МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ,
ЧИСЛЕННЫЕ МЕТОДЫ И КОМПЛЕКСЫ ПРОГРАММ

DOI 10.21672/2074-1707.2020.49.4.094-111
УДК 519.178/.245:004.94

НАХОЖДЕНИЕ ПРОВОДЯЩЕГО ОСТОВА В ДВУМЕРНОЙ РЕШЕТКЕ МЕТОДОМ ЗАЛИВКИ

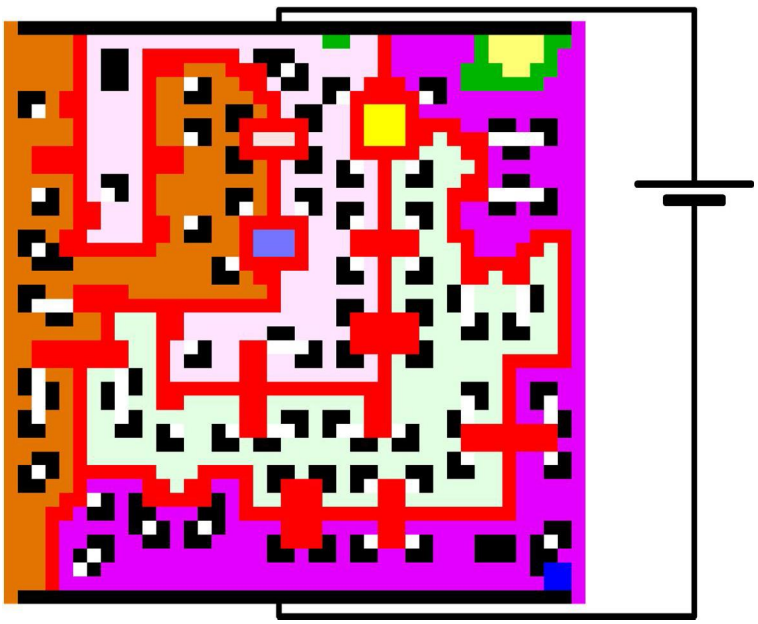
Статья поступила в редакцию 21.12.2019, в окончательном варианте – 02.2020.

Гордеев Иван Иванович, Астраханский государственный университет, 414056, Российская Федерация, г. Астрахань, ул. Татищева, 20а,
кандидат физико-математических наук, e-mail: g2i@mail.ru, ORCID https://orcid.org/0000-0001-5036-4791; elibrary: https://elibrary.ru/author_profile.asp?authorid=581330
Овчаренко Сергей Сергеевич, Астраханский государственный университет, 414056, Российская Федерация, г. Астрахань, ул. Татищева, 20а,
студент, e-mail: obchapehko@mail.ru
Сизова Анастасия Александровна, Астраханский государственный университет, 414056, Российская Федерация, г. Астрахань, ул. Татищева, 20а,
студент, e-mail: lucky_girl_zz@mail.ru

Показано место проблематики, рассматриваемой в настоящей статье, в структуре исследований по физике и вычислительной математике, практическая значимость этого направления исследовательских работ. Рассмотрена модель перколяции узлов в двумерной решетке с открытыми граничными условиями. Для этой модели проведен анализ основанного на заливке алгоритма, позволяющего находить остов соединяющего кластера. Предлагаемый алгоритм позволяет отделить от соединяющего кластера так называемые висячие части, которые не проводят ток. Приводится подробная классификация висячих частей (висячие концы, висячие циклы и висячие дуги). Охарактеризовано, какие именно висячие части обрабатываются на разных этапах алгоритма. Предлагается исправление некоторых неточностей, допущенных предыдущими авторами, описывавшими алгоритм ранее. Обосновывается, что при реализации алгоритма можно обойтись рассмотрением меньшей окрестности каждой из ячеек, соответствующих узлам. Обсуждаются особые случаи расположения узлов, алгоритмическая обработка которых не была описана ранее в литературе. Представлены особенности реализации исправленного (модифицированного) алгоритма на языке программирования C++. Указаны некоторые усовершенствования в реализации алгоритма, позволяющие увеличить скорость работы программы, построенной на его основе.

Ключевые слова: идентификация остова, соединяющий кластер, перколяция узлов, двумерная решетка, открытые граничные условия, заливка, окрестность фон Неймана, окрестность Мура, информационные технологии, алгоритмы вычислений

Графическая аннотация (Graphical annotation)



THE DETERMINATION OF THE CONDUCTING BACKBONE IN A TWO-DIMENSIONAL LATTICE BY THE FLOODING METHOD

The article was received by the editorial board on 21.12.2019, in the final version – 20.02.2020.

Gordeev Ivan I., Astrakhan State University, 20a Tatishchev St., Astrakhan, 414056, Russian Federation,
Cand. Sci. (Physics and Mathematics), e-mail: g2i@mail.ru, ORCID https://orcid.org/0000-0001-5036-4791; https://elibrary.ru/author_profile.asp?authorid=581330
Ovcharenro Sergey S., Astrakhan State University, 20a Tatishchev St., Astrakhan, 414056, Russian Federation,
student, e-mail: obchapehko@mail.ru
Sizova Anastasia A., Astrakhan State University, 20a Tatishchev St., Astrakhan, 414056, Russian Federation,
student, e-mail: lucky_girl_zz@mail.ru

In this paper, we give an analysis of a flood-based algorithm that makes possible to find the backbone of spanning cluster for the site percolation model in a two-dimensional lattice with open boundary conditions. The algorithm allows separating from the spanning cluster the so-called dangling parts, which do not conduct current. A detailed classification of the dangling parts (dangling ends, dangling cycles and dangling arcs) is given and it is specified which dangling parts are processed at different stages of the algorithm. Corrections are suggested for some inaccuracies of previous algorithm description. We found possible to consider a smaller neighborhood with the implementation of the algorithm. Cases in the implementation of the algorithm, not discussed earlier, are discussed. The implementation of the corrected algorithm in the C++ programming language is described. Some improvements allowing speed up the program are made to the algorithm.

Key words: backbone identification, spanning cluster, site percolation, 2D lattice, open boundary condition, flooding, von Neumann neighborhood, Moore neighborhood, information technology, computing algorithms

Введение. В настоящее время актуальной задачей является моделирование проводимости в неупорядоченных средах, состоящих из случайно расположенных проводящих и непроводящих элементов. Эта задача важна, в частности, для оценки проводимости при прохождении электрического тока в случайной среде [22]; при прохождении газа или жидкости в пористой среде [25]. В отношении последнего направления отметим задачи геофильтрации в неоднородных породах, а также исследование характеристик геологических пород на основе методов электропроводности [3, 8].

Несмотря на значительное количество работ, посвященных проблематике перколяции [5, 6, 7, 26, 27], некоторые направления исследований и разработок остаются исследованными недостаточно полно. Поэтому основной целью данной работы было устранение указанного недочета применительно к одному из направлений исследований по перколяции.

Общая характеристика проблематики исследований. Для моделирования случайных неупорядоченных сред часто используются периодические решетки, в которых рассматривается случайное заполнение *узлов* (sites) проводящими и непроводящими элементами. С точки зрения теории графов периодическая решетка является *периодическим графом* (periodic graph) без *петель* (loops). Вершины графа в подобных моделях называют узлами, а рёбра графа называют *связями* (bonds). При этом вершины графа являются случайным образом *занятыми* (occupied) или *свободными* (vacant) [20]. Обычно занятые узлы считаются проводящими, свободные – непроводящими. Следует отметить, что в англоязычной математической литературе по теории графов, петлей называют *ребро* (edge) графа, соединенное обоими концами с одной и той же *вершиной* (vertex) [23]. В то же время в англоязычной физической литературе, посвященной перколяции [25, 31, 35], термин *loop* часто используется в значении, которое в теории графов называется *циклом* (cycle) [23]. Модель, в которой рассматриваются случайным образом занятые узлы, называется моделью *перколяции узлов* (site percolation). Возможна также модель *перколяции связей* (bond percolation), в которой рассматриваются случайным образом занятые связи [20, 25]. В теории перколяции рассматриваются *связные компоненты* (connected components) графа из однотипных (занятых или незанятых) узлов или связей, называемые *кластерами* (clusters) [20, 25].

В данной статье дается анализ основанного на *заливке* (flood fill [30] или flooding [31]) алгоритма, позволяющего находить остов соединяющего кластера для модели перколяции узлов в двумерной решетке. *Соединяющий кластер* (spanning cluster) [10, 15, 25, 31] – это множество узлов, для которых существует непрерывная цепочка проводящих узлов от каждого из узлов до двух противоположных краев решётки. Иногда словосочетание *spanning cluster* переводится на русский язык как *стягивающий кластер* [18]. Однако, на наш взгляд, такой перевод не очень удачен, поскольку соединяющий кластер не обязательно «натянут», он может «висеть» достаточно

свободно. Для соединяющего кластера используются также названия *перколяционный кластер* (percolation cluster) [25] и *бесконечный кластер* (infinite cluster) [31]. Последнее название предполагает, что при увеличении размера решетки до бесконечности количество узлов, входящих в этот кластер, также стремится к бесконечности.

При моделировании конечных квадратных решеток обычно уточняют, с какими краями решётки соединяется перколяционный кластер, например, с верхним и нижним краями.

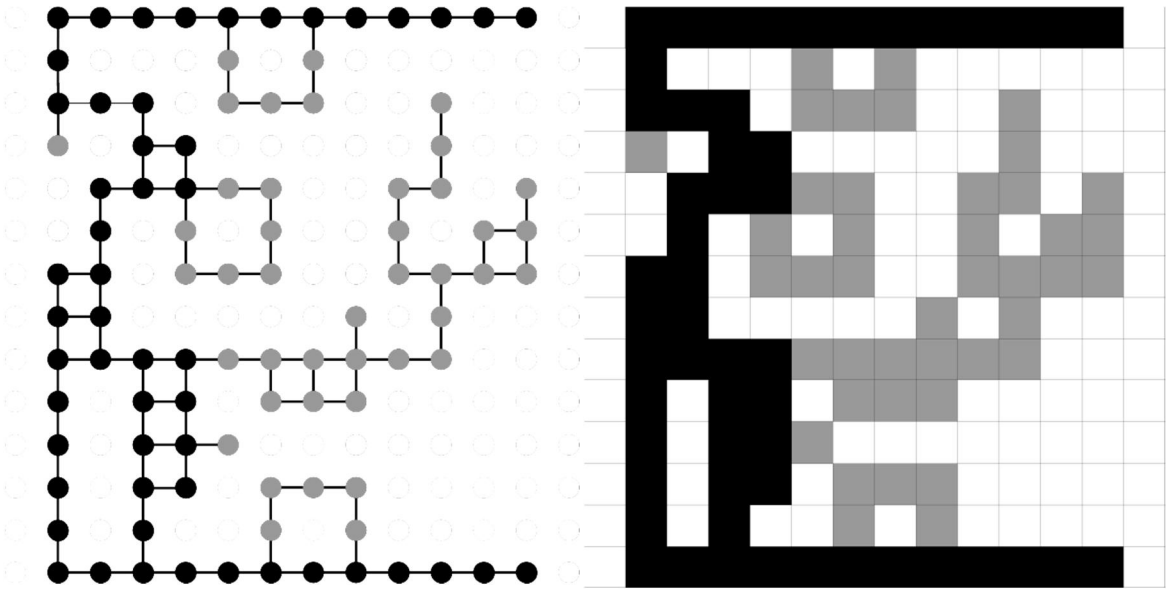
Перколяционный кластер может проводить электрический ток, жидкость или газ, если к противоположным краям прикладывается разность потенциалов или разность давлений соответственно. Для рассматриваемого класса задач в обеспечении проводимости электрического тока, жидкости или газа участвуют не все проводящие элементы перколяционного кластера, а только те, которые входят в так называемый *остов* (backbone) [4, 28] или *скелет* (в английском переводе также *backbone*) [9]. Остов можно определить как подграф перколяционного кластера, содержащий все вершины, от которых имеется как минимум два непересекающихся пути, ведущих к противоположным краям решётки. Здесь подразумеваются *вершинно-непересекающиеся* (vertex-disjoint) пути, в литературе по теории графов такие пути часто называют просто *непересекающимися* (disjoint) или *независимыми путями* (independent paths) [11]. Вершинно-непересекающиеся пути следует отличать от *рёберно-непересекающихся* (edge-disjoint) путей [11], поскольку вершинно-непересекающиеся пути всегда являются рёберно-непересекающимися, а обратное утверждение неверно. Проводящие элементы соединяющего кластера, которые не участвуют в проводимости, относятся к так называемым *висячим частям* (dangling parts) [35] перколяционного кластера.

Для плоских графов используют два варианта наглядного изображения. Первый вариант, который используется и для неплоских графов: граф изображают в виде жирных точек (кругов) и соединяющих точки линий, где жирные точки соответствуют вершинам графа, а линии соответствуют рёбрам графа. Второй вариант: граф изображают в виде многоугольников, разделяющих плоскость на части, где многоугольники соответствуют вершинам графа, а общая сторона каждой пары соседних многоугольников соответствует ребру между соответствующими вершинами. При втором варианте изображения многоугольники часто называют также *ячейками* (cells) [31, 34]. В одних публикациях по перколяции на плоских графах придерживаются первого способа изображения, например, в [16, 36]. В других – придерживаются второго способа, например, в [31, 34]. В третьих – комбинируют первый и второй способы, например, в [25].

На рисунке 1а показан первый способ изображения графа, где чёрные круги соответствуют проводящим узлам, входящим в остов, серые круги соответствуют висячим частям перколяционного кластера (проводящим узлам, входящим в перколяционный кластер, но не входящим в остов), а белые круги соответствуют непроводящим узлам. В данной статье мы будем далее придерживаться второго способа изображения графов. Поскольку рассматривается квадратная решётка, где у каждой вершины может быть четыре смежных, то вершинам графа на рисунках будут соответствовать квадраты. На рисунке 1б приведён второй способ изображения графа с чёрными, серыми и белыми квадратами соответственно.

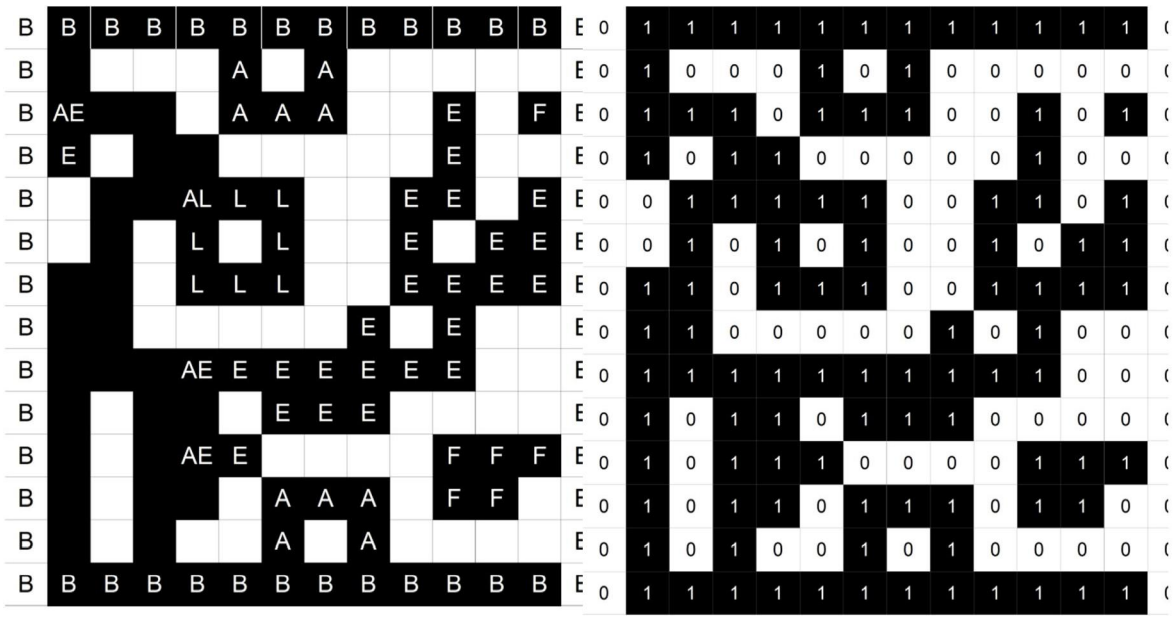
Описание алгоритма для идентификации остова было дано в статье Тробека и Стаматович [31]. При этом идея алгоритма в свою очередь основывается на статье Иня и Тао [34]. Следует отметить, что рассматриваемый алгоритм предполагает так называемую «*геометрию шины*» (bus-bar geometry) [28]. При использовании геометрии шины для конечной решетки рассматриваются перколяционные кластеры, соединенные с двумя любыми точками на противоположных краях (шинах) решетки. С физической точки зрения *шина* (bus-bar) [28] обычно представляет собой проводящую *полосу* (bar) [25], присоединенную к краю решетки. Возможно также рассмотрение непроводящих шин, которые расположены вдоль двух других противоположных краёв. На рисунке 2а ячейки, относящиеся к проводящим шинам, помечены белыми буквами «В», а ячейки, относящиеся к непроводящим шинам, помечены чёрными буквами «В».

Непроводящие шины по краям соответствуют так называемым «*открытым граничным условиям*» (open boundary conditions) [31, 34]. В качестве синонима открытых граничных условий иногда говорят об *отсутствии периодических граничных условий* (without periodic boundary conditions) [25, 35]. С точки зрения теории графов, шина представляет собой дополнительный ряд узлов вдоль края решетки, которые не являются случайно занятыми, а всегда заняты (или всегда свободны).



а) б)

Рисунок 1 – Изображение плоского графа в виде: а) жирных точек (кругов) и соединяющих линий; б) квадратов (ячеек)



а) б)

Рисунок 2 – Исходная решетка: а) ячейки, соответствующие шинам (B), висячим концам (E), висячим циклам (L), висячим дугам (A), сочленяющие ячейки в остове для висячих концов (AE) и висячих циклов (AL), ячейки, не входящие в перколяционный кластер (F); б) с пометкой числами проводящих (1) и непроводящих (0) ячеек

Описываемый далее алгоритм позволяет отделить от соединяющего кластера висячие части, которые не проводят ток. В статье Иня и Тао [35] дается подробная классификация висячих частей: *висячие концы* (dangling ends), *висячие циклы* (dangling loops) и *висячие дуги* (dangling arcs). На рисунке 2а буквами помечены ячейки, соответствующие висячим концам («E»), висячим циклам («L») и висячим дугам («A»). Висячими концами и висячими циклами считаются части перколяционного кластера, которые соединены с ним только через один *сочленяющий узел* (articulation site [34]). Сочленяющий узел в теории графов называют *точкой сочленения* (articulation point) [1, 11], (cutpoint) [17], либо *шарниром* [1]; используются также термины *разделяющая вершина* (separating vertex) или *разрезающая вершина* (cut vertex) [23], в некоторых источниках *cutvertex* пишется слитно [11]. В статье Тробека и Стаматович [31] соответствующие узлы называются

сочленяющими ячейками (articulation cells). На рисунке 2а на остове помечены сочленяющие ячейки, к которым присоединены висячие концы («AE») и висячие циклы («AL»). При этом висячие концы не содержат циклов, а висячие циклы содержат. С точки зрения теории графов, висячие циклы содержат цикл, проходящий через точку сочленения, но не принадлежащий остову, остову принадлежит только точка сочленения. Висячие концы соединены с остовом при помощи *мостов* (bridges) [11, 17, 32], при этом конец моста, принадлежащий остову, всегда является точкой сочленения. В некоторых источниках мост называется *разделяющим ребром* (separating edge) [23] или *разрезающим ребром* (cut edge) [23], в русскоязычных публикациях и переводах используется также название *перешеек* [1, 32].

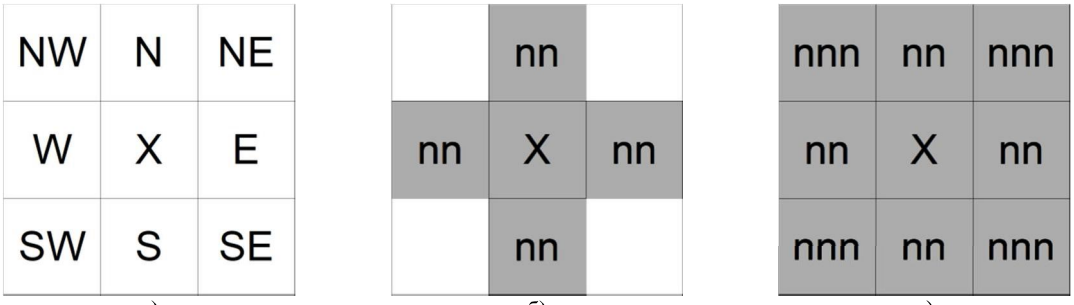
Висячими дугами называются части кластера, присоединенные через два или более узлов к одной из шин, но не имеющие соединения с другой шиной, не проходящего через первую шину. Поскольку в некоторых модификациях алгоритмов нахождения остова шины не совсем аккуратно рассматриваются тоже как часть остова, то висячие дуги могут ошибочно присоединяться к остову. В частности, в предыдущей версии того же алгоритма, предложенного в статье Стаматович и Тробека [24], висячие дуги присоединялись к остову.

Занятые ячейки, не соединённые с перколяционным кластером, а образующие отдельные кластеры, на рисунке 2б помечены белыми буквами «F».

При моделировании проводящих и непроводящих узлов их обычно кодируют различными числами. Например, проводящие узлы кодируют числом «1», а непроводящие – числом «0». При анализе различных частей решетки некоторые узлы могут помечаться и другими кодами, отличающимися от «0» и «1». Чтобы наглядно показать части решётки на рисунке, различным кодам ставят в соответствие разные цвета. На рисунке 2б показана заполненная случайным образом исходная решётка, в которой висячие части ещё не отделены, с пометкой проводящих и непроводящих узлов как цветами, так и числами.

Согласно описанию Тробека и Стаматович [31], алгоритм нахождения проводящего остова делится на 7 шагов. Нами проанализировано описание каждого шага в алгоритме Тробека, выявлены и исправлены некоторые неточности в описании этого алгоритма; выполнена реализация алгоритма на языке программирования C++ [2]. Также сделаны некоторые усовершенствования алгоритма, позволяющие ускорить работу программы.

Алгоритм основывается на анализе локальных свойств ячейки и ее соседей. На рисунке 3а показаны используемые обозначения восьми соседей ячейки (X) по сторонам света: северный (N), западный (W), южный (S), восточный (E), северо-западный (NW), юго-западный (SW), юго-восточный (SE), северо-восточный (NE), использованы стандартные сокращения по английским названиям сторон света (North, West, South, East).



а) б) в)

Рисунок 3 – Обозначения соседних ячеек и окрестности: а) обозначение соседних ячеек по сторонам света; б) окрестность Неймана (nn-окрестность); в) окрестность Мура (nnn-окрестность)

В теории клеточных автоматов используются понятия *окрестности фон Неймана* (von Neumann neighborhood) и *окрестности Мура* (Moore neighborhood) [29]. В окрестность фон Неймана (рис. 3б) входит ячейка X, а также её N, W, S и E соседи, называемые nn-соседями (nn – сокращение от nearest neighbors – ближайшие соседи) [31]. В окрестность Мура (рис. 3в), кроме ячеек, входящих в окрестность фон Неймана, входят также NW, SW, SE и NE соседи, называемые nnn-соседями (nnn – сокращение от next nearest neighbors – следующие ближайшие соседи) [31]. Для окрестности фон Неймана и окрестности Мура используются также названия nn-окрестность и nnn-окрестность, соответственно. В nn-окрестность (nnn-окрестность) входит сама ячейка и 4 ячейки (8 ячеек) вокруг, соответственно. Множества ячеек одного типа (занятых или незанятых), в которых любые две ячейки, либо входят в общую nn-окрестность (nnn-окрестность), либо имеют связывающую их цепочку

из соседей того же цвета, имеющих общую nn-окрестность (nnn-окрестность), образуют nn-кластеры (nnn-кластеры), соответственно. В рассматриваемой модели в качестве перколяционного рассматривается nn-кластер из проводящих узлов, соединяющий верхнюю и нижнюю шины. В качестве вспомогательных рассматриваются непроводящие nnn-кластеры.

В статье Тробека и Стаматович [31] предлагается рассматривать для нахождения остова также 16 еще более дальних nnnn-соседей (not next-nearest neighbors), которые вместе с 9 ячейками nnn-окрестности образуют окрестность Мура второго порядка (или nnnn-окрестность) из 25 ячеек. Тробек и Стаматович, указывая необходимость использования nnnn-окрестности, ссылаются на статью Иня и Тао [34]. Однако в [34] не рассматриваются nnnn-окрестности, анализируются только nnn-окрестности. Наш анализ также показал, что для нахождения остова достаточно рассмотрения nnn-окрестности.

В предлагаемом алгоритме анализируется случайно заполненная квадратная решетка размера «L · L», L предполагается без учёта шин, на рисунке 2 L = 12. Первоначально все ячейки решетки случайным образом закрашены в два цвета: белый (0 – непроводящие узлы) и черный (1 – проводящие узлы). С учётом шин размер решётки оказывается «(L+2) · (L+2)», на рисунке 2 изображена решётка с шинами, где L+2 = 14. Чтобы у каждой анализируемой ячейки было восемь nn-соседей, к решетке вокруг шин добавляется ещё один вспомогательный слой по всему периметру (постоянно заполнен значением 0, на рисунке 2 не показан), с этим слоем решетка имеет размер «(L+4) · (L+4)».

Характеристика алгоритма. При описании алгоритма Тробек и Стаматович [31] используют понятие *такта времени* (time-step), соответствующее *дискретному шагу* (discrete step) в публикациях по клеточным автоматам [29]. Клеточный автомат рассматривается Тробеком и Стаматович как вычислительная машина, которая может реализовываться на реальном компьютере, в частности на суперкомпьютере. Тробек и Стаматович предполагают, что для обработки каждой ячейки может использоваться отдельный *вычислительный элемент* (computing element) [31] высокопроизводительного компьютера, например, отдельное ядро центрального или графического процессора. Понятие вычислительного элемента применяется, чтобы пояснить синхронизацию обработки ячеек разными вычислительными элементами. Каждый вычислительный элемент имеет собственную память и арифметический блок. При достаточно большом размере решётки даже на самых высокопроизводительных суперкомпьютерах нереально выделить отдельный вычислительный элемент для каждой ячейки, поэтому Тробек и Стоматович отмечают, что возможна обработка одним вычислительным элементом более чем одной ячейки. Однако вопрос об обработке нескольких ячеек одним вычислительным элементом не получил сколько-нибудь подробного освещения у Тробека и Стаматович.

В данной публикации мы подробно рассматриваем обработку большого числа ячеек на одном вычислительном элементе и возможность оптимизации соответствующей программной реализации. В качестве предельного случая нами рассматривается реализация обработки всех ячеек на одном вычислительном элементе.

Изначально вся решетка помечена двумя цветами – белым (0) и черным (1). За один такт каждая ячейка решетки может либо однократно менять свое состояние (цвет), либо не менять.

Также используется большое количество уникальных цветов для пометки непроводящих ячеек и нахождения непроводящих кластеров. Кроме того, применяется еще три особых цвета для раскраски остова. Эти три цвета условно называются синим, зеленым и красным. Вопрос о кодировке данных цветов в статье Тробека и Стаматович не обсуждается. Поэтому, чтобы избежать совпадений с другими цветами, нами была выбрана кодировка отрицательными числами. Далее приводится соответствующее кодировке цветов перечисляемый тип, используемый в программе.

```
enum{blue=-1,green=-2,red=-3};
```

В описываемом алгоритме выделяются 7 этапов, называемых «Шагами» (Steps). На рисунке 4 показана блок-схема алгоритма.

На «этапе 1» за один такт делается закраска так называемых затравочных ячеек внутри непроводящих nnn-кластеров уникальными цветами, зависящими от номера ячейки. В качестве затравочных выбираются все белые ячейки, у которых 3 соседа (S, SE и E) черные, а также нижние ячейки на левой и правой белых шинах. Такой способ выбора затравочных ячеек обеспечивает, что в каждом непроводящем nnn-кластере будет хотя бы одна затравочная ячейка (возможно несколько затравочных ячеек в одном кластере). Номер цвета, в который закрашивается затравочная ячейка, вычисляется по формуле: $m = i \times n + j$, где i – номер строки, j – номер столбца ячейки.

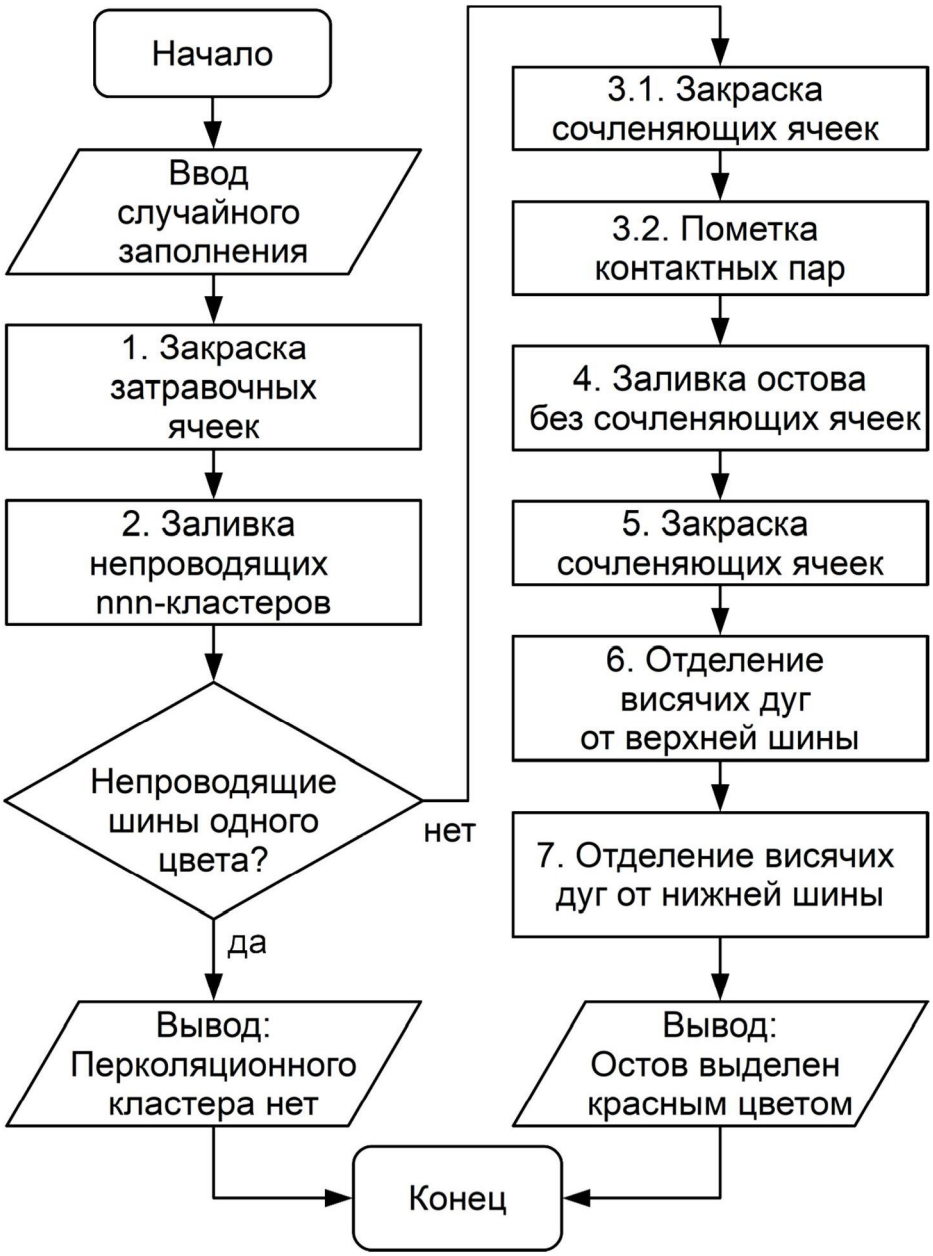


Рисунок 4 – Блок-схема алгоритма нахождения остова

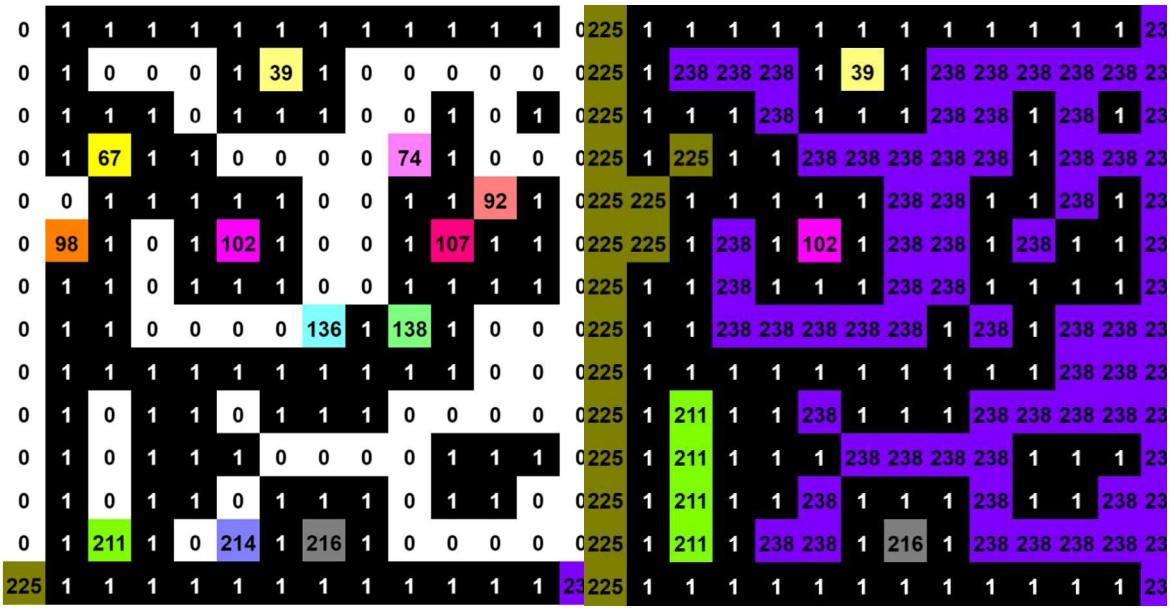
На рисунке 5а показана решетка после обработки по «этапу 1» исходной решётки (рис. 2). В данном примере закрашено четырнадцать затравочных ячеек, в соответствии с описанным правилом получивших номера цветов от 39 до 238.

На «этапе 2» происходит заливка каждого непроводящего nnn-кластера максимальным номером цвета из затравочных ячеек, имеющихся в данном кластере.

Заливка непроводящих nnn-кластеров означает, что на каждом такте перекрашиваются все нечерные ячейки, которые имеют в nnn-окрестности соседей с бо́льшими номерами цветов. Этап 2 сводится к многократной обработке всей решетки, которая повторяется до тех пор, пока меняется состояние хотя бы одной ячейки в решетке. На каждом такте заливки для всех нечерных ячеек (номер цвета не равен «1») из их nnn-окрестностей выбирается наибольший номер цвета. Если этот номер больше того номера цвета, который сейчас в ячейке, то ячейка перекрашивается в цвет с наибольшим номером.

На рисунке 5б показано состояние решетки после «этапа 2». Различными номерами цветов (39, 102, 211, 216, 225, 238) помечены шесть непроводящих nnn-кластеров, соответствующих незанятым узлам. Можно заметить, что в nnn-кластере, закрашенном номером цвета 225, перекрасились затравочные ячейки с номерами цветов 67 и 98, поскольку наибольшим в данном кластере

является номер 225. То же самое имеет место для кластера, закрашенного номером цвета 238. Все занятые узлы пока по-прежнему помечены черным. Левая и правая непроводящие шины оказались закрашены разными цветами – это говорит о том, что существует перколяционный кластер.



а) б)

Рисунок 5 – Решетка: а) после выполнения «этапа 1»; б) после выполнения «этапа 2»

При отсутствии распараллеливания вычислений все ячейки обрабатываются одним вычислительным элементом. Если один вычислительный элемент обрабатывает больше чем одну ячейку, то работа алгоритма на «этапе 2» может быть заметно ускорена за счет выбора порядка обработки ячеек одним вычислительным элементом на одном такте. В частности, поскольку затравочные ячейки с большим номером строки и большим номером столбца имеют бо́льшие номера цветов, то более эффективно на «этапе 2» делать обработку ячеек в порядке уменьшения номеров строк и столбцов. В этом случае происходит более быстрое распространение заливки, поскольку за один такт максимальный номер цвета в непроводящем nnn-кластере может распространиться не только на соседние, но и на гораздо более дальние ячейки.

Основным результатом «этапа 2» является выяснение того, существует ли перколяционный кластер. Если в результате заливки левая и правая непроводящие шины перекрасились в один и тот же цвет, то перколяционного кластера не существует, поскольку между проводящими шинами находится непрерывный изолирующий кластер. В случае, когда перекрывающего кластера не существует, то работа алгоритма может быть закончена уже на «этапе 2». Иначе следует переход к этапу 3. Дополнительным результатом «этапа 2» является то, что висячие части перекрывающего кластера оказываются окруженными оболочками из одноцветных непроводящих узлов. Этот результат используется на следующих этапах.

Этап 3. Тробек и Стаматович предлагают разбить этот этап на два *подэтапа* (sub-steps) [31]. На первом подэтапе закрашиваются сочленяющие ячейки. Алгоритм выполняет закрашивание сочленяющих ячеек, как принадлежащих остову, так и не принадлежащих остову.

На втором подэтапе помечаются так называемые контактные пары. Контактной парой называются два проводящих nn-соседа сочленяющей ячейки, через которых сочленяющая ячейка может быть присоединена к остову.

На «этапе 3» Тробек и Стаматович предлагают использовать для анализа nnnn-окрестность [24, 31]. Однако при этом они не приводят сколько-нибудь подробного описания алгоритма анализа nnnn-окрестности.

В общем случае при программной реализации клеточного автомата необходимо на каждом временном шаге формировать новую копию решетки на каждом временном шаге. В нашей реализации при использовании одного вычислительного элемента для нескольких ячеек возникает необходимость делать новую копию всей решетки только на этапах 3 и 5. На «этапе 3» в копии помечаются сочленяющие ячейки, и используется анализ только nnn-окрестности. Необходимость создания копии решётки связана с тем, что при закраске сочленяющих ячеек на первом подэтапе меняется состояние решетки. Это может приводить к некорректному определению контактных пар на втором подэтапе.

Во всех проводящих черных кластерах выделяются два вида сочленяющих ячеек. Сочленяющие ячейки *первого вида* входят в висячие концы. Такие ячейки не могут принадлежать остову и закрашиваются на «этапе 3» цветом, которым залит окружающий непроводящий nnn-кластер. Более точно, закрашиваются цветом окружающего непроводящего nnn-кластера те черные ячейки, которые имеют не более трех черных nn-соседей – при условии, что расположение черных ячеек и ячеек окружающего непроводящего nnn-кластера соответствует одной из конфигураций, показанных на рисунке 6, либо поворотам этих конфигураций на 90, 180 и 270 градусов.

В конфигурациях, изображенных на рисунке 6 (и далее на рис. 7, 8, 9), используются следующие метки: «1» – для проводящих ячеек черного цвета; «*» – для ячеек окружающего непроводящего кластера; «X» – для ячеек любого цвета (проводящих или непроводящих); «O» – для ячеек любого цвета, отличающегося от «*».

X	X	X
*	1	*
X	X	X

а)

X	1	*
*	1	1
X	*	X

б)

X	1	*
*	1	1
X	1	*

в)

Рисунок 6 – Сочленяющие ячейки первого вида

Три изображенные на рисунке 6 конфигурации соответствуют сочленяющим ячейкам первого вида. Во всех конфигурациях центральная ячейка закрашивается цветом окружающего непроводящего кластера, поскольку эта ячейка не может принадлежать остову. После закраски цветом непроводящего кластера такие ячейки не отличаются от непроводящих.

Следует отметить, что первая из конфигураций (рис. 6а) может соответствовать не только сочленяющим ячейкам, но также висячим ячейкам (если у центральной ячейки ровно один проводящий nn-сосед) или изолированным ячейкам (если у центральной ячейки нет проводящих nn-соседей). Для упрощения алгоритма все висячие и изолированные ячейки также закрашиваются цветом окружающего непроводящего кластера (хотя в закраске изолированных ячеек необходимости нет).

Сочленяющие ячейки *второго вида* присоединяют висячие циклы. Для таких ячеек ответить на вопрос о принадлежности их к остову сразу нельзя. Поэтому сочленяющие ячейки второго вида помечаются цветом «0» (не используемым после второго этапа), и рядом с каждой сочленяющей ячейкой второго вида помечаются так называемые контактные пары. Цветом «0» закрашиваются те черные ячейки, которые имеют трех или четырех черных nn-соседей и не менее двух nnn-соседей, относящихся к одному окружающему непроводящему кластеру при условии, что расположение черных ячеек и ячеек окружающего непроводящего nnn-кластера соответствует одной из конфигураций, показанных на рисунке 7, либо поворотам этих конфигураций на 90, 180 и 270 градусов.

Три изображенные на рисунке 7 конфигурации соответствуют сочленяющим ячейкам второго вида. Во всех конфигурациях центральная ячейка закрашивается значением «0», которое после заливки белых кластеров на «этапе 2» оказывается неиспользованным.

X	1	O
*	1	1
X	1	*

а)

X	1	*
*	1	1
X	1	O

б)

*	1	X
1	1	1
O	1	*

в)

Рисунок 7 – Сочленяющие ячейки второго вида

Все ячейки, не являющиеся сочленяющими, висячими или изолированными, закрашиваются на «этапе 3» тем же цветом, что и во входной копии решетки.

Одновременно с пометкой сочленяющих ячеек на шаге 3 в нашей программе помечаются контактные пары для сочленяющих ячеек второго вида. К контактной паре принадлежат две ячейки, являющиеся одновременно nn-соседями для ячейки, помеченной на рисунке 7 «O» и сочленяющей ячейки. В контактную пару может входить ячейка на шине.

Заметим, что одна ячейка может входить в более чем одну контактную пару, но этот важный

вопрос не рассмотрен Тробеком и Стаматович. Например, на рисунке 8а центральная ячейка входит в две контактные пары одновременно. Также Тробеком и Стаматович не рассмотрен важный вопрос, касающийся того, что сочленяющая ячейка может входить в контактную пару для другой сочленяющей ячейки. Например, на рисунке 8б две центральные ячейки одновременно являются сочленяющими и входят в контактные пары.

Возможны и еще более сложные сочетания вхождения в контактные пары с сочленяющими ячейками. Например, на рисунке 9 центральная ячейка входит одновременно в две контактные пары и является сочленяющей.

1	1	*	1	1
1	1	1	1	1
*	1	1	1	*

а)

1	1	*	*	1	1
1	1	1	1	1	1
*	*	1	1	*	*

б)

Рисунок 8 – Специфические расположения контактных пар: а) центральная ячейка входит в две контактные пары; б) две центральные ячейки одновременно являются сочленяющими и входят в контактные пары

1	1	*	1	1	*	*
1	1	1	1	1	1	1
*	*	1	1	*	1	1

Рисунок 9 – Центральная ячейка входит в две контактные пары и одновременно является сочленяющей

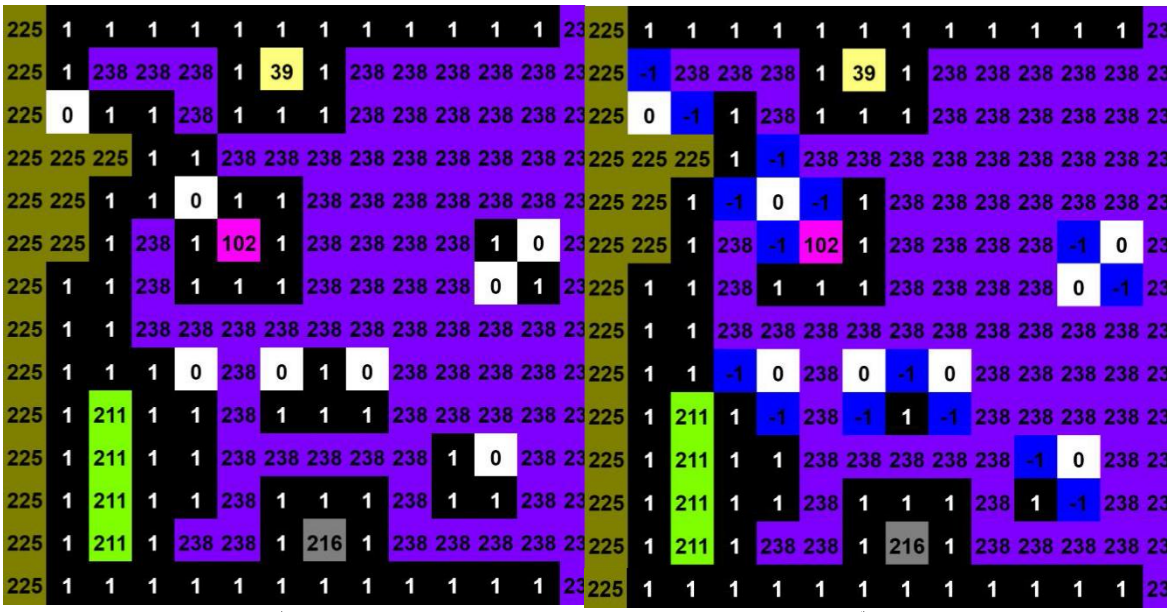
В статье [24] Стаматович и Тробека упоминают о 12 случаях (вариантах) пометки контактных ячеек, но, к сожалению, не поясняют, какие именно это случаи. Исходя из приведенных выше примеров, количество возможных сочетаний (с учетом ориентации для контактных пар) может оказаться существенно больше 12. Поэтому для упрощения алгоритма в нашей программе для каждой ячейки хранится отдельная вспомогательная структура данных, которая учитывает, что одна ячейка может входить в несколько контактных пар. Далее нами приводится определение соответствующей структуры данных в виде фрагмента программного кода на языке C++ (рис. 10)

```
typedef struct
{
	char NW, SW, NE, SE;
} Pairs;
```

Рисунок 10 – Фрагмент программного кода, описывающий структуру данных

Поля каждой ячейки NW, SW, NE и SE содержат «1», если соответствующий nnn-сосед ячейки входит с ней в контактную пару и содержит «0» в противном случае. Общий размер подобной структуры в C++ составляет 4 байта, хотя, используя битовые поля, можно уложиться и в 4 бита.

На рисунке 11а показаны помеченные белым (0) сочленяющие ячейки второго вида после первого подэтапа для этапа 3. На рисунке 11б синим цветом (-1) помечены ячейки, входящие в контактные пары после второго подэтапа. Можно заметить, что на рисунке 11б имеется нечетное количество ячеек, входящих в контактные пары (15), поскольку одна из ячеек входит в две контактные пары.



а) б)

Рисунок 11 – Решетка на этапе 3: а) сочленяющие ячейки второго вида после выполнения первого подэтапа этапа 3; б) ячейки контактных пар

Наша реализация алгоритма на «этапе 3» существенно отличается от описания алгоритма, предлагаемого Тробеком и Стаматович. Поэтому приведем фрагмент кода, отвечающий за пометку контактных пар – для определенности, когда у сочленяющей ячейки второго вида есть три проводящих nnn-соседа (рис. 12).

```
if(cell[m-1]==max1&&cell[m+n+1]==max1&&cell[m-n+1]!=max1||//пара NW иSE
cell[m+n]==max1&&cell[m-n-1]==max1&&cell[m-n+1]!=max1)
   q[m+1].NW=q[m-n].SE=1;
else if(cell[m+n]==max1&&cell[m-n+1]==max1&&cell[m-n-1]!=max1||//-90 градусов
cell[m+1]==max1&&cell[m+n-1]==max1&&cell[m-n-1]!=max1)
   q[m-1].NE=q[m-n].SW=1;
else if(cell[m+1]==max1&&cell[m-n-1]==max1&&cell[m+n-1]!=max1||//-180 градусов
cell[m-n]==max1&&cell[m+n+1]==max1&&cell[m+n-1]!=max1)
   q[m-1].SE=q[m+n].NW=1;
else if(cell[m-n]==max1&&cell[m+n-1]==max1&&cell[m+n+1]!=max1||//-270 градусов
cell[m-1]==max1&&cell[m-n+1]==max1&&cell[m+n+1]!=max1)
   q[m+1].SW=q[m+n].NE=1;
```

Рисунок 12 – Фрагмент программного кода в случае, когда у сочленяющей ячейки второго вида есть три проводящих nnn-соседа

В приведенном фрагменте кода max1 содержит цвет четвертого (непроводящего) nnn-соседа. Решетка для повышения эффективности хранится в одномерном массиве cell. Одномерный массив q типа Pairs содержит для каждой ячейки информацию о том, с какими ячейками она входит в контактную пару. Алгоритмом анализируются соседи-ячейки с индексом m. При этом N, W, S, E, NW, SW, SE, NE соседи имеют соответственно индексы *m-n, m-1, m+n, m+1, m-n-1, m+n-1, m+n+1*, *m-n+1*, где $n = L+4$ (размер решетки с учетом дополнительных слоев).

Следующий фрагмент кода (рис. 13) обрабатывает случаи, когда у сочленяющей ячейки второго вида есть четыре проводящих nnn-соседа.

```
if(cell[m-n-1]!=1 && cell[m-n-1]==cell[m+n+1])
{
   if(cell[m+n-1]!=cell[m-n-1])
          q[m-1].SE=q[m+n].NW=1;
   if(cell[m-n+1]!=cell[m-n-1])
          q[m+1].NW=q[m-n].SE=1;
}
else if(cell[m-n+1]!=1&&cell[m-n+1]==cell[m+n-1])
{
   if(cell[m-n-1]!=cell[m-n+1])
          q[m-1].NE=q[m-n].SW=1;
   if(cell[m+n+1]!=cell[m-n+1])
          q[m+1].SW=q[m+n].NE=1;
}
```

Рисунок 13 – Фрагмент программного кода для случаев, когда у сочленяющей ячейки второго вида есть четыре проводящих nnn-соседа

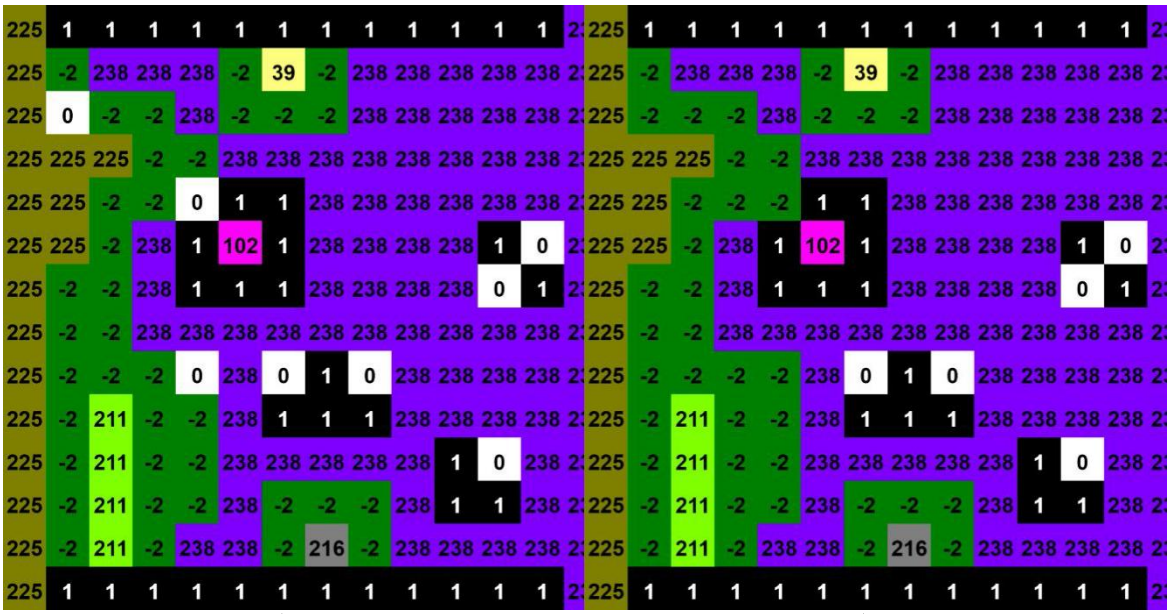
На этапе 4 делается комбинированная заливка, с одновременным применением nn- и nnn-правил, следующим образом. От верхней и нижней шин зеленым цветом (код цвета «-2») заливаются черные ячейки, для которых выполняется одно из следующих условий: они являются nn-соседом либо одной из шин; либо они уже являются закрашенной зеленым ячейки (nn-правило); либо они входят в контактную пару с уже закрашенной зеленым цветом ячейкой (nnn-правило, поскольку входящие в контактную пару ячейки являются nnn-соседями). Сами шины при этом не закрашиваются.

Так же как и на «этапе 2», «этап 4» реализуется многократным повторением описанных выше действий до тех пор, пока меняется состояние (характеристика) хотя бы одной ячейки в решетке. В статье [31] говорится о необходимости анализа nnnn-окрестностей на этапе 4. Однако предлагаемая нами реализация требует только анализа nnn-окрестностей. Поэтому приведем наш вариант реализации «этапа 4» в виде фрагмента программного кода (рис. 14).

```
int flag=1;
while(flag)
{
       flag=0;
       for(int i=2;i<n-2;i++)
              for(int j=2;j<n-2;j++)
              {
                     int m=i*n+j;
                     if(cell[m]==1)
                     {
                            if(i==(n-3)||i==2||//отшин
                            cell[m-n]==green||cell[m-1]==green||//nn-соседи
                            cell[m+n]==green||cell[m+1]==green||
                            cell[m-n-1]==green&&q[m].NW||//nnn-соседи
                            cell[m+n-1]==green&&q[m].SW||
                            cell[m+n+1]==green&&q[m].SE||
                            cell[m-n+1]==green&&q[m].NE)
                            {
                                   cell[m]=green;
                                   flag++;
                            }
                     }
              }
}
```

Рисунок 14 – Фрагмент реализации программного кода для «этапа 4»

В приведенной реализации один проход для внешнего цикла while соответствует одному такту клеточного автомата. Переменная flag позволяет определить, было ли изменение характеристик (состояния) ячеек на такте. На рисунке 15а показан вид решетки после выполнения «этапа 4».

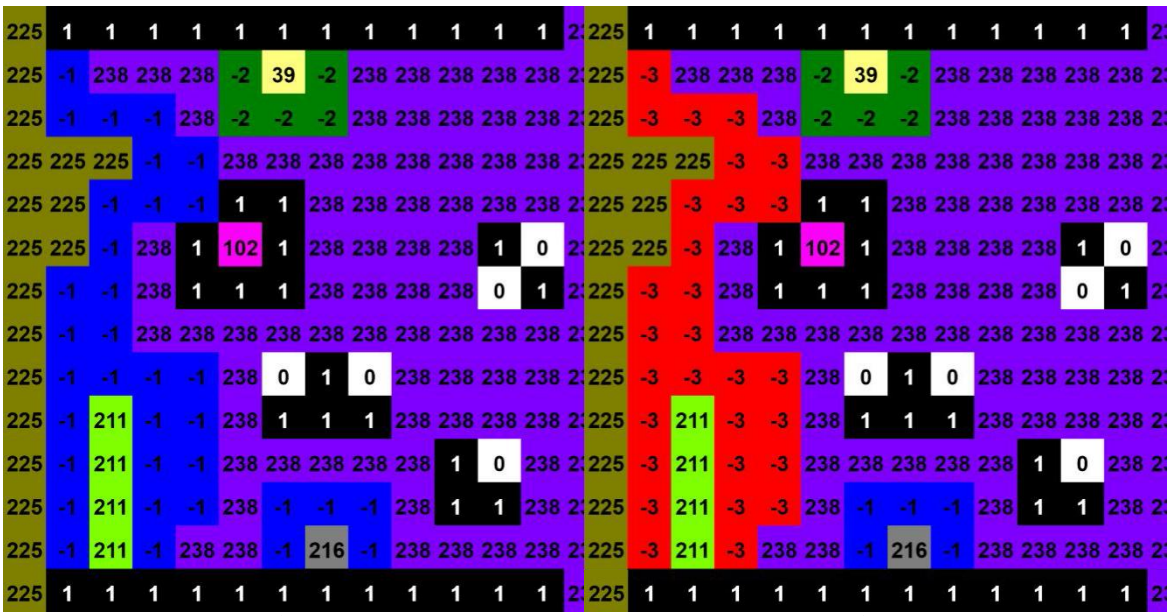


а) б)

Рисунок 15 – Вид решетки после выполнения: а) «этапа 4»; б) «этапа 5»

На этапе 5 закрашиваются зеленым те ячейки, помеченные символом «0», у которых есть nn-сосед, закрашенный зеленым. Иными словами, закрашиваются сочленяющие ячейки, принадлежащие остову. Действие выполняется однократно. На рисунке 15б показан вид решетки после выполнения «этапа 5». На нем остов уже в основном определен, хотя к верхней и нижней шине могут быть прикреплены закрашенные зеленым цветом висячие дуги, не относящиеся к остову. Чтобы их отбросить, выполняются этапы 6 и 7.

На «этапе 6» от нижней шины делается nn-заливка (заливка nn-кластеров) тех ячеек, которые закрашены зеленым, синим цветом (-1). Выражаясь более точно, на «этапе 6» заливаются синим цветом те из ячеек, помеченных зеленым, которые либо прилегают к нижней шине, либо имеют nn-соседа, уже закрашенного синим цветом. В результате после закраски остова от нижней шины синим цветом этим цветом уже не будут закрашены висячие дуги, которые прикреплены к верхней шине (рис. 16а).



а) б)

Рисунок 16 – Вид решетки после выполнения: а) «этапа 6»; б) «этапа 7»

На этапе 7 делается заливка от верхней шины красным цветом (-3) тех ячеек, которые закрашены синим цветом. В результате отбрасываются висячие дуги, которые могли быть прикреплены к нижней шине. В результате получается остов в чистом виде, закрашенный красным цветом (рис. 16б).

Результаты вычислительных экспериментов. Объём выполняемого файла написанной нами программы составляет 40 килобайт (в машинных кодах). Использовался компилятор C++ Microsoft Visual Studio 2012.

В примере заполнения решётки, который иллюстрирует этапы выполнения алгоритма на рисунках 2, 5, 11, 15, 16, из-за малого размера решётки встречаются не все варианты расположения сочленяющих ячеек, показанные на рисунках 6–9. Для полноценного тестирования алгоритма нами специально был подготовлен файл [14] с заполнением решётки проводящими и непроводящими узлами, в котором встречаются все показанные на рисунках 6–9 случаи расположения сочленяющих ячеек и их повороты на 90, 180 и 270 градусов. Для случаев, показанных на рисунках 8, 9, в данном файле имеются варианты крепления конфигураций ячеек к остову за угол и за середину. Для рисунка 9 в файле имеются также зеркальные конфигурации. Общий размер «случайно» заполненной части в тестовом примере L = 40. Однако поскольку в файле показаны шины, то общее количество строк и столбцов L+2 = 42. Результат обработки тестового файла на каждом этапе можно посмотреть в файле, приведенном в [13]. В графической аннотации к статье показан окончательный вариант раскраски.

Нами было протестировано среднее время выполнения алгоритма на заполненных при помощи генератора случайных чисел решетках размером L = 200, 280, 400, 560, 800, 1120, 1600. Шаг размера решетки был выбран так, что на каждой следующей решетке число узлов было примерно в два раза больше, чем на предыдущей. На всех размерах решеток тестировались различные доли заполнения решетки p с шагом 0,01 от 0 до 1. Для каждой доли заполнения p выполнялось по 1000 (для размеров решеток от 200 до 800) или 500 (для размеров решеток от 1120 до 1600) моделирований и определялось среднее время работы алгоритма. Тестирование проводилось на компьютере с процессором Intel Core i3-3210 с тактовой частотой 3,2 ГГц, программа задействовала одно ядро процессора. На рисунке 17 показаны пять графиков зависимости среднего времени t (в секундах) работы программы от доли заполнения решетки «p».

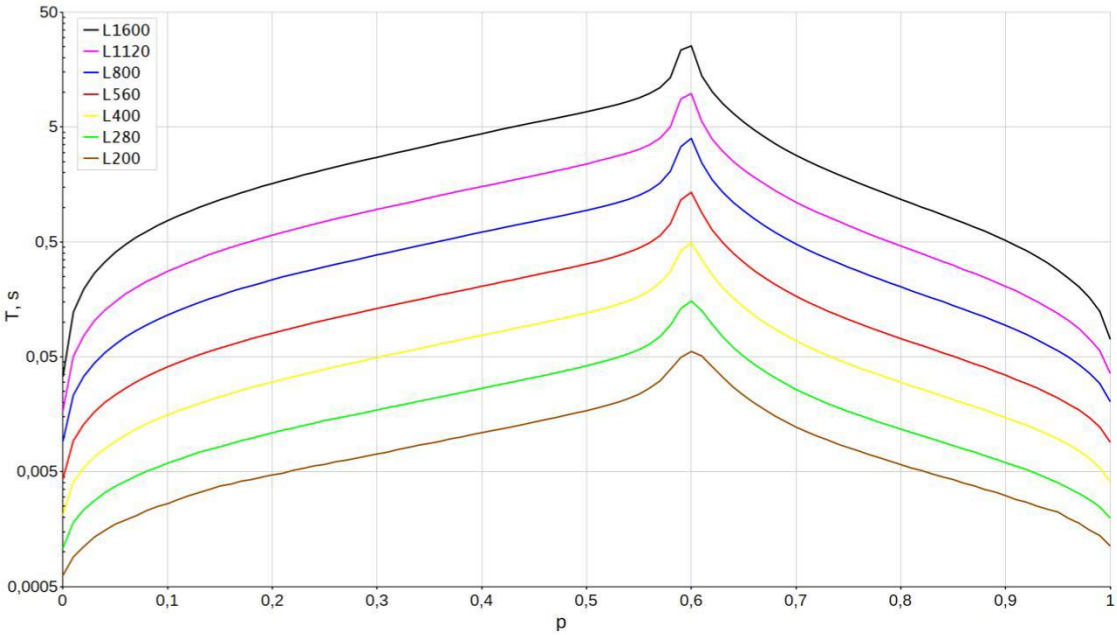


Рисунок 17 – Графики зависимости среднего времени T (в секундах) работы программы от доли заполнения решетки p для разных размеров решёток. Снизу вверх: L = 200 (коричневый цвет), L = 280 (зеленый), L = 400 (желтый), L = 560 (красный), L = 800 (красный), L = 1120 (фиолетовый), L = 1600 (чёрный)

Тробеком и Стаматович также делалась оценка трудоемкости алгоритма, но только для значений 0,62 <= p <= 0,65 [31]. К сожалению, Тробек и Стаматович не указывают, почему был выбран именно данный диапазон вероятностей. Можно предположить, что в [31] были взяты доли заполнения решеток, для которых еще не возникало отсутствия перколяции. Нами было обнаружено при моделировании решеток с L = 200, что для p = 0,61 в 3,2 % случаев отсутствовал перколяционный

кластер, тогда как при p = 0,62 случаев отсутствия перколяционного кластера не встречалось.

Несмотря на то, что при p = 0,60 для всех размеров решеток встречались случаи отсутствия перколяционного кластера, когда алгоритм выполняет только два из семи этапов, наиболее трудоемкими оказались случаи с долей заполнения p = 0,60. Это значение чуть выше порога перколяции узлов на квадратной решетке, равного, согласно публикациям последних лет [12, 19, 21, 33,], с точностью до шести знаков после запятой p = 0,592746.

На рисунке 18 на графике с логарифмическим масштабом по обоим осям показана зависимость среднего времени расчетов «T» от размера решетки «L» на основе экспериментальных данных для наиболее трудоемкого случая при p = 0,60. Нами была проделана (с использованием метода наименьших квадратов) аппроксимация трудоемкости алгоритма в зависимости от L функцией вида $T(L)\sim L^k$. В результате мы получили коэффициент k = 3,0, то есть $T(L)\sim L^{3,0}$. Тробек и Стаматович указывают асимптотическую сложность алгоритма вблизи порога перколяции $\Theta(L)$ при реализации на параллельных платформах [31]. Однако, поскольку Тробек и Стаматович предполагают использование отдельного вычислительного элемента для каждой ячейки, а количество ячеек растет как L^2, то реальная трудоёмкость алгоритма в расчете на один вычислительный элемент растет как $\Theta(L^3)$, хотя Тробек и Стаматович в явной форме этого не указывают. Грубая оценка трудоёмкости по точкам графика, которую приводят Тробек и Стаматович для p = 0,62 даёт примерно $\Theta(L^{1,0})$ или, в расчете на один вычислительный элемент, $\Theta(L^{3,0})$, что совпадает с нашей оценкой.

Коэффициент детерминации для регрессии, показанной на рисунке 18, составил $R^2 = 0,9988$.

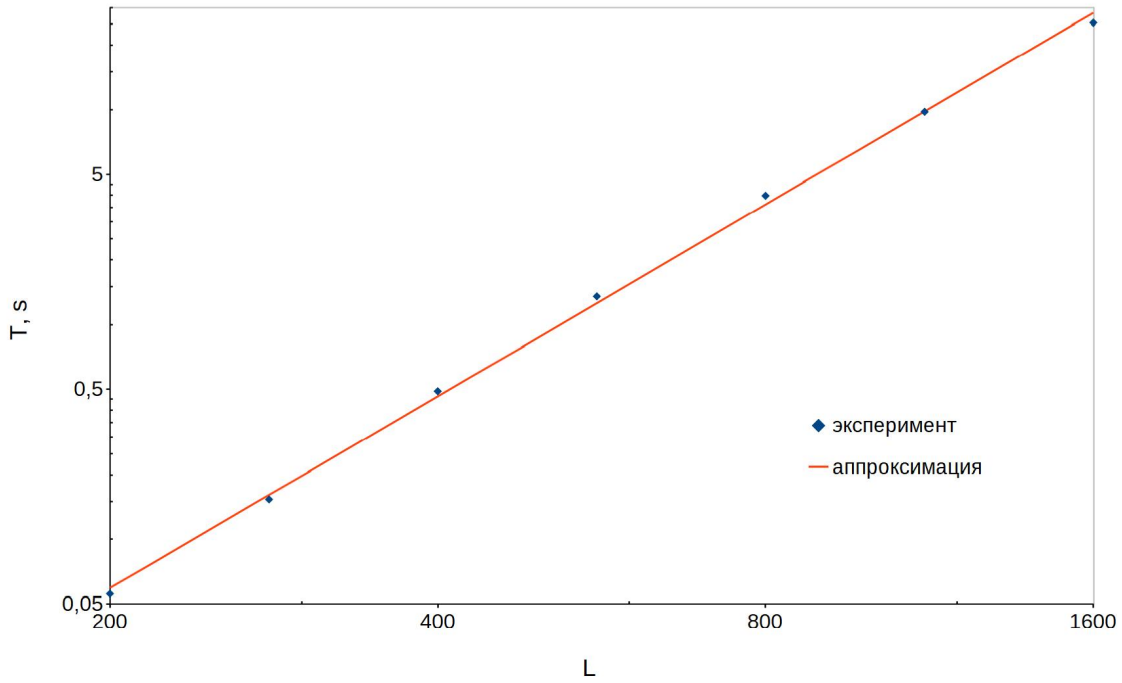


Рисунок 18 – Зависимость среднего времени T (в секундах) работы программы от размера решетки L при p = 0,60 (эксперимент) и аппроксимация среднего времени функцией $T(L)\sim L^k$

Заключение. Авторами был проведен анализ содержания предложенного недавно алгоритма для идентификации остова [31], основанного на заливке ячеек. Были выявлены неточности в описании алгоритма, имеющиеся в опубликованных работах. Нами предложен вариант алгоритма с исправлением этих неточностей. Кроме того, выполнена реализация алгоритма на языке программирования C++, и предложены тестовые примеры со специфическими конфигурациями ячеек.

Описанная нами реализация алгоритма на языке C++ предполагает выполнение всех вычислений на одном вычислительном элементе. Однако, как отмечают Тробек и Стаматович [31], скорость работы алгоритма должна быть существенно выше при параллельной реализации алгоритма на многих вычислительных элементах. Поэтому нами была также подготовлена параллельная реализация алгоритма с использованием интерфейса программирования MPI (Message Passing Interface). Обсуждение деталей такой реализации и её вычислительной эффективности предполагается выполнить в следующей работе.

Библиографический список

1. Алексеев В. Е. Графы и алгоритмы. Структуры данных. Модели вычислений / В. Е. Алексеев. – Москва : Интернет-университет информационных технологий ; БИНОМ. Лаборатория знаний, 2012. – 320 с.

2. Гордеев И. И. Программа для нахождения геометрического остова перколяционного кластера. Свидетельство о государственной регистрации программы для ЭВМ № 2018610093. Зарегистрировано в Реестре программ для ЭВМ 09 января 2018 года / И. И. Гордеев, А. А.Сизова.

3. Кадет В. В. Перколяционный анализ гидродинамических и электрокинетических процессов в пористых средах / В. В. Кадет. – Москва : ИНФРА-М, 2017. – 256 с.

4. Москалев П. В. Анализ структуры перколяционного кластера / П. В. Москалев // Журнал технической физики. – 2009. – Т. 79, № 6. – С. 1–7.

5. Москалев П. В. Оценки порога и мощности перколяционных кластеров на квадратных решетках с (1, π)-окрестностью / П. В. Москалев // Компьютерные исследования и моделирование. – 2014. – Т. 6, № 3. – С. 405–414. DOI: 10.20537/2076-7633-2014-6-3-405-414.

6. Москалев П. В. Перколяционное моделирование пористых структур / П. В. Москалев. – Москва : ЛЕНАНД, 2018. – 240 с.

7. Москалев П. В. Структура моделей перколяции узлов на трехмерных квадратных решетках / П. В. Москалев // Компьютерные исследования и моделирование. – 2013. – Т. 5, № 4. – С. 607–622. DOI: 10.20537/2076-7633-2013-5-4-607-622.

8. Селяков В. И. Перколяционные модели переноса в микронеоднородных средах / В. И. Селяков, В. В. Кадет. – Москва : Недра, 1995. – 224 с.

9. Эфрос А. Л. Физика и геометрия беспорядка / А. Л. Эфрос. – Москва : Наука. Главная редакция физико-математической литературы, 1982. – 176 с.

10. Binder K. Monte Carlo simulation in statistical physics: an introduction / K. Binder, D. W. Heermann. – 4th ed. – Berlin ; New York : Springer, 2002.

11. Diestel R. Graph Theory / R. Diestel. – New York : Springer, 2000.

12. Feng X. Percolation transitions in two dimensions / X. Feng, Y. Deng, H. W. J. Blöte // Physical Review E. – 2008. – Vol. 78, 031136. DOI: 10.1103/PhysRevE.78.031136.

13. G2ii2g. Backbone-flooding/output40.txt. – Режим доступа: https://github.com/G2ii2g/backbone-flooding/blob/master/output40.txt, свободный. – Заглавие с экрана. – Яз. рус. (дата обращения: 27.08.2019).

14. G2ii2g. Backbone-flooding/test40.txt. – Режим доступа: https://github.com/G2ii2g/backbone-flooding/blob/master/test40.txt, свободный. – Заглавие с экрана. – Яз. рус. (дата обращения: 27.08.2019).

15. Gould H., Tobochnik J. An Introduction to Computer Simulation Methods: Applications to Physical Systems / H. Gould, J. Tobochnik. – Part 2. Reading. – MA: Addison-Wesley, 1987.

16. Grimmet G. Percolation / G. Grimmet // Geoffrey Grimmett. – 2nd ed. – Berlin ; Heidelberg ; New York ; Barcelona ; Hong Kong ; London ; Milan ; Paris ; Singapore ; Tokyo : Springer, 1999. (Grundlehren der mathematischen Wissenschaften; 321).

17. Harary F. Graph Theory / F. Harary. – Boca Raton ; London ; New York : CRC Press, 2018.

18. Heermann D. W. Computer Simulation Methods in Theoretical Physics / D. W. Heermann. – Berlin ; Heidelberg ; New York ; London ; Paris ; Tokyo : Springer, 1986. – 154 p.

19. Jacobsen J. L. Critical points of Potts and O(N) models from eigenvalue identities in periodic Temperley–Lieb algebras / J. L. Jacobsen // Journal of Physics A: Mathematical and Theoretical. – 2015. – Vol. 48, 454003. DOI: 10.1088/1751-8113/48/45/454003.

20. Kesten H. Percolation theory for mathematicians / H. Kesten. – Boston ; Basel ; Stuttgart : Birkhauser, 1982. – 423 p.

21. Lee M. J. Pseudo-random-number generators and the square site percolation threshold / M. J. Lee // Physical Review E. – 2008. – Vol. 78, 031131. DOI: 10.1103/PhysRevE.78.031131.

22. Mutiso R. M. Electrical properties of polymer nanocomposites containing rod-like nanofillers / R. M. Mutiso, K. I. Winey // Progress in Polymer Science. – 2015. – Vol. 40. – P. 63–84. DOI: 10.1016/j.progpolymsci.2014.06.002.

23. Ore O. Theory of graphs. Providence, Rhode Island: American mathematical society / O. Ore. – 1974. – 270 p.

24. Stamatovic B. Data parallel algorithm in finding 2-D site percolation backbones / B. Stamatovic, R. Trobec // Proceeding of the First International Workshop on Sustainable Ultrascale Computing Systems (NESUS 2014). – Porto, Portugal ; Madrid : Computer Architecture Communications, and System Group (ARCOS) University Carlos III, 2014. – P. 65–70.

25. Stauffer D. Introduction to Percolation Theory / D. Stauffer, A. Aharony. – 2nd ed., rev. – London : Taylor & Francis, 2003. – 180 p.

26. Tarasevich Yu. Yu. Percolation of linear k-mers on a square lattice: From isotropic through partially ordered to completely aligned states / Yu. Yu. Tarasevich, N. I. Lebovka, V. V. Laptev // Physical Review E. – 2012. – Vol. 86, 061116. DOI: 10.1103/PhysRevE.86.061116.

27. Tarasevich Yu. Yu., Lebovka N. I., Vodolazskaya I. V. et al. Anisotropy in electrical conductivity of two-dimensional films containing aligned nonintersecting rodlike particles: Continuous and lattice models // Physical Review E. 2018. Vol. 98, 012105. DOI: 10.1103/PhysRevE.98.012105.

28. Tarasevich Yu.Yu. Identification of current-carrying part of a random resistor network: electrical approaches vs. graph theory algorithms / Yu. Yu. Tarasevich, A. S. Burmistrov, V. A. Goltseva, I. I. Gordeev, V. I. Serbin,

A. A. Sizova, I. V. Vodolazskaya, D. A. Zholobov // Journal of Physics: Conference Series. – 2018. –Vol. 955, 012021. DOI:10.1088/1742-6596/955/1/012021.

29. Toffoli T., Margolus N. Cellular Automata Machines: A new environment for modeling / T. Toffoli, N. Margolus. – Cambridge, Massachusetts ; London, England : The MIT Press, 1987.

30. Torbert S. Applied Computer Science / S. Torbert. – 2nd ed. – Springer, 2016. – 279 p.

31. Trobec R. Analysis and classification of flow-carrying backbones in two-dimensional lattices / R. Trobec, B. Stamatovic // Advances in Engineering Software. – 2017. – Vol. 103. – P. 38–45. DOI: 10.1016/j.advengsoft.2015.11.002.

32. Wilson R. J. Introduction to Graph Theory / R. J. Wilson. – 4th ed. – Edinburgh Gate, Harlow : Addison Wesley Longman Limited, 1998.

33. Yang Y. Square++: Making a connection game win-lose complementary and playing-fair / Y. Yang, S. Zhou, Y. Li // Entertainment Computing. – 2013. – Vol. 4. – P. 105–113. DOI: 10.1016/j.entcom.2012.10.004.

34. Yin W.-G. Algorithm for finding two-dimensional site percolation backbones / W.-G. Yin, R. Tao // Physica B: Condensed Matter. – 2000. – Vol. 279. – P. 84–86. DOI: 10.1016/S0921-4526(99)00675-4.

35. Yin W.-G. Rapid algorithm for identifying backbones in the two-dimensional percolation model / W.-G. Yin, R. Tao // International Journal of Modern Physics C. – 2003. – Vol. 14. – P. 1427–1437. DOI: 10.1142/S0129183103005509.

References

1. Alekseev V. E. *Grafy i algoritmy. Struktury dannykh. Modeli vychisleniy* [Graphs and algorithms. Data structures. Computation Models]. Moscow, 2012. 320 p. ISBN 978-5-94774-543-6.

2. Gordeev I. I., Sizova A. A. *Programma dlya nakhozhdeniya geometricheskogo ostova perkolyacionnogo klastera Svidetelstvo o gosudarstvennoy registratsii programmy dlya EVM № 2018610093. Zaregistrirovano v Reestre programm dlya EVM 09 yanvarya 2018 goda* [The program for finding the geometric backbone of a percolation cluster. Certificate of state registration of computer programs No. 2018610093. Registered in the Register of computer programs on January 9, 2018].

3. Kadet V. V. *Perkolyatsionnyh analiz gidrodinamicheskikh i elektrokineticheskikh protsessov v poristykh sredakh* [Percolation analysis of hydrodynamic and electrokinetic processes in porous media]. Moscow, 2017. 256 p.

4. Moskalev P. V. Analiz struktury perkolyacionnogo klastera [Analysis of the percolation cluster structure]. *Zhurnal tekhnicheskoy fiziki* [Technical Physics], 2009, vol. 79, no. 6, pp. 1–7.

5. Moskalev P. V. Otsenki poroga i moshchnosti perkolyatsionnykh klasterov na kvadratnykh reshetkakh s (1, π)-okrestnostyu [Estimates of the threshold and power of percolation clusters on square lattices with a (1, π) neighborhood]. *Kompyuternye issledovaniya i modelirovanie* [Computer Research and Modeling], 2014, vol. 6, no. 3, pp. 405–414. DOI: 10.20537/2076-7633-2014-6-3-405-414.

6. Moskalev P. V. *Perkolyatsionnoe modelirovanie poristykh struktur* [Percolation modeling of porous structures]. Moscow, 2018. 240 p.

7. Moskalev P. V. Struktura modeley perkolyatsii uzlov na trekhmernykh kvadratnykh reshetkakh [The structure of site percolation models on three-dimensional square lattices]. *Kompyuternye issledovaniya i modelirovanie* [Computer Research and Modeling], 2013, vol. 5, no. 4, pp. 607–622. DOI: 10.20537/2076-7633-2013-5-4-607-622.

8. Selyakov V. I., Kadet V. V. *Perkolyatsionnye modeli perenosa v mikroneodnorodnykh sredakh* [Percolation Models for Transport in Porous Media]. Moscow, Nedra Publ., 1995. 224 p.

9. Efros A. L. *Fizika i geometriya besporyadka* [Physics and Geometry of Disorder]. Moscow, Nauka. Glavnaya redaktsiya fiziko-matematicheskoy literatury Publ., 1982. 176 p.

10. Binder K., Heermann D. W. *Monte Carlo simulation in statistical physics: an introduction*. 4th ed. Berlin; New York, Springer, 2002.

11. Diestel R. *Graph Theory*. New York, Springer, 2000.

12. Feng X., Deng Y., Blöte H. W. J. Percolation transitions in two dimensions. *Physical Review E*, 2008, vol. 78, 031136. DOI: 10.1103/PhysRevE.78.031136.

13. *G2ii2g. Backbone-flooding/output40.txt.* Available at: https://github.com/G2ii2g/backbone-flooding/blob/master/output40.txt.

14. *G2ii2g. Backbone-flooding/test40.txt.* Available at: https://github.com/G2ii2g/backbone-flooding/blob/master/test40.txt.

15. Gould H., Tobochnik J. *An Introduction to Computer Simulation Methods: Applications to Physical Systems*. Part 2. Reading, MA, Addison-Wesley, 1987.

16. Grimmet G. *Percolation*. 2nd ed. Berlin; Heidelberg; New York; Barcelona; Hong Kong; London; Milan; Paris; Singapore; Tokyo, Springer, 1999. (Grundlehren der mathemati-schen Wissenschaften; 321).

17. Harary F. *Graph Theory*. Boca Raton, London, New York, CRC Press, 2018.

18. Heermann D. W. *Computer Simulation Methods in Theoretical Physics*. Berlin; Heidelberg; New York; London; Paris; Tokyo, Springer, 1986. 154 p.

19. Jacobsen J. L. Critical points of Potts and O(N) models from eigenvalue identities in periodic Tem-perley–Lieb algebras. *Journal of Physics A: Mathematical and Theoretical*, 2015, vol. 48, 454003. DOI: 10.1088/1751-8113/48/45/454003.

20. Kesten H. *Percolation theory for mathematicians*. Boston; Basel; Stuttgart, Birkhauser, 1982. 423 p.

21. Lee M. J. Pseudo-random-number generators and the square site percolation threshold. *Physical Review E*, 2008, vol. 78, 031131. DOI: 10.1103/PhysRevE.78.031131.

22. Mutiso R. M., Winey K. I. Electrical properties of polymer nanocomposites containing rod-like nanofillers. *Progress in Polymer Science*, 2015, vol. 40, pp. 63–84. DOI: 10.1016/j.progpolymsci.2014.06.002.

23. Ore O. *Theory of graphs.* Providence, Rhode Island: American mathematical society, 1974. 270 p.

24. Stamatovic B., Trobec R. Data parallel algorithm in finding 2-D site percolation backbones. *Proceeding of the First International Workshop on Sustainable Ultrascale Computing Systems (NESUS 2014).* Porto, Portugal; Madrid, Computer Architecture Communications, and System Group (ARCOS) University Carlos III, 2014, pp. 65–70.

25. Stauffer D., Aharony A. *Introduction to Percolation Theory.* 2nd ed., rev. London, Taylor & Francis, 2003. 180 p.

26. Tarasevich Yu. Yu., Lebovka N. I., Laptev V. V. Percolation of linear k-mers on a square lattice: From isotropic through partially ordered to completely aligned states. *Physical Review E*, 2012, vol. 86, 061116. DOI: 10.1103/PhysRevE.86.061116.

27. Tarasevich Yu. Yu., Lebovka N. I., Vodolazskaya I. V. et al. Anisotropy in electrical conductivity of two-dimensional films containing aligned nonintersecting rodlike particles: Continuous and lat-tice models. Physical Review E. 2018. Vol. 98, 012105. DOI: 10.1103/PhysRevE.98.012105.

28. Tarasevich Yu.Yu., Burmistrov A.S., Goltseva V.A., Gordeev I.I., Serbin V.I., Sizova A.A., Vodolazskaya I.V. and Zholobov D.A. Identification of current-carrying part of a random resistor network: electrical approaches vs. graph theory algorithms. *Journal of Physics: Conference Series*, 2018, vol. 955, 012021. DOI:10.1088/1742-6596/955/1/012021.

29. Toffoli T., Margolus N. *Cellular Automata Machines: A new environment for modeling.* Cambridge, Massachusetts; London, England: The MIT Press, 1987.

30. Torbert S. *Applied Computer Science.* 2nd ed. Springer, 2016. 279 p.

31. Trobec R., Stamatovic B. Analysis and classification of flow-carrying backbones in two-dimensional lattices. *Advances in Engineering Software*, 2017, vol. 103, pp. 38–45. DOI: 10.1016/j.advengsoft.2015.11.002.

32. Wilson R. J. *Introduction to Graph Theory.* 4th ed. Edinburgh Gate, Harlow, Addison Wesley Longman Limited, 1998.

33. Yang Y., Zhou S., Li Y. Square++: Making a connection game win-lose complementary and playing-fair. *Entertainment Computing*, 2013, vol. 4, p. 105–113. DOI: 10.1016/j.entcom.2012.10.004.

34. Yin W.-G., Tao R. Algorithm for finding two-dimensional site percolation backbones. *Physica B: Condensed Matter*, 2000, vol. 279, pp. 84–86. DOI: 10.1016/S0921-4526(99)00675-4.

35. Yin W.-G., Tao R. Rapid algorithm for identifying backbones in the two-dimensional percolation model. *International Journal of Modern Physics C*, 2003, vol. 14, pp. 1427–1437. DOI: 10.1142/S0129183103005509.

УДК 621.548.4

РОТОР ДАРЬЕ И ОПТИМИЗИРОВАННЫЕ ВЕТРОТУРБИНЫ: СРАВНИТЕЛЬНЫЙ АНАЛИЗ

Статья поступила в редакцию 23.02.2020, в окончательном варианте – 04.03.2020.

Роткин Владимир Михайлович, НТА «Экологический императив», 3321214, Израиль, г. Хайфа, ул. Герцль, 63 алэф,

кандидат технических наук, Ph.D, доцент, e-mail: ricentr@mail.ru

Соколовский Юлий Борисович, НТА «Экологический императив», 3321214, Израиль, г. Хайфа, ул. Герцль, 63 алэф,

кандидат технических наук, Ph.D, старший научный сотрудник, e-mail: sokol1937y@gmail.com

Ажмухамедов Искандар Маратович, Астраханский государственный университет, 414056, Российская Федерация, г. Астрахань, Татищева, 20а,

доктор технических наук, профессор, заведующий кафедрой информационной безопасности, e-mail: iskander_agm@mail.ru

Рассматриваются возможности повышения энергопоказателей эффективности вертикально-осевых ветроэнергетических установок как альтернативных источников энергии. При значительном потенциале масштабирования вертикально-осевых ветротурбин научно-техническая информация о них в открытых источниках существенно ограничена. Предлагается методология оценки энергоэффективности ортогональных роторов на основе сравнительного анализа путем сопоставления расчетных моделей различных типов роторов с эталонной оптимизированной ветротурбиной. Методология представлена на примере анализа энергетического потенциала ротора Дарье. Адекватность методологии подтверждается практикой применения базовых аэродинамических зависимостей в расчетных моделях как в ветроэнергетике, так и в смежных отраслях. Модель взаимодействия воздушного потока с крыльчатой лопастью основана на допущении о суперпозиции лобовых сил и подъемной силы. Энергия, извлекаемая из взаимодействующего с лопастью воздушного потока, равна работе приложенных к лопасти сил на перемещении этой лопасти за полный оборот турбины. Исследование этой зависимости на экстремум даёт оптимальную конфигурацию лопастей. Расчетная модель