УДК 004.722.23 DOI: 10.17587/it.26.22-29

**А. Ю. Романов**, канд. техн. наук, доц., e-mail: a.romanov@hse.ru,
**М. В. Сидоренко**, студент, e-mail: mvsidorenko@edu.hse.ru,
Национальный исследовательский университет "Высшая школа экономики", г. Москва,
**Э. А. Монахова**, канд. техн. наук, доц., ст. науч. сотр., e-mail: emilia@rav.sscc.ru,
Институт вычислительной математики и математической геофизики СО РАН, г. Новосибирск

# Маршрутизация в сетях-на-кристалле с топологией трехмерный циркулянт

*Представлена реализация динамического алгоритма маршрутизации, предназначенного для использования в сетях-на-кристалле с топологией трехмерный циркулянт (размерности 3). По сравнению с классическими алгоритмами A* или Дейкстры предложенный алгоритм не требует рассчитывать весь путь прохождения пакета, а проводит расчет номера порта, в который надо направить пакет, чтобы он гарантированно достиг узла назначения. Алгоритм может быть реализован в виде цифрового автомата для выбора маршрута, что позволяет значительно упростить структуру маршрутизаторов в сетях-на-кристалле.*

***Ключевые слова:*** *сеть-на-кристалле, алгоритм Дейкстры, циркулянт размерности 3, трехмерный циркулянт, алгоритм маршрутизации в трехмерных циркулянтах*

## Введение

В настоящее время одним из важнейших направлений исследований в области информатики и вычислительных систем является построение многоядерных процессоров. Переход к многоядерным процессорам позволяет преодолеть снижение производительности при проектировании все более сложных одноядерных систем [1]. В условиях растущего интереса к технологиям построения систем-на-кристалле (System-on-Chip, SoC) и мультипроцессорных систем-на-кристалле (Multi-Processor System-on-Chip, MPSoC) [2—4] приобретают широкое распространение сети-на-кристалле (Network-on-Chip, NoC, СтнК) [5—7].

Одной из актуальных проблем в исследовании СтнК является поиск оптимальных топологий, поскольку классические регулярные топологии (mesh [8], torus [9], hypercube [10], spidergon [11]) не удовлетворяют современным требованиям к сетям на кристалле, особенно с увеличением числа узлов [12]. К таким требованиям можно отнести высокую масштабируемость и независимое от топологии число узлов сети (число узлов не обязательно должно являться степенью какого-либо числа, либо простым числом, как, например, для топологий mesh и torus). Попытка сохранения основных характеристик таких топологий приводит к большим затратам ресурсов. Сравнительно недавно в ряде работ [13, 14] было предложено для реализации сетей-на-кристалле использовать топологии циркулянтов. Достичь приемлемых значений приведенных выше параметров для циркулянтов с тремя образующими позволяет их высокая степень связности и относительно небольшое среднее расстояние между узлами [15, 16].

Циркулянт — это неориентированный граф, состоящий из множества вершин и множества образующих. Образующие циркулянта — это фиксированные числа, составляющие конфигурацию графа. По ребрам, соответствующим образующим, осуществляется переход из одной вершины в другую, и таким образом формируется маршрут из начальной вершины в конечную. Циркулянтные топологии имеют лучшие характеристики по сравнению со стандартными топологиями, например, гиперкубами: они обладают показателями лучшей структурной живучести, надежности и связности, а также требуют меньшего числа межпроцессорных обменов при решении вычислительных задач и задач системного управления [14]. Данные ха-

рактеристики обусловлены структурой самого циркулянтного графа и его симметричностью. Это позволяет использовать циркулянты в сетях с большим числом узлов, насчитывающим десятки и сотни вычислительных узлов, что уже сейчас с появлением систем с 48, 80 и больше ядрами [17] является насущной необходимостью. Ряд известных семейств циркулянтов хорошо описан в научной литературе, например, рекурсивные циркулянты [18], а также их подвиды — мультипликативные циркулянты [19], кольцевые циркулянты и другие. Для некоторых типов циркулянтов (размерности 2) известны формулы для нахождения оптимальных циркулянтов, т. е. циркулянтов с минимальным диаметром (диаметр графа — это наибольшее расстояние между всеми парами вершин графа), для других необходима разработка программных средств для их поиска. Применительно к топологии сети граф должен быть оптимальным, чтобы маршруты из одной вершины в другую имели меньшую длину. Само понятие "оптимального графа" определяется с точки зрения реализации структуры графа.

Для применения циркулянтов в качестве топологий для сетей-на-кристалле важно разработать алгоритмы маршрутизации в них, учитывающие особенности данных семейств графов и организации сетей на кристалле. В связи с тем что топология циркулянтов обладает приведенными выше характеристиками, значительно превосходящими характеристики графов с другими топологиями, можно предположить, что их использование будет более эффективным, чем остальных, однако для более точной оценки преимущества таких графов необходимо разработать и провести анализ алгоритмов маршрутизации в таких графах применительно к сетям-на-кристалле.

## 1. Циркулянты с тремя образующими

Циркулянтной сетью называется неориентированный граф $C(N; s_1, s_2, ..., s_k)$, где $1 \leqslant s_1 < s_2 < ... < s_k < N$ — целые числа, имеет $N$ вершин, перенумерованных числами 0, 1, ..., $N - 1$, и каждая вершина с номером $i$ связана ребрами с $2k$ вершинами с номерами $i + s_j (\mathrm{mod} N)$ и $i - s_j (\mathrm{mod} N)$ для всех $j$, $1 \leqslant j \leqslant k$. Числа $s_1$, $s_2$, ..., $s_k$ называют образующими циркулянтного графа, число его образующих — размерностью графа. Размерность графа равна полустепени его вершин. Здесь и далее рассматриваются циркулянтные графы четной степени. На рис. 1 (см. третью сторону обложки) изображены примеры циркулянтов размерности три.

Основной характеристикой циркулянтного графа является его диаметр. Диаметром графа $C$ называется число $d(C) = \max\limits_{i,j \in V} \mathrm{len}(i, j)$, где $\mathrm{len}(i, j)$ — длина наименьшего пути из вершины $i$ в вершину $j$ графа $C$; $V = \{1, ..., N\}$ — множество вершин графа [20]. В связи с необходимостью минимизации данной характеристики существует фундаментальная проблема теории графов — синтез оптимальных графов, т. е. таких графов, диаметр которых минимален.

В настоящей работе рассматриваются неориентированные циркулянты типа $C(N; s_1, s_2, s_3)$. Они имеют три типа ребер ($s_1$, $s_2$ и $s_3$). Из каждой вершины выходит по паре ребер каждого типа, симметричных относительно линии, разделяющей граф пополам из вершины, откуда исходит ребро (линия является биссектрисой угла, образуемого парой самых длинных образующих). Примерами циркулянтов такого типа могут служить графы, приведенные на рис. 1.

## 2. Разработка алгоритма маршрутизации для сетей-на-кристалле с топологией трехмерный циркулянт

Как было отмечено ранее, необходима разработка алгоритма маршрутизации в трехмерных циркулянтах. Типичный подход — использовать алгоритмы, подобные алгоритму Дейкстры [21]. В таком случае на каждом узле на каждом шаге пакета надо будет рассчитывать все возможные пути для достижения узла назначения, хранить таблицы маршрутизации в узлах или путь в пакете, предварительно его рассчитав. Все эти способы достаточно ресурсозатратны, поэтому необходим алгоритм, который бы позволил на каждом узле вычислить следующий шаг для движения пакета на пути в узел назначения.

Рассмотрим циркулянт типа $C(N; s_1, s_2, s_3)$. Пусть $s_1$, $s_2$, $s_3$ — образующие, упорядоченные по возрастанию, причем если хотя бы для одного $s_i$ ($i = 1, ..., 3$) не выполняется неравенство $s_i < [N/2]$, где квадратные скобки обозначают целую часть от деления, тогда, если $N - s_i \geqslant 0$, то значение образующей становится равной $s_i = N - s_i$.

Из любой вершины графа можно перейти в шесть других вершин по ребрам, соответствующим значениям образующих $s_1$, $s_2$, $s_3$, как в направлении часовой стрелки, так и в направлении против часовой стрелки (рис. 2, см. тре-

тью сторону обложки), пройдя по $s_1$, $s_2$, $s_3$ и $-s_1$, $-s_2$, $-s_3$ соответственно.

Таким образом, при переходе в следующую вершину по одному из ребер, соответствующих образующим, из начальной вершины для поиска дальнейшего пути становятся доступными только пять образующих, так как по одному ребру уже был осуществлен переход в текущую вершину.

Пусть далее $N_1$ — текущая вершина, $N_2$ — конечная. Для того, чтобы определить, по какому из ребер двигаться дальше, в алгоритме при каждом изменении вершины рассчитываются четыре расстояния:

$$L_1 = \min(|N_2 - N_1|, N - |N_2 - N_1|);$$
$$L_2 = \max(|N_2 - N_1|, N - |N_2 - N_1|) = N - L_1;$$
$$L_3 = N + L_1;$$
$$L_4 = N + L_2,$$

где $L_1$ — минимальное расстояние от $N_1$ до $N_2$; $L_2$ — максимальное расстояние от $N_1$ до $N_2$ (при движении в другую сторону); $L_3$ — расстояние $L_1$ после полного обхода (цикла) вокруг графа в сторону наименьшего расстояния от $N_1$ до $N_2$; $L_4$ — расстояние $L_2$ после полного обхода вокруг графа в сторону наибольшего расстояния от $N_1$ до $N_2$.

Если одно из четырех расстояний от каждой из пяти вершин (в начальной вершине $N_1$ — из шести), в которые можно перейти по ребрам единичной длины, соответствующим образующим единичной длины, делится на одну из образующих нацело, то устанавливается "приоритет" частному — "high"; если расстояние делится на сумму образующих (некоторые члены суммы могут не входить в сумму, например $s_2 + s_1$, или быть со знаком минус, например $s_1 - s_2 + s_3$), то устанавливается приоритет частному "low". Само частное умножается на 2 или на 3 для случаев суммы двух или трех элементов, так как потребуется в 2 или 3 раза больше шагов алгоритма соответственно.

Среди всех частных выбирается минимальное и сохраняется в паре с соответствующей ему образующей. Среди всех рассчитанных пар выбирается пара с наивысшим приоритетом, которой для достижения конечной вершины $N_2$ требуется минимальное число шагов. В данной паре результатом является образующая $s_i$, взятая со знаком плюс или минус в зависимости от того, по какому из ребер был осуществлен переход в текущую вершину.

Для того чтобы избежать зацикливания, в алгоритме проверяется вхождение текущей вершины в список уже "посещенных" вершин. Если вершина входит в данный список, следующий шаг происходит уже по одной из тех образующих, которые не равны текущей.

На основе приведенных выше утверждений разработан следующий алгоритм:

**algorithm Find_Route_Sequent** is

**Input:** $N_1$ — start node, $N_2$ — end node, $N$ — count of nodes, $s_1$ — first generatrix, $s_2$ — second generatrix, $s_3$ — third generatrix, *previosStep* — last algorithm step (0 by default), *theFirst* — start vertex in the graph.

**Output:** $N_1$ — next start node.

```
1:  sList ← [s1, s2, s3, −s1, −s2, −s3]
2:  removeFlag ← false
3:  if −previousStep in sList then
4:  removePosition ← sList.IndexOf(−previousStep)
5:  delete sList[−previousStep]
6:  removeFlag ← true
7:  x ← [N1 + s1, N1 + s2, N1 + s3, N1 − s1, N1 − s2, N1 − s3]
8:  if removeFlag then
9:  delete x[removePosition]
10: distances ← [ ]
11: for i ← 0; i < x.Count; i ← i + 1 do
12: if x[i] ≤ 0 then
13: x[i] ← x[i] + N
14: if x[i] > N then
15: x[i] ← x[i] mod N
16: distances ← [
17: min(|N2 − N1|, N − |N2 − N1|),
18: max(|N2 − N1|, N − |N2 − N1|),
19: N + min(|N2 − N1|, N − |N2 − N1|),
20: N + max(|N2 − N1|, N − |N2 − N1|)
21: ]
22: if theFirst in x then
23: removeTfPos ← x.IndexOf(theFirst)
24: delete sList[removeTfPos]
25: delete x[removeTfPos]
26:
27: minDistances ← [ ], minLengths ← [ ]
28: for i ← 0; i < distances.Count; i ← i + 1 do
29: minDistances.Clear()
30: d ← distances[i]
31: for idx ← 0; idx < d.Count; idx ← idx + 1 do
32: if d[idx] mod s3 = 0 then
33:         minDistances.Add(d[idx] div s3, sList[i], "high")
34: if d[idx] mod s2 = 0 then
35:         minDistances.Add(d[idx] div s2, sList[i], "high")
36: if d[idx] mod s1 = 0 then
37:         minDistances.Add(d[idx] div s1, sList[i], "high")
38: if d[idx] mod (s1 + s2) = 0 then
39:         minDistances.Add(d[idx] div(s1 + s2)*2, sList[i], "low")
40: if d[idx] mod (s1 + s3) = 0 then
```

```
        minDistances.Add(d[idx] div(s1 + s3)*2, sList[i], "low")
if d[idx] mod (s2 + s3) = 0 then
        minDistances.Add(d[idx] div(s2 + s3)*2, sList[i], "low")
if d[idx] mod (s2 − s1) = 0 . then
        minDistances.Add(d[idx] div(s2 − s1)*2, sList[i], "low")
if d[idx] mod (s3 − s2) = 0 then
        minDistances.Add(d[idx] div(s3 − s2)*2, sList[i], "low")
if d[idx] mod (s3 − s1) = 0 then
        minDistances.Add(d[idx] div(s3 − s1)*2, sList[i], "low")
if d[idx] mod (s1 + s2 + s3) = 0 then
        minDistances.Add(d[idx] div(s1 + s2 + s3)*3, sList[i], "low")
if d[idx] mod (s1 − s2 + s3) = 0 then
        minDistances.Add(d[idx] div(s1 − s2 + s3)*3, sList[i], "low")
if d[idx] mod (−s1 + s2 + s3) = 0 then
        minDistances.Add(d[idx] div(−s1 + s2 + s3)*3, sList[i], "low")
if s1 + s2 − s3 > 0 and d[idx] mod (s1 + s2 − s3) = 0 then
        minDistances.Add(d[idx] div(s1 + s2 − s3)*3, sList[i], "low")
if −s1 − s2 + s3 > 0 and d[idx] mod (−s1 − s2 + s3) = 0 then
        minDistances.Add(d[idx] div(−s1 − s2 + s3)*3, sList[i], "low")
minLenghts.Add(GetMinLength(minDistances))
resultGen ← min(minLengths, argument = 0)[1]
if resultGen = 0 then
    if prevStep > 0 then
            if s3 in sList then
            resultGen ← s3
    else if s2 in sList then
            resultGen ← s2
    else if s1 in sList then
            resultGen ← s1
else
    if −s3 in sList then
            resultGen ← −s3
    else if −s2 in sList then
            resultGen ← −s2
    else if −s1 in sList then
            resultGen ← −s1
return resultGen
```

**function GetMinLength** is

**Input:** *minLengths* — list of triples: (division_result, generatrix, priority).

**Output:** $(a,b)$ — tuple with minimal steps to $N_2$ $(a)$ and with corresponding generatrix $(b)$.

```
minTHigh ← 10e5
minTLow ← 10e5
secArgHigh ← 0
secArgLow ← 0
for i ← 0; i < minLengths; i ← i + 1 do
    if minLengths[i].priority = "high" and
     minLengths[i].divisionResult < minTHigh then
            minTHigh ← minLengths[i].divisionResult
            secArgHigh ← minLengths[i].generator
    else
            minTLow ← minLengths[i].divisionResult
            secArgLow ← minLengths[i].generator
if minTHigh = 10e5 or minTLow < minTHigh then
    return (minTLow, secArgLow)
return (minTHigh, secArgHigh)
```

## 3. Тестирование разработанного алгоритма маршрутизации

Разработанному алгоритму присвоено название "Sequent". Для тестирования данного алгоритма был сгенерирован набор оптимальных циркулянтов (графов с минимальным диаметром) размерности 3 для различного числа вершин (9, 15, 16, 21 23, 25 36, 49, 64, 81 и 100) с помощью алгоритма Дейкстры [22]. Зависимость диаметра от числа вершин в циркулянтах из выборки [23] приведена на рис. 3.

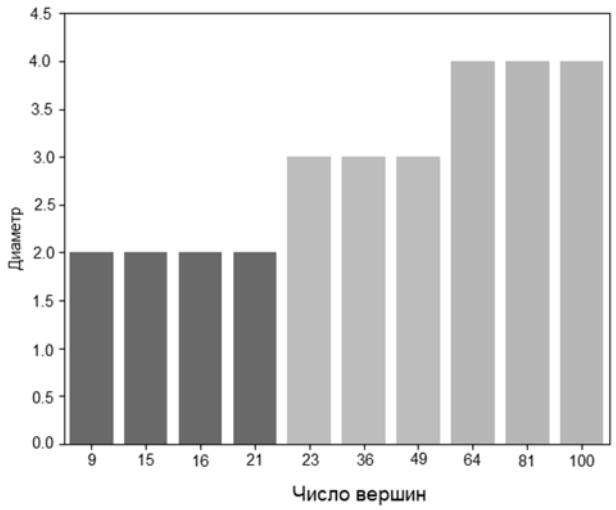



**Рис. 3. Зависимость диаметра от числа вершин графов из выборки оптимальных циркулянтов**

## 4. Оценка эффективности алгоритма "Sequent"

С использованием программных средств для синтеза циркулянтных топологий из работы [24] было сгенерировано 416 описаний оптимальных циркулянтов [23], которые были использованы для тестирования алгоритма "Sequent". На рис. 4 представлен график разностей максимальных расстояний для разработанного алгоритма с диаметром, рассчитанным по алгоритму Дейкстры [22]. Для всех отсутствующих на графике циркулянтов разность равна нулю.

Из графика следует, что в 20 из 416 тестов максимальное расстояние между узлами для алгоритма "Sequent" отличается от диаметра, а на 396 тестах — совпадает. Более того, среди

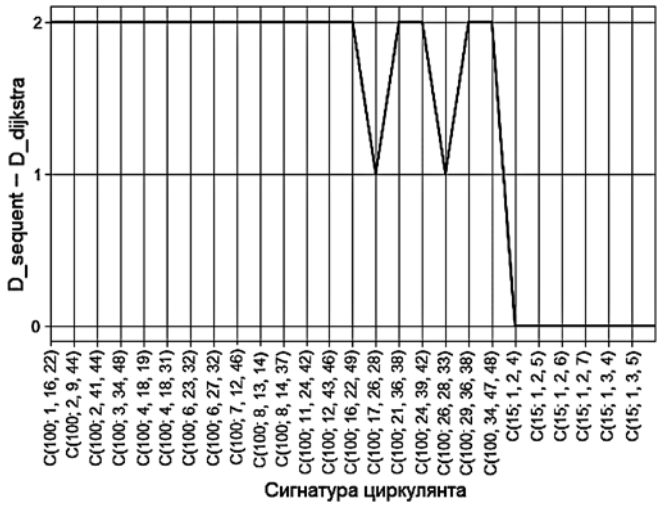



**Рис. 4. График разностей максимальных расстояний для алгоритмов "Sequent" и Дейкстры (тест 1)**

20 случаев разница между диаметром и рассчитанным расстоянием не превышает единицы в двух случаях и равна двум — в 18 случаях. Таким образом, эффективность алгоритма можно вычислить по формуле:

$$Alg\_efficiency = \frac{\sum_{i=1}^{416} D_{i\,Dijkstra}}{\sum_{i=1}^{416} D_{i\,Sequent}},$$

где $D_{i_{Dijkstra}}$ — диаметр $i$-го циркулянта, рассчитанного по алгоритму Дейкстры; $D_{i_{Sequent}}$ — максимальное расстояние между узлами для $i$-го циркулянта, рассчитанное по алгоритму "Sequent".

Эффективность разработанного алгоритма на основе приведенного выше набора оптимальных циркулянтов равна

$$Alg\_efficiency = \frac{1351}{1389} \approx 0,973.$$

Далее алгоритм "Sequent" был запущен для 296 циркулянтов, приведенных в работе [23]. Они распределены так, что для каждого числа вершин берется по одному оптимальному циркулянту, при этом у 96 % циркулянтов $s_1 = 1$, т. е. они являются кольцевыми. График разностей максимальных расстояний с диаметрами для данного теста приведен на рис. 5, *а* (см. третью сторону обложки).

Из графика следует, что для 209 из 296 (70 %) циркулянтов алгоритм "Sequent" дает максимальное расстояние, отличное от диаметра. Изменение разности максимального расстояния и диаметра находится в следующих пределах: 10 — для порядков графов от 5 до 154; от 3 до 20 — для порядков от 155 до 250; от 4 до 40 — для порядков от 251 до 300. Полученные в результате тестирования значения могут отличаться при повторении эксперимента из-за особенностей алгоритма. Это может происходить из-за того, что при попадании текущей вершины в список посещенных вершин, для устранения зацикливания предусмотрен переход по одному из трех ребер, отличных от того, по которому уже осуществлялся переход.

Затем алгоритм "Sequent" был реализован на выборке из 28 299 циркулянтов из набора, представленного в [23]. Результат приведен на рис. 5, *б* (см. третью сторону обложки). Исходя из полученных результатов можно сделать вывод о том, что в циркулянтах, имеющих число вершин от 300 и выше, расхождения между алгоритмом Дейкстры и алгоритмом "Sequent" становятся значительными: в 99 % случаях диаметр графа не совпадает с максимальным расстоянием, рассчитанным по разработанному алгоритму "Sequent", а среднее значение расхождения составляет 154.

Таким образом, применение данного алгоритма для сетей на кристалле с числом узлов больше 300 неэффективно. Тем не менее, порог в 300 узлов вполне достаточен для современных сетей на кристалле, где число узлов обычно не превышает 100 [17].

## 5. Анализ конфигураций циркулянтов, при которых алгоритм "Sequent" работает неэффективно

Анализируя устройство сетей-на-кристалле с циркулянтной топологией, можно обнаружить такие сигнатуры циркулянтов, при которых маршрут из одной вершины в другую будет слишком длинным. К таким сигнатурам относят следующий тип циркулянтов:

$$C(N;\ s_1,\ s_2,\ s_3),\ \text{при четном } N \text{ и НОД } (s_1,\ s_2,\ s_3) \neq 1,$$

где НОД $(s_1, s_2, s_3)$ — наибольший общий делитель чисел $s_1, s_2, s_3$ [25].

Также возникают случаи, когда путь существует, но является большим (в десятки или сотни раз выше оптимального). К данным сигнатурам относятся следующие типы циркулянтов:

1) $C(N; s_1, s_2, s_3)$ при $s_2 = 2s_1$;

2) $C(N; s_1, s_2, s_3)$ при $s_2 - s_1 = s_3 - s_2$ и $N \bmod s_2 = 0$.

Сети-на-кристалле с таким набором вершин и образующих будут заметно менее эффектив-

ны, чем сети с циркулянтами, приведенными в работе [23].

Для наборов графов тестов 2 и 3 был проведен анализ на наличие сигнатур, для которых алгоритм "Sequent" имеет циклы и недостижимые пути. Так, для теста 2 среди общего числа циркулянтов обнаружено 2 % сигнатур, при которых алгоритм возвращает длинные маршруты, а недостижимые пути отсутствуют. В тесте 3 с числом вершин от 300 были обнаружены 285 графов с недостижимыми путями (1 % от размера выборки).

Таким образом, для реализации сетей-на-кристалле необходимо учитывать условия появления недостижимых маршрутов. Более того, для повышения эффективности работы алгоритма следует исключать длинные маршруты, применяя соответствующие условия.

## 6. Оценка времени выполнения алгоритма "Sequent"

Рассмотрим зависимости времени выполнения и максимального расстояния между узлами, рассчитанного по алгоритму "Sequent", от числа вершин для различных циркулянтов. Все вычисления времени выполнения алгоритма проводили на 64-разрядном компьютере с тактовой частотой 2,5ГГц с процессором Intel Core i7 и с 8 Гб оперативной памяти (рис. 6, см. третью сторону обложки).

На основе графиков на рис. 6 можно сделать вывод о том, что время выполнения алгоритма зависит от числа вершин нелинейно. Эта зависимость близка к квадратичной.

Несколько другая ситуация с максимальным расстоянием между узлами (рис. 7, см. третью сторону обложки).

Из рис. 7 следует, что зависимость максимального расстояния графа от числа вершин при увеличении числа вершин линейная. Но это не всегда так, поскольку для различных образующих при малых $N$ зависимость отличается. Сначала она выглядит как ступенчатая функция (рис. 8, *а*, см. четвертую сторону обложки), а затем вырождается в линейную при больших $N$ (рис. 8, *б*, см. четвертую сторону обложки).

## 7. Оценка сложности алгоритма "Sequent"

На каждой итерации алгоритм совершает расчет четырех расстояний для пяти вершин, в которые можно перейти по одному из пяти ребер, соответствующих образующим (для первой итерации — шесть вершин). Как было показано выше, зависимость $D(N)$ — линейная при больших $N > 100$, где $D$ — диаметр графа. Это означает, что диаметр $D$ можно заменить на выражение $kN$, где $k$ — коэффициент, причем $0 < k < 1$. Таким образом, для одной итерации сложность алгоритма [26] составляет $O(20kN)$, что равносильно $O(N)$. Для $N$ таких итераций сложность алгоритма равна $O(N^2)$, что не превосходит сложность вычислений по алгоритму Дейкстры [22].

В разработанном алгоритме не требуется хранить большие таблицы с данными, следовательно, использование памяти в нем минимальное.

## Заключение

Разработанный алгоритм для общего случая оптимального циркулянта размерности 3 был протестирован на специально сгенерированном наборе данных из 416 оптимальных циркулянтов, и была рассчитана его эффективность. Для данного набора она равна 0,973, что является достаточным показателем эффективности алгоритма для задачи реализации алгоритма на HDL на уровне маршрутизатора сети-на-кристалле, поскольку данный алгоритм имеет линейную сложность и может быть легко описан в виде цифрового автомата.

В 95 % рассмотренных случаев для набора из графов с числом вершин, меньшим 300, алгоритм показал результат, аналогичный результату, полученному с помощью алгоритма Дейкстры. Вычислительная временная сложность разработанного алгоритма совпадает со сложностью алгоритма Дейкстры, который считается эталонным при нахождении маршрутов в сетях-на-кристалле. При числе вершин больше 300 алгоритм становится неэффективным, так как максимальное расстояние между узлами, рассчитанное по алгоритму "Sequent", может в десятки раз превышать диаметр, рассчитанный по алгоритму Дейкстры.

По сравнению с классическими алгоритмами предложенный алгоритм не требует рассчитывать весь путь прохождения пакета, а определяет номер порта, в который надо направить пакет, чтобы он гарантированно достиг узла назначения. Это позволяет значительно упростить структуру маршрутизатора СтнК.

*Публикация подготовлена в ходе проведения исследования (№ 18-01-0074) в рамках Про-*

*граммы "Научный фонд Национального исследовательского университета "Высшая школа экономики" (НИУ ВШЭ)" в 2018—2019 гг. и в рамках государственной поддержки ведущих университетов Российской Федерации "5-100".*

*Исследование Монаховой Э. А. выполнено в рамках проекта ИВМиМГ СО РАН 0315-2016-0006.*

**Список литературы**

1. **Nychis G.** et al. Next generation on-chip networks: What kind of congestion control do we need? // Proc. 9th ACM SIGCOMM Work. Hot Top. Networks. 2010. P. 1—6.

2. **Paul J.** et al. Resource-awareness on heterogeneous MPSoCs for image processing // J. Syst. Archit. 2015. Vol. 61, N. 10. P. 668—680.

3. **Wolf W., Jerraya A. A., Martin G.** Multiprocessor system-on-chip (MPSoC) technology // IEEE Trans. Comput. Des. Integr. Circuits Syst. 2008. Vol. 27, N. 10. P. 1701—1713.

4. **Phanibhushana B., Kundu S.** Network-on-chip design for heterogeneous multiprocessor system-on-chip // Proceedings of IEEE Computer Society Annual Symposium on VLSI, ISVLSI. 2014. P. 486—491.

5. **Abdelfattah M. S., Bitar A., Betz V.** Design and Applications for Embedded Networks-on-Chip on FPGAs // IEEE Trans. Comput. 2017. Vol. 66, N. 6. P. 1008—1021.

6. **Benini L., De Micheli G.** Networks on Chip // Networks on Chips. 2006.

7. **Jantsch A.** Models of computation for networks on chip // Proceedings — International Conference on Application of Concurrency to System Design, ACSD. 2006.

8. **Deb D.** et al. Cost effective routing techniques in 2D mesh NoC using on-chip transmission lines // J. Parallel Distrib. Comput. Academic Press, 2019. Vol. 123. P. 118—129.

9. **Ansari A. Q., Ansari M. R., Khan M. A.** Modified quadrant-based routing algorithm for 3D Torus Network-on-Chip architecture // Perspect. Sci. 2016. Vol. 8. P. 718—721.

10. **Marvasti M. B., Szymanski T. H.** The performance of hypermesh NoCs in FPGAs // IEEE International Conference on Computer Design: VLSI in Computers and Processors. 2012. P. 492—493.

11. **Bishnoi R.** et al. Distributed adaptive routing for spidergon NoC // 18th International Symposium on VLSI Design and Test, VDAT 2014. 2014. P. 1—6.

12. **Dally W. J., Towles B. P.** Principles and Practices of Interconnection Networks. Elsevier, 2003. 581 p.

13. **Boesch F. T., Jhing-Fa Wang, Wang J. F.** Reliable Circulant Networks with Minimum Transmission Delay // IEEE Transactions on Circuits and Systems. 1985. Vol. 32, N. 12. P. 1286—1291.

14. **Monakhova E. A.** A Survey on Undirected Circulant Graphs // Discret. Math. Algorithms Appl. World Scientific Publishing Company, 2012. Vol. 4, N. 1. P. 1250002.

15. **Kotsis G., Awiuni B. A.** Interconnection Topologies for Parallel Processing Systems // PARS Mitteilungen. 1993. Vol. 11. P. 1—6.

16. **Lu R.** Fast methods for designing circulant network topology with high connectivity and survivability // J. Cloud Comput. Journal of Cloud Computing, 2016. Vol. 5, N. 5.

17. **Comprés Ureña I. A., Riepen M., Konow M.** RCKMPI — Lightweight MPI implementation for intel's single-chip cloud computer (SCC) // Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Springer, Berlin, Heidelberg, 2011. Vol. 6960 LNCS. P. 208—217.

18. **Park J.-H., Chwa K.-Y.** Recursive Circulant: A New Topology for Multicomputer Networks // International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN 1994). 1994. P. 73—80.

19. **Stojmenović I.** Multiplicative circulant networks topological properties and communication algorithms // Discret. Appl. Math. 1997. Vol. 77, N. 3. P. 281—305.

20. **Chung F. R. K.** Diameters and Eigenvalues // J. Am. Math. Soc. 2006.

21. **Dijkstra E. W.** A note on two problems in connexion with graphs // Numer. Math. 1959. Vol. 1. P. 269—271.

22. **Deng Y.** et al. Fuzzy Dijkstra algorithm for shortest path problem under uncertain environment // Appl. Soft Comput. J. 2012. Vol. 12, N. 3. P. 1231—1237.

23. **Romanov A. Y.** Optimal Circulants Dataset [Electronic resource]. URL: https://github.com/RomeoMe5/circulantGraphs/ (accessed: 21.03.2019).

24. **Romanov A. Y., Romanova I. I., Glukhikh A. Y.** Development of a Universal Adaptive Fast Algorithm for the Synthesis of Circulant Topologies for Networks-on-Chip Implementations // 2018 IEEE 38th International Scientific Conference on Electronics and Nanotechnology, ELNANO 2018. 2018. P. 110—115.

25. **Eisenbrand F.** The euclidean algorithm // Algorithms Unplugged. 2011.

26. **Fortnow L., Homer S.** Computational Complexity // Handbook of the History of Logic. 2014.

**A. Yu. Romanov,** PhD, Associate Professor, e-mail: a.romanov@hse.ru,
**M. V. Sidorenko,** Student, e-mail: mvsidorenko@edu.hse.ru,
National Research University Higher School of Economics, Moscow, Russian Federation,
**E. A. Monakhova,** PhD, Associate Professor, Senior Researcher, e-mail: emilia@rav.sscc.ru,
ICMMG SB RAS, Novosibirsk, Russian Federation

# Routing in Networks-on-Chip with a Three-Dimensional Circulant Topology

*The paper presents the implementation of a dynamic routing algorithm intended for use in networks-on-chip with a three-dimensional circulant topology of type C(N; s1, s2, s3). Compared with the classical algorithms A* or Dijkstra, the proposed algorithm does not require to calculate the entire path of the packet, but calculates the port number to which the packet should be sent so that it can reach the destination node. This makes it possible to significantly simplify the structure of the NoC router.*

*The algorithm can be implemented as RTL state machine in routers for NoCs for finding the shortest routes between any two network nodes. The proposed algorithm was tested on sets of optimal circulants. For this set, it is equal to 0.973 which is a sufficient indicator of efficiency of the algorithm for the task of implementing the algorithm in HDL at the level of a network-on-chip router, since this algorithm has linear complexity and can be easily implemented. In 95 % of the cases considered, for a set of graphs with the number of vertices less than 300, the algorithm showed a result similar to that obtained using the Dijkstra algorithm. The computational time complexity of the created algorithm is similar to the complexity of the Dijkstra algorithm which is considered to be the reference when finding routes in networks-on-chip. When the number of vertices is more than 300, the algorithm becomes inefficient, since the diameters, calculated by "Sequent" algorithm, can be ten times higher than the optimal diameters calculated by the Dijkstra algorithm.*

***Keywords:*** *network-on-chip, Dijkstra algorithm, circulant of the third dimension, three-dimensional circulant, routing algorithm in three-dimensional circulants*

DOI: 10.17587/it.26.22-29

**References**

1. **Nychis G.** et al. Next generation on-chip networks: What kind of congestion control do we need?, *Proc. 9th ACM SIGCOMM Work. Hot Top. Networks*, 2010, pp. 1—6.

2. **Paul J.** et al. Resource-awareness on heterogeneous MPSoCs for image processing, *J. Syst. Archit.*, 2015, vol. 61, no. 10, pp. 668—680.

3. **Wolf W., Jerraya A. A., Martin G.** Multiprocessor system-on-chip (MPSoC) technology, I*EEE Trans. Comput. Des. Integr. Circuits Syst.,* 2008, vol. 27, no. 10, pp. 1701—1713.

4. **Phanibhushana B., Kundu S.** Network-on-chip design for heterogeneous multiprocessor system-on-chip, *Proceedings of IEEE Computer Society Annual Symposium on VLSI, ISVLSI,* 2014, pp. 486—491.

5. **Abdelfattah M. S., Bitar A., Betz V.** Design and Applications for Embedded Networks-on-Chip on FPGAs, I*EEE Trans. Comput.*, 2017, vol. 66, no. 6, pp. 1008—1021.

6. **Benini L., De Micheli G.** Networks on Chip, *Networks on Chips*, 2006.

7. **Jantsch A.** Models of computation for networks on chip, *Proceedings — International Conference on Application of Concurrency to System Design*, *ACSD,* 2006.

8. **Deb D.** et al. Cost effective routing techniques in 2D mesh NoC using on-chip transmission lines, *J. Parallel Distrib. Comput. Academic Press*, 2019, vol. 123, pp. 118—129.

9. **Ansari A. Q., Ansari M. R., Khan M. A.** Modified quadrant-based routing algorithm for 3D Torus Network-on-Chip architecture, *Perspect. Sci.*, 2016, vol. 8, pp. 718—721.

10. **Marvasti M. B., Szymanski T. H.** The performance of hypermesh NoCs in FPGAs, I*EEE International Conference on Computer Design: VLSI in Computers and Processors,* 2012, pp. 492—493.

11. **Bishnoi R.** et al. Distributed adaptive routing for spidergon NoC, *18th International Symposium on VLSI Design and Test, VDAT 2014*, 2014, pp. 1—6.

12. **Dally W. J., Towles B. P.** Principles and Practices of Interconnection Networks. Elsevier, 2003, 581 p.

13. **Boesch F. T., Jhing-Fa Wang, Wang J. F.** Reliable Circulant Networks with Minimum Transmission Delay, *IEEE Transactions on Circuits and Systems,* 1985, vol. 32, no. 12, pp. 1286—1291.

14. **Monakhova E. A.** A Survey on Undirected Circulant Graphs, *Discret. Math. Algorithms Appl. World Scientific Publishing Company,* 2012, vol. 4, no. 1, pp. 1250002.

15. **Kotsis G., Awiuni B. A.** Interconnection Topologies for Parallel Processing Systems, *PARS Mitteilungen*, 1993, vol. 11, pp. 1—6.

16. **Lu R.** Fast methods for designing circulant network topology with high connectivity and survivability, *J. Cloud Comput. Journal of Cloud Computing,* 2016, vol. 5, no. 5.

17. **Comprés Ureña I. A., Riepen M., Konow M.** RCKMPI — Lightweight MPI implementation for intel's single-chip cloud computer (SCC), *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics),* Springer, Berlin, Heidelberg, 2011, vol. 6960 LNCS, pp. 208—217.

18. **Park J.-H., Chwa K.-Y.** Recursive Circulant: A New Topology for Multicomputer Networks, *International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN 1994),* 1994, pp. 73—80.

19. **Stojmenović I.** Multiplicative circulant networks topological properties and communication algorithms, *Discret. Appl. Math.,* 1997, vol. 77, no. 3, pp. 281—305.

20. **Chung F. R. K.** Diameters and Eigenvalues, *J. Am. Math. Soc.*, 2006.

21. **Dijkstra E. W.** A note on two problems in connexion with graphs, *Numer. Math.*, 1959, vol. 1, pp. 269—271.

22. **Deng Y.** et al. Fuzzy Dijkstra algorithm for shortest path problem under uncertain environment, *Appl. Soft Comput. J.*, 2012, vol. 12, no. 3, pp. 1231—1237.

23. **Romanov A. Y.** Optimal Circulants Dataset, available at: https://github.com/RomeoMe5/circulantGraphs/ (accessed: 21.03.2019).

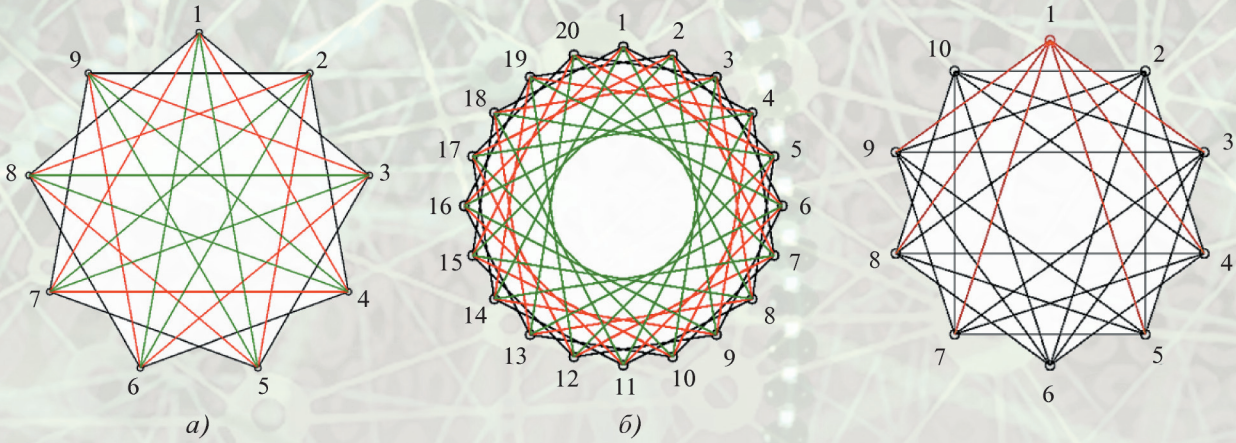
24. **Romanov A. Y., Romanova I. I., Glukhikh A. Y.** Development of a Universal Adaptive Fast Algorithm for the Synthesis of Circulant Topologies for Networks-on-Chip Implementations, *2018 IEEE 38th International Scientific Conference on Electronics and Nanotechnology, ELNANO 2018,* 2018, pp. 110—115.

25. **Eisenbrand F.** The euclidean algorithm, *Algorithms Unplugged,* 2011.

26. **Fortnow L., Homer S.** Computational Complexity, Handbook of the History of Logic, 2014.

Рисунки к статье А. Ю. Романова, М. В. Сидоренко, Э. А. Монаховой

# «МАРШРУТИЗАЦИЯ В СЕТЯХ-НА-КРИСТАЛЛЕ С ТОПОЛОГИЕЙ ТРЕХМЕРНЫЙ ЦИРКУЛЯНТ»

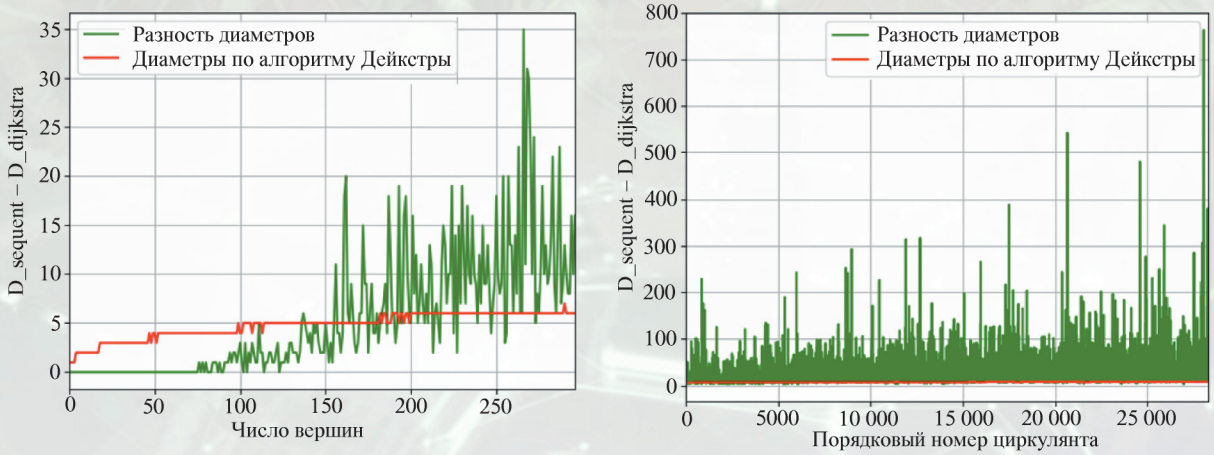



*а)* *б)*

**Рис. 1. Циркулянты с тремя образующими:** *а* – *C*(9; 2, 3, 4); *б* – *C*(20; 3, 5, 7)




**Рис. 2. Единичные маршруты из одной вершины в циркулянте *C*(10; 2, 3, 4)**




*а)* *б)*

**Рис. 5. График разностей максимальных расстояний, рассчитанных с помощью алгоритмов «Sequent» и Дейкстры:** *а* – тест 2; *б* – тест 3

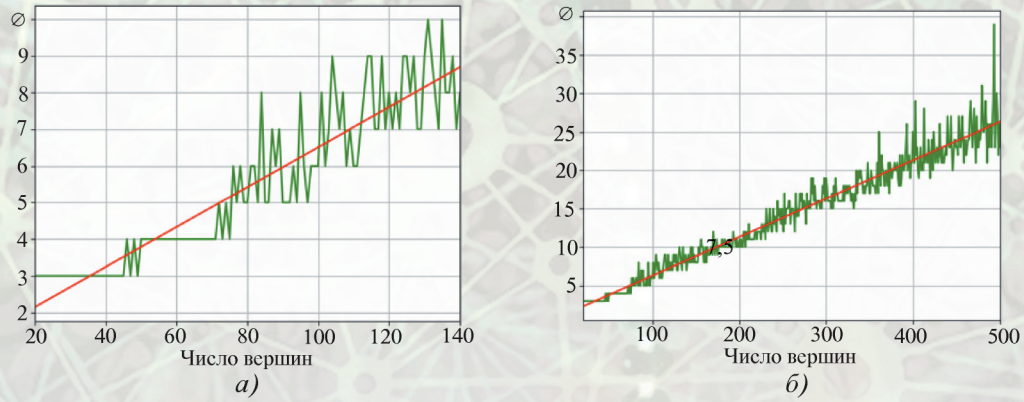

*а)* *б)*

**Рис. 6. Зависимость времени выполнения алгоритма от числа вершин в циркулянтах типа:** *а* – *C*(*N*; 2, 3, 7); *б* – *C*(*N*; 10, 17, 39)

**Рис. 7. Зависимость максимального расстояния от числа вершин для циркулянтов типа *C*(*N*; 2, 3, 7) при изменении *N* от 20 до 220**

Рисунки к статье А. Ю. Романова, М. В. Сидоренко, Э. А. Монаховой

## «МАРШРУТИЗАЦИЯ В СЕТЯХ-НА-КРИСТАЛЛЕ С ТОПОЛОГИЕЙ ТРЕХМЕРНЫЙ ЦИРКУЛЯНТ»




*а)* *б)*

**Рис. 8. Зависимость максимального расстояния циркулянта, рассчитанного по разработанному алгоритму, от числа вершин для циркулянтов типа *C*(*N*; 4, 7, 13) при изменении *N*: *а* – от 20 до 140; *б* – от 20 до 500**

Рисунки к статье А. И. Лепского

## «СРАВНИТЕЛЬНЫЙ АНАЛИЗ АЛГОРИТМОВ КЛАСТЕРИЗАЦИИ ЛЕЙКОЦИТОВ ПО FS И SS ПАРАМЕТРАМ ПРИ ЦИТОФЛУОРИМЕТРИЧЕСКОМ ИССЛЕДОВАНИИ КРОВИ»

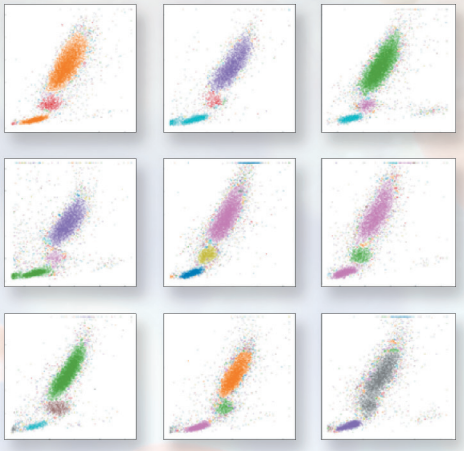


**Рис. 2. Результаты численного моделирования процесса кластеризации лейкоцитов методом одиночной связи**

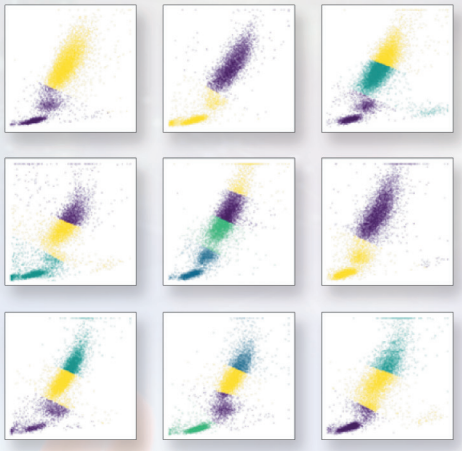


**Рис. 3. Результаты численного моделирования методом К-средних**

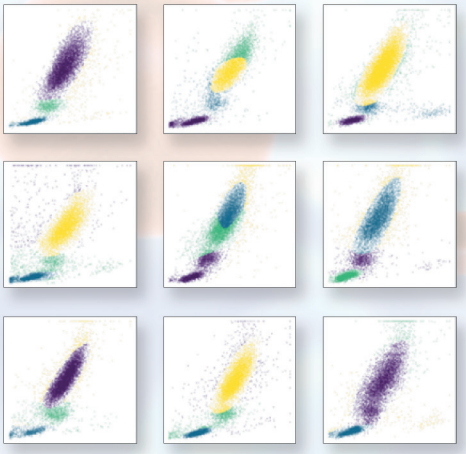


**Рис. 4. Результаты кластеризации с помощью *EM*-алгоритма**

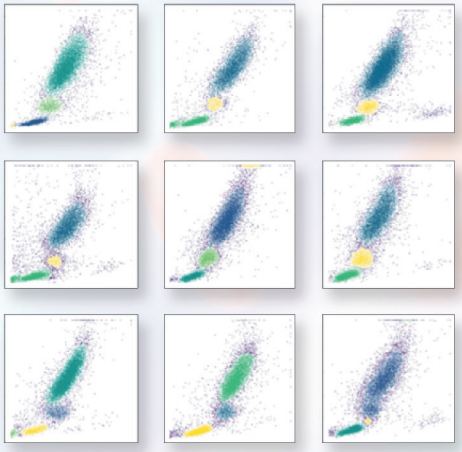
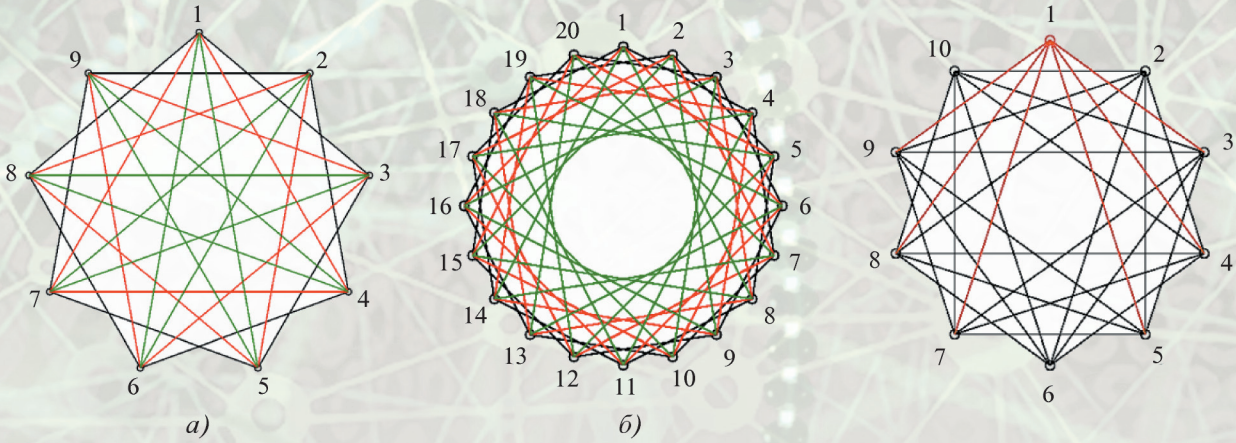


**Рис. 5. Результаты кластеризации лейкоцитов методом *DBSCAN***

ISSN 1684-6400. Информационные технологии. 2020. Том 26. № 1. 1–64. Индекс: 72656 (Объединенный каталог «Пресса России»)

Рисунки к статье А. Ю. Романова, М. В. Сидоренко, Э. А. Монаховой

# «МАРШРУТИЗАЦИЯ В СЕТЯХ-НА-КРИСТАЛЛЕ С ТОПОЛОГИЕЙ ТРЕХМЕРНЫЙ ЦИРКУЛЯНТ»

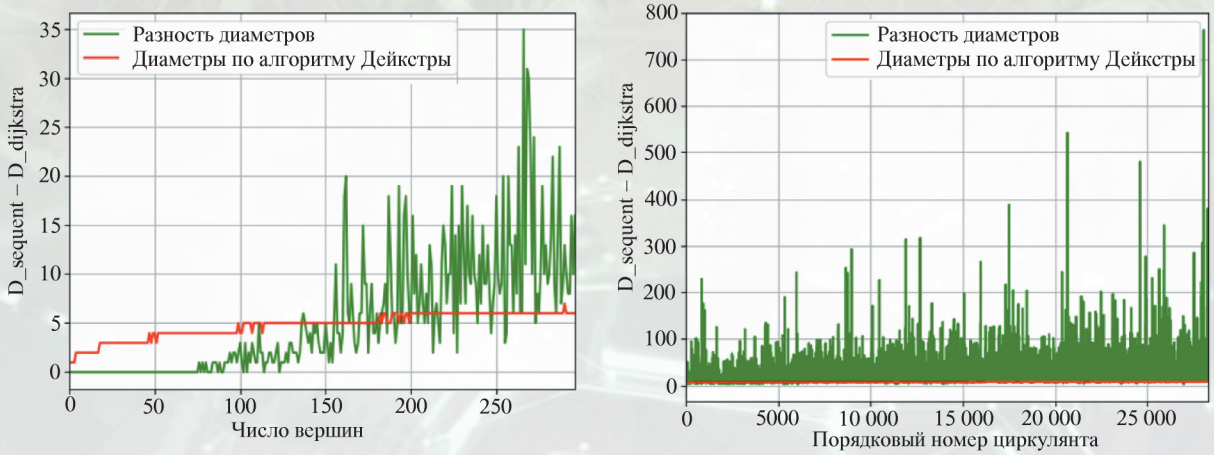



*а)* *б)*

**Рис. 1. Циркулянты с тремя образующими:** *а* – *C*(9; 2, 3, 4); *б* – *C*(20; 3, 5, 7)




**Рис. 2. Единичные маршруты из одной вершины в циркулянте *C*(10; 2, 3, 4)**




*а)* *б)*

**Рис. 5. График разностей максимальных расстояний, рассчитанных с помощью алгоритмов «Sequent» и Дейкстры:** *а* – тест 2; *б* – тест 3

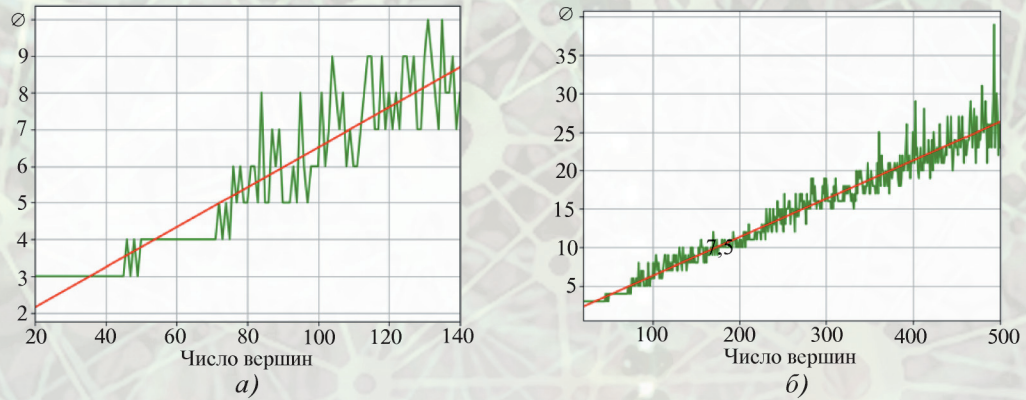

*а)* *б)*

**Рис. 6. Зависимость времени выполнения алгоритма от числа вершин в циркулянтах типа:** *а* – *C*(*N*; 2, 3, 7); *б* – *C*(*N*; 10, 17, 39)

**Рис. 7. Зависимость максимального расстояния от числа вершин для циркулянтов типа *C*(*N*; 2, 3, 7) при изменении *N* от 20 до 220**

Рисунки к статье А. Ю. Романова, М. В. Сидоренко, Э. А. Монаховой

## «МАРШРУТИЗАЦИЯ В СЕТЯХ-НА-КРИСТАЛЛЕ С ТОПОЛОГИЕЙ ТРЕХМЕРНЫЙ ЦИРКУЛЯНТ»




*а)* *б)*

**Рис. 8. Зависимость максимального расстояния циркулянта, рассчитанного по разработанному алгоритму, от числа вершин для циркулянтов типа *C*(*N*; 4, 7, 13) при изменении *N*: *а* – от 20 до 140; *б* – от 20 до 500**