



УДК 519.1

ВЫЧИСЛЕНИЕ ИНДЕКСА ДОСТУПНОСТИ И ОКРЕСТНОСТНОЙ ЦЕЛОСТНОСТИ ГРАФА

П. Дюндар, А. Айтак, В. Айтак

Вычисляются значения устойчивости графов типа “Бабочка” и “Омега”, используемых при создании компьютерных сетей. Получены соотношения между окрестностной целостностью и индексом доступности сети.

Библиография: 11 названий.

Надежность и эффективность служат важными критериями при проектировании коммуникационных сетей. Уязвимость коммуникационной сети показывает способность сети поддерживать связь между узлами при разрушении некоторых узлов или линий связи. При изменении числа узлов в подсетях данной сети ее уязвимость также меняется, а это означает, что надо повышать устойчивость системы или, что тоже самое, понижать ее уязвимость. При изображении сети графом G анализ наихудшего случая некоторых аспектов всего процесса разрушения связей осуществляется с помощью детерминированных мер. Для описания устойчивости коммуникационных сетей используются различные параметры графов. При потере узла в шпионской сети связанные с ним узлы оказываются преданными и, тем самым, бесполезными для всей остальной сети. Определение окрестностной целостности графа основано на этом представлении.

Для измерения устойчивости графа используются такие понятия как связность, число покрытия и индекс доминантности. В этой статье мы рассматриваем окрестностную целостность и индекс доступности. Сначала мы дадим некоторые определения и обозначения и приведем алгоритм вычисления индекса доступности графа. Затем мы вычислим значения устойчивости графов Бабочка и Омега, используемых при создании компьютерных сетей. Мы также опишем некоторые соотношения между окрестностной целостностью и индексом доступности в этих сетях.

1. Введение. Сеть можно моделировать графом, вершины которого соответствуют станциям сети, а ребра – линиям связи. Уязвимость измеряет способность сети сопротивляться разрушению при отказе некоторых станций или линий связей. Потеря звеньев и узлов постепенно приводит к потере эффективности сети. Поэтому коммуникационные сети должны быть по возможности устойчивы, причем не только к разрушению, но и к возможному перепроектированию. Для характеристики устойчивости коммуникационных сетей используются разнообразные параметры графов, включая связность, целостность, жесткость и индекс связи. Однако эти параметры не отражают воздействия удаляемой вершины на ее соседей. При разрушении станции примыкающие к ней

станции оказываются преданными и бесполезными для остальной сети. Окрестностная целостность измеряет уязвимость графа по отношению к разрушениям, вызванным удалением вершин вместе со всеми их соседями [1]–[7].

Граф G – это пара $G = (V(G), E(G))$, где $V(G)$ и $E(G)$ – это соответственно множество вершин и множество ребер графа G . Пусть u – вершина графа G . Обозначим через $N(u) = \{\nu \in V(G) \mid u \neq \nu, \nu \text{ примыкает к } u\}$ открытую окрестность вершины u и через $N[u] = \{u\} \cup N(u)$ замкнутую окрестность вершины. Удаление замкнутой окрестности $N[u]$ вершины u графа G называется *разрушением* вершины. Набор вершин графа G , замкнутые окрестности которых должны быть удалены из графа, называется *стратегией разрушения* графа. Пусть S – такая стратегия. Обозначим через $G \setminus S$ выживший подграф; тогда *окрестностной целостностью* $NI(G)$ графа G называется величина

$$NI(G) = \min_{S \subseteq V(G)} \{|S| + c(G \setminus S)\}.$$

где S пробегает все вершинные стратегии разрушения, а $c(G \setminus S)$ обозначает максимум порядков компонент связности графа $(G \setminus S)$ [1]. Стратегия S называется *NI-множеством*, если минимум достигается на ней.

Сети Бабочка и Омега строятся из $\log p$ уровней. При этом вершины сети Омега можно переупорядочить таким образом, что она будет выглядеть как сеть Бабочка – важный класс параллельных сетей.

В этой статье мы обсуждаем введенные в [3] понятия окрестностной целостности и индекса доступности для сетей Бабочка и Омега. Сначала мы дадим некоторые определения и обозначения, а затем вычислим значения некоторых параметров устойчивости для сетей Бабочка и Омега.

Через $\lceil y \rceil$ мы обозначаем наименьшее целое число, не меньшее чем y , а через $\lfloor y \rfloor$ – наибольшее целое число, не превосходящее y .

Говорят, что вершина и ребро в графе G покрывают друг друга, если они инцидентны. *Покрытием* графа G называется подмножество его вершин, покрывающее все его ребра. Минимальное количество вершин в покрытии называется его *индексом покрытия* и обозначается через $\alpha(G)$ [8].

Набор вершин графа G называется *независимым*, если его элементы попарно не примыкают друг к другу. *Индексом независимости* $\beta(G)$ графа G называется максимальное число элементов в независимом множестве вершин. Порядок n графа равен сумме индекса покрытия и индекса независимости, $\alpha(G) + \beta(G) = n$ [8].

Множество S вершин графа G называется *вершинно доминирующим*, если всякая вершина графа либо лежит в S , либо примыкает к вершине из S . Минимальное количество вершин в доминирующем множестве называется *индексом доминантности* графа и обозначается через $\sigma(G)$. Для всякого графа G выполняется неравенство $\sigma(G) \leq \beta(G)$ [8].

ОПРЕДЕЛЕНИЕ 1. Подмножество S множества $V(G)$ называется *множеством доступности* для графа G , если всякая вершина $x \in \{V(G) - S\}$ примыкает к $N[S]$, где через $N[S]$ обозначена замкнутая окрестность множества S . *Индексом доступности* называется минимальное количество вершин в множестве доступности; это число обозначается через $\eta(G)$ [6].

2. Основные свойства индекса доступности и окрестностной целостности. В этом разделе мы приводим ряд известных результатов о свойствах индекса доступности и окрестностной целостности.

Ниже выписаны индексы доступности для некоторых основных семейств графов [6]:

- 1) для цепочки P_n на n вершинах $\eta(P_n) = \lfloor n/5 \rfloor + 1$;
- 2) для цикла C_n на n вершинах $\eta(C_n) = \lfloor n/5 \rfloor + 1$;
- 3) для полного графа K_n на n вершинах $\eta(K_n) = 1$;
- 4) для звезды $K_{1,n}$ на $n + 1$ вершинах $\eta(K_{1,n}) = 1$;
- 5) для полного многодольного графа $K_{m,n,\dots,t}$ на $m + n + \dots + t$ вершинах $\eta(K_{m,n,\dots,t}) = 1$;
- 6) для колеса $W_{1,n}$ на $n + 1$ вершинах $\eta(W_{1,n}) = 1$;

Индекс доступности графа обладает следующими свойствами [6].

ТЕОРЕМА 1. *В том и только в том случае $\eta(G) = 1$, если G содержит остовный подграф, являющийся звездой.*

ТЕОРЕМА 2. *Всякое множество доступности в графе является либо независимым, либо покрывающим, либо доминирующим.*

ТЕОРЕМА 3. *Пусть G – связный простой граф. Тогда $\eta(G) < \beta(G)$, где через $\beta(G)$ обозначен индекс независимости графа G .*

ТЕОРЕМА 4. *Для всякого связного графа G выполняется неравенство $\eta(G) < \alpha(G)$, где через $\alpha(G)$ обозначен индекс покрытия графа G .*

ТЕОРЕМА 5. *Пусть G – связный простой граф. Тогда $\eta(G) < \sigma(G)$, где через $\sigma(G)$ обозначен индекс доминантности графа G .*

Окрестностная целостность графа обладает следующими свойствами.

СЛЕДСТВИЕ 1. *Индекс независимости графа G служит верхней оценкой для $NI(G)$ [1].*

СЛЕДСТВИЕ 2. *Если для данного графа G существует независимое множество I , замкнутая окрестность которого совпадает со всем множеством вершин $V(G)$, то $NI(G)$ не совпадает с индексом независимости графа G [1].*

СЛЕДСТВИЕ 3. *В том и только в том случае $NI(G) = 1$, если либо в G есть остовный подграф, являющийся звездой, либо G представляет собой множество изолированных вершин [1].*

Ниже выписаны значения окрестностной целостности для некоторых основных семейств графов:

- 1) для полного графа K_n на n вершинах $NI(K_n) = 1$;
- 2) для колеса $W_{1,n}$ на $n + 1$ вершинах, состоящего из n -цикла и одной дополнительной вершины, соединенной со всеми вершинами цикла, $NI(W_{1,n}) = 1$;
- 3) для цепочки P_n на n вершинах

$$NI(P_n) = \begin{cases} \lfloor 2\sqrt{n+3} \rfloor - 4, & \text{если } n \geq 2, \\ 1, & \text{если } n = 1; \end{cases}$$

4) для цикла C_n на n вершинах

$$NI(C_n) = \begin{cases} \lceil 2\sqrt{n} \rceil - 3, & \text{если } n \geq 5, \\ 2, & \text{если } n = 4, \\ 1, & \text{если } n = 3. \end{cases}$$

3. Алгоритм построения множества доступности и подсчета индекса доступности графа. В этом разделе мы приводим алгоритм вычисления индекса доступности графа.

T : array of V B : $\{0, 1\}$ L, M : subset of $q + B^*$
 E : array $n^+ \times n^+$ of B Q : $\{1, 2, \dots, \text{piece}\}$ σ : element of B
 v : array B of V q : element of Q x : element of M
 $\eta, i, j, k, \text{piece}$: positive integer $FLoft$: array of V
 n : positive integer (число вершин в G) FL : set of $FLoft$

```

begin
   $T \leftarrow \emptyset$ ;  $T[\epsilon] \leftarrow V$ ;  $L \leftarrow \{\epsilon\}$ ;
  firstvertex  $\leftarrow v_1$ ;  $FLoft[1] \leftarrow v_1$ ; piece  $\leftarrow 1$ ;
  for  $i \leftarrow 2$  to  $n$  do
    if  $E[i, v_1] = 1$  then begin
      piece  $\leftarrow$  piece + 1;
       $FLoft[i] \leftarrow v_i$ ;
    end;

   $FL \leftarrow \{\epsilon\}$ ;
  for  $i \leftarrow 1$  to piece do  $FL \leftarrow FL \cup FLoft[i]$ ;
  for  $i \leftarrow 1$  to piece do begin
     $T[i] \leftarrow FLoft[i] + (T[\epsilon] \sim FL)$ ;
     $L \leftarrow L \cup i$ ;
  end;

  for  $j \leftarrow 1$  to  $n - 1$  do
  for  $i \leftarrow j + 1$  to  $n$  do
    if  $E[i, j] = 1$  then begin
      for  $k \leftarrow i + 1$  to  $n$  do
        if  $E[k, i] = 1$  then begin
           $M \leftarrow \{x \in L : \{v_k, v_j\} \subseteq T[u] \text{ for all } u \in L\}$ ;
           $L \leftarrow L \sim M$ ;
           $v[0] \leftarrow v_k$ ;
           $v[1] \leftarrow v_j$ ;
          for  $x \in M$  do
            for  $\sigma \in B$  do
              begin
                 $T[x\sigma] \leftarrow T[x] - \{v[\sigma]\}$ ;
                if  $T[x\sigma] \not\subseteq T[y]$  for all  $y \in L$  then
                   $L \leftarrow L \cup [x\sigma]$ ;
              end;
            end;
          end;
        end;
      end;
    end;
  end;
end;
```

```

 $\eta \leftarrow \min_{x \in L} \{|T[X]|\};$ 
end.

```

Этот алгоритм вычисляет индекс доступности η для любого графа $G = (V, E)$ одновременно со списками множеств доступности. Мы выбираем минимальный из них.

Вершины дерева помечены подмножествами в V , множестве вершин графа $G = (V, E)$. Изначально корень дерева T помечен множеством V целиком. Подмножество $L \subseteq q + B^*$ содержит список листьев в процессе разметки. Первый цикл `for` заносит первые листья дерева (*FLoFT*). Их число не превосходит n . Поэтому при первой пометке имеем $L = \{1, 2, 3, \dots, \text{piece}\}$, $\text{piece} \leq n$. Затем мы можем пометить ребра таким образом, чтобы ориентировать ветви слева направо. После завершения первого цикла мы получаем двоично-помеченное дерево. В циклах `for $j \leftarrow 1$ to $n - 1$ do` и `for $i \leftarrow j + 1$ to n do` идет поиск (по нижне-треугольной матрице представления графа) пар вершин, для которых $v_i E v_j$ ($E[i, j] = 1$). Это означает, что мы исследуем вершины v_k , примыкающие к вершине v_i , и подсчитываем соответствующие индексы доступности.

Множество M представляет собой множество всех листьев дерева, метки которых содержат построенную пару в качестве подмножества. Затем множество M удаляется из L , поскольку множество, содержащее такую пару, не может быть независимым. Пусть $x \in q + B^*$ — одна из таких помеченных вершин. Последовательность команд, следующая за циклами `for $j \leftarrow 1$ to $n - 1$ do` и `for $i \leftarrow j + 1$ to n do`, помечает вершины ниже, левее и правее вершины x максимальными подмножествами в $T[x]$, не содержащими обеих вершин v_i и v_j одновременно (см. рис. 1).

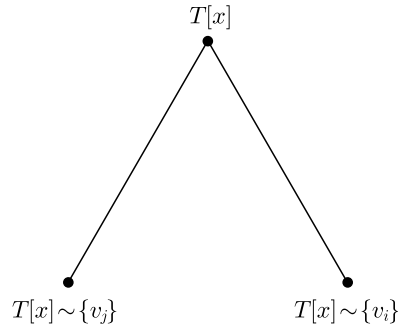


Рис. 1

На этом же этапе мы проверяем, не содержатся ли пометки этих новых листьев в пометках остальных помеченных листьев дерева. Интересуют же нас лишь минимальные множества доступности. Множество L модифицируется в зависимости от ответа на этот запрос. Затем мы готовы исследовать новый элемент $E[i, j] = 1$ такой, что $E[k, j] = 1$. В целом бинарное дерево следует представлять себе как состоящее из тех узлов, которые в тот или иной момент оказывались элементами множества L , листьями строящегося дерева.

Видно, что в алгоритме использована техника, аналогичная методу ветвей и границ. Подобно всем реализациям метода ветвей и границ в наихудшем случае число операций пропорционально экспоненте длины входных данных задачи. Однако проведенные нами эксперименты с малыми размерами входных данных показывают, что алгоритм работает быстро. Нам представляется, что для большинства конкретных классов графов алгоритм будет работать полиномиальное время.

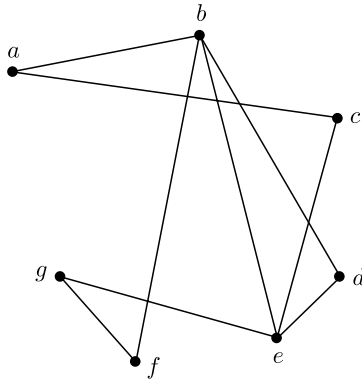


Рис. 2

ПРИМЕР 1. Для иллюстрации различных шагов алгоритма рассмотрим граф с рис. 2. Сначала помечен только корень будущего дерева, как показано на рис. 3а. В этот момент $L = \{e\}$. Затем мы начнем исследовать содержимое нижнетреугольной матрицы:

b	1						
c	1	0					
d	0	1	0				
e	0	1	1	1			
f	0	1	0	0	0		
g	0	0	0	0	1	1	
		a	b	c	d	e	f

Встреченные нами в первом столбце единицы показывают, что aEb и aEc . Посмотрим, помечены ли какие-либо из листьев подмножествами, содержащими множество $\{a, b, c\}$. На рис. 3b показано, как выполняется необходимая операция расщепления, упомянутая на рис. 2. Теперь наше множество листьев имеет вид $L = \{1, 2, 3\}$. Первая встреченная нами единица во втором столбце, в котором указаны вершины, примыкающие к b , соответствует отношению aEd . Имеется один лист ($x = 1$), для которого $\{a, d\} \subseteq T[x] = \{a, d, e, f, g\}$. Поэтому $M = \{1\}$, и этот узел удаляется из L . Теперь алгоритм выполняет присваивания

$$T[10] = T[x0] \leftarrow T[x] \sim \{d\} = \{a, e, f, g\},$$

$$T[11] = T[x1] \leftarrow T[x] \sim \{a\} = \{d, e, f, g\}.$$

Следующая строка проверяет, содержатся ли новые метки в метках других листьев дерева. Поскольку $T[11] \subseteq T[2]$ при $0 \in L$, узел 11 не становится новым узлом дерева (см. рис. 3c). Теперь мы получаем $L = \{2, 3, 10\}$. Результат расщепления для следующего отношения aEe показан на рис. 3d, и в этот момент $L = \{2, 3, 101\}$.

При исполнении алгоритма вручную все дерево может быть изображено на одной схеме, которая в рассматриваемом нами примере приведена на рис. 4.

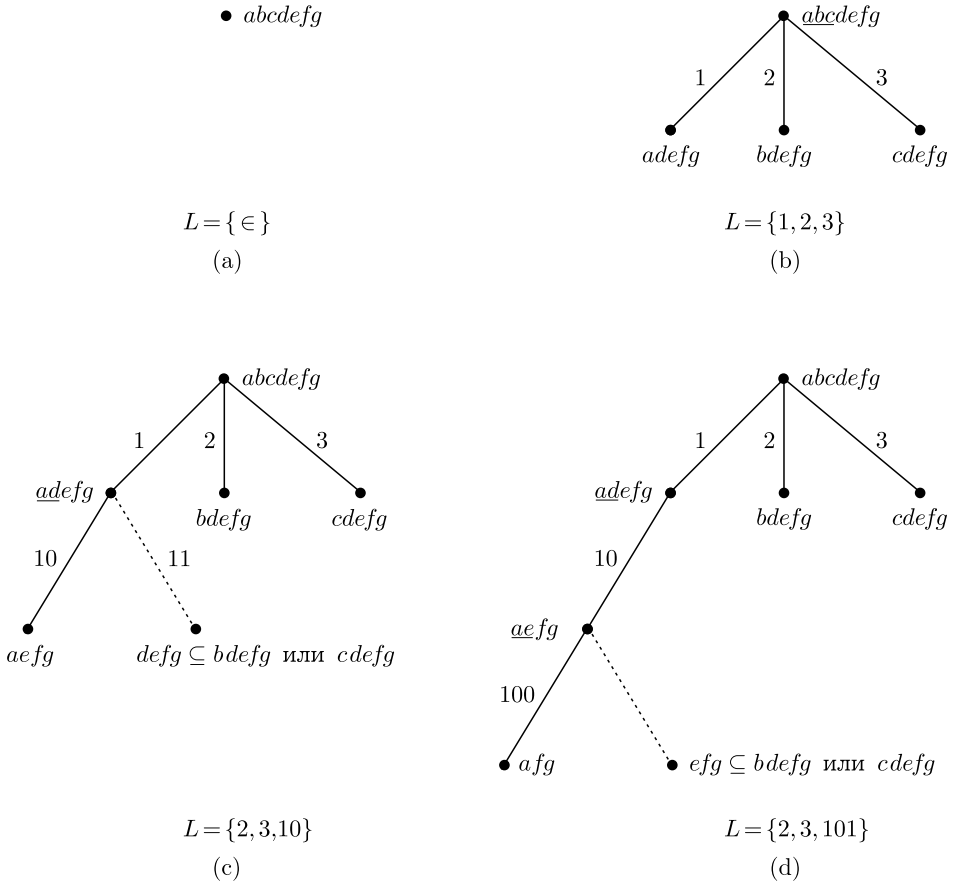


Рис. 3

В заключение мы получаем $L = \{1000, 20000, 300, 3110\}$ и, наконец,

$$\eta = \min_{x \in L} \{|T[x]|\} = 1.$$

На самом деле метки всех листьев, найденные по завершении работы алгоритма, соответствуют минимальным подмножествам доступности. Здесь это множества доступности

$$\{a, g\}, \{b\}, \{c, d, f\}, \{e, f\}$$

соответственно [9].

4. Вычисление индекса доступности и окрестностной целостности сетей Бабочка и Омега. В этом разделе мы рассматриваем результаты об окрестностной целостности и индексе доступности сетей Бабочка и Омега.

ОПРЕДЕЛЕНИЕ 2 [10]. В литературе сети Бабочка известны так же, как банановые сети [10]. *Сеть Бабочка* B_n – это граф с вершинами $x = (x_0, x_1, \dots, x_n)$, $0 \leq x_0 \leq n$ и $x_i \in \{0, 1\}$ при $1 \leq i \leq n$, причем две вершины x и y примыкают друг к другу, если и только если $y_0 = x_0 + 1$ и $x_i = y_i$ при $1 \leq i \leq n, i \neq y_0$. Отметим, что B_1 – это 4-цикл.

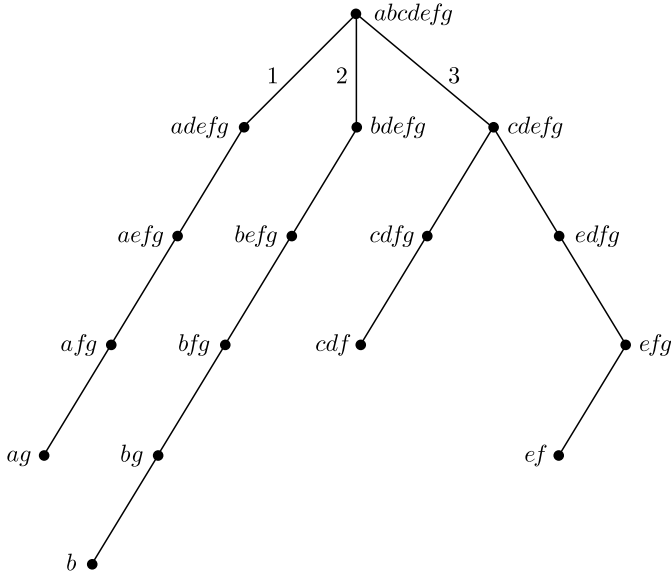


Рис. 4

Для вершины $x = (x_0, x_1, \dots, x_n)$ в B_n будем говорить, что x находится на уровне x_0 сети B_n и назовем значение x_i i -й координатой вершины x .

На рис. 5 приведен пример сети B_3 , верхняя строка которой указывает уровни, а левый столбец – имена (x_0, x_1, \dots, x_n) .

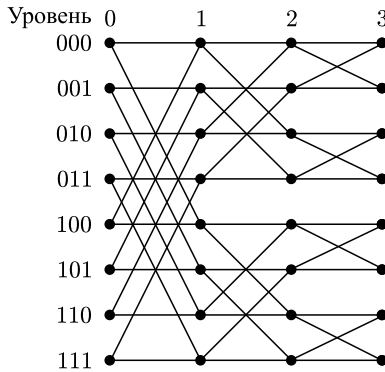


Рис. 5. Сеть Бабочка B_3

ОПРЕДЕЛЕНИЕ 3. Обычно используемая многоуровневая коммуникационная сеть это *сеть Омега* [11]. Эта сеть состоит из $\log p$ уровней. Каждый уровень в сети Омега состоит из внутриуровневой схемы, соединяющей p входов с p выходами; вход i соединяется с выходом j , если выполняются соотношения

$$j = \begin{cases} 2i, & \text{если } 0 \leq i \leq p/2 - 1, \\ 2i + 1 - p, & \text{если } p/2 \leq i \leq p - 1. \end{cases}$$

Эти равенства означают, что для получения j по i необходимо совершить левое вращение двоичного представления числа i . Такая схема связи называется *совершенным*

тасованием. На каждом уровне сети Омега схема совершенного тасования порождает набор из $p/2$ узлов.

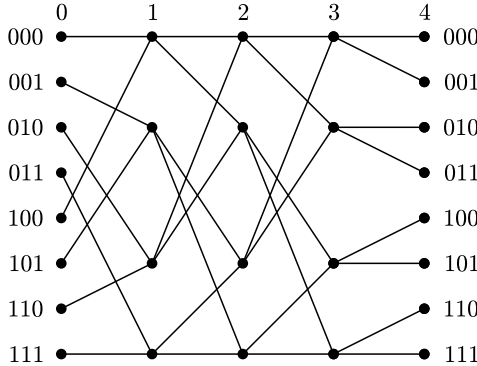


Рис. 6. Сеть Омега O_3 для восьми процессоров

Рисунок 6 показывает сеть Омега для восьми процессоров. Процессоры являются входными узлами сети и банками памяти для выходных узлов. Рутинные сообщения в сети Омега создаются по простой схеме. Пусть s и t – двоичные представления источника и адресата сообщения. Сообщение проходит по линии связи к первому узлу. Если ведущие биты адресов s и t совпадают, то узел просто пропускает это сообщение. Если эти биты различны, то сообщение переадресуется. Затем та же процедура повторяется на следующем уровне вершин с использованием следующего по значимости бита. Сеть Омега обозначается через (O_p) , где p – число уровней.

ТЕОРЕМА 6. Пусть B_n – сеть Бабочка с $2^n(n + 1)$ вершинами. Тогда

$$NI(B_n) = 2^{n-1} + 2^{n-4}(n - 3) \quad \text{для всех } n \geq 5.$$

ДОКАЗАТЕЛЬСТВО. Сеть Бабочка имеет рекурсивную структуру. Так, сеть B_n состоит из двух копий сети B_{n-1} . При удалении r вершин со второго уровня сети Бабочка B_n возможны три ситуации в зависимости от величины r . Пусть S – стратегия вершинного разрушения сети B_n .

Случай 1. При $|S| = 2^{n-1}$ одна из оставшихся компонент связности графа имеет вид B_{n-4} . Поэтому

$$NI(B_n) = \min_{S \subseteq V(G)} \{|S| + c(G \setminus S)\} = 2^{n-1} + V(B_{n-4}).$$

Случай 2. При $|S| > 2^{n-1}$ в одной из оставшихся компонент связности графа, по меньшей мере, $V(B_{n-4})$ вершин. Поэтому

$$NI(B_n) = \min_{S \subseteq V(G)} \{|S| + c(G \setminus S)\} > 2^{n-1} + V(B_{n-4}).$$

Случай 3. При $|S| < 2^{n-1}$, т.е. при $|S| = 2^{n-1} - a$ (a – постоянная, $a = 1, 2, \dots, 2^{n-1} - 1$), справедливо равенство $c(G/S) = 4V(B_{n-4}) + 4a$, поскольку одна из вершин на втором уровне сети B_n примыкает по меньшей мере к трем вершинам. Поэтому

$$\begin{aligned} NI(B_n) &= \min_{S \subseteq V(G)} \{|S| + c(G \setminus S)\} = 2^{n-1} - a + 4V(B_{n-4}) + 4a \\ &= 2^{n-1} + 4V(B_{n-4}) + 3a. \end{aligned}$$

ТАБЛИЦА 1

n	1	2	3	4
$NI(B_n)$	2	3	5	9

Следовательно, $NI(B_n) = 2^{n-1} + 2^{n-4}(n - 3)$ для всех $n \geq 5$.

Значения величины $NI(B_n)$ при $n < 5$ приведены в таблице 1.

ТЕОРЕМА 7. Пусть O_p – сеть Омега с $2^{p+1} + 2^{p-1}p$ вершинами. Тогда

$$NI(O_p) = \begin{cases} 2^{p-2} \lceil p/4 \rceil + 3, & \text{если } p = 4m \quad (m = 1, 2, 3, \dots), \\ 2^{p-2} \lceil p/4 \rceil + 1 & \text{в противном случае.} \end{cases}$$

ДОКАЗАТЕЛЬСТВО. Всякая сеть Омега O_p состоит из тех же уровней, что и сеть Бабочка. При удалении r вершин с уровнем сети O_p возможны три варианта. По определению окрестностной целостности, число уровней равно $\lceil p/4 \rceil$. Пусть S – стратегия вершинного разрушения сети O_p .

Случай 1. При $|S| = 2^{p-2} \lceil p/4 \rceil$ одна из оставшихся компонент связности содержит, по крайней мере, одну или три вершины в зависимости от p , откуда и вытекают выписанные выше равенства.

Случай 2. При $|S| > 2^{p-2} \lceil p/4 \rceil$ имеем $c(G/S) = 3$ или 1. Поэтому значение окрестностной целостности больше, чем в случае 1.

Случай 3. При $|S| < 2^{p-2} \lceil p/4 \rceil$ имеем $c(G/S) > 3$. Например, при $|S| = 2^{p-2} \lceil p/4 \rceil - 1$ справедливо равенство $c(G/S) = 4 * 3 + 1 + A$, где A – число вершин, примыкающих к удаляемой вершине.

ТЕОРЕМА 8. Пусть B_n – сеть Бабочка с $2^n(n + 1)$ вершинами. Тогда

$$\eta(B_n) = \left(\left\lfloor \frac{n}{5} \right\rfloor + 1 \right) 2^{n-1}.$$

ДОКАЗАТЕЛЬСТВО. Сеть Бабочка представляет собой многоуровневый граф. В этом графе каждый уровень соединен с каждым. Из определения доступности вытекает, что удаление вершин на любом из уровней оказывает влияние самое большее на пять уровней, включая уровень удаляемых вершин. Для того чтобы найти индекс доступности в сети Бабочка, нужно, в первую очередь, рассмотреть удаление минимального числа вершин со второго уровня. Со всех уровней надо удалить 2^{n-1} вершин. Поэтому

$$\eta(B_n) = \left(\left\lfloor \frac{n}{5} \right\rfloor + 1 \right) 2^{n-1}.$$

СЛЕДСТВИЕ 4. Окрестностная целостность и индекс доступности в сети Бабочка удовлетворяют неравенству

$$NI(B_n) \leq \eta(B_n) + 1.$$

ТЕОРЕМА 9. Для сети Омега O_p имеем

$$\eta(B_n) = \begin{cases} \lceil p/5 \rceil 2^{p-2} + 2^{p-1}, & \text{если } p \bmod 5 = 4, \\ \lceil p/5 \rceil 2^{p-2} & \text{в противном случае.} \end{cases}$$

ДОКАЗАТЕЛЬСТВО. Аналогично доказательству теоремы 2 при удалении из сети Омега $S = \lceil p/5 \rceil 2^{p-2}$ вершин в оставшихся структурах имеются компоненты с одной или тремя вершинами. В сети Омега с $p + 1$ уровнями при $p \bmod 5 = 4$ при удалении S вершин остаются компоненты с тремя вершинами (случай А). В противном случае имеются компоненты с одной вершиной (случай В). В случае А мы должны удалить из сети также 2^{p-1} вершин. Поэтому все вершины в оставшейся сети являются соседними либо с удаленной вершиной, либо с замкнутой окрестностью удаленной вершины.

СЛЕДСТВИЕ 5. *Окрестностная целостность и индекс доступности в сети Омега удовлетворяют неравенству*

$$NI(O_p) \leq \eta(O_p) + 1.$$

5. Заключение. При разрушении узлов или линий связи коммуникационной сети ее эффективность падает. При проектировании коммуникационных сетей следует обеспечить их устойчивость не только к возможным разрушениям, но и к возможному перепроектированию. Уязвимость коммуникационной сети характеризует ее сопротивляемость к разрушению узлов и связей. В теории графов для измерения уязвимости используются самые разные параметры. Понятие окрестностной целостности графа основано на следующей идее: при разрушении узла все примыкающие к нему вершины считаются преданными и бесполезными для оставшейся части сети. В настоящей статье мы приводим алгоритм для вычисления индекса доступности графа, определенного нами в предыдущих статьях, и находим окрестностную целостность сетей Бабочка и Омега.

СПИСОК ЦИТИРОВАННОЙ ЛИТЕРАТУРЫ

- [1] Cozzens M. B. Stability measures and data fusion networks // Graph Theory Notes of Network. 1994. V. 26. P. 8–14.
- [2] Cozzens M. B., Wu S. Y. Vertex neighbor-integrity of trees // Ars Combinatoria. 1996. V. 43. P. 169–180.
- [3] Dündar P. The neighbour-integrity of boolean graphs and its compounds // Intern. J. Computer Math. 1999. V. 72. P. 441–447.
- [4] Dündar P. New notions in network reliability: stability numbers of sequential joined graphs // Neural Network World. 1999. V. 5. P. 403–411.
- [5] Dündar P., Ozan A. On the neighbour-integrity of sequential joined graphs // Intern. J. Computer Math. 2000. V. 74. P. 45–52.
- [6] Dündar P. Accessibility number and the neighbour-integrity of generalised Petersen graphs // Neural Network World. 2001. V. 2. P. 167–174.
- [7] Dündar P. Stability measures of some static interconnection networks // Intern. J. Computer Math. 2001. V. 76. №4. P. 455–462.
- [8] Chartrand G., Lesniak L. Graphs and Digraphs. California: Wadsworth & Brooks, 1986.
- [9] Prather R. E. Discrete Mathematical Structures for Computer Science. Boston: Houghton Mifflin Comp., 1976.
- [10] Liaw S. C., Chang G. J. Rabin numbers of Butterfly networks // Discrete Math. 1999. V. 196. P. 219–227.
- [11] Kumar V., Grama A., Gupta A., Karpis G. Introduction to Parallel Computing. Redwood City, CA: The Benjaming/Cumminngs Publ., 1994.