ИПМ им.М.В.Келдыша РАН

Абрау-2019 • Труды конференции

И.Б. Бурдонов

Распределенный алгоритм самотрансформации топологии распределенной сети с целью минимизации индекса Винера

Рекомендуемая форма библиографической ссылки

Бурдонов И.Б. Распределенный алгоритм самотрансформации топологии распределенной сети с целью минимизации индекса Винера // Научный сервис в сети Интернет: труды XXI Всероссийской научной конференции (23-28 сентября 2019 г., г. Новороссийск). — М.: ИПМ им. М.В.Келдыша, 2019. — С. 166-176. — URL: http://keldysh.ru/abrau/2019/theses/52.pdf doi:10.20948/abrau-2019-52

Размещена также презентация к докладу

Распределенный алгоритм самотрансформации топологии распределенной сети с целью минимизации индекса Винера

И.Б. Бурдонов

Институт системного программирования им. В.П. Иванникова РАН

Аннотация. Рассматривается распределенная сеть, топология которой описывается неориентированным деревом без кратных ребер и петель. Сеть может сама менять свою топологию, используя специальные «команды», подаваемые ее узлами. В работе предложена предельно локальная атомарная трансформация: добавление ребра, соединяющего разные концы двух смежных ребер, и одновременное удаление одного из этих ребер. Такая трансформация выполняется по «команде» от общей вершины двух смежных ребер. Показывается, что из любого дерева можно получить любое другое дерево с помощью только атомарных трансформаций. Если степени вершин дерева ограничены числом d ($d \geq 3$), то трансформация не нарушает этого ограничения. В качестве примера цели такой трансформации рассматриваются 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 определяется его вид и способ вычисления числа вершин в ветвях соседей корня. Предлагаются два распределенных алгоритма: трансформация дерева в линейное дерево и трансформация линейного дерева в дерево с минимальным индексом Винера. Доказывается, что оба алгоритма имеют сложность не выше $2n - 2$, где n число вершин дерева.

Ключевые слова: распределенная сеть, трансформация графов, индекс Винера.

Distributed algorithm of self-transformation of the distributed network topology in order to minimize the Wiener index

I.B. Burdonov

Ivannikov Institute for System Programming of the RAS

Abstract. We consider a distributed network, the topology of which is described by a undirected tree without multiple edges and loops. The network itself can change its topology using special “commands” supplied by its nodes. The paper proposed an extremely local atomic transformation: the addition of an edge connecting the different ends of two adjacent edges, and the simultaneous removal of one of these edges. This transformation is performed by “command” from the common vertex of two adjacent edges. It is shown that any other tree can be obtained from any tree using only atomic transformations. If the degrees of the tree vertices are limited by the number d ($d \geq 3$), then the transformation does not violate this restriction. As an example of the goal of such a transformation, the tasks 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 the pairwise distances between the vertices of the 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 of calculating the number of vertices in the branches of the neighbors of the root is determined. Two distributed algorithms are proposed: the transformation of a tree into a linear tree and the transformation of a linear tree into a tree with a minimum Wiener index. It is proved that both algorithms have complexity not higher than $2n - 2$, where n is the number of tree vertices.

Keywords: distributed network, transformation of graphs, Wiener index.

1. Введение

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

Рассматривается распределенная сеть, топология которой – динамически меняющееся дерево. Динамические графы [2] моделируют самоорганизующиеся сети [3, 4, 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 статье предлагается атомарная трансформация, которая предельно локальна, затрагивая минимум вершин и ребер, максимально близких друг другу. Ранее [10, 11, 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 2 определяются модель распределенной сети и атомарная трансформация. Раздел 3 содержит основные понятия и утверждения, связанные с индексом Винера. В разделе 4 предложен алгоритм трансформации дерева в линейное дерево, а в разделе 5 – из линейного дерева в дерево с минимальным индексом Винера и заданным ограничением на степени вершин. Даются оценки сложности.

2. 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 инцидентных a ребер. Далее при трансформации графа вершина a сама корректирует переменную $E(a)$.

Сообщение задается типом и параметрами: $Tun(p_1, \ldots, p_k)$. 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 $a \to c \to b$ – 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 меняет множество вершин дерева и оставляет его деревом.

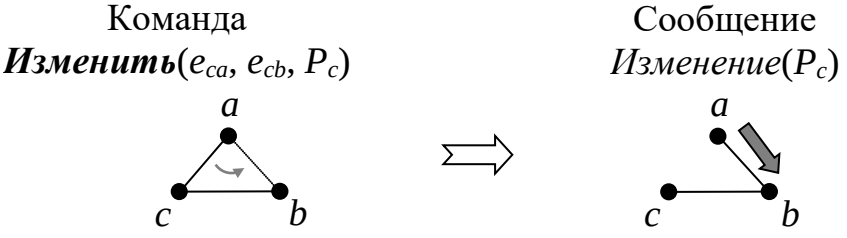

Рис. 1. Трансформация $a \to c \to b$: замена ребра ac на ребро ab.

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.

3. 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 $h \geq 3$ все вершины на высоте $1 .. h - 2$ имеют степень d, и дерево можно так упорядочить, что для $h \geq 2$ 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 на рис. 2.

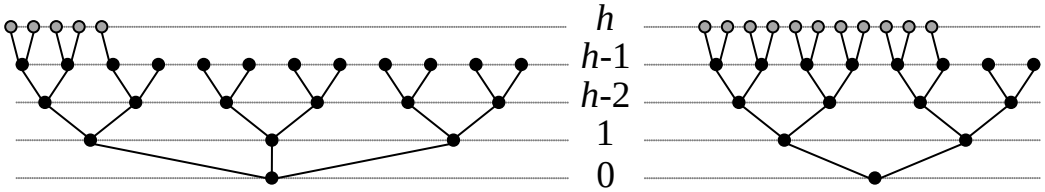

Рис. 2. Хорошее дерево (слева) и почти хорошее дерево (справа).

Утверждение 1. (Теорема 2.2. в [14]) Дерево со степенью вершин не более d ($d \geq 3$) 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 $N(d, h) = 1 + (d - 1) + (d - 1)^2 + \ldots + (d - 1)^h = ((d - 1)^{h+1} - 1) / (d - 2)$. Пусть в 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

$M(d, h)$: $M(d, 0) = 1$ и $M(d, h) = 1 + d + d(d - 1) + \ldots + d(d - 1)^{h - 1} = 1 + dN(d, h - 1)$ для $h \geq 1$. 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: $N(d, n, 1) \geq \ldots \geq N(d, n, min\{d - 1, n - 1\})$,
для хорошего дерева: $M(d, n, 1) \geq \ldots \geq M(d, n, min\{d, n - 1\})$.

Утверждение 2. 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.

4. 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 – 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.

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

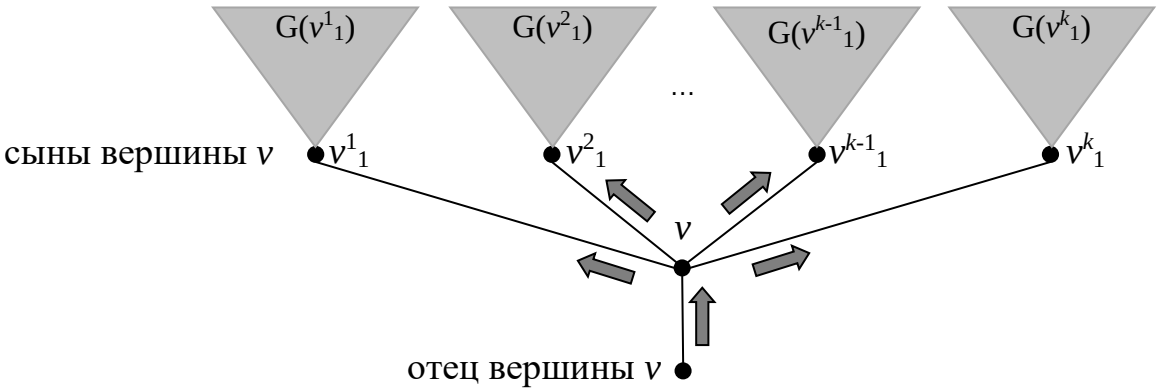

Рис. 3. Этап 1: Сообщения *Старт* и ветви дерева.

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*.

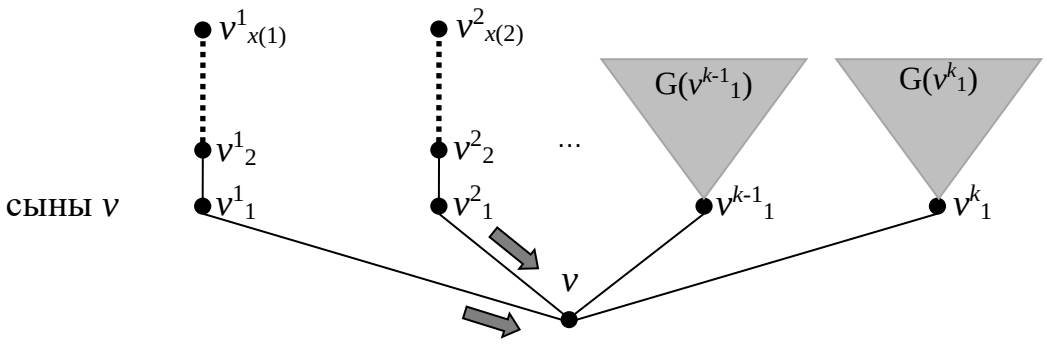

Рис. 4. Этап 2: Сообщения *Линия* и линейки.

В начале этапа 3 (рис. 5) ветвь $G(v)$ звездообразное дерево с k линейками. Длина i-й линейки $x(i)$. 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 $G(v)$.

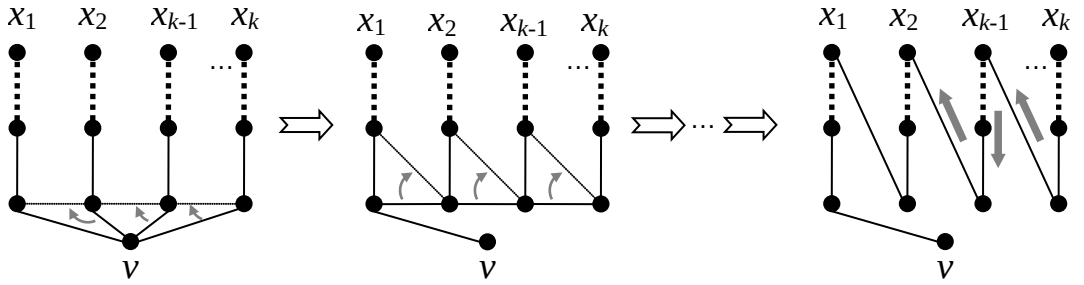

Рис. 5. Этап 3: Сообщения *Финиш* и трансформации.

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 $2n - 2$.

5. 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 $v \neq r$. 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 ($v \neq r$) 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 $a \rightarrow c \rightarrow b$: 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:
   $v \to v^j_1 \to v^j_2$, $v \to v^j_2 \to v^j_3$, …, $v \to v^j_{M(d, l, j)-1} \to v^j_{M(d, l, j)}$;
   $t = M(d, l, j) - 1$, $t = M(d, l, j) - 2$, …, $t = 1$.
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*$(d, M(d, l, 1))$.

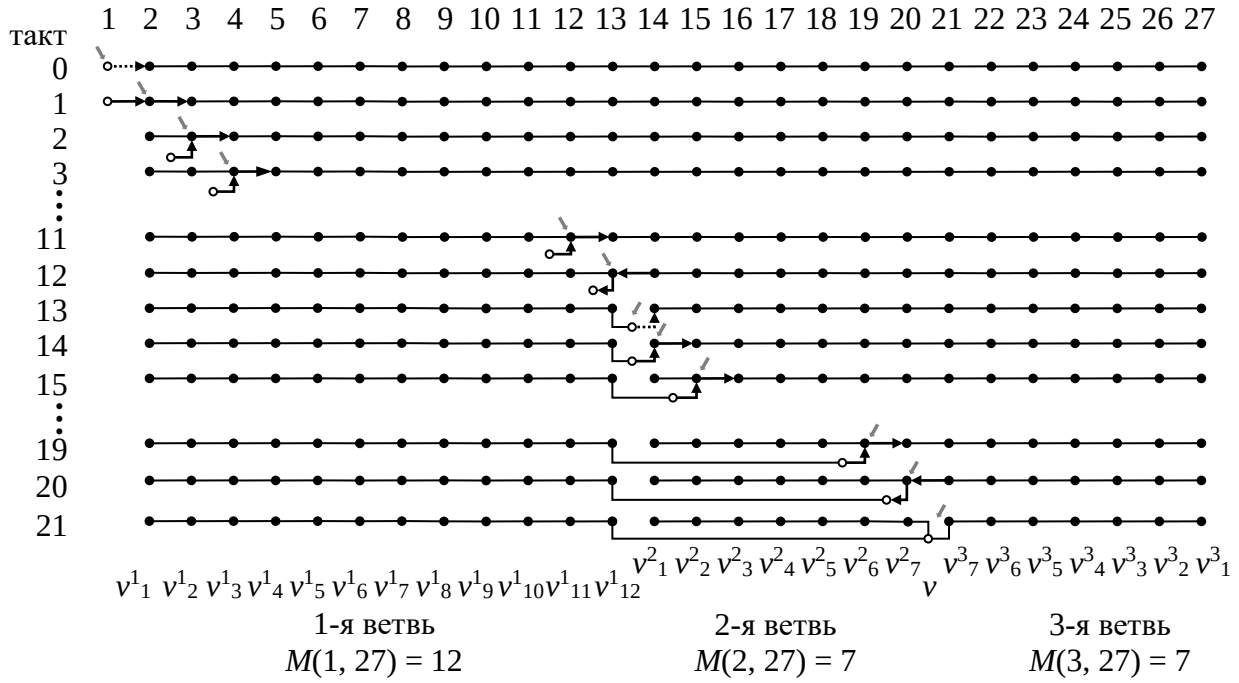

Рис. 6. Построение звездообразного хорошего дерева для $n = 27$ и $d = 3$.

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 $t(n) \leq 2n - 2$.

Работа выполнена при поддержке Российского фонда фундаментальных исследований, проект 17-07-00682-а.

Литература

1. H. Wiener. Structural determination of paraffin boiling points. // J. Am. Chem. Soc, 1947, № 69 (1), p. 17–20.
2. А.А. Кочкаров, Л.И. Сенникова, Р.А. Кочкаров. 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, № 19 (216), 2015, с. 177-186.
4. A.S.K. Pathan (ed.). Security of self-organizing networks: MANET, WSN, WMN, VANET. CRC press, 2010, p. 638.
5. A. Boukerche (ed.). Algorithms and protocols for wireless, mobile Ad Hoc networks. // John Wiley & Sons, 2008, p. 496.
6. Z. Chen, S. Li, W. Yue. SOFM Neural Network Based Hierarchical Topology Control for Wireless Sensor Networks. // Hindawi Publishing Corporation, Journal of Sensors, v. 2014, article ID 121278, 6 pp. http://dx.doi.org/10.1155/2014/121278
7. S. Mo, J.-C. Zeng, Y. Tan. 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, 13, p. 23–26.
8. C.-Y. Wen, H.-K. Tang. Autonomous distributed self-organization for mobile wireless sensor networks. // Sensors (Basel, Switzerland) v. 9,11, 2009, p. 8961-8995.
9. J. Llorca, S.D. Milner, C. Davis. Molecular System Dynamics for Self-Organization in Heterogeneous Wireless Networks. // EURASIP Journal on Wireless Communications and Networking, 2010, 10.1155/2010/548016, p. 13.
10. C. Wai-kai. Net Theory And Its Applications: Flows In Networks. Imperial College Press (26 May 2003), p. 672.
11. H. Wang. On the Extremal Wiener Polarity Index of Hückel Graphs. // Computational and Mathematical Methods in Medicine, volume 2016, article ID 3873597, p. 6. http://dx.doi.org/10.1155/2016/3873597.

12. X. Xu, Y. Gao, Y. Sang, Y. Liang. On the Wiener Indices of Trees Ordering by Diameter-Growing Transformation Relative to the Pendent Edges. // Mathematical Problems in Engineering, v. 2019, article ID 8769428, p. 11. https://doi.org/10.1155/2019/8769428.
13. The On-Line Encyclopedia of Integer Sequences (OEIS). http://oeis.org/.
14. M. Fischerman, A. Hoffmann, D. Rautenbach, L. Székely, L. Volkmann. Wiener index versus maximum degree in trees. // Discrete Applied Mathematics, v. 122, is. 1–3, 15 October 2002, p. 127–137.

References

1. H. Wiener. Structural determination of paraffin boiling points. // J. Am. Chem. Soc, 1947, № 69 (1), p. 17–20.
2. A. A. Kochkarov, L. I. Sennikova, R. A. Kochkarov. Nekotorye osobennosti primeneniia dinamicheskikh grafov dlia konstruirovaniia algoritmov vzaimodeistviia podvizhnykh abonentov. // «Izvestiia IuFU. Tekhnicheskie nauki», razdel V, cistemy i punkty upravleniia, 2015, №1, s. 207–214.
3. A.V. Proskochilo, A.V. Vorobev, M.S. Zriakhov, A.S. Kravchuk. Analiz sostoianiia i perspektivy razvitiia samoorganizuiushchikhsia setei. // Nauchnye vedomosti, seriia ekonomika, informatika, vypusk 36/1, № 19 (216), 2015, s. 177-186.
4. A.S.K. Pathan (ed.). Security of self-organizing networks: MANET, WSN, WMN, VANET. CRC press, 2010, p. 638.
5. A. Boukerche (ed.). Algorithms and protocols for wireless, mobile Ad Hoc networks. // John Wiley & Sons, 2008, p. 496.
6. Z. Chen, S. Li, W. Yue. SOFM Neural Network Based Hierarchical Topology Control for Wireless Sensor Networks. // Hindawi Publishing Corporation, Journal of Sensors, v. 2014, article ID 121278, 6 pp. http://dx.doi.org/10.1155/2014/121278
7. S. Mo, J.-C. Zeng, Y. Tan. 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, 13, p. 23–26.
8. C.-Y. Wen, H.-K. Tang. Autonomous distributed self-organization for mobile wireless sensor networks. // Sensors (Basel, Switzerland) v. 9,11, 2009, p. 8961-8995.
9. J. Llorca, S.D. Milner, C. Davis. Molecular System Dynamics for Self-Organization in Heterogeneous Wireless Networks. // EURASIP Journal on Wireless Communications and Networking, 2010, 10.1155/2010/548016, p. 13.
10. C. Wai-kai. Net Theory And Its Applications: Flows In Networks. Imperial College Press (26 May 2003), p. 672.
11. H. Wang. On the Extremal Wiener Polarity Index of Hückel Graphs. // Computational and Mathematical Methods in Medicine, volume 2016, article ID 3873597, p. 6. http://dx.doi.org/10.1155/2016/3873597.

12. X. Xu, Y. Gao, Y. Sang, Y. Liang. On the Wiener Indices of Trees Ordering by Diameter-Growing Transformation Relative to the Pendent Edges. // Mathematical Problems in Engineering, v. 2019, article ID 8769428, p. 11. https://doi.org/10.1155/2019/8769428.
13. The On-Line Encyclopedia of Integer Sequences (OEIS). http://oeis.org/.
14. M. Fischerman, A. Hoffmann, D. Rautenbach, L. Székely, L. Volkmann. Wiener index versus maximum degree in trees. // Discrete Applied Mathematics, v. 122, is. 1–3, 15 October 2002, p. 127–137.