УДК 004.021 + 004.42

МОДЕЛЬ САМОТРАНСФОРМАЦИИ ГРАФОВ, ОСНОВАННАЯ НА ОПЕРАЦИИ ИЗМЕНЕНИЯ КОНЦА РЕБРА

И. Б. Бурдонов

Институт системного программирования им. В.П. Иванникова Российской академии наук, г. Москва

igorburdonov@yandex.ru, igor@ispras.ru

Аннотация

Рассмотрена распределенная сеть, топология которой описана неориентированным графом. Сеть может сама изменять свою топологию, используя специальные «команды», подаваемые ее узлами. В работе предложена предельно локальная атомарная трансформация $a \to c \to b$ изменения конца c ребра ac, «движущегося» вдоль ребра cb от вершины c к вершине b. В результате этой операции ребро ac удаляется, а ребро ab добавляется. Такая трансформация выполняется по «команде» от общей вершины c двух смежных ребер ac и cb. Показано, что из любого дерева можно получить любое другое дерево с тем же множеством вершин, использовав только атомарные трансформации. Если степени вершин дерева ограничены числом d ($d \geq 3$), то трансформация не нарушает этого ограничения. В качестве примера цели такой трансформации рассмотрены задачи максимизации и минимизации индекса Винера дерева с ограниченной степенью вершин без изменения множества его вершин. Индекс Винера – это сумма попарных расстояний между вершинами графа. Максимальный индекс Винера имеет линейное дерево (дерево с двумя листовыми вершинами). Для корневого дерева с минимальным индексом Винера определены его вид и способ вычисления числа вершин в ветвях соседей корня. Предложены два распределенных алгоритма: трансформации дерева в линейное дерево и трансформации линейного дерева в дерево с минимальным индексом Винера. Доказано, что оба алгоритма имеют сложность не выше $2n–2$, где n – число вершин дерева. Также рассмотрена трансформация произвольных неориентированных графов, в которых могут быть циклы, кратные ребра и петли, без ограничения на степени вершин. Показано, что любой связный граф с n вершинами может быть преобра-

зован в любой другой связный граф с k вершинами и тем же числом ребер за время не более $2(n+k)–2$.

***Ключевые слова**: распределенная сеть, самотрансформация графов, индекс Винера*

ВВЕДЕНИЕ

Индекс Винера [1] – топологический индекс молекулярных графов, используемый во многих приложениях, особенно в математической и компьютерной химии и хемоинформатике.

Рассмотрим распределенную сеть, топология которой – динамически изменяющееся дерево. Динамические графы [2] моделируют самоорганизующиеся сети [3–5], в том числе, социальные сети, нейронные сети [6] и роевой интеллект [7]. Особенность этих сетей – их однородность, без разделения узлов на коммутаторы, хосты и контроллеры. Основное внимание уделяется вопросам маршрутизации, пропускной способности, помехоустойчивости, безопасности, распределения нагрузки и сетевых ресурсов и т. п. Изменение топологии сети понимается как внешний фактор, который надо учитывать, но которым сама сеть не управляет или управляет лишь частично [8, 9].

С другой стороны, в литературе много работ, посвященных как раз целенаправленной трансформации графа, в частности, деревьев, с целью оптимизации по тем или иным критериям. В настоящей статье предложена атомарная трансформация, которая предельно локальна, затрагивая минимум вершин и ребер, максимально близких друг другу. Ранее [10–12] рассматривались другие трансформации деревьев, но они недостаточно локальны и сводятся к цепочкам атомарных трансформаций.

Предлагаемые в статье алгоритмы распределенные и параллельные. Дерево трансформирует само себя по «командам» от вычислительных единиц, соотносимых с вершинами. Для этого и нужна локальность трансформации.

Структура статьи следующая. В разделе 1 определены модель распределенной сети и атомарная трансформация. Раздел 2 содержит основные понятия и утверждения, связанные с индексом Винера. В разделе 3 предложен алгоритм трансформации дерева в линейное дерево, а в разделе 4 – из линейного дерева в дерево с минимальным индексом Винера и заданным ограничением на степе-

ни вершин. Даны оценки сложности. Доказательства утверждений можно найти в [15].

В разделе 5 рассмотрен общий случай трансформации неориентированных связных графов, в которых могут быть циклы, кратные ребра и петли, без ограничения на степени вершин. С помощью атомарной трансформации изменения конца ребра смоделированы удаление и добавление вершин. Показано, что любой связный граф с n вершинами может быть преобразован в любой другой связный граф с k вершинами и тем же числом ребер за время не более $2(n+k)-2$.

1. МОДЕЛЬ

Пусть G – неориентированное дерево без кратных ребер и петель с ограничением d ($d \geq 3$) на степени вершин, лежащее в основе распределенной сети. Дерево *упорядоченное*: ребрам, инцидентным вершине, присвоены различные ненулевые номера. Ребро ab имеет два номера: e_{ab} в вершине a и e_{ba} в вершине b.

Вершины отождествляются с вычислительными единицами, которые посылают друг другу сообщения по ребрам графа. Память вершины – набор переменных. Вначале в каждой вершине a переменная $E(a)$ инициализирована множеством номеров ребер, инцидентных a. Далее при трансформации графа вершина a сама корректирует переменную $E(a)$.

Сообщение задается типом и параметрами: $Тип(p_1, \ldots, p_k)$. Вершина a, посылая сообщение по ребру ab, указывает его номер e_{ab}. Вершина b получает сообщение вместе с номером ребра e_{ba}.

Дерево трансформируется по «командам» от своих вершин. Атомарная трансформация $a \rightarrow c \rightarrow b$ – это замена ребра ac на ребро ab при наличии ребра cb (рис. 1).

Выполняется по команде ***Изменить***(e_{ca}, e_{cb}, P), которую подает вершина c, где P – дополнительные параметры. Ребро ab получает в вершине a тот же номер, который имело удаляемое ребро ac, т. е. e_{ac}, а в вершине b – любой «свободный» номер e_{bc}. Для того чтобы вершина b «узнала» этот номер, по ребру из a в b автоматически посылается сообщение *Изменение*(P). Другие сообщения, передаваемые по изменяемому ребру, не теряются, но сообщение, направлявшееся в вершину c, получит вершина b. Вершина c сама удаляет номер e_{ca} из $E(c)$,

а вершина *b* сама добавляет номер e_{ba} в $E(b)$. Атомарная трансформация не изменяет множество вершин дерева и оставляет его деревом.

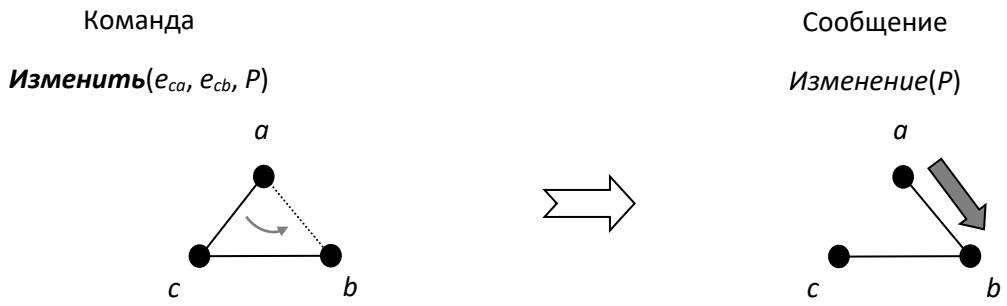

Рис. 1. Трансформация $a \to c \to b$: замена ребра *ac* на ребро *ab*

Для оценки времени работы алгоритмов будем пренебрегать временем вычислений в вершине и предполагать, что время пересылки сообщения по ребру и время атомарной трансформации, включая пересылку сообщения *Изменение*, не превышает 1 такта.

Утверждение 1: Для любых двух деревьев с ограничением d ($d \geq 3$) на степени вершин одно можно получить из другого с помощью цепочки атомарных трансформаций, причем в процессе трансформации ограничение d на степени вершин не будет нарушено.

2. ИНДЕКС ВИНЕРА

Индекс Винера – это сумма всех попарных расстояний между вершинами. Для данного числа вершин максимальный индекс Винера имеет линейное дерево (дерево с двумя листьями) (A000292 в [13]). Вид дерева с ограничением на степени вершин и минимальным индексом Винера определен в [14]. Это разновидность сбалансированного дерева (высоты листьев отличаются не более чем на 1) с жестким требованием к степени вершины, что отличает его от B-деревьев (в которых все листья находятся на одной высоте, а степени вершин могут быть разными) и от АВЛ-деревьев (которые двоичны).

Дерево *G* с выделенной вершиной – *корнем* – называется *корневым*. *Высота вершины* – расстояние от нее до корня. *Высота дерева* – максимальная высота вершины. *Ветвь вершины v* – подграф $G(v)$, порожденный множеством вершин, связанных с корнем путем, проходящим через *v*. Для ребра *ab* вершина

a – *отец* вершины *b*, а вершина *b* – *сын* вершины *a*, если путь из корня в *b* проходит через *a*. У каждой вершины, кроме корня, ровно один отец.

В упорядоченном корневом дереве вершины одной высоты линейно упорядочены: вершина *v левее* вершины *w* (*w правее v*), если после максимального общего префикса путей, ведущих из корня в *v* и *w*, номер следующего ребра на пути в *v* меньше номера следующего ребра на пути в *w*.

Корневое дерево высотой *h* с *n* вершинами *почти хорошее*, если степень корня равна min{*d*–1, *n*–1}, для *h*≥3 все вершины на высоте 1 .. *h*–2 имеют степень *d*, и дерево можно так упорядочить, что для *h*≥2 на высоте *h*–1 самая правая внутренняя вершина *u* имеет степень не больше *d*, вершины левее *u* имеют степень *d*, а вершины правее *u* – листья. *Хорошее* дерево отличается только степенью корня, она равна min{*d*, *n*–1}. Примеры даны на рис. 2.

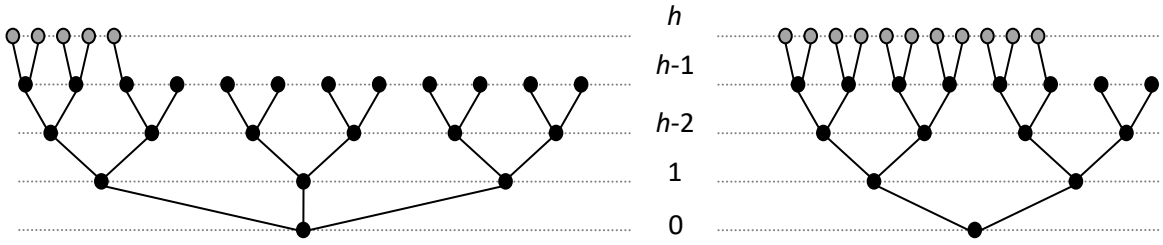

Рис. 2. Хорошее дерево (слева) и почти хорошее дерево (справа)

Утверждение 2 (Теорема 2.2 в [14]). Дерево со степенью вершин не более *d* (*d*≥3) имеет минимальный индекс Винера тогда и только тогда, когда это хорошее дерево.

Пусть в почти хорошем дереве высотой *h* степень корня равна 0 или *d*–1, а степени всех вершин на высоте *h*–1 равны *d*. Число вершин этого дерева обозначим $N(d, h)=1+(d–1)+(d–1)^2+…+(d–1)^h = ((d–1)^{+1} –1)/(d–2)$. Пусть в хорошем дереве высотой *h* степень корня равна 0 или *d*, а степени всех вершин на высоте *h*–1 равны *d*. Число вершин этого дерева обозначим *M*(*d*, *h*): *M*(*d*, 0)=1 и $M(d, h)=1+d+d(d–1)+…+d(d–1)^{h-1}=1+dN(d, h–1)$ для *h*≥1. Примеры – на рис. 2 при удалении «серых» вершин на высоте *h*.

Пусть задано (почти) хорошее дерево с *n* вершинами. Ветвь соседа корня – почти хорошее дерево. Упорядочим соседей корня по невозрастанию числа вершин в их ветвях и обозначим эти числа:

для почти хорошего дерева: $N(d, n, 1)≥…≥N(d, n, \min\{d–1, n–1\})$,

для хорошего дерева: $M(d, n, 1) \geq \ldots \geq M(d, n, \min\{d, n-1\})$.

Утверждение 3. Пусть $L(d, i)=N(d, i)$, если G – почти хорошее дерево, и $L(d, i)=M(d, i)$, если G – хорошее дерево. Пусть число вершин $n=L(d, h-1)+m(d-1)+s<L(d, h)$, где $0 \leq s<d-1$ и $m=p(d-1)^{h-2}+q$, где $0 \leq q<(d-1)^{h-2}$. Тогда для p самых левых соседей корня их ветви имеют по $N(d, h-1)$ вершин, для следующего справа соседа корня его ветвь имеет $N(d, h-2)+q(d-1)+s$ вершин, а для остальных соседей корня их ветви имеют по $N(d, h-2)$ вершин.

3. АЛГОРИТМ $\mathcal{A}$ ТРАНСФОРМАЦИИ В ЛИНЕЙНОЕ ДЕРЕВО

Если для вершины v ветвь ее сына w – линейное дерево, то путь от v через w до листа назовем *линейкой из v*. *Звездообразное* дерево – это корневое дерево, состоящее из линеек, ведущих из корня.

Пусть G – дерево с n вершинами с корнем r. Для удобства будем считать, что у корня есть фиктивное ребро с номером 0, ведущее к фиктивному отцу. Алгоритм стартует при получении корнем от его (фиктивного) отца сообщения *Старт*() и завершается посылкой из корня по этому ребру сообщения *Линия*(n). Алгоритм рекурсивный, на уровне рекурсии h алгоритм работает на каждой ветви $G(v)$, где v имеет высоту h, и состоит из трех этапов.

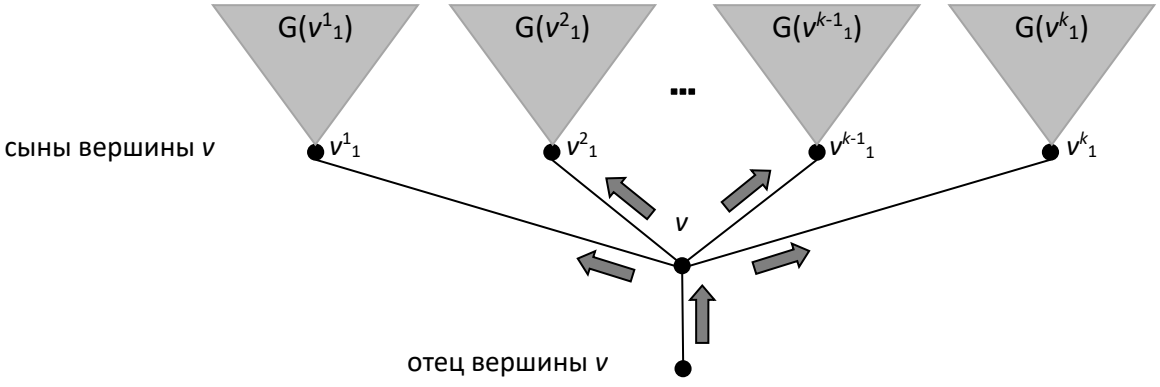

Рис. 3. Этап 1: Сообщения *Старт* и ветви дерева

Этап 1 (рис. 3). Вершина v получает от своего отца сообщение *Старт*, запоминает номер ребра, ведущего к отцу, и рассылает *Старт* всем своим сынам.

Этап 2 (рис. 4). Вершина v ожидает от своих сынов получения сообщений *Линия*, подсчитывая число вершин ветви $G(v)$ как 1 + сумма параметров получаемых сообщений *Линия*.

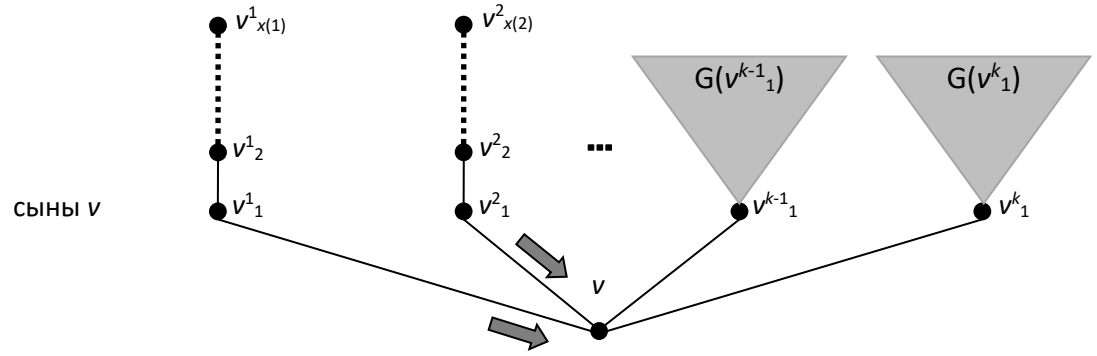

Рис. 4. Этап 2: Сообщения *Линия* и линейки

В начале этапа 3 (рис. 5) ветвь $G(v)$ – звездообразное дерево с k линейками. Длина i-й линейки равна $x(i)$. Вершина v запускает k–1 параллельных цепочек атомарных трансформаций так, что для i=1 .. k–1 у первого ребра i+1-ой линейки его конец $v^{i+1}{}_1$ фиксируется, а другой конец двигается по i-ой линейке от v до листа $v^i_{x(i)}$. Для этого вершина v выполняет сразу k–1 трансформаций $v^{i+1}{}_1 \to v \to v^i{}_1$ в последовательности $i=k$–1, ..., 1. Каждая из этих трансформаций – первая в цепочке трансформаций вдоль i-й линейки: $v^{i+1}{}_1 \to v \to v^i{}_1$, $v^{i+1}{}_1 \to v^i{}_1 \to v^i{}_2$, …, $v^{i+1}{}_1 \to v^i{}_{x(i)-1} \to v^i{}_{x(i)}$. Эти цепочки трансформаций выполняются параллельно вдоль k–1 линеек. Когда цепочка трансформаций вдоль i-й линейки заканчивается в ее листе, происходит конкатенация i-й и i+1-й линеек. Когда это случится для всех i=1..k–1, ветвь $G(v)$ станет линейным деревом. Об этом вершину v извещает сообщение *Финиш*(). Оно посылается после конкатенации k-й и k–1-й линеек, и далее проходит по линейкам k–1,…,1, причем i-я линейка проходится от конца $v^i_{x(i)}$ к началу v^i_1. Если *Финиш* переходит на i–1-ю линейку до конкатенации ее с i-й линейкой, пересылка приостанавливается до завершения конкатенации. В конце *Финиш* посылается по ребру (v^1_1, v). Вершина v посылает своему отцу сообщение *Линия*, завершая работу на ветви $G(v)$.

Утверждение 4. Алгоритм $\mathcal{A}$ трансформирует дерево с n вершинами в линейное дерево, не нарушая ограничения d на степени вершин, за время $t(n) \le 2n-2$. Из корня по его фиктивному отцу послано сообщение *Линия*(n).

Оценка $2n$–2 достижима на линейном дереве, когда корень – один из листов. Оценка не улучшаема: чтобы выяснить, что дерево линейное, сообщение

должно дойти от корня до другого листа и обратно, т. е. пройти путь длиной 2*n*–2.

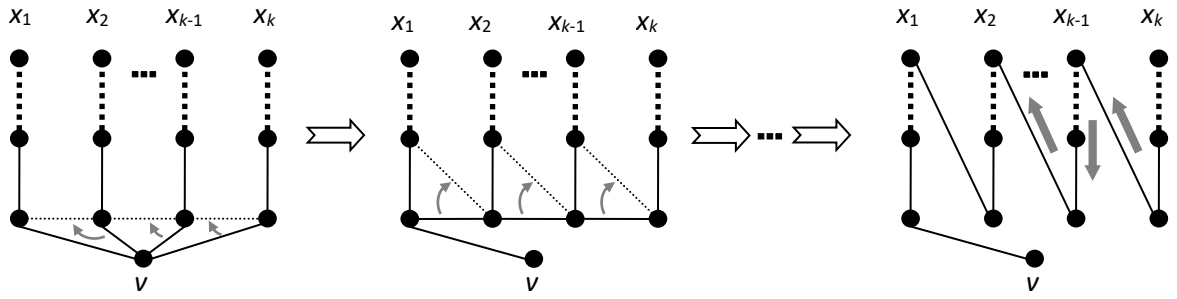

Рис. 5. Этап 3: Сообщения *Финиш* и трансформации

4. АЛГОРИТМ $\mathcal{B}$ – ТРАНСФОРМАЦИЯ ЛИНЕЙНОГО ДЕРЕВА В ХОРОШЕЕ ДЕРЕВО

Пусть *G* – линейное дерево с *n* вершинами и корнем *r* в листе. Алгоритм стартует при получении корнем по фиктивному ребру с номером 0 сообщения *Начало*(*d*, *n*), а завершается посылкой из корня по этому ребру сообщения *Конец*().

Алгоритм выполняется рекурсивно, уровень рекурсии равен высоте вершины в целевом хорошем дереве. На уровне рекурсии *h* построена часть хорошего дерева на высотах от 0 до *h*. Вначале *h*=0, и построенная часть состоит из одного корня. На уровне *h* алгоритм выполняется на каждой ветви *G*(*v*), где вершина *v* имеет в *G* высоту *h*, и состоит из двух этапов.

(*Почти*) *хорошим звездообразным* деревом назовем звездообразное дерево, в котором степень корня и числа вершин ветвей соседей корня такие же, как у (почти) хорошего дерева с тем же числом вершин.

Этап 1. Ветвь *G*(*v*) – линейка из *v*. Строим звездообразное дерево с корнем в *v*: хорошее, если *v*=*r*, и почти хорошее, если *v*≠*r*. Число вершин ветви *G*(*v*) – параметр сообщения *Начало*, с получения которого стартует этап 1 на ветви *G*(*v*).

Этап 2. Ветвь *G*(*v*) – хорошее (*v*=*r*) или почти хорошее (*v*≠*r*) звездообразное дерево. Вершина *v* посылает каждому сыну *w* сообщение *Начало*(*d*, *l*), где *l* – число вершин на ветви *G*(*w*), инициируя работу алгоритма на следующем уровне рекурсии. Это можно делать, как только построена линейка нужной длины из *w*. Вершина *v* ожидает от всех своих сынов сообщений *Конец*(), а затем посылает своему отцу сообщение *Конец*(). Если *v*=*r*, алгоритм заканчивается.

Как строить звездообразное дерево на этапе 1? Используем понятие *текущей вершины* (вначале вершина v) и две операции: *перемещение* и *трансформация*. Параметр t – число трансформаций, которые осталось сделать для построения линейки звездообразного дерева.

Перемещение $c \dashrightarrow b$: c – текущая вершина, есть ребро $\{c, b\}$. Вершина c посылает в вершину b сообщение *Перемещение*(t). Получив сообщение, вершина b становится текущей.

Трансформация $a \to c \to b$: c – текущая вершина, есть ребра $\{a, c\}$ и $\{c, b\}$. Величина t уменьшается на 1: $t:=t-1$. Вершина c подает команду ***Изменить***(e_{ca}, e_{cb}, t). Получив сообщение *Изменение*(t), вершина b становится текущей.

Построение показано на рис. 6: серая стрелка указывает текущую вершину, белый кружок – вершину v. Пусть $l>2$ – число вершин ветви $G(v)$. Вначале есть линейка $v=v_1, v_2, \ldots, v_l$, текущая вершина v. Обозначим:

$x=\min\{d, l-1\}$ – степень вершины v в хорошем звездообразном дереве;

$SM(d, l, 0)=1$;

$SM(d, l, j) = 1+M(d, l, 1)+M(d, l, 2)+\ldots+M(d, l, j)$, для $j=1 .. x$, – число вершин в хорошем звездообразном дереве на первых j линейках плюс единица (корень);

$v^j_i=v_{SM(d, l, j-1)+i}$, для $j=1 .. x-1$ и $i=1 .. M(d, l, j)$ – i-я вершина j-й линейки;

$v^x_i=v_{SM(d, l, x)-i+1}$ для $i=1 .. M(d, l, x)$ – i-я вершина x-й линейки.

Строим линейки «справа налево», начиная от x-й и заканчивая 2-й.

Построение j-й линейки хорошего звездообразного дерева для $j=x .. 2$:

1. $t=M(d, l, j)$.
2. Перемещение $v \dashrightarrow v^j_1$.
3. Цепочка $M(d, l, j)-1$ трансформаций:

$$v \to v^j_1 \to v^j_2, \quad v \to v^j_2 \to v^j_3, \quad \ldots, v \to v^j_{M(d, l, j)-1} \to v^j_{M(d, l, j)};$$

$$t=M(d, l, j)-1, \quad t=M(d, l, j)-2, \quad \ldots, \quad t=1.$$

4. $M(d, l, j)$-я трансформация: $v^{j+1}_1 \to v^j_{M(d, l, j)} \to v$.
5. Пуск алгоритма построения почти хорошего звездообразного дерева на j-й ветви: вершина v посылает вершине $v^j_{M(d, l, j)}$ сообщение *Начало*(d, $M(d, l, j)$).

Когда построена 2-я линейка, построена и 1-я линейка, поэтому одновременно запускается алгоритм на 1-й линейке: вершина v посылает вершине $v^1_{M(d, l, 1)}$ сообщение *Начало*(d, $M(d, l, 1)$).

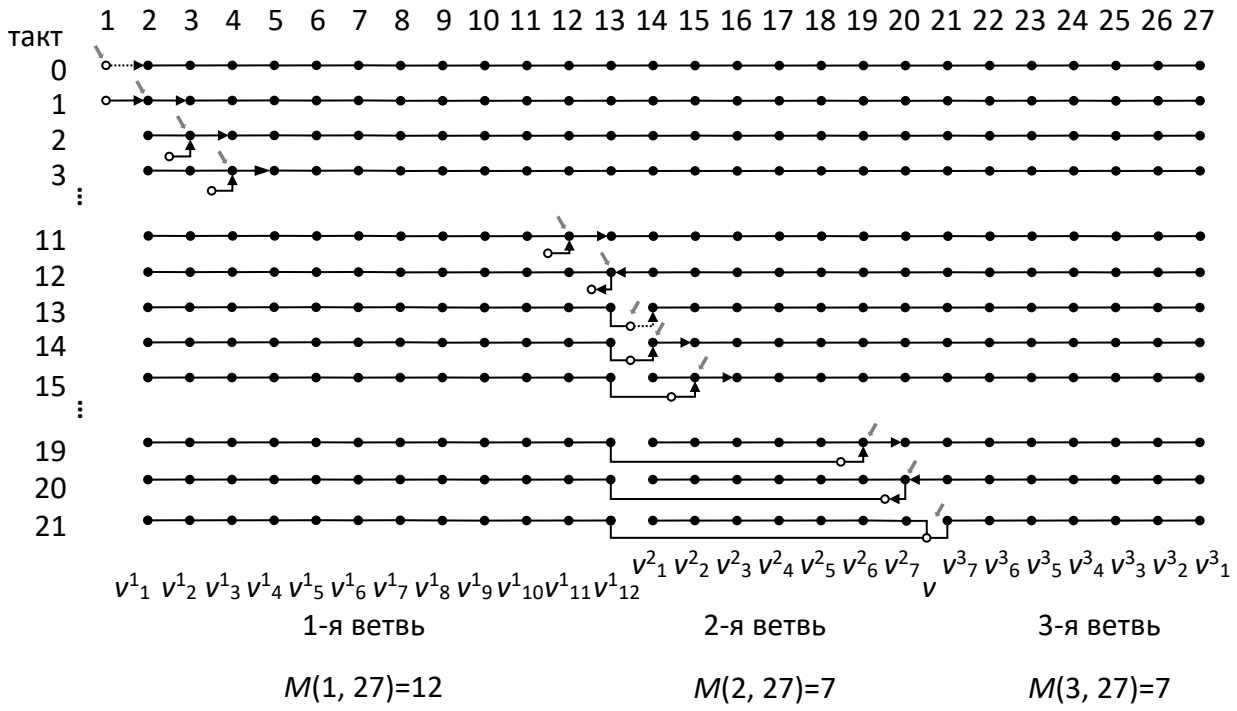

Рис. 6. Построение звездообразного хорошего дерева для n=27 и d=3

Почти хорошее звездообразное дерево строится так же, но число линеек x равно не $\min\{d, l–1\}$, а $\min\{d–1, l–1\}$, и число вершин в j-й линейке равно не $M(d, l, j)$, а $N(d, l, j)$.

Утверждение 5. Алгоритм $\mathcal{B}$ трансформирует линейное дерево с n вершинами в хорошее дерево без нарушения ограничения d на степени вершин за время $t(n)\leq 2n–2$.

5. ТРАНСФОРМАЦИЯ ПРОИЗВОЛЬНЫХ НЕОРИЕНТИРОВАННЫХ ГРАФОВ

В этом разделе рассмотрена трансформация произвольных неориентированных графов, в которых могут быть циклы, кратные ребра и петли, без ограничения на степени вершин.

В предыдущих разделах неявно предполагалось, что атомарная трансформация $a\rightarrow c\rightarrow b$ выполняется тогда, когда ребро cb не изменяется в результате одновременного выполнения других атомарных трансформаций. Здесь мы это ограничение снимаем.

Кроме асинхронной сети, в которой время выполнения атомарных трансформаций и пересылки сообщений по ребрам недетерминировано (но ограни-

чено сверху), мы также будем рассматривать синхронную сеть, в которой это время фиксировано.

В синхронной сети две трансформации $a \to c \to b$ и $c \to b \to d$ могут выполняться одновременно: конец c ребра ac перемещается по ребру cb от вершины c к вершине b, и одновременно конец b ребра cb перемещается по ребру bd от вершины b к вершине d. В результате этих двух трансформаций ребро bd не изменяется, а рёбра ac и cb заменяются рёбрами ad и cd. (рис. 7 справа). Тот же результат получается в асинхронной сети, если трансформации выполняются последовательно в порядке $c \to b \to d$, $a \to c \to b$. Однако при другом порядке последовательного выполнения трансформаций результат получается другой (рис. 7 слева).

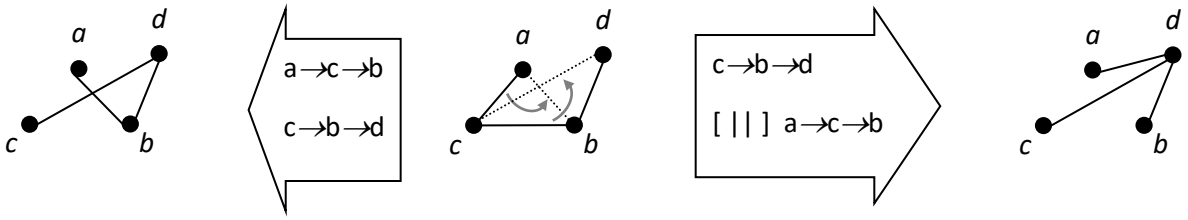

Рис. 7. Порядок трансформаций влияет на результат

При трансформациях могут появляться кратные рёбра: либо в результате одной трансформации при наличии циклов (рис. 8 слева), либо в результате одновременного выполнения одной и той же вершиной нескольких трансформаций (рис. 8 справа).

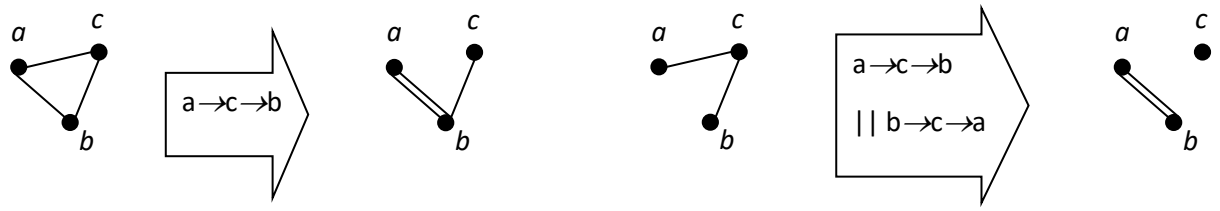

Рис. 8. Появление кратных рёбер

Два кратных ребра образуют цикл длины 2, но могут возникать и циклы большей длины (рис. 9).

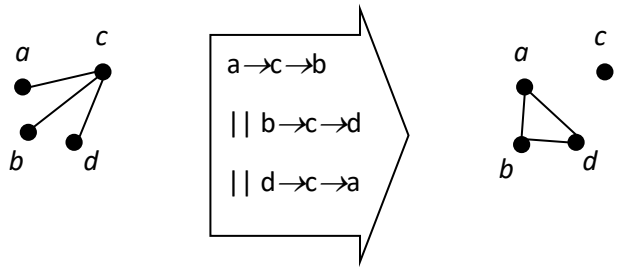

Рис. 9. Появление циклов

Рис. 8 (справа) и рис. 9 демонстрируют также, что в результате трансформаций граф может стать несвязным, в частности, могут появляться изолированные вершины без петель. В таких случаях естественно считать, что результатом трансформации связного корневого графа является компонента связности результирующего графа, которой принадлежит корень.

Особая ситуация возникает при одновременном выполнении трансформаций по циклу (рис. 10). Это можно интерпретировать как рождение новой вершины, соответствующей циклу; в результате трансформаций цикл исчезает, а его рёбра будут вести в эту новую вершину.

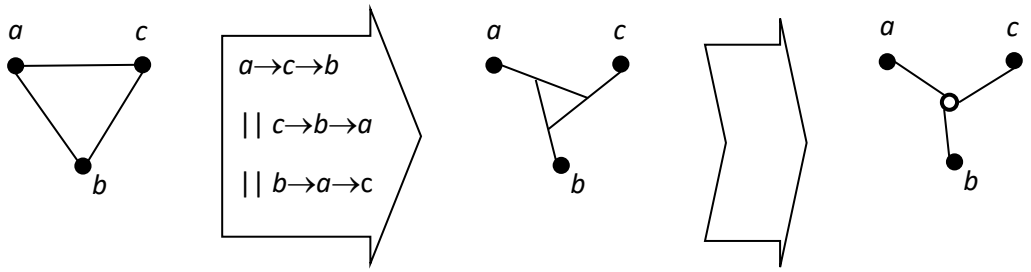

Рис. 10. Рождение вершины из цикла

Конечно, одновременной выполнение трансформаций по циклу требует синхронного выполнения, поскольку команды на эти трансформации подают разные вершины, лежащие на цикле. Однако есть одно исключение, когда цикл имеет длину 1, т. е. состоит из одной петли в одной вершине (рис. 11 справа). Петля, если её не было с самого начала, порождается трансформацией вида $a \to b \to a$ (рис. 11 слева), когда конец b ребра ab двигается вдоль самого этого ребра к вершине a, в которой и образуется петля. Обратная трансформация формально должна была бы записываться как $a \to a \to b$, однако, во-первых, вер-

шины *a* и *b* могут быть не соединены ребром, а, во-вторых, речь идёт о проведении ребра не в старую вершину *b*, а в новую вершину. Поэтому эта трансформация имеет вид $a \to a \to a$, она порождает новую вершину, например, *c,* и превращает петлю *aa* в ребро *ac* (рис. 11 справа). Мы будем записывать эту трансформацию как $a \to a \to a[c]$.

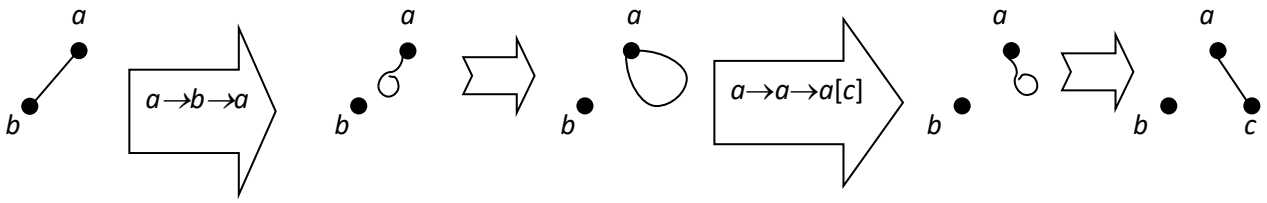

Рис. 11. Превращение ребра в петлю и петли в ребро, ведущее в новую вершину

В качестве примера предложим алгоритм «схлопывания» графа в изолированную вершину с петлями. Рассмотрим произвольный граф *G* с *m* рёбрами и корнем *r*, в котором могут быть циклы, петли и кратные рёбра. При этом петля имеет два номера в вершине. При «схлопывании» граф *G* с *m* рёбрами превращается в граф с *m* петлями в одной вершине — корне *r*.

Идея алгоритма «схлопывания» заключается в следующем: строится остов графа, после чего концы всех рёбер двигаются к корню по этому остову, пока не превратятся в петли в корне.

В вершинах имеются следующие переменные:

d(*v*)=степень вершины *v* в начале (до трансформаций);

s(*v*) – число сообщений *Старт*, полученных вершиной *v*, вначале *s*(*v*)=0;

e(*v*) – число рёбер, инцидентных вершинам ветви *G*(*v*), причем ребра остова считаются дважды, вначале *e*(*v*)=0.

Кроме того, в корне *r* имеется переменная *M* – число полученных корнем *r* сообщений *Изменение* плюс степень корня до трансформаций, вначале *M*=*d*(*r*).

Используются сообщения двух типов: 1) *Старт*(*p*), где *p* – целочисленный параметр в диапазоне [0..*m*], 2) *Изменение* – сообщение, автоматически посылаемое в вершину *b* при трансформации $a \to c \to b$ по ребру *ab*, в которое превращается ребро *ac*.

Алгоритм «схлопывания»:

1. Вначале из корня *r* посылается сообщение *Старт*(1) по всем инцидентным корню рёбрам.

2. Если $c \neq r$, $s(c)=0$ и вершина c получает (первый раз) сообщение *Старт* по ребру cb, вершина c запоминает ребро cb как *обратное* и посылает сообщение *Старт*(1) по всем остальным инцидентным ей рёбрам: $s(c):=1$, $e(c):=2$.
3. Если $c \neq r$, $s(c)>0$ и вершина c получает (повторно) сообщение *Старт*(p) по ребру ca, вершина c делает трансформацию $a \to c \to b$, где ребро cb — обратное. Посылается сообщение *Изменение* по ребру ab, в которое превращается ребро ac, $s(c)=(c)+1$, $e(c):=e(c)+p$.
4. Если $c \neq r$ и вершина c получает сообщение *Изменение* по ребру ac, вершина c делает трансформацию $a \to c \to b$, где ребро cb — обратное. Посылается сообщение *Изменение* по ребру ab, в которое превращается ребро ac.
5. Если $c \neq r$ и $s(c)=d(c)$ (вершина c получила сообщение *Старт* по всем инцидентным ей рёбрам), вершина c посылает сообщение *Старт*($e(c)$) по обратному ребру cb и делает трансформацию $b \to c \to b$. Посылается сообщение *Изменение* по ребру bb, в которое превращается ребро bc.
6. Если корень r получает сообщение *Изменение*, $M:=M+1$.
7. Если корень r получает сообщение *Старт*(p), $e(r):=e(r)+p$.
8. Если $s(r)=d(r)$ (корень r получил сообщение *Старт* по всем инцидентным ему рёбрам) и $e(r)=M$, конец алгоритма.

Время работы алгоритма не превосходит $2n-1$, где n – число вершин. Действительно, время достижения сообщением *Старт* любой вершины не превосходит $n-1$. Тогда сообщение *Старт* пройдёт по каждому ребру ac из вершины a в вершину c через время не более n от начала работы алгоритма. После этого начинается цепочка трансформаций, перемещающая конец c этого ребра до корня. Число трансформаций в этой цепочке равно расстоянию по остову от вершины c до корня, которое не превосходит $n-1$. Тем самым, через время не более $2n-1$, каждый конец каждого ребра перемещается в корень. Следовательно, каждое ребро становится петлёй в корне.

Каждая вершина c, получая сообщения *Старт*, подсчитывает число $e(c)$ ребер, инцидентных вершинам поддерева $G(c)$, причём рёбра остова считаются дважды. После этого вершина c посылает по обратному ребру сообщение *Старт*($e(c)$). Когда корень r получит сообщение *Старт* по каждому инцидентному ему ребру, переменная $e(r)$ будет содержать удвоенное число рёбер графа $2m$. С другой стороны, от каждого из двух концов каждого ребра, кроме конца r

ребра, инцидентного корню до трансформаций, в корень придёт одно сообщение *Изменение*. Тем самым корень получит 2*m*–*d*(*r*) сообщений *Изменение*, и переменная *M* станет равной 2*m*. Поэтому конец алгоритма правильно определяется по равенству *e*(*r*)=*M*.

Обратная последовательность обратных трансформаций алгоритма «схлопывания» графа *G* с *m* ребрами выполняет «расхлопывание» графа, т. е. превращение изолированного корня с *m* петлями в граф *G*. На рис. 12 показан пример «схлопывания» и «расхлопывания» графа.

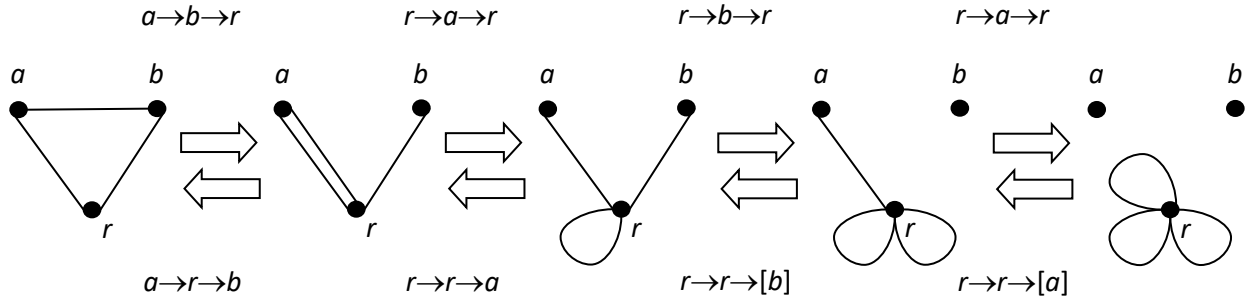

Рис. 12. «Схлопывание» и «расхлопывание» графа

Таким образом, любой связный граф с *n* вершинами может быть преобразован в любой другой связный граф с *k* вершинами и тем же числом ребер за время не более (2*n*–1)+(2*k*–1)=2(*n*+*k*)–2.

ЗАКЛЮЧЕНИЕ

В статье предложены два алгоритма самотрансформации дерева, лежащего в основе распределенной сети, с помощью локальных атомарных трансформаций, выполняемых по «командам» от узлов дерева. Это алгоритм трансформации любого дерева в линейное дерево и алгоритм трансформации линейного дерева в хорошее дерево, имеющее минимальный индекс Винера при том же числе вершин. Трансформации не изменяют множество вершин дерева и не нарушают ограничение *d* (*d*≥3) на степени вершин.

Оба алгоритма имеют верхнюю оценку времени работы 2*n*–2, где *n* – число вершин дерева. Однако, если для алгоритма $\mathcal{A}$ трансформации в линейное дерево эта оценка достижима, то для алгоритма $\mathcal{B}$ трансформации в хорошее дерево это не так. Это объясняется тем, что для алгоритма $\mathcal{A}$ оценка достигается на линейном дереве, а хорошее дерево алгоритм $\mathcal{A}$ трансформирует в линейное за меньшее время. Соответственно, и «обратный» алгоритм $\mathcal{B}$ трансформирует ли-

нейное дерево в хорошее меньше чем за 2*n*–2 тактов. Поэтому верхняя оценка для алгоритма $\mathcal{B}$ нуждается в уточнении (в сторону уменьшения).

Другое отличие алгоритмов $\mathcal{A}$ и $\mathcal{B}$ состоит в следующем. Алгоритм $\mathcal{A}$ (если из него удалить функцию вычисления числа вершин дерева) не зависит от *n* – числа вершин дерева. Поэтому алгоритм $\mathcal{A}$ может выполняться конечными автоматами в вершинах дерева. Однако алгоритм $\mathcal{B}$ существенно зависит от *n*. Остаются вопросы:

1) Можно ли сделать алгоритм трансформации линейного дерева в хорошее дерево не зависящим от *n*?

2) Можно ли сделать алгоритм трансформации дерева в хорошее дерево «напрямую», т.е. не через линейное дерево?

В статье также рассмотрена трансформация произвольных неориентированных графов, в которых могут быть циклы, кратные ребра и петли. Такая трансформация сохраняет число ребер графа. В качестве примера предложен алгоритм преобразования графа в изолированную вершину с петлями. Ограничение на степени вершин при этом, очевидно, не сохраняется. Интерес представляет исследование таких трансформаций, которые сохраняют ограничение на степени вершин. В частности, трансформации, имеющие целью минимизацию индекса Винера для графов с циклами.

СПИСОК ЛИТЕРАТУРЫ

1. *Wiener H.* Structural determination of paraffin boiling points // J. Am. Chem. Soc. 1947. No 69 (1). P. 17–20.

2. *Кочкаров А.А., Сенникова Л.И., Кочкаров Р.А.* Некоторые особенности применения динамических графов для конструирования алгоритмов взаимодействия подвижных абонентов // Известия ЮФУ. Технические науки, раздел V, системы и пункты управления. 2015. №1. С. 207–214.

3. *А.В. Проскочило, А.В. Воробьев, М.С. Зряхов, А.С. Кравчук*. Анализ состояния и перспективы развития самоорганизующихся сетей // Научные ведомости, серия экономика, информатика, выпуск 36/1. 2015. № 19 (216). С. 177–186.

4. *Pathan A.S.K.* (ed.). Security of self-organizing networks: MANET, WSN, WMN, VANET. CRC press, 2010. 638 p.

5. *Boukerche A.* (ed.). Algorithms and protocols for wireless, mobile Ad Hoc networks // John Wiley & Sons, 2008. 496 p.

6. *Chen Z., Li S., Yue W*. SOFM Neural Network Based Hierarchical Topology Control for Wireless Sensor Networks // Hindawi Publishing Corporation. J. of Sensors. 2014. article ID 121278. 6 p. http://dx.doi.org/10.1155/2014/121278

7. *Mo S., Zeng J.-C., Tan Y.* Particle Swarm Optimization Based on Self-organizing Topology Driven by Fitness // International Conference on Computational Aspects of Social Networks, CASoN 2010, Taiyuan, China, 10.1109/CASoN. 2010. No 13. P. 23–26.

8. *Wen C.-Y., Tang H.-K.* Autonomous distributed self-organization for mobile wireless sensor networks // Sensors (Basel, Switzerland). 2009. V. 9, 11. P. 8961–8995.

9. *Llorca J., Milner S.D., Davis C.* Molecular System Dynamics for Self-Organization in Heterogeneous Wireless Networks // EURASIP J. on Wireless Communications and Networking. 2010. 10.1155/2010/548016. 13 p.

10. *Wai-kai C*. Net Theory And Its Applications: Flows In Networks. Imperial College Press (26 May 2003). 672 p.

11. *Wang H.* On the Extremal Wiener Polarity Index of Hückel Graphs // Computational and Mathematical Methods in Medicine. 2016. article ID 3873597. 6 p. http://dx.doi.org/10.1155/2016/3873597

12. *Xu X., Gao Y., Sang Y., Liang Y.* On the Wiener Indices of Trees Ordering by Diameter-Growing Transformation Relative to the Pendent Edges // Mathematical Problems in Engineering. 2019. article ID 8769428. 11 p. https://doi.org/10.1155/2019/8769428

13. The On-Line Encyclopedia of Integer Sequences (OEIS). http://oeis.org/

14. *Fischerman M., Hoffmann A., Rautenbach D., Székely L., Volkmann L.* Wiener index versus maximum degree in trees // Discrete Applied Mathematics. 2002. V. 122. Is. 1–3. P. 127–137.

15. *Бурдонов И.* Самотрансформация деревьев с ограниченной степенью вершин с целью минимизации или максимизации индекса Винера// Труды ИСП РАН. 2019. Т. 31. Вып. 4. С. 189–210.

GRAPH SELF-TRANSFORMATION MODEL BASED ON THE OPERATION OF CHANGE THE END OF THE EDGE

I. B. Burdonov

Ivannikov Institute for System Programming of the RAS, Moscow

igorburdonov@yandex.ru,igor@ispras.ru

Abstract

We consider a distributed network whose topology is described by an undirected graph. The network itself can change its topology, using special “commands” provided by its nodes. The work proposes an extremely local atomic transformation $a \rightarrow c \rightarrow b$ of a change the end c of the edge ac, “moving” along the edge cb from vertex c to vertex b. As a result of this operation, the edge ac is removed, and the edge ab is added. Such a transformation is performed by a “command” from a common vertex c of two adjacent edges ac and cb. It is shown that from any tree you can get any other tree with the same set of vertices using only atomic transformations. If the degrees of the tree vertices are bounded by the number d ($d \geq 3$), then the transformation does not violate this restriction. As an example of the purpose of such a transformation, the problems of maximizing and minimizing the Wiener index of a tree with a limited degree of vertices without changing the set of its vertices are considered. The Wiener index is the sum of pairwise distances between the vertices of a

graph. The maximum Wiener index has a linear tree (a tree with two leaf vertices). For a root tree with a minimum Wiener index, its type and method for calculating the number of vertices in the branches of the neighbors of the root are determined. Two distributed algorithms are proposed: transforming a tree into a linear tree and transforming a linear tree into a tree with a minimum Wiener index. It is proved that both algorithms have complexity no higher than 2*n*–2, where *n* is the number of tree vertices. We also consider the transformation of arbitrary undirected graphs, in which there can be cycles, multiple edges and loops, without restricting the degree of the vertices. It is shown that any connected graph with *n* vertices can be transformed into any other connected graph with *k* vertices and the same number of edges in no more than 2(*n*+*k*)–2.

Keywords*: distributed network, self-transformation of graphs, Wiener index*

REFERENCES

1. *Wiener H.* Structural determination of paraffin boiling points // J. Am. Chem. Soc. 1947. No 69 (1). P. 17–20.

2. *Kochkarov A.A., Sennikova L.I., Kochkarov R.A.* Nekotorye osobennosti primeneniia dinamicheskikh grafov dlia konstruirovaniia algoritmov vzaimodeistviia podvizhnykh abonentov // Izvestiia IuFU. Tekhnicheskie nauki, razdel V, cistemy i punkty upravleniia. 2015. No 1, S. 207–214 (in Russian).

3. *Proskochilo A.V., Vorobev A.V., Zriakhov M.S., Kravchuk A.S.* Analiz sostoianiia i perspektivy razvitiia samoorganizuiushchikhsia setei // Nauchnye vedomosti, seriia ekonomika, informatika. 2015. Vypusk 36/1. No 19 (216). S. 177–186 (in Russian).

4. *Pathan A.S.K.* (ed.). Security of self-organizing networks: MANET, WSN, WMN, VANET. CRC press, 2010. 638 p.

5. *Boukerche A.* (ed.). Algorithms and protocols for wireless, mobile Ad Hoc networks // John Wiley & Sons, 2008. 496 p.

6. *Chen Z., Li S., Yue W*. SOFM Neural Network Based Hierarchical Topology Control for Wireless Sensor Networks // Hindawi Publishing Corporation. J. of Sensors. 2014. article ID 121278. 6 p. http://dx.doi.org/10.1155/2014/121278

7. *Mo S., Zeng J.-C., Tan Y.* Particle Swarm Optimization Based on Self-organizing Topology Driven by Fitness // International Conference on Computational

Aspects of Social Networks, CASoN 2010, Taiyuan, China, 10.1109/CASoN. 2010. No 13. P. 23–26.

8. *Wen C.-Y., Tang H.-K.* Autonomous distributed self-organization for mobile wireless sensor networks // Sensors (Basel, Switzerland). 2009. V. 9, 11. P. 8961–8995.

9. *Llorca J., Milner S.D., Davis C.* Molecular System Dynamics for Self-Organization in Heterogeneous Wireless Networks // EURASIP J. on Wireless Communications and Networking. 2010. 10.1155/2010/548016. 13 p.

10. *Wai-kai C*. Net Theory And Its Applications: Flows In Networks. Imperial College Press (26 May 2003). 672 p.

11. *Wang H.* On the Extremal Wiener Polarity Index of Hückel Graphs // Computational and Mathematical Methods in Medicine. 2016. article ID 3873597. 6 p. http://dx.doi.org/10.1155/2016/3873597

12. *Xu X., Gao Y., Sang Y., Liang Y.* On the Wiener Indices of Trees Ordering by Diameter-Growing Transformation Relative to the Pendent Edges // Mathematical Problems in Engineering. 2019. article ID 8769428. 11 p. https://doi.org/10.1155/2019/8769428

13. The On-Line Encyclopedia of Integer Sequences (OEIS). http://oeis.org/

14. *Fischerman M., Hoffmann A., Rautenbach D., Székely L., Volkmann L.* Wiener index versus maximum degree in trees // Discrete Applied Mathematics. 2002. V. 122. Is. 1–3. P. 127–137.

15. *Bourdonov I.* Self-transformation of trees with a limited degree of vertices in order to minimize or maximize the Wiener index// Proceedings of ISP RAS. 2019. V. 31. No 4. P. 189–210 (in Russian).

СВЕДЕНИЯ ОБ АВТОРЕ

БУРДОНОВ Игорь Борисович – ведущий научный сотрудник Института системного программирования им. В.П. Иванникова РАН. Сфера научных интересов – моделирование и верификация программных систем, теория графов, теория автоматов

Igor Borisovich BURDONOV – *Leading Researcher, Ivannikov Institute for System Programming of the RAS. Research interests - modeling and verification of software systems, graph theory, automata*

email: igorburdonov@yandex.ru, igor@ispras.ru

Материал поступил в редакцию 30 октября 2019 года