ISSN 2076-2429 (print)
ISSN 2223-3814 (on line)

УДК 519.16:621.39

В.Г. Ткаченко, канд. техн. наук, доц.,
Д.Г. Ларин, канд. техн. наук,
Одес. нац. акад. связи им. А.С.Попова

МЕТОДЫ ПРЯМОГО И ОБРАТНОГО ПЕРЕБОРА ОСТОВОВ ГРАФА ТЕЛЕКОММУНИКАЦИОННЫХ СЕТЕЙ

В.Г. Ткаченко, Д.Г. Ларін. **Методи прямого і зворотного перебору остовів графа телекомунікаційних мереж.** Розроблені методи перебору остовів графа, що дозволяють значно скоротити кількість обчислювальних операцій при аналізі телекомунікаційних мереж. Наведено приклади алгоритмів повного перебору підграфів графу з заданою кількістю ребер.
Ключеві слова: метод перебору, остов, граф, підграф, телекомунікаційна мережа, монотонна бульова функція.

В.Г. Ткаченко, Д.Г. Ларин. **Методы прямого и обратного перебора остовов графа телекоммуникационных сетей.** Разработаны методы перебора остовов графа, позволяющие значительно сократить количество вычислительных операций при анализе телекоммуникационных сетей. Приведены примеры алгоритмов полного перебора подграфов графа с заданным количеством ребер.
Ключевые слова: метод перебора, остов, граф, подграф, телекоммуникационная сеть, монотонная булева функция.

V.G Tkachenko, D.G. Larin. **The methods of direct and reverse search of the graph telecommunication networks skeletons.** The graph skeleton search techniques allowing to considerably shorten computing operations cycle in analyzing telecommunication networks are developed. Examples of subgfaphs exhaustive search algorithms with the given amount of graph ribs are exposed.
Keywords: search technique, skeleton, graph, subgraph, telecommunication network, monotonous Boolean function.

Для исследования и оптимизации телекоммуникационных сетей и систем очень часто применяется аппарат теории графов [1…3], одним из исследуемых понятий которого является “остов графа” (“каркас графа” [4]) — связный подграф без циклов и петель, содержащий все вершины графа.

Известны такие методы определения остова графа: построение остова минимальной стоимости [4, 5]; нахождение остова на основе поиска в глубину; нахождение остова на основе поиска в ширину; метод Краскала [6].

Необходимость получения всех остовов возникает при:

— отсутствии или неопределенности весов ребер графа;

— наличии у ребер нескольких весов, соответствующих несвязанным между собой характеристикам, из-за чего невозможно привести весовые значения ребер к одному параметру;

— наличии у ребер характеристик, определяемых диапазоном значений, а не конкретным числом.

Задача нахождения оптимального остова в этих условиях решается методом полного перебора.

Известны комбинаторные методы упорядоченного и неупорядоченного полного перебора подграфов графа [5, 7], на основе анализа которых находятся остовы графа. Однако эти методы недостаточно эффективны, т.к. с увеличением размерности задачи количество вычислительных операций растет в экспоненциальной зависимости.

Предлагаются методы прямого и обратного полного перебора остовов графа телекоммуникационной сети с использованием монотонных булевых функций (МБФ).

Пусть задан некоторый связный граф $G = (V, E)$, где V — множество вершин v_n, E — множество ребер e_m. Каждое ребро инцидентно двум вершинам. Топологию графа можно задать в виде перечня ребер, где каждое ребро описывается совокупностью вершин, которое оно связывает.

В итоге граф телекоммуникационной сети (см. рисунок) можно описать с помощью МБФ:

— инциденции , в виде минимальной дизъюнктивной формы [8]

$$v_1v_2 \vee v_1v_4 \vee v_1v_5 \vee v_2v_3 \vee v_2v_5 \vee v_3v_5 \vee v_4v_5;$$

— кликовой, конъюнкции которой соответствуют кликам — полным подграфам или кликам — полным подграфам, состоящим из вершин, любая пара которых соединена ребром,

$$v_1v_2v_5 \vee v_1v_4v_5 \vee v_2v_3v_5;$$

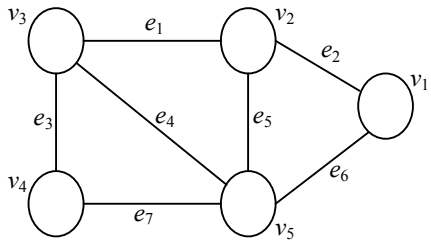
— вершинно-реберной, число конъюнкций которой равно числу вершин

$$e_1e_3e_4 \vee e_1e_2e_5 \vee e_2e_6 \vee e_3e_7 \vee e_4e_5e_6e_7. \quad (1)$$

Поскольку число ребер в остове графа n–1, где n — число вершин графа, то все остовы данного графа являются подмножеством всех подграфов данного графа из n–1 ребра. Для выделения этого подмножества необходимо сформулировать условие, которому должен удовлетворять подграф, чтобы быть остовом.

Выбор подграфа из n–1 ребер предлагается осуществлять следующим образом: т.к. каждое ребро может или входить или не входить в подграф, можно обозначить состояние каждого ребра как 1 (входит), или 0 (не входит). Таким образом, множество E ребер графа можно представить в виде двоичного числа, количество разрядов которого равно количеству ребер графа.

Условием того, что выбранная связная группа ребер образует остов, будет наличие в ней всех вершин графа. Для данного графа (см. рисунок) условия наличия проверяются следующим образом:



Пример графа телекоммуникационной сети

— наличие вершины v_1 в выборке, если $e_2 \vee e_6 = 1$;

— наличие вершины v_2 в выборке, если $e_1 \vee e_2 \vee e_5 = 1$;

— наличие вершины v_3 в выборке, если $e_1 \vee e_3 \vee e_4 = 1$;

— наличие вершины v_4 в выборке $e_3 \vee e_7 = 1$;

— наличия вершины v_5 в выборке $e_4 \vee e_5 \vee e_6 \vee e_7 = 1$.

Условием наличия всех вершин в выборке является двойственная функция от (1)

$$(e_1 \vee e_3 \vee e_4)\,(e_1 \vee e_2 \vee e_5)\,(e_2 \vee e_6)\,(e_3 \vee e_7)\,(e_4 \vee e_5 \vee e_6 \vee e_7) = 1. \quad (2)$$

ISSN 2076-2429 (print)
ISSN 2223-3814 (on line)

В предлагаемых методах прямого и обратного перебора остовов графа можно выделить следующие этапы:

— запись топологии исходного графа в виде вершинно-реберной МБФ; определение количества вершин n и количества ребер m;

— определение вершины с минимальным числом ребер — базовой;

— упорядочивание ребер: ребра, входящие в базовую вершину, в методе прямого перебора записываются первыми, а в методе обратного перебора – последними; порядок записи остальных ребер не имеет значения;

— задание первоначального подграфа: выборка его ребер описывается как двоичное число (далее — *PodGraf)*, разряды которого соответствуют ребрам в порядке их записи, а значения 1 или 0 — наличию или отсутствию определенного ребра в подграфе соответственно; количество разрядов *PodGraf* равно m, число единиц — n–1; первоначально это двоичное число равно для метода обратного перебора 2^{n-1}–1, а для метода прямого перебора $2^{m-k}+2^{n-2}$–1, где k — количество ребер входящих в базовую вершину;

— для метода обратного перебора в подграфе, описываемом *PodGraf*, определяется наличие ребер, входящих в базовую вершину; *PodGraf* побитно умножается на число, соответствующее маске указанной вершины; маска вершины определяется следующим образом: всем разрядам, соответствующим ребрам, входящим в вершину присваивается значение 1, остальным разрядам — 0; если результат умножения равен 0, то подграф, описываемый *PodGraf* , не содержит базовую вершину, т.е не является остовом, и следует конец работы; иначе — переход к следующему этапу; для метода прямого перебора этот этап пропускается;

— последовательная проверка вхождения всех остальных вершин, кроме базовой, в полученный подграф (выполняется аналогично описанной на предыдущем этапе), если все вершины входят в подграф и этот подграф связный, то подграф — остов, описание которого фиксируется;

— выбор описания следующего подграфа и возврат для метода прямого перебора к предыдущему этапу, а для метода обратного перебора к этапу определения наличия ребер, входящих в базовую вершину.

Наиболее сложным для реализации является этап выбора описания следующего подграфа. В зависимости от метода полного перебора, предлагается два алгоритма реализации этого этапа. В алгоритме метода прямого перебора в *PodGraf* производится перемещение нулевого бита слева направо, а метода обратного перебора — перемещение единичного бита справа налево.

В обоих алгоритмах текущее описание задается двоичным числом *PodGraf*, количество ребер в остове — *kRebOst*, общее количество ребер в графе — *kReb*, проверочная однобитная маска для определения наличия соответствующего ребра в описании подграфа — *bit.* Результатом работы обоих алгоритмов служит число *res*, описывающее следующий подграф, если *res*≠0. Если *res*=0, то предыдущий подграф был последним, и перебор закончен.

Алгоритм реализации этапа выбора описания следующего подграфа в методе прямого перебора:

1. *res* присваивается первичное значение 1, *bit* — 1. Рабочим элементам i и k — значение 0.

2. Проверяется условие *bit*&*PodGraf*≠0 (проверка наличия единицы в младшем разряде *PodGraf*). Если условие не выполняется, то переход на шаг 6.

3. До тех пор пока выполняются условия *bit*&*PodGraf*=1 и *i*<*kRebOst*, единичный элемент в *bit* сдвигается влево на один разряд и инкрементируется значение i (т.е подсчитывается количество единиц, находящихся подряд в младших разрядах *PodGraf*).

4. Если *kRebOst*=*kReb* (т.е. исходный граф является остовом), то *re*s=0.

5. Если *res*≠0, то *PodGraf*=*PodGraf*⊕*bit*, в *bit* сдвигается вправо единичный элемент на один разряд, и еще раз выполняется операция *PodGraf*=*PodGraf*⊕*bit* (тем самым происходит сдвиг единичного разряда влево) и *res*=*PodGraf*. Переход на шаг 12.

6**.** Если *bit*&*PodGraf*=0, то в *bit* сдвигается влево единичный элемент на один разряд, k=k+1, и снова выполняется этот шаг.

7. Если *k*=*kReb*–*kRebOst*, т.е. проверяемый подграф последний, то *res*=0 — переход на шаг 12.

8. Если *res*≠0 то, пока *bit*&*PodGraf*≠0 и *i*<*kRebOst*, выполняются следующие действия *PodGraf* =*PodGraf*⊕*bit*, в *bit* сдвигается влево единичный элемент на один разряд, *i*=*i*+1.

9. *PodGraf*=*PodGraf*⊕*bit*, *bit*=1.

10. *PodGraf*=*PodGraf*⊕*bit*, в *bit* сдвигается влево единичный элемент на один разряд, шаг повторяется *i*–1 раз.

11. *res*=*PodGraf*.

12. Возвращается значение результата *res* в основную программу.

Результаты работы алгоритма метода прямого перебора графа (см. рисунок) приведены в табл. 1.

Таблица 1

Результат работы алгоритма метода прямого перебора остовов графа сети

№ итерации	Ребра							Наличие остова
	e_2	e_6	e_1	e_3	e_4	e_5	e_7	
1	2	3	4	5	6	7	8	9
1	0	0	0	1	1	1	1	Итерации 1...5 в алгоритме не выполняются
2	0	0	1	0	1	1	1	
3	0	0	1	1	0	1	1	
4	0	0	1	1	1	0	1	
5	0	0	1	1	1	1	0	
6	0	1	0	0	1	1	1	$e_4e_5e_6e_7$
7	0	1	0	1	0	1	1	$e_3e_5e_6e_7$
8	0	1	0	1	1	0	1	—
9	0	1	0	1	1	1	0	$e_3e_4e_5e_6$
10	0	1	1	0	0	1	1	$e_1e_5e_6e_7$
11	0	1	1	0	1	0	1	$e_1e_4e_6e_7$
12	0	1	1	0	1	1	0	—
13	0	1	1	1	0	0	1	$e_1e_3e_6e_7$
14	0	1	1	1	0	1	0	$e_1e_3e_5e_6$
15	0	1	1	1	1	0	0	$e_1e_3e_4e_6$
16	1	0	0	0	1	1	1	$e_2e_4e_5e_7$
17	1	0	0	1	0	1	1	$e_2e_3e_5e_7$
18	1	0	0	1	1	0	1	—
19	1	0	0	1	1	1	0	$e_2e_3e_4e_5$
20	1	0	1	0	0	1	1	$e_1e_2e_5e_7$
21	1	0	1	0	1	0	1	$e_1e_2e_4e_7$
22	1	0	1	0	1	1	0	—
23	1	0	1	1	0	0	1	$e_1e_2e_3e_7$
24	1	0	1	1	0	1	0	$e_1e_2e_3e_5$
25	1	0	1	1	1	0	0	$e_1e_2e_3e_4$
26	1	1	0	0	0	1	1	—
27	1	1	0	0	1	0	1	$e_2e_4e_6e_7$
28	1	1	0	0	1	1	0	—
29	1	1	0	1	0	0	1	$e_2e_3e_6e_7$
30	1	1	0	1	0	1	0	—
31	1	1	0	1	1	0	0	$e_2e_3e_4e_6$
32	1	1	1	0	0	0	1	$e_1e_2e_6e_7$
33	1	1	1	0	0	1	0	—
34	1	1	1	0	1	0	0	—
35	1	1	1	1	0	0	0	$e_1e_2e_3e_6$

ISSN 2076-2429 (print)
ISSN 2223-3814 (on line)

Итоговая функция описания остовов данного графа примет вид

$$e_4e_5e_6e_7 \vee e_3e_5e_6e_7 \vee e_3e_4e_5e_6 \vee e_1e_5e_6e_7 \vee e_1e_4e_6e_7 \vee e_1e_3e_6e_7 \vee e_1e_3e_5e_6 \vee$$
$$\vee e_1e_3e_4e_6 \vee e_2e_4e_5e_7 \vee e_2e_3e_5e_7 \vee e_2e_3e_4e_5 \vee e_1e_2e_5e_7 \vee e_1e_2e_4e_7 \vee e_1e_2e_3e_7 \vee \quad (3)$$
$$\vee e_1e_2e_3e_5 \vee e_1e_2e_3e_4 \vee e_2e_4e_6e_7 \vee e_2e_3e_6e_7 \vee e_2e_3e_4e_6 \vee e_1e_2e_6g \vee e_1e_2e_3e_6,$$

или в алфавитной последовательности

$$e_1e_2e_3e_4 \vee e_1e_2e_3e_5 \vee e_1e_2e_3e_6 \vee e_1e_2e_3e_7 \vee e_1e_2e_4e_7 \vee e_1e_2e_5e_7 \vee e_1e_2e_6e_7 \vee$$
$$\vee e_1e_3e_4e_6 \vee e_1e_3e_5e_6 \vee e_1e_3e_6e_7 \vee e_1e_4e_6e_7 \vee e_1e_5e_6e_7 \vee e_2e_3e_4e_5 \vee e_2e_3e_4e_6 \vee \quad (4)$$
$$\vee e_2e_3e_5e_7 \vee e_2e_3e_6e_7 \vee e_2e_4e_5e_7 \vee e_2e_4e_6e_7 \vee e_3e_4e_5e_6 \vee e_3e_5e_6e_7 \vee e_4e_5e_6e_7$$

Алгоритм реализации этапа выбора описания следующего подграфа в методе обратного перебора:

1. *res* присваивается первичное значение 1, *bit* присваивается значение 2^{m-1}, рабочему элементу *i* присваивается значение 1.

2. Проверяется *bit*&*PodGraf*≠0, Если условие не выполняется, то переход на шаг 5.

3. До тех пор пока выполняются условия *bit*&*PodGraf*≠0, и *i*<*kRebOst*+1, то *PodGraf*=*PodGraf*⊕*bit*, единичный элемент в *bit* сдвигается вправо на один разряд, инкрементируется значение *i*.

4. Если *i*=*kRebOst*+1, то *res*=0.

5. Если *res*≠1, то переход на шаг 10.

6. Если *bit*&*PodGraf*=0, в *bit* единичный элемент сдвигается вправо на один разряд, и снова выполняется этот шаг.

7. *PodGraf* =*PodGraf*⊕*bit*.

8. В *bit* единичный элемент сдвигается влево на один разряд, *PodGraf*=*PodGraf*⊕*bit*, шаг повторяется *i* раз.

9. *res*=*PodGraf*.

10. Возвращается значение результата *res* в основную программу.

Поскольку разряды ребер базовой вершины являются последними, то невыполнения условия вхождения базовой вершины в подграф (итерация 31) служит концом работы алгоритма. Все остальные комбинации уже не будут содержать ребра, связывающие базовую вершину с остальными.

Результаты работы алгоритма метода обратного перебора графа (см. рисунок) приведены в табл. 2.

Итоговая функция описания остовов данного графа методом обратного перебора

$$e_2e_4e_6e_7 \vee e_2e_3e_6e_7 \vee e_1e_2e_6e_7 \vee e_2e_3e_4e_6 \vee e_1e_2e_3e_6 \vee e_4e_5e_6e_7 \vee e_3e_5e_6e_7 \vee$$
$$e_1e_5e_6e_7 \vee e_1e_4e_6e_7 \vee e_1e_3e_6e_7 \vee e_3e_4e_5e_6 \vee e_1e_3e_5e_6 \vee e_1e_3e_4e_6 \vee e_2e_4e_5e_7 \vee \quad (5)$$
$$e_2e_3e_5e_7 \vee e_1e_2e_5e_7 \vee e_1e_2e_4e_7 \vee e_1e_2e_3e_7 \vee e_2e_3e_4e_5 \vee e_1e_2e_3e_5 \vee e_1e_2e_3e_4,$$

после упорядочения в алфавитной последовательности, аналогично (4).

Таким образом, оба метода приводят к одной функции, описывающей остовы графа (см. рисунок), которую назовем остовной МБФ [9].

В методе прямого перебора последовательность проверяемых подграфов соответствует возрастающей последовательности всех *m* — разрядных двоичных чисел с *n*–1 единицей. Докажем, что обратный алгоритм также перебирает все подграфы с *n*-1 ребром рассматриваемого графа, как и прямой.

Лемма 1. В методе обратного перебора перебираются все возможные подграфы из заданного количества ребер.

Доказательство. Сравниваем результаты работы рассмотренных алгоритмов. Каждая *i*-я итерация (см. таблицу 2) является зеркальным отражением (*k*–*i*+1)-й итерации (см. таблицу 1), где *k* — общее количество итераций в таблице.

Таким образом, можно сделать вывод, что метод прямого перебора более эффективен, чем метод обратного перебора. Однако последний проще реализуется.

Таблица 2

Реализация обратного перебора остовов графа сети

№ итерации	Ребра							Наличие остова
	e_1	e_3	e_4	e_5	e_7	e_2	e_6	
1	2	3	4	5	6	7	8	9
1	0	0	0	1	1	1	1	—
2	0	0	1	0	1	1	1	$e_2e_4e_6e_7$
3	0	1	0	0	1	1	1	$e_2e_3e_6e_7$
4	1	0	0	0	1	1	1	$e_1e_2e_6e_7$
5	0	0	1	1	0	1	1	—
6	0	1	0	1	0	1	1	—
7	1	0	0	1	0	1	1	—
8	0	1	1	0	0	1	1	$e_2e_3e_4e_6$
9	1	0	1	0	0	1	1	—
10	1	1	0	0	0	1	1	$e_1e_2e_3e_6$
11	0	0	1	1	1	0	1	$e_4e_5e_6e_7$
12	0	1	0	1	1	0	1	$e_3e_5e_6e_7$
13	1	0	0	1	1	0	1	$e_1e_5e_6e_7$
14	0	1	1	0	1	0	1	—
15	1	0	1	0	1	0	1	$e_1e_4e_6e_7$
16	1	1	0	0	1	0	1	$e_1e_3e_6e_7$
17	0	1	1	1	0	0	1	$e_3e_4e_5e_6$
18	1	0	1	1	0	0	1	—
19	1	1	0	1	0	0	1	$e_1e_3e_5e_6$
20	1	1	1	0	0	0	1	$e_1e_3e_4e_6$
21	0	0	1	1	1	1	0	$e_2e_4e_5e_7$
22	0	1	0	1	1	1	0	$e_2e_3e_5e_7$
23	1	0	0	1	1	1	0	$e_1e_2e_5e_7$
24	0	1	1	0	1	1	0	—
25	1	0	1	0	1	1	0	$e_1e_2e_4e_7$
26	1	1	0	0	1	1	0	$e_1e_2e_3e_7$
27	0	1	1	1	0	1	0	$e_2e_3e_4e_5$
28	1	0	1	1	0	1	0	—
29	1	1	0	1	0	1	0	$e_1e_2e_3e_5$
30	1	1	1	0	0	1	0	$e_1e_2e_3e_4$
31	0	1	1	1	1	0	0	Не выполнена проверка по наличию в подграфе 1 узла
32	1	0	1	1	1	0	0	Итерации 32...35 в алгоритме не выполняются
33	1	1	0	1	1	0	0	
34	1	1	1	0	1	0	0	
35	1	1	1	1	0	0	0	

При сравнении существующих комбинаторных методов [5, 7] с предлагаемыми видно, что для указанной задачи реализация полного неупорядоченного перебора потребует 7^4 или 2401 итерацию, метод упорядоченного перебора потребует 7·6·5·4=840 итераций.

Рассмотрим задачу выбора решения о проведении последовательности ремонта в телекоммуникационной сети, описанной графом (см. рисунок). Перебор всех остовов позволяет найти ребра, чаще всего входящие в остовы, т.е. ребра (связи, каналы, маршруты), выход которых из строя может привести к максимальным потерям при функционировании телекоммуникационных сетей.

Исследуя итоговую функцию описания остовов (4), можно отметить, что ребро e_1 встречается в функции 12, e_2 — 13, e_3 — 13, e_4 — 10, e_5 — 10, e_6 — 13, e_7 — 13 раз, т.е. в первую очередь необходимо уделить внимание ребрам e_2, e_3, e_6, e_7, затем ребру e_1, и в конце — ребрам e_4 и e_5.

ISSN 2076-2429 (print)
ISSN 2223-3814 (on line)

Для реализации данных методов написаны программы для ПЭВМ, позволяющие находить остовы графов содержащих до 32 ребер.

В целом, применение предлагаемых методов поиска остовов графа позволяет упростить решение задач исследования и проектирования телекоммуникационных сетей.

Литература

1. Свами, М. Графы, сети и алгоритмы / М. Свами, К. Тхуласираман. — М.: Мир, 1984. — 455 с.
2. Захарченко, Н.В. Оптимизация и моделирование систем связи: Учеб. пособие / Н.В. Захарченко, Н.А. Князева. — Одес. электротехн. ин-т связи. — Одесса, 1990. Ч. 2. — 76 с.
3. Тихонов, В.И. Фрактальная топологическая модель открытой телекоммуникационной сети / В.И. Тихонов // Наук. пр. ОНАЗ ім. О.С. Попова. — 2010. — № 1. — С. 31 — 38 .
4. Липский, В. Комбинаторика для программистов: пер. с пол. / В. Липский. — М.: Мир, 1998. — 213 с.
5. Кристофидес, Н. Теория графов: алгоритмический подход / Н. Кристофидес. — М.: Мир, 1978. — 430 с.
6. Костюкова, Н.И. Графы и их применение. Комбинаторные алгоритмы для программистов: Учеб. пособие / Н.И. Костюкова, интернет-ун-т информ. технологий. — М.: БИНОМ: Лаборатория знаний, 2010. — 311 с.
7. Виленкин, Н.Я. Комбинаторика / Н.Я. Виленкин. — М.: Наука, 1969. — 328 с.
8. Ткаченко, В.Г. Отказы цифровых схем и представления монотонных булевых функций / В.Г. Ткаченко // Наук. пр. ОНАЗ ім. О.С. Попова. — 2006. — № 2. — С. 54 — 69.
9. Иваницкий, А.М. Взаимосвязь между матроидами и монотонными булевыми функциями электрических цепей / А.М. Иваницкий, В.Г. Ткаченко // Наук. пр. ОНАЗ ім. О.С. Попова. — 2009. — № 1. — С. 18 — 26.

References

1. Svami, M. Grafy, seti i algoritmy [Graphs, networks and algorithms] / M. Svami, K. Tkhulasiraman. — Moscow, 1984. — 455 pp.
2. Zakharchenko, N.V. Optimizatsiya i modelirovanie sistem svyazi: Ucheb. Posobie [Optimization and modeling of communication systems: tutorial] / N.V. Zakharchenko, N.A. Knyazeva. — Odes. elektrotekhn. in-t svyazi. [Odessa electrical engineering institute of communications] — Odessa, 1990. Part 2. — 76 pp.
3. Tikhonov, V.I. Fraktal’naya topologicheskaya model’ otkrytoy telekommunikatsionnoy seti [Fractal topological model of an open telecommunication network] / V.I. Tikhonov // Nauk. pr. ONAZ im. O.S. Popova. [Proceedings of ONAC named after O.S. Popov] — 2010. — № 1. — P. 31 — 38 .
4. Lipskiy, V. Kombinatorika dlya programmistov: per. s pol. [Combinatorics for programmers: transl. from Polish] / V. Lipskiy. — Moscow, 1998. — 213 pp.
5. Kristofides, N. Teoriya grafov: algoritmicheskiy podkhod [Graph theory: algorithmic approach] / N. Kristofides. — Moscow, 1978. — 430 pp.
6. Kostyukova, N.I. Grafy i ikh primenenie. Kombinatornye algoritmy dlya programmistov: Ucheb. posobie [Graphs and their application. Combinatoric algorithms for programmers: tutorial] / N.I. Kostyukova, internet-un-t inform. tekhnologiy. [IT internet university] — Moscow, 2010. — 311 pp.
7. Vilenkin, N.Ya. Kombinatorika [Combinatorics] / N.Ya. Vilenkin. — Moscow, 1969. — 328 pp.
8. Tkachenko, V.G. Otkazy tsifrovykh skhem i predstavleniya monotonnykh bulevykh funktsiy [Digital circuits failures and monotonous Boolean functions presentations]/ V.G. Tkachenko // Nauk. pr. ONAZ im. O.S. Popova [Proceedings of ONAC named after O.S. Popov] — 2006. — № 2. — P. 54 — 69.
9. Ivanitskiy, A.M. Vzaimosvyaz’ mezhdu matroidami i monotonnymi bulevymi funktsiyami elektricheskikh tsepey [Interrelation between matroids and monotonous Boolean functions in electric circuits] / A.M. Ivanitskiy, V.G. Tkachenko // Nauk. pr. ONAZ im. O.S. Popova [Proceedings of ONAC named after O.S. Popov] — 2009. — № 1. — P. 18 — 26.

Рецензент д-р техн. наук, проф. Одес. нац. акад. связи им. А.С. Попова Лесовой И.П.

Поступила в редакцию 4 мая 2011 г.