, что и итерации комбинаторного BFS. Это значит, что мы получаем тот же самый обход графа. Отсюда следует

Теорема 1. Для графа G и заданной стартовой вершины s комбинаторный BFS и итерации метода Якоби с правой частью $b = e_s$ и $x^{(0)} = d \cdot e_s$ дают один и тот же обход.

6. Итерации метода Гаусса — Зейделя как реализация комбинаторного CCS

6.1. Вычисление компонент вектора состояния с помощью обходов цепей, исходящих из вершин текущего фронтира

Рассматривая $(k+1)$-ю итерацию комбинаторного CCS и итерацию вида (7) с тем же номером, будем использовать следующие обозначения. В отличие от п. 5, через $C(\tilde{s}, i)$ обозначим множество простых цепей, соединяющих вершины $\tilde{s} \in \mathcal{F}^{(k)}$ с вершиной $i \in \mathcal{F}^{(k+1)}$. На $(k+1)$-й итерации комбинаторного CCS производятся переходы по рёбрам этих цепей. Пусть $c + (j, i)$ обозначает цепь, получаемую из c соединением её последней вершины j с вершиной i ребром $(j, i) \in E$.

Все вычисления на итерациях (6) и (7) могут быть представлены как действия, производимые в соответствии с обходами отдельных цепей (не обязательно простых), соединяющих стартовую вершину и вершину i, для которой вычисляется значение $x_i^{(k+1)}$. Будем рассматривать только простые цепи, поскольку этого достаточно для доказательств леммы и теоремы, сформулированных далее. В действительности при реализации вычислений вида (6) или (7) множество всех арифметических операций, выполняемых на одной итерации, состоит из вычислений, соответствующих всем маршрутам, соединяющим стартовую вершину с другими вершинами в графе. Мы иллюстрируем это далее примером на рис. 1, *а* и *б*.

Цепи, которые обходятся в ходе одной итерации комбинаторного CCS, то есть цепи $c = (i_0, i_1, \ldots, i_{\ell(c)}) \in C(\tilde{s}, i)$, где $i_0 = \tilde{s}$, могут быть двух типов. Цепь $c \in C(\tilde{s}, i)$ — цепь типа (I), если $i_0 < i_1$, и типа (II), если $i_0 > i_1$. Для цепей обоих типов, возможно, за исключением первого ребра (i_0, i_1), имеем $i_{j-1} < i_j$ для $j = 1, \ldots, \ell(c)$. Переход по ребру (i_0, i_1) производится на шаге 3 комбинаторного CCS, переходы по последующим рёбрам цепи — на шаге 4.

Для $\tilde{s}, i \in V$ пусть $\ell(\tilde{s}, i) = \max\{\ell(c) : c \in C(\tilde{s}, i)\}$ — максимальная длина простой цепи, исходящей из $\tilde{s} \in \mathcal{F}^{(k)}$, в результате переходов по рёбрам которой достигаеся вершина $i \in \mathcal{F}^{(k+1)}$ на $(k+1)$-й итерации.

Пусть $i_0 = \tilde{s}$; $i_{\ell(c)} = i$ — первая и последняя вершины в цепи $c = (i_0, \ldots, i_{\ell(c)})$; $v_{\tilde{s}}$ — значение, которое в ходе одной итерации передаётся по цепи c, соединяющей $\tilde{s}$ и вершину i, при подстановке $v_{\tilde{s}}$ в уравнения (7) в соответствии с номерами вершин из c. Вычисление этого значения производится с помощью алгоритма 5 (обхода цепи).

Алгоритм 5. Алгоритм обхода цепи

Вход: цепь c, $v_{\tilde{s}}$.
Выход: $x_{i,c}^{(k+1)}$.

1: $c' := \varnothing$; $\quad x_{i_0,c'}^{(k+1)} := v_{\tilde{s}}$.
2: **Для** $t = 1, \ldots, \ell(c)$:
3: $\quad c'' := c' + (i_{t-1}, i_t)$; $\quad x_{i_t,c''}^{(k+1)} := x_{i_{t-1},c'}^{(k+1)}(-d)$; $\quad c' := c''$.
4: **Вернуть** $x_{i_t,c'}^{(k+1)}$.

В результате передачи $v_{\tilde{s}}$ при обходе цепи c мы получаем вклад $x_{i,c}^{(k+1)}$ в значение $x_i^{(k+1)}$, который передаётся по этой цепи. Для того чтобы выполнялось $x_i^{(k+1)} \neq 0$, должны иметься цепи, по которым в вершину i передаётся ненулевое значение $v_{\tilde{s}}$.

Таким образом, для $\tilde{s} \in \mathcal{F}^{(k)}$ и $c \in C(\tilde{s}, i)$ мы определяем $x_{i,c}^{(k+1)}$ как вклад значения $v_{\tilde{s}}$, которое передаётся по цепи $c \in C(\tilde{s}, i)$ из $\tilde{s}$ в i:

$$x_{i,c}^{(k+1)} = v_{\tilde{s}}(-d)^{\ell(c)}, \tag{8}$$

где

$$v_{\tilde{s}} = \begin{cases} x_{\tilde{s}}^{(k+1)}, & \text{если } c \text{ — цепь типа (I)}, \\ x_{\tilde{s}}^{(k)}, & \text{если } c \text{ — цепь типа (II)}. \end{cases}$$

Все вычисления на итерации (7) могут быть представлены как вычисления вкладов отдельных цепей, которые производятся по алгоритму 5.

Для $\tilde{s} \in \mathcal{F}^{(k)}$ определим $x_{i(\tilde{s})}^{(k+1)}$ как

$$x_{i(\tilde{s})}^{(k+1)} = \sum_{c \in C(\tilde{s},i)} x_{i,c}^{(k+1)}. \tag{9}$$

Если $C(\tilde{s}, i) = \varnothing$, то $x_{i(\tilde{s})}^{(k+1)} = 0$, поскольку в этом случае нет цепей, через которые вклад $v_{\tilde{s}}$ может быть в ходе $(k+1)$-й итерации (7) передан в вершину i из вершины $\tilde{s}$. Значение $x_{i(\tilde{s})}^{(k+1)}$ равно сумме всех вкладов в значение $x_i^{(k+1)}$, которые передаются через все цепи из $C(\tilde{s}, i)$ в ходе $(k+1)$-й итерации (7).

Отметим, что при проведении итераций (7) вершины из $\mathcal{F}^{(k+1)}$ могут достигаться не только в результате обхода простых цепей. Обход цепи типа (II) может привести к тому, что правильная цепь, исходящая из вершины j, уже достигнутой на итерации (7), на той же итерации будет повторно проходить через вершину $\tilde{s}$, из которой j была достигнута. Значение, полученное при обходе таких циклов, будет добавлено к исходному значению $x_{\tilde{s}}^{(k)}$ и передано далее через все цепи из $C(\tilde{s}, i)$. То есть оно будет включено в значение $v_{\tilde{s}}$ из (8) при повторном посещении $\tilde{s}$. Это может происходить, если $\tilde{s} > j$.

На рис. 1 показана такая ситуация. В данном случае (7) имеет следующий вид:

$$\begin{cases} x_1^{(k+1)} = x_2^{(k)}(-d), \\ x_2^{(k+1)} = \left(-1 + x_1^{(k+1)} + x_3^{(k)}\right)(-d), \\ x_3^{(k+1)} = x_2^{(k+1)}(-d). \end{cases}$$

Стартовая вершина — вершина 2. На первой итерации $x_1^{(0)} = 0$, $x_2^{(0)} = d$, $x_3^{(0)} = 0$, поэтому

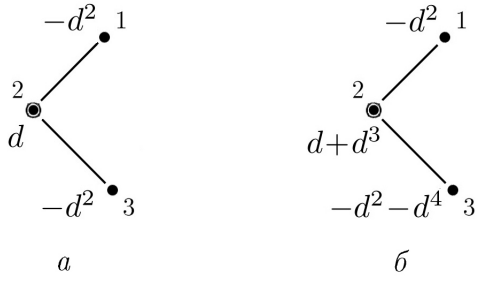
$$x_3^{(1)} = x_2^{(1)}(-d) = \left(-1 + x_1^{(1)}\right) d^2 = \left(-1 + x_2^{(0)}(-d)\right) d^2 = (-1 - d^2)d^2 = -d^2 - d^4.$$

Переданное по циклу $(2, 1, 2)$ значение $v_s = d$ вносит вклад $-d^4$ в значение $x_3^{(1)}$ при обходе всего маршрута $c_1 = (2, 1, 2, 3)$ на первой итерации. Суммируя его с вкладом, который передаётся по простой цепи $c_2 = (2, 3)$, состоящей из одного ребра, получаем итоговое значение $x_3^{(1)}$ на первой итерации как сумму вкладов, передаваемых из вершины 2 в вершину 3 по маршруту c_1 и по простой цепи c_2:

$$x_3^{(1)} = x_{3,c_1}^{(1)} + x_{3,c_2}^{(1)} = -d^4 - d^2.$$

Для графа на рис. 2 такие петли (циклы) отсутствуют в случае стартовой вершины 1. На рис. 2 также показано, как складываются вклады, передаваемые в вершину 5 по двум правильным цепям, которые обходятся в ходе одной итерации CSS. Это цепи $c_1 = (1, 2, 5)$ и $c_2 = (1, 3, 4, 5)$;

$$x_5^{(1)} = x_{5,c_1}^{(1)} + x_{5,c_2}^{(1)} = d^3 - d^4.$$



а *б*

Рис. 1. Получение значения $x_3^{(1)}$ для цепи на трёх вершинах в ходе одной итерации CCS

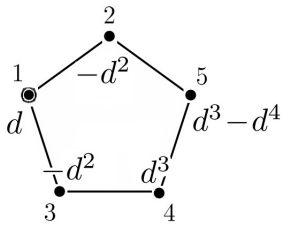


Рис. 2. Получение значения $x_5^{(1)}$ для цикла на пяти вершинах в ходе одной итерации CCS

6.2. Итерации метода Гаусса — Зейделя как реализация комбинаторного CCS

Для $i \neq s$ из (7) имеем

$$x_i^{(k+1)} = \Big(\sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} + \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\Big)(-d) = \sum_{\substack{j<i \\ (i,j)\in E}} x_j^{(k+1)}(-d) + \sum_{\substack{j>i \\ (i,j)\in E}} x_j^{(k)}(-d). \quad (10)$$

Доказываемая далее лемма 2 утверждает, что получение на $(k+1)$-й итерации (7) вектора $x^{(k+1)}$ эквивалентно его получению по простым цепям из $C(\tilde{s},i)$ (9) для всех $\tilde{s} \in \mathcal{F}^{(k)}$.

Отметим, что мы не рассматриваем ситуацию наличия циклов, начинающихся и заканчивающихся в вершине $\tilde{s}$, пример которой приведён выше. Поскольку распространяющийся далее после прохождения такой петли (цикла) вклад будет одинаково распространяться по всем исходящим из вершины правильным простым цепям после повторного прохождения вершины, это не поменяет равенства или неравенства нулю суммы вкладов, переданных по цепям в ходе итерации (7).

Лемма 2. Пусть $i \in \mathcal{F}^{(k+1)}$ для комбинаторного CCS и $x_i^{(k+1)}$ — значение, вычисляемое на $(k+1)$-й итерации (7). Тогда

$$x_i^{(k+1)} = \sum_{\tilde{s}\in\mathcal{F}^{(k)}} x_{i(\tilde{s})}^{(k+1)} = \sum_{\tilde{s}\in\mathcal{F}^{(k)}} \sum_{c\in C(\tilde{s},i)} x_{i,c}^{(k+1)} = \sum_{\tilde{s}\in\mathcal{F}^{(k)}} \sum_{c\in C(\tilde{s},i)} v_{\tilde{s}}(-d)^{\ell(c)},$$

где $v_{\tilde{s}} = x_{\tilde{s}}^{(k)}$ — передаваемое на $(k+1)$-й итерации из $\tilde{s}$ значение.

Доказательство. Индукция по количеству итераций комбинаторного CCS и итераций (7). Доказательство основания индукции и индуктивного предположения проведём индукцией по длине цепей из $C(\tilde{s},i)$ для $\tilde{s} \in \mathcal{F}^{(k)}$ и $i \in \mathcal{F}^{(k+1)}$.

Покажем, что лемма 2 верна для первой итерации. На этой итерации фронтир содержит только стартовую вершину s: $\mathcal{F}^{(0)} = \{s\}$. Пусть $i \in \mathcal{F}^{(1)}$ — такая вершина, что $\ell(s,i) = 1$. В этом случае $C(s,i)$ состоит из единственной цепи $c = (s,i)$ длины 1, содержащей только одно ребро $(s,i) \in E$. Таким образом, по (8) получаем $x_{i,c}^{(1)} = -d^2$, поскольку $v_s = d$ на первой итерации (7).

На первой итерации (7) имеем $x_j^{(0)} = 0$ для всех $j > i$, $(j,i) \in E$, $j \neq s$. Кроме того, $x_j^{(1)} = 0$ и для $j < i$, $(j,i) \in E$, $j \neq s$. Допустим, это не так и $x_j^{(1)} \neq 0$ для такого j. Значит, существует цепь типа (I) или типа (II), соединяющая вершины s и j, такая, что при подстановках на первой итерации значения $x_s^{(0)} = d$ в уравнения (7) с номерами, равными номерам вершин из этой цепи, мы получили бы $x_j^{(1)} \neq 0$. Но в этом случае $\ell(s,i) > 1$, поскольку $(j,i) \in E$, что противоречит предположению $\ell(s,i) = 1$.

Следовательно, все слагаемые в (10) нулевые, кроме $x_s^{(0)}$, если $i < s$, или кроме $x_s^{(1)}$, если $i > s$. Поэтому имеем

$$x_i^{(1)} = x_{i(s)}^{(1)} = x_{i,c}^{(1)} = -d^2,$$

где $c = (s,i)$. Таким образом, лемма 2 верна для вершины i, достигаемой на первой итерации (7) по цепям длины 1.

Предположим, что лемма 2 верна на первой итерации (7) для всех $i \in \mathcal{F}^{(1)}$, таких, что $\ell(s,i) = l$. Покажем, что она верна и для всех $i \in \mathcal{F}^{(1)}$, таких, что $\ell(s,i) = l+1$.

Поскольку на первой итерации $x_j^{(0)} = 0$ для $j > i$, $j \neq s$, то из (10) имеем

$$x_i^{(1)} = \sum_{\substack{j<i,\, j\neq s,\\ (i,j)\in E}} x_j^{(1)}(-d) + \alpha v_s(-d), \tag{11}$$

где $\alpha = 1$, если $(s,i) \in E$, $s > i$, и в этом случае $C(s,j) = \{(s,i)\}$. В случае $\alpha = 0$, $j \neq s$ выполняется $\ell(c) \leqslant \ell(s,j) \leqslant l$ для всех $c \in C(s,j)$, поскольку $\ell(s,i) = l+1$. Следовательно, по индукционному предположению для l имеем

$$x_j^{(1)} = \sum_{c' \in C(s,j)} x_{j,c'}^{(1)}. \tag{12}$$

Поскольку $x_{i,c}^{(1)} = x_{j,c'}^{(1)}(-d)$ для $c' \in C(s,j)$, такой, что $c = c' + (j,i)$, $c \in C(s,i)$, то, подставляя (12) в (11) и имея $v_s(-d) = x_{i,c}^{(1)}$ для $c = (s,i)$, получаем

$$x_i^{(1)} = \sum_{c \in C(s,i)} x_{i,c}^{(1)} = x_{i(s)}^{(1)}$$

для обоих возможных значений α.

Таким образом, для первой итерации лемма 2 верна для вершин $i \in \mathcal{F}^{(1)}$, достигаемых по цепям из $C(\tilde{s},i)$ любой длины.

Предположим, лемма 2 верна для всех итераций с номерами от 1 до k. Покажем, что она верна и для $(k+1)$-й итерации.

Пусть вершина $i \in \mathcal{F}^{(k+1)}$ такова, что $\ell(\tilde{s},i) = 1$ для вершин $\tilde{s} \in \mathcal{F}^{(k)}$. В этом случае все цепи $c \in C(\tilde{s},i)$ — это рёбра $(\tilde{s},i) \in E$, каждое из которых в (10) соответствует ненулевому слагаемому вида $x_{i,c}^{(k+1)} = x_{\tilde{s}}^{(k+1)}(-d)$, если $\tilde{s} < i$, или $x_{i,c}^{(k+1)} = x_{\tilde{s}}^{(k)}(-d)$, если $\tilde{s} > i$. Поэтому имеем

$$x_i^{(k+1)} = \sum_{\tilde{s} \in \mathcal{F}^{(k)}} \sum_{c \in C(\tilde{s},i)} x_{i,c}^{(k+1)} = \sum_{\tilde{s} \in \mathcal{F}^{(k)}} x_{i(\tilde{s})}^{(k+1)}.$$

Таким образом, лемма 2 верна для $i \in \mathcal{F}^{(k+1)}$, таких, что $\ell(\tilde{s}, i) = 1$ для $\tilde{s} \in \mathcal{F}^{(k)}$.

Предположим, что лемма 2 верна для $i \in \mathcal{F}^{(k+1)}$, таких, что $\ell(\tilde{s}, i) \leqslant l$, $\tilde{s} \in \mathcal{F}^{(k)}$. Покажем, что она верна для всех $i \in \mathcal{F}^{(k+1)}$, таких, что $\ell(\tilde{s}, i) = l + 1$ (индукция по l).

Пусть $\mathcal{F}_1^{(k)} = \{\tilde{s} \in \mathcal{F}^{(k)} : \tilde{s} < i\}$, $\mathcal{F}_2^{(k)} = \{\tilde{s} \in \mathcal{F}^{(k)} : \tilde{s} > i\}$, $\mathcal{F}_1^{(k)} \cup \mathcal{F}_2^{(k)} = \mathcal{F}^{(k)}$. Рассмотрим первое слагаемое S_1 в правой части (10):

$$S_1 = \sum_{\substack{j<i,\\(i,j)\in E}} x_j^{(k+1)}(-d).$$

Если $j < i$ и $x_j^{(k+1)} \neq 0$, то $C(\tilde{s}, j) \neq \varnothing$, поскольку по предположению индукции по k ненулевыми могут быть только те компоненты вектора состояния, которые соответствуют вершинам, достигнутым на итерациях до k-й включительно. Для получения ненулевого значения $x_j^{(k+1)}$ из правой части (10) требуются простые цепи, соединяющие такие вершины и j. Длина любой цепи из $C(\tilde{s}, j)$ меньше или равна l, поскольку иначе $\ell(\tilde{s}, i) > l + 1$. Поэтому по индукционному предположению по l имеем

$$S_1 = \sum_{\tilde{s}\in\mathcal{F}_1^{(k)}} \sum_{c\in C(\tilde{s},j)} x_{j,c}^{(k+1)}(-d).$$

Поскольку по (8) верно $x_{j,c}^{(k+1)}(-d) = x_{i,c'}^{(k+1)}$, где $c' = c + (j, i)$, с учётом (9) получаем

$$S_1 = \sum_{\tilde{s}\in\mathcal{F}_1^{(k)}} \sum_{c\in C(\tilde{s},i)} x_{i,c}^{(k+1)} = \sum_{\tilde{s}\in\mathcal{F}_1^{(k)}} x_{i(\tilde{s})}^{(k+1)}. \tag{13}$$

Рассмотрим второе слагаемое S_2 из правой части (10):

$$S_2 = \sum_{\substack{j>i,\\(i,j)\in E}} x_j^{(k)}(-d).$$

По индукционному предположению по k для $j > i$ имеем $x_j^{(k)} \neq 0$, если $j \in \mathcal{F}_2^{(k)}$. Поэтому

$$S_2 = \sum_{\tilde{s}\in\mathcal{F}_2^{(k)}} x_{i(\tilde{s})}^{(k+1)}, \tag{14}$$

поскольку $x_{\tilde{s}}^{(k)}(-d) = x_{i,c}^{(k+1)} = x_{i(\tilde{s})}^{(k+1)}$ для цепей $c = (\tilde{s}, i)$, $\tilde{s} \in \mathcal{F}_2^{(k)}$, $\ell(c) = 1$.

Складывая S_1 и S_2, из (13) и (14) получаем

$$x_i^{(k+1)} = S_1 + S_2 = \sum_{\tilde{s}\in\mathcal{F}_1^{(k)}} x_{i(\tilde{s})}^{(k+1)} + \sum_{\tilde{s}\in\mathcal{F}_2^{(k)}} x_{i(\tilde{s})}^{(k+1)} = \sum_{\tilde{s}\in\mathcal{F}^{(k)}} x_{i(\tilde{s})}^{(k+1)}.$$

Значит, лемма 2 верна для всех $i \in \mathcal{F}^{(k+1)}$, таких, что $\ell(\tilde{s}, i) = l + 1$. Таким образом, по индукции по l утверждение леммы 2 верно для всех вершин из $\mathcal{F}^{(k+1)}$, т. е. лемма верна для $(k + 1)$-й итерации, и по индукции по k она верна для всех k ($k \leqslant n$). ■

Теорема 2. Для графа G и стартовой вершины s комбинаторный CCS и итерации метода Гаусса — Зейделя с правой частью $b = e_s$ и $x^{(0)} = d \cdot e_s$ дают один и тот же обход.

Доказательство. Необходимо показать, что в ходе выполнения итераций (7) условие (5) для вершины $i \in V$ выполнено тогда и только тогда, когда $i \in \mathcal{F}^{(k+1)}$ для комбинаторного CCS.

По лемме 2

$$x_i^{(k+1)} = \sum_{\tilde{s}\in\mathcal{F}^{(k)}} \sum_{c\in C(\tilde{s},i)} v_{\tilde{s}}(-d)^{\ell(c)},$$

где $v_{\tilde{s}} = O(d^{\ell(s,\tilde{s})}) \neq 0$ и $\ell(s,\tilde{s})$ — максимальная длина простой цепи, соединяющей стартовую вершину s и вершину $\tilde{s} \in \mathcal{F}^{(k)}$. По (8), при переходе по каждому ребру такой цепи значение $v_{\tilde{s}}$, которое является суммой степеней d с некоторыми коэффициентами перед ними, умножается на $-d$. В этом случае одни и те же степени d в итоговом значении $x_{i,c}^{(k+1)}$ имеют один и тот же знак, поскольку они получаются в результате одного и того же количества умножений на $-d$, производимых в соответствии с алгоритмом 5 для цепей одной длины. Поэтому если $C(\tilde{s},i) \neq \varnothing$, то $x_i^{(k+1)} \neq 0$ и условие (5) выполнено для $i \in \mathcal{F}^{(k+1)}$ комбинаторного CCS после $(k+1)$-й итерации.

Пусть $i \in V \setminus \mathcal{C}^{(k+1)}$ для комбинаторного CCS, т. е. вершина i не достигается после $(k+1)$-й итерации. Это значит, что $C(\tilde{s},i) = \varnothing$ для всех $\tilde{s} \in \mathcal{F}^{(k)}$. Поскольку $x_{i(\tilde{s})}^{(k+1)} = 0$, если $C(\tilde{s},i) = \varnothing$, то по лемме 2 из этого следует, что

$$x_i^{(k+1)} = \sum_{\tilde{s}\in\mathcal{F}^{(k)}} x_{i(\tilde{s})}^{(k+1)} = 0,$$

т. е. условие (5) не выполнено для $i \in V \setminus \mathcal{C}^{(k+1)}$.

Таким образом, каждая итерация комбинаторного CCS даёт те же фронтиры $\mathcal{F}^{(k)}$, $k = 1, 2, \ldots$, что и итерации (7). Значит, мы получим те же самые обходы графа при их выполнении. ■

7. Примеры обходов графов, реализуемых итерациями методов Якоби и Гаусса — Зейделя

В рассматриваемых примерах стартовая вершина — это вершина 1; $d = 2$.

Сравним обходы графов, реализуемые BFS и CCS, для графа G_1 (рис. 3). Для этого графа преобразование $\mathbf{F}$ векторов состояния $x^{(k)}$ вида (6) задаётся следующими уравнениями:

$$\begin{cases} x_1^{(k+1)} = \left(-1 + x_2^{(k)}\right)(-d), \\ x_2^{(k+1)} = \left(x_1^{(k)} + x_3^{(k)} + x_6^{(k)}\right)(-d), \\ x_3^{(k+1)} = \left(x_2^{(k)} + x_4^{(k)} + x_7^{(k)}\right)(-d), \\ x_4^{(k+1)} = \left(x_3^{(k)}\right)(-d), \\ x_5^{(k+1)} = \left(x_6^{(k)}\right)(-d), \\ x_6^{(k+1)} = \left(x_2^{(k)} + x_5^{(k)} + x_7^{(k)}\right)(-d), \\ x_7^{(k+1)} = \left(x_3^{(k)} + x_6^{(k)} + x_8^{(k)}\right)(-d), \\ x_8^{(k+1)} = \left(x_7^{(k)}\right)(-d). \end{cases} \tag{15}$$

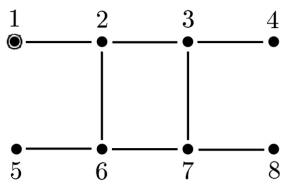


Рис. 3. Граф G_1

Обход, который эти итерации дадут для G_1, представлен на рис. 4. Требуется четыре итерации для того, чтобы завершить обход графа, используя алгебраический BFS, реализуемый итерациями метода Якоби (6).

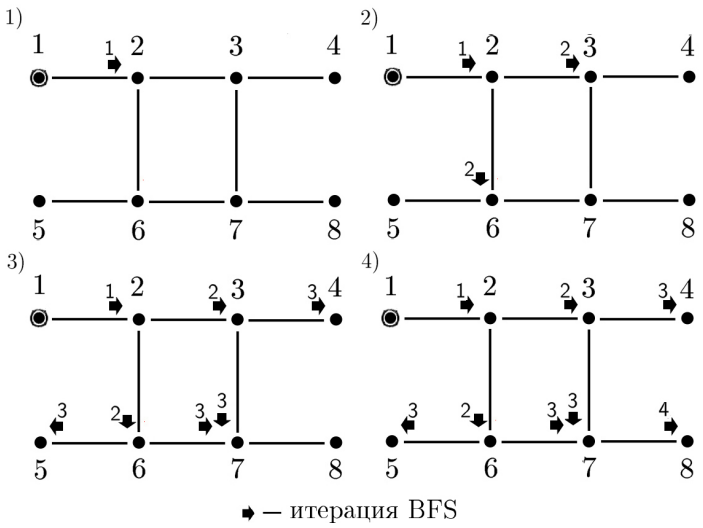


Рис. 4. Обход графа G_1, реализуемый BFS

Для итераций алгебраического CCS преобразование $\mathbf{F}$ векторов состояния $x^{(k)}$ задаётся следующими уравнениями:

$$\begin{cases} x_1^{(k+1)} = \left(-1 + x_2^{(k)}\right)(-d), \\ x_2^{(k+1)} = \left(x_1^{(k+1)} + x_3^{(k)} + x_6^{(k)}\right)(-d), \\ x_3^{(k+1)} = \left(x_2^{(k+1)} + x_4^{(k)} + x_7^{(k)}\right)(-d), \\ x_4^{(k+1)} = \left(x_3^{(k+1)}\right)(-d), \\ x_5^{(k+1)} = \left(x_6^{(k)}\right)(-d), \\ x_6^{(k+1)} = \left(x_2^{(k+1)} + x_5^{(k+1)} + x_7^{(k)}\right)(-d), \\ x_7^{(k+1)} = \left(x_3^{(k+1)} + x_6^{(k+1)} + x_8^{(k)}\right)(-d), \\ x_8^{(k+1)} = \left(x_7^{(k+1)}\right)(-d). \end{cases} \tag{16}$$

Производя итерации (16), получаем обход графа за две итерации. Как можно видеть (рис. 5), в отличие от BFS, на первой итерации CCS производятся переходы по рёбрам правильных цепей, исходящих из вершины 2, которая достигается из вершины 1 на той же итерации.

Векторы состояния $x^{(k)}$ и множества $\mathcal{C}^{(k)}$ вершин, достигаемых после k-й итерации BFS и CCS, представлены далее.

BFS (реализумый итерациями (15))

Инициализация: $x^{(0)} = (2, 0, 0, 0, 0, 0, 0, 0)$, $\mathcal{F}^{(0)} = \{1\}$, $\mathcal{C}^{(0)} = \{1\}$;
$x^{(1)} = (2, -4, 0, 0, 0, 0, 0, 0)$, $\mathcal{F}^{(1)} = \{2\}$, $\mathcal{C}^{(1)} = \{1, 2\}$;
$x^{(2)} = (10, -4, 8, 0, 0, 8, 0, 0)$, $\mathcal{F}^{(2)} = \{3, 6\}$, $\mathcal{C}^{(2)} = \{1, 2, 3, 6\}$;
$x^{(3)} = (10, -52, 8, -16, -16, 8, -32, 0)$, $\mathcal{F}^{(3)} = \{5, 7\}$, $\mathcal{C}^{(3)} = \{1, 2, 3, 4, 5, 6, 7\}$;
$x^{(4)} = (106, -52, 200, -16, -16, 200, -32, 64)$, $\mathcal{F}^{(4)} = \{8\}$, $\mathcal{C}^{(4)} = \{1, 2, 3, 4, 5, 6, 7, 8\}$.

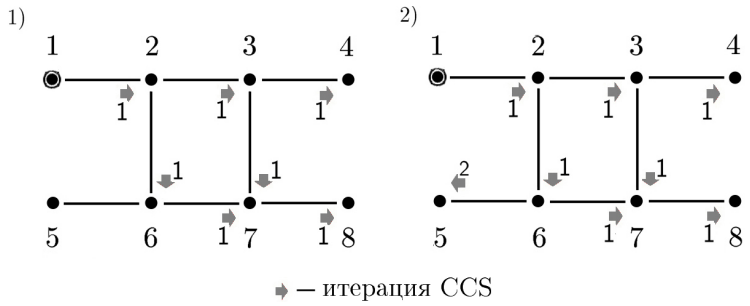


Рис. 5. Обход графа G_1, реализуемый CCS

CCS (реализуемый итерациями (16))

Инициализация: $x^{(0)} = (2,0,0,0,0,0,0,0)$, $\mathcal{F}^{(0)} = \{1\}$, $\mathcal{C}^{(0)} = \{1\}$;
$x^{(1)} = (2,-4,8,-16,0,8,-32,64)$, $\mathcal{F}^{(1)} = \{2,3,4,6,7,8\}$, $\mathcal{C}^{(1)} = \{1,2,3,4,6,7,8\}$;
$x^{(2)}$=$(10,-52,200,-400,-16,200,-928,1856)$, $\mathcal{F}^{(2)}$=$\{5\}$, $\mathcal{C}^{(2)}$=$\{1,2,3,4,5,6,7,8\}$.

Для графа G_2 (рис. 6) преобразование $\mathbf{F}$ вида (6) вектора состояния $x^{(k)}$ задаётся уравнениями

$$\begin{cases} x_1^{(k+1)} = \left(-1 + x_2^{(k)}\right)(-d), \\ x_2^{(k+1)} = \left(x_1^{(k)} + x_3^{(k)}\right)(-d), \\ x_3^{(k+1)} = \left(x_2^{(k)} + x_4^{(k)}\right)(-d), \\ x_4^{(k+1)} = \left(x_3^{(k)} + x_5^{(k)}\right)(-d), \\ x_5^{(k+1)} = \left(x_4^{(k)}\right)(-d). \end{cases} \tag{17}$$

Преобразование $\mathbf{F}$, которое производится CCS для этого графа, имеет вид

$$\begin{cases} x_1^{(k+1)} = \left(-1 + x_2^{(k)}\right)(-d), \\ x_2^{(k+1)} = \left(x_1^{(k+1)} + x_3^{(k)}\right)(-d), \\ x_3^{(k+1)} = \left(x_2^{(k+1)} + x_4^{(k)}\right)(-d), \\ x_4^{(k+1)} = \left(x_3^{(k+1)} + x_5^{(k)}\right)(-d), \\ x_5^{(k+1)} = \left(x_4^{(k+1)}\right)(-d). \end{cases} \tag{18}$$

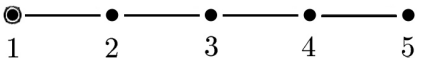


Рис. 6. Граф G_2 — правильная цепь

В графе G_2 имеется правильная цепь, исходящая из вершины 2, достигаемой на первой итерации, которой принадлежат все вершины графа, за исключением стартовой. Поэтому все вершины графа будут посещены в ходе одной итерации CCS (рис. 7). BFS для этого потребуется четыре итерации, что составляет максимально возможное количество итераций для всех возможных вариантов стартовой вершины.

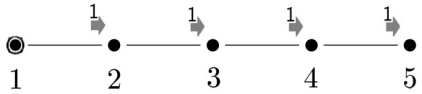


Рис. 7. Обход графа G_2, реализуемый CCS

BFS (реализуемый итерациями (17))

Инициализация: $x^{(0)} = (2, 0, 0, 0, 0)$, $\mathcal{F}^{(0)} = \{1\}$, $\mathcal{C}^{(0)} = \{1\}$;
$x^{(1)} = (2, -4, 0, 0, 0)$, $\mathcal{F}^{(1)} = \{2\}$, $\mathcal{C}^{(1)} = \{1, 2\}$;
$x^{(2)} = (10, -4, 8, 0, 0)$, $\mathcal{F}^{(2)} = \{3\}$, $\mathcal{C}^{(2)} = \{1, 2, 3\}$;
$x^{(3)} = (10, -36, 8, -16, 0)$, $\mathcal{F}^{(3)} = \{4\}$, $\mathcal{C}^{(3)} = \{1, 2, 3, 4\}$;
$x^{(4)} = (74, -36, 104, -16, 32)$, $\mathcal{F}^{(3)} = \{5\}$, $\mathcal{C}^{(4)} = \{1, 2, 3, 4, 5\}$.

CCS (реализуемый итерациями (18))

Инициализация: $x^{(0)} = (2, 0, 0, 0, 0)$, $\mathcal{F}^{(0)} = \{1\}$, $\mathcal{C}^{(0)} = \{1\}$;
$x^{(1)} = (2, -4, 8, -16, 32)$, $\mathcal{F}^{(1)} = \{2, 3, 4, 5\}$, $\mathcal{C}^{(1)} = \{1, 2, 3, 4, 5\}$.

При другой нумерации вершин (рис. 8) на каждой итерации CCS нет правильных цепей, исходящих из достигнутых на этих итерациях вершин. Поэтому для того, чтобы посетить все вершины графа, и BFS, и CSS требуется произвести четыре итерации; обход, который даёт CCS, полностью повторяет обход, который даёт BFS (рис. 9).

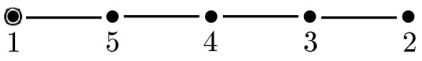


Рис. 8. Неправильная цепь

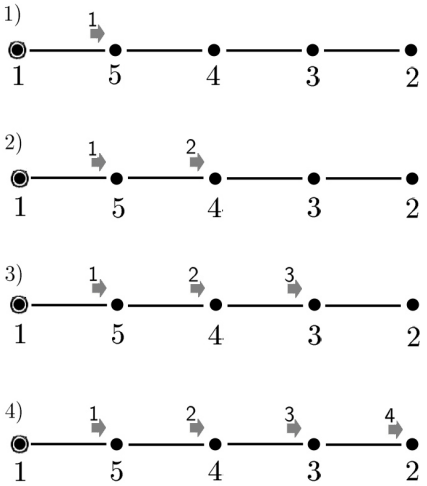


Рис. 9. Обход графа, реализуемый CCS

8. Беззнаковый CCS

В качестве преобразования $\mathbf{F}$ вектора состояния в алгоритме 3 обхода компоненты связности может быть использовано следующее преобразование:

$$x_i^{(k+1)} = \left(b_i + \sum_{l=1}^{i-1} a_{ij} x_j^{(k+1)} + \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right) d. \tag{19}$$

Назовём алгоритм 3, который использует (19) в качестве преобразования $\mathbf{F}$, *беззнаковым CCS*.

В ходе выполнения беззнакового CCS, который даёт тот же самый обход графа, что и CCS, вычисления производятся с использованием только неотрицательных целых чисел, если $d \in \mathbb{Z}$. Отсутствие операций деления и, возможно, умножения в случае $d = 1$ при численной реализации позволяет значительно уменьшить время, требуемое для обхода графа.

Преобразование (19) представляет собой модификацию преобразования $\mathbf{F}$, используемого CSS. Вместо преобразования $x^{(k+1)} = (b - Lx^{(k+1)} - Ux^{(k)})\, d$ используется преобразование $x^{(k+1)} = (b + Lx^{(k+1)} + Ux^{(k)})\, d$.

Если не учитывать умножения на d, итерация беззнакового CCS может быть рассмотрена как итерация BFS, в которой каждая компонента вектора состояния вычисляется с использованием компонент с меньшими индексами, уже вычисленными на текущей итерации. В то же время умножение (деление) на d и операция обращения по сложению может быть принципиальной для применения алгоритмов обхода графа при решении задач дискретного анализа и дискретной оптимизации. Например, реализация CCS для возмущённых матриц графов с нецелочисленными элементами может быть использована в подходе к решению задачи проверки изомофизма графов [11, 12].

9. Регуляризация вектора состояния и маскировка вершин

Использование умножения вместо деления в преобразовании (7) позволяет уменьшить время, требуемое для проведения обхода графа. Для того чтобы предотвратить как переполнение значений компонент, представляемых числами с ограниченной длиной мантиссы, так и их зануление в случае $d \in (0, 1)$, после заданного количества итераций M вектор состояния может быть регуляризован:

$$x_i^{(k)} \to \frac{1}{d^M} x_i^{(k)}.$$

Кроме того, без изменения порядка вычислительной сложности CCS существенное ускорение может быть получено при использовании маскировки вершин. *Маскировка вершины* означает исключение её из дальнейших вычислений. Маскируемые вершины — это вершины, из которых не могут быть достигнуты вершины, не достигнутые на предыдущих итерациях. Маскироваться могут как вершины, принадлежащие уже полученным компонентам связности, так и уже посещённые вершины компоненты связности, обход которой производится на текущей итерации алгоритма 4. Во втором случае при вычислениях на $(k + 1)$-й итерации (7) маскируются вершины, принадлежащие фронтирам, полученным на предыдущих $(k - 1)$ итерациях алгоритма 3.

Для экономии памяти и ускорения вычисления должны проводиться с *портретом* матрицы смежности. Портрет матрицы смежности содержит только ненулевые элементы матрицы, то есть он представляет собой список рёбер графа. При использовании портрета матрицы смежности одна итерация вида (6) или (7) имеет вычислительную сложность $O(m)$, поскольку для одной такой итерации требуется один проход по списку рёбер.

10. Обход графа и нумерация его вершин

Наличие правильных цепей в графе определяется нумерацией его вершин. Нумерация вершин и выбор стартовой вершины определяют вычислительную сложность проведения обхода графа с помощью CCS. Нумерация вершин графа — это обязательный параметр индивидуальной задачи обхода графа.

Предположим, что информация о графе, для которого необходимо провести обход, регулярно обновляется добавлением или удалением его вершин и рёбер. Оптимизация нумерации вершин может быть произведена, исходя из расстояния от них до вершины, которая выбирается как стартовая. При оптимальной нумерации вершин, когда все цепи, исходящие из стартовой вершины, правильные, CCS будет иметь минимально возможную для алгоритма обхода графа сложность $O(m)$.

Отметим, что назначение номеров вершин в соответствии с удалённостью представляемых ими объектов от некоторого заданного объекта часто является естественным подходом. Это имеет место, например, для транспортных сетей. Случайность в нумерации вершин неизбежна в таких приложениях, но она может быть минимизирована. Граф, в котором все вершины пронумерованы случайно, может быть интерпретирован как граф транспортной сети, хранящийся в базе данных со случайно пронумерованными записями, что может встретиться в таких приложениях только гипотетически при условии, что данные вводятся в течение долгого времени по мере развития сети. И даже при совершенно случайной нумерации вершин графа в большинстве случаев в таком графе присутствуют правильные цепи.

11. Количество итераций BFS и CCS, требуемых для обхода связного графа

Сравним количество итераций BFS и CCS, требуемых для обхода графа. Если не принимать во внимание такие детали реализации алгоритмов, как распараллеливание вычислений и маскировка вершин, то вычислительная сложность обхода графа определяется количеством итераций, требуемых для посещения всех его вершин.

Пусть N_{BFS} — количество итераций, требуемых для того, чтобы в ходе BFS посетить все вершины графа, и N_{CCS} — количество итераций, требуемых CCS для обхода того же графа из той же стартовой вершины. Значения N_{BFS} и N_{CCS} определяются выбором стартовой вершины, но, кроме того, N_{CCS} определяется также нумерацией вершин графа.

Поскольку и BFS, и CCS производят переходы через рёбра, инцидентные вершинам фронтиров, можно утверждать, что для любого графа $N_{\mathrm{CCS}} \leqslant N_{\mathrm{BFS}}$. Вместе с тем если в ходе выполнения CCS достигается хотя бы одна вершина, из которой исходит хотя бы одна правильная цепь, то достижимость вершин графа из стартовой вершины определяется CCS за меньшее количество итераций, чем требуется для этого BFS, то есть в этом случае $N_{\mathrm{CCS}} < N_{\mathrm{BFS}}$. Поэтому верно следующее утверждение.

Утверждение 1. Для одной и той же стартовой вершины CCS требует не больше итераций, чем BFS.

12. Вычислительный эксперимент

Цель эксперимента — оценить влияние диаметра графа на разницу в количестве проводимых в среднем итераций CCS и BFS. Для этого мы рассчитывали общее количество итераций, потребовавшееся для обходов случайных графов, и находили отношение этих значений для CCS и BFS для одних и тех же стартовых вершин. Случайные графы — это графы, в которых при их генерации:

1) случайно задаётся нумерация вершин;
2) случайно выбранные несмежные вершины соединяются ребром.

Как *расширенную звезду* определим граф, в котором одна вершина является общей для некоторого количества исходящих простых цепей. Назовём эти цепи *лучами* расширенной звезды. Чтобы оценить влияние диаметра на разницу в количестве про-

водимых в среднем итераций CCS и BFS, все графы генерировались на основе графов двух типов:

(I) расширенных звёзд с лучами разной (случайной) длины;
(II) расширенных звёзд с лучами одинаковой длины.

К сгенерированному графу (расширенной звезде со случайной нумерацией вершин) добавлялись рёбра, соединяющие несмежные до этого вершины. Пусть $E(G)$ — множество рёбер графа G. Алгоритм генерирации случайного графа следующий:

1) Генерируем на множестве вершин $V = \{1, \ldots, n\}$ расширенную звезду $G^{(0)}$ типа (I) или типа (II).
2) Добавляем случайные рёбра к $G^{(0)}$:

$$E(G^{(i+1)}) := E(G^{(i)}) \cup \{e_{i+1}\},$$

где $e_{i+1} \notin E(G^{(i)})$, $i = 1, \ldots, \tilde{m} - 1$; $\tilde{m}$ — количество добавляемых к $G^{(0)}$ случайных рёбер. Количество рёбер в сгенерированном графе $G = G^{(\tilde{m})}$ составляет

$$m = |E(G^{(\tilde{m})})| = |E(G^{(0)})| + \tilde{m} = n - 1 + \tilde{m}.$$

Параметры графов, используемых в эксперименте, следующие: n — количество вершин расширенной звёзды, на основе которой генерируется случайный граф; $\tilde{m}$ — количество добавляемых случайных рёбер, характеризующее плотность графа. Для графов, генерируемых на основе расширенных звёзд с лучами одинаковой длины, параметры следующие: ℓ — длина луча расширенной звёзды; r — количество лучей.

Заметим, что чем плотнее граф, то есть чем больше значение $\tilde{m}$, тем, как правило, меньше диаметр сгенерированного графа. При добавлении случайных рёбер уменьшение диаметра графа может происходить очень быстро, что можно оценить по резкому уменьшению количества итераций и CCS, и BFS при добавлении в расширенную звезду случайных рёбер, как показано далее в таблицах для значений $\tilde{m} = 0$ и $2n$. Кроме того, диаметр генерируемого графа типа (II), как правило, тем меньше, чем меньше длина ℓ луча расширенной звезды, из которой получен граф.

Пусть $N^{(i)}_{\mathrm{CCS}}$ и $N^{(i)}_{\mathrm{BFS}}$ — количества итераций, которые необходимо провести с помощью CCS и BFS соответственно для обхода i-го графа из набора, состоящего из M случайных графов с заданными параметрами. Пусть

$$\overline{N}_{\mathrm{CCS}} = \sum_{i=1}^{M} N^{(i)}_{\mathrm{CCS}}, \quad \overline{N}_{\mathrm{BFS}} = \sum_{i=1}^{M} N^{(i)}_{\mathrm{BFS}}.$$

В табл. 1–4 приведены результаты экспериментов, в ходе которых для вычисления $\overline{N}_{\mathrm{CCS}}/\overline{N}_{\mathrm{BFS}}$ с помощью CCS и BFS производились обходы $M = 10\,000$ случайно сгенерированных графов типов (I) и (II) с $n = 101$ и указанными параметрами.

Таблица 1
Разреженные графы типа (I), $n = 101$

$\tilde{m}$	$\overline{N}_{\mathrm{CCS}}/\overline{N}_{\mathrm{BFS}}$
0	$315\,990/619\,604 \approx 0{,}51$
$2n$	$44\,646/60\,148 \approx 0{,}56$
$5n$	$30\,288/52\,269 \approx 0{,}58$
$10n$	$25\,285/40\,431 \approx 0{,}63$

Таблица 2
Разреженные графы типа (II), $n = 101$, $\ell = 50$, $r = 2$

$\tilde{m}$	$\overline{N}_{\mathrm{CCS}}/\overline{N}_{\mathrm{BFS}}$
0	$381\,702/752\,844 \approx 0{,}51$
$2n$	$24\,563/41\,462 \approx 0{,}59$
$5n$	$20\,007/30\,059 \approx 0{,}67$
$10n$	$19\,925/23\,346 \approx 0{,}85$

Таблица 3
Разреженные графы типа (II),
$n = 101$, $\ell = 20$, $r = 5$

$\tilde{m}$	$\overline{N}_{\mathrm{CCS}}/\overline{N}_{\mathrm{BFS}}$
0	95 746/154 515 ≈ 0,62
$2n$	24 705/41 494 ≈ 0,59
$5n$	20 005/30 062 ≈ 0,67
$10n$	19 933/23 360 ≈ 0,85

Таблица 4
Разреженные графы типа (II),
$n = 101$, $\ell = 10$, $r = 10$

$\tilde{m}$	$\overline{N}_{\mathrm{CCS}}/\overline{N}_{\mathrm{BFS}}$
0	170 693/304 627 ≈ 0,56
$2n$	24 637/41 540 ≈ 0,59
$5n$	20 005/30 052 ≈ 0,67
$10n$	19 939/23 429 ≈ 0,85

Для плотных графов с количеством рёбер $n+n^2/4$, $n = 101$, что составляет 60 % от количества рёбер в полном графе, имеем $\overline{N}_{\mathrm{CCS}}/\overline{N}_{\mathrm{BFS}} = 16\,851/20\,000 \approx 0{,}84$. Усреднение получено по $M = 10\,000$ случайных графов.

Для вычисления каждого отношения $\overline{N}_{\mathrm{CCS}}/\overline{N}_{\mathrm{BFS}}$, представленного в табл. 5–8, с помощью CCS и BFS производились обходы $M = 1\,000$ случайно сгенерированных графов указанных типов с указанными параметрами.

Таблица 5
Разреженные графы типа (I),
$n = 1001$

$\tilde{m}$	$\overline{N}_{\mathrm{CCS}}/\overline{N}_{\mathrm{BFS}}$
0	310 023/618 989 ≈ 0,5
$2n$	3 140/6 006 ≈ 0,52
$5n$	2 102/4 046 ≈ 0,52
$10n$	2 000/3 189 ≈ 0,63

Таблица 6
Разреженные графы типа (II),
$n = 1001$, $\ell = 500$, $r = 2$

$\tilde{m}$	$\overline{N}_{\mathrm{CCS}}/\overline{N}_{\mathrm{BFS}}$
0	376 913/752 406 ≈ 0,5
$2n$	3 119/6 001 ≈ 0,52
$5n$	2 076/4 045 ≈ 0,51
$10n$	2 000/3 185 ≈ 0,63

Таблица 7
Разреженные графы типа (II),
$n = 1001$, $\ell = 200$, $r = 5$

$\tilde{m}$	$\overline{N}_{\mathrm{CCS}}/\overline{N}_{\mathrm{BFS}}$
0	154 000/298 341 ≈ 0,52
$2n$	3 114/6 011 ≈ 0,52
$5n$	2 079/4 041 ≈ 0,51
$10n$	2 000/3 220 ≈ 0,62

Таблица 8
Разреженные графы типа (II),
$n = 1001$, $\ell = 100$, $r = 10$

$\tilde{m}$	$\overline{N}_{\mathrm{CCS}}/\overline{N}_{\mathrm{BFS}}$
0	79 238/148 825 ≈ 0,53
$2n$	3 122/5 993 ≈ 0,52
$5n$	2 089/4 063 ≈ 0,51
$10n$	2 000/3 202 ≈ 0,62

Результаты вычислительного эксперимента показывают, что чем больше диаметр графа, тем больше в среднем разница в количестве итераций CCS и BFS, требуемых для обхода графа. Для разреженных графов эта разница в среднем составляет от 15 до 49 % процентов от количества итераций, требуемых для обхода BFS, при $n = 101$ (табл. 1–4) и от 37 до 50 % процентов при $n = 1001$ (табл. 5–8). Для плотных графов при $n = 101$ разница составляет в среднем 16 %.

Выводы

Рассмотрены методы простой итерации решения систем линейных алгебраических уравнений с модифицированными матрицами смежности графов и заданной правой частью как реализации обходов графа. Такой подход даёт два варианта алгоритмов обхода графа. Один из них реализуется итерациями метода Якоби, второй — итерациями метода Гаусса — Зейделя. Обход, реализуемый в ходе проведения итераций метода

Якоби, эквивалентен поиску в ширину, тогда как обход, реализуемый с помощью метода Гаусса — Зейделя, не эквивалентен ни поиску в ширину, ни поиску в глубину. Для любой индивидуальной задачи нахождения компонент связности графа количество итераций, требуемых для такого алгоритма, не превышает количества итераций, требуемых для поиска в ширину для одной и той же стартовой вершины в графе. Для многих индивидуальных задач нахождения компонент связности графа алгоритм обхода, ассоциированный с итерациями метода Гаусса — Зейделя, требует меньшего количества итераций.

ЛИТЕРАТУРА

1. *Cormen T. H., Leiserson C. E., Rivest R. L., and Stein C.* Introduction to Algorithms (3rd ed). MIT Press, 2009.
2. *Zuse K.* Der Plankalkül. Konrad Zuse Internet Archive. 1972. S. 96–105.
3. *Moore E. F.* The shortest path through a maze // Proc. Intern. Symp. Theory of Switching. P. II. Cambridge, MA: Harvard University Press, 1959. P. 285–292.
4. *Lee C. Y.* An algorithm for path connections and its applications // IRE Trans. Electronic Comput. 1961. V. EC-10. No. 3. P. 346–365.
5. *Beamer S., Asanović K., and Patterson D.* Direction-optimized breadth-first search // Proc. SC'12. Salt Lake City, Utah, 2012. Article 12. P. 1–10.
6. *Bücker H. M. and Sohr C.* Reformulating a breadth-first search algorithm on an undirected graph in the language of linear algebra // Proc. MCSI'14. IEEE Computer Society, 2014. P. 33–35.
7. *Azad A. and Buluç A.* A work-efficient parallel sparse matrix-sparse vector multiplication algorithm // Proc. IPDPS'17. Orlando, Florida, USA, 2017. P. 688–697.
8. *Yang C., Buluç A., and Owens J. D.* Graphblast: A high-performance linear algebra-based graph framework on the GPU // ACM Trans. Math. Software. 2022. V. 48 (1). P. 1–51.
9. *Yang C., Wang Y., and Owens J. D.* Fast sparse matrix and sparse vector multiplication algorithm on the GPU // Proc. IPDPSW'15. IEEE Computer Society, 2015. P. 841–847.
10. *Burkhardt P.* Optimal algebraic Breadth-First Search for sparse graphs // ACM Trans. Knowl. Discov. Data. 2021. V. 15 (5). Article 77.
11. *Пролубников А. В.* Сведение задачи проверки изоморфизма графов к задаче проверки равенства полиномов от n переменных // Тр. ИММ УрО РАН. 2016. Т. 22. № 1. С. 235–240.
12. *Пролубников А. В.* Точность и сложность вычислений, необходимые для проверки изоморфизма графов сравнением полиномов // Вычислительные технологии. 2016. Т. 21. № 6. С. 71–88.

REFERENCES

1. *Cormen T. H., Leiserson C. E., Rivest R. L., and Stein C.* Introduction to Algorithms (3rd ed). MIT Press, 2009.
2. *Zuse K.* Der Plankalkül. Konrad Zuse Internet Archive, 1972, S. 96–105. (in German)
3. *Moore E. F.* The shortest path through a maze. Proc. Intern. Symp. Theory of Switching. P. II, Cambridge, MA, Harvard University Press, 1959, pp. 285–292.
4. *Lee C. Y.* An algorithm for path connections and its applications. IRE Trans. Electronic Comput., 1961, vol. EC-10, no. 3, pp. 346–365.
5. *Beamer S., Asanović K., and Patterson D.* Direction-optimized breadth-first search. Proc. SC'12, Salt Lake City, Utah, 2012, article 12, pp. 1–10.

6. *Bücker H. M. and Sohr C.* Reformulating a breadth-first search algorithm on an undirected graph in the language of linear algebra. Proc. MCSI'14, IEEE Computer Society, 2014, pp. 33–35.
7. *Azad A. and Buluç A.* A work-efficient parallel sparse matrix-sparse vector multiplication algorithm. Proc. IPDPS'17, Orlando, Florida, USA, 2017, pp. 688–697.
8. *Yang C., Buluç A., and Owens J. D.* Graphblast: A high-performance linear algebra-based graph framework on the GPU. ACM Trans. Math. Software, 2022, vol. 48 (1), pp. 1–51.
9. *Yang C., Wang Y., and Owens J. D.* Fast sparse matrix and sparse vector multiplication algorithm on the GPU. Proc. IPDPSW'15, IEEE Computer Society, 2015, pp. 841–847.
10. *Burkhardt P.* Optimal algebraic Breadth-First Search for sparse graphs // ACM Trans. Knowl. Discov. Data. 2021. V. 15 (5). Article 77.
11. *Prolubnikov A. V.* Svedenie zadachi proverki izomorfizma grafov k zadache proverki ravenstva polinomov ot n peremennykh [Reduction of the graph isomorphism problem to checking the equality of polynomials of n variables]. Trudy Instituta Matematiki i Mekhaniki UrO RAN, 2016, vol. 22, no. 1, pp. 235–240. (in Russian)
12. *Prolubnikov A. V.* Tochnost' i slozhnost' vychisleniy, neobkhodimye dlya proverki izomorfizma grafov sravneniem polinomov [Accuracy and complexity of computations needed to check graph isomorphism checking polynomials]. Vychislitel'nye Tekhnologii, 2016, vol. 21, no. 6, pp. 71–88. (in Russian)