УДК 621.039.56

С.Д. Винничук, В.Д. Самойлов, доктора техн. наук
Ин-т проблем моделирования в энергетике им. Г.Е. Пухова НАН Украины
(Украина, 03164, Киев, ул. Генерала Наумова, 15,
тел. (044) 4241063, e-mail: vynnychuk@i.ua; samoylov.vd@gmail.com)

Определение токовых ребер графов коммутационных структур на основе анализа фундаментальной системы циклов

Предложен метод определения ребер в графе коммутационной структуры электрической подстанции, по которым протекает ток неопределенного направления (токовые ребра), на основе анализа фундаментальной системы циклов (контуров). Описанный алгоритм реализации метода предлагается использовать на этапе подготовки СЛАУ $AX + B = 0$ с матрицей Максвелла минимального порядка. В алгоритме использована модификация псевдокода, не зависящая от языка программирования. Выполнена оценка вычислительной сложности данного алгоритма.

Запропоновано метод визначення ребер у графі комутаційної структури електричної підстанції, по яким протікає ток невизначеного напряму (токові ребра), на основі аналізу фундаментальної системи циклів (контурів). Алгоритм реалізації методу пропонується використовувати на етапі підготовки СЛАУ $AX + B = 0$ з матрицею Максвелла мінімального порядку. В алгоритмі використано модифікований псевдокод, який не залежить від мови програмування. Отримано оцінку обчислювальної складності даного алгоритму.

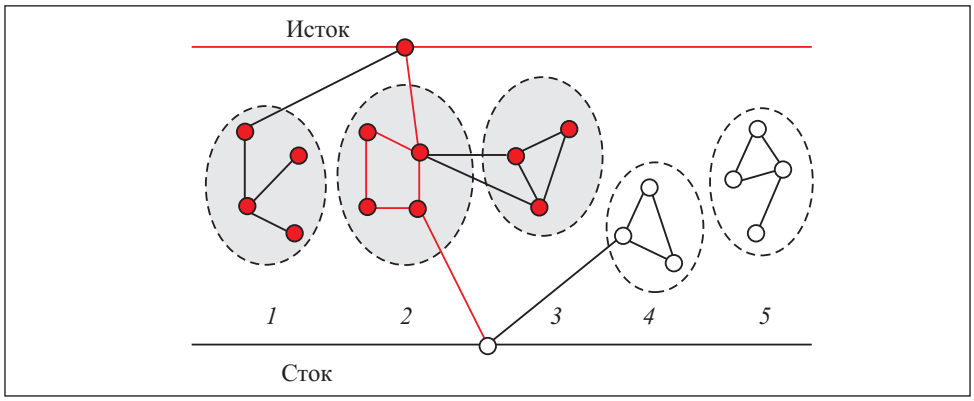
Ключевые слова: *информационные технологии, коммутационная структура, конструирование тренажеров, конструирование моделей.*

Модели коммутационных структур (КС) являются основной составляющей тренажеров оперативных переключений (ТОП), разрабатываемых для подготовки и поддержки квалификации персонала подстанций электроэнергетической системы. В работах [1, 2] показано, что основной задачей при создании моделей является разработка эффективных методов определения наличия токов и (или) их значений в коммутационных элементах (КЭ) КС с учетом следующих условий:

необходимость пересчета модели КС при переключении любого КЭ;

значительное число КЭ в КС (порядка нескольких тысяч);

обеспечение комфортного времени пересчета модели КС (до 1 с.) при ее использовании в тренажерах.



Варианты подструктур КС

Задача определения значений токов в линиях КС в общем случае может быть решена посредством построения СЛАУ $AX + B = 0$ и ее решения одним из известных методов. В [2] для решения СЛАУ использован метод квадратных корней (метод Холецкого). Время решения СЛАУ пропорционально числу неизвестных в третьей степени, которое при использовании матрицы Максвелла A равно числу узлов КС.

В ряде случаев нулевые значения токов можно определить заблаговременно. Так, в работе [2] при построении матрицы A использованы только узлы, доступные по напряжению (УДН) из узла ИСТОК. Следовательно, из числа неизвестных можно исключить все узлы, недоступные по напряжению (УнДН).

Если в число неизвестных СЛАУ включено все множество узлов, то СЛАУ может оказаться недоопределенной. Тогда определение значений токов возможно посредством исключения части узлов (по одному из каждой связной компоненты графа). При этом заранее известно, что ток в ребре будет нулевым, если один из описывающих узлов изолирован. Такие узлы можно исключить из СЛАУ и ускорить процесс ее решения.

Число неизвестных СЛАУ можно уменьшить и другими способами. Возможные варианты специфических подструктур КС представлены на рисунке, где из множества всех узлов КС выделены два подмножества, а именно УДН из узла ИСТОК (варианты *1*—*3*) и УнДН (варианты *4*, *5*) [2, рис. 4]. В работе [2] для каждого состояния КС с помощью алгоритма поиска УДН из узла ИСТОК, найдено множество УДН. Число этих узлов определяет порядок СЛАУ, в результате решения которой находим значения напряжений в узлах.

Очевидно, что токи в ребрах будут ненулевыми в тех случаях, когда существует простой путь от узла ИСТОК к узлу СТОК. Все такие ребра

будем называть активными, или токовыми. Если простого пути не существует, то ток в ребре будет нулевым. Этому соответствуют подмножества *1* и *3*, которые подключены к простым путям только в одном узле. В [2] такие подмножества названы «гроздьями».

Следовательно, с задачей определения значений токов в ребрах связана задача определения множества ребер, в которых ток гарантированно будет нулевым независимо от значений сопротивлений ребер. Такая задача становится основной, если необходимо определить множество ребер, в которых возможен ненулевой ток. Ее можно решить с помощью алгоритмов анализа графов, вычислительная сложность которых меньше, чем решение СЛАУ.

Рассмотрим способ решения указанной задачи на основе анализа структуры фундаментальной системы циклов графа.

Спецификация на основе псевдокода. Для разработки алгоритмов решения задач с помощью псевдокода в работе [1, с. 61—63] приняты определенные соглашения, обеспечивающие их наглядность и компактность. Для русскоязычного пользователя предложена следующая коррекция соглашений.

Использование русскоязычного варианта записи операторов. Выполнен перевод англоязычных операторов: for → **Для**, do → **Вып** (выполнить), while → **Пока**, if — then — else → **Если** — **То** — **Иначе**.

Использование англо- и русскоязычных имен переменных и массивов.

Использование русскоязычного названия атрибута объекта. Например, число элементов массива: length [A] → длина [A].

Применение специального значения ПУСТО *для указания на несуществующий объект* (вместо NIL).

В логических соотношениях допускается выражение $x \neq$ ПУСТО.

Пример применения скорректированных соглашений представлен в виде алгоритма поиска в ширину [1 с. 614, 615].

П с е в д о к о д 1.

Поиск_Шир(*СписСмеж*)

1 *Цвет* ← []
2 *Цвет*[*Исток*] ← СЕРЫЙ
3 *Предок* ← []
4 *Дист* ← []
5 *Очередь* ← []
6 Занести(*Очередь, Исток*)
7 **Пока** *Очередь* ≠ ПУСТО
8 **Вып** *и* ← Извлечь(*Очередь*)

```
9                  Для каждой v ∈ СписСмеж[u]
10                     Вып Если Цвет[u] =        БЕЛЫЙ
11                           То    Цвет[u] ← СЕРЫЙ
12                                 Дист[v] ← Дист[u] +1
13                                 Предок[v] ← u
14                                 Занести(Очередь, v)
15          Цвет[u] =ЧЕРНЫЙ
```

Дополнительные пункты соглашения следующие:

Возможность создания пустых массивов и объектов. $X \leftarrow [\,]$ означает создание массива *X*, элементы которого отсутствуют.

Использование некоторых типовых функций. При работе с массивом <переменная очередь> используются функции «Занести» (<переменная очередь>, <простая переменная>), которая заносит значение <простая переменная> в конец массива <переменная очередь>). Функция «Извлечь» (<переменная очередь >) извлекает первое значение из очереди.

Представление итерационных циклов в псевдокоде как циклов-функций. Стандартный набор циклов-функций, характерный для языков программирования, и соответствующий код циклов в Action Script представлены в таблице.

Псевдокод 1 аналогичен приведенному в [1] , но имеет такие упрощения. Для цвета БЕЛЫЙ используется признак *Цвет*[*u*] = ПУСТО, поэтому для массива *Цвет* нет необходимости занесения признака цвета (например, значения нуль) для всех вершин графа. Аналогично массивы для предков (*Предок*) и дистанций узлов от ИСТОКА(*Дист*) также вначале пустые.

Псевдокод алгоритма поиска в ширину имеет следующий вид.

П с е в д о к о д 2.

Модель_Поиск_Шир(*СписСмеж,Исток*)

```
1  Цвет ← []                          > Цвет всех узлов  БЕЛЫЙ
2  Цвет[Исток] ← СЕРЫЙ
3  Предок ← []                        > Предок[Исток] отсутствует
4  Дист ← []
```

Тип цикла-функции	Код в Action Script
ЦиклПока (<условие>) <тело цикла>	while(<условие>){<тело цикла>}
ЦиклВып_Пока (<условие>) <тело цикла>	do{<тело цикла>} while(<условие>)
ЦиклДляВсех(<переменная> ∈ <объект>) <телоцикла>	for(var <переменная> in <объект>){<тело цикла>}

```
5  Очередь ← []
6  Поиск_Шир(Исток)

   Поиск_Шир(Исток)
1  Занести(Очередь, Исток)
2  Дист[Исток] ← 0
3  ЦиклПока(Очередь ≠ ПУСТО)
4          и ← Извлечь(Очередь)
5          ЦиклДляВсех( v ∈ СписСмеж [и])
6                  (Цвет[и] =БЕЛЫЙ) И
7                  Цвет[и] ← СЕРЫЙ
8                  Дист[v] ← Дист [и] +1
9                  Предок[v] ← и
10                 Занести(Очередь,v)
11         Цвет[и] =ЧЕРНЫЙ
```

Метод поиска активных токовых ребер выявлением и просмотром всех контуров в КС.

П с е в д о к о д 3.

```
Модель_Токи_Контур(СписСмеж)
1  Цвет ← []                          > Цвет всех узлов БЕЛЫЙ
2  Цвет[Исток]← СЕРЫЙ
3  Предок ← []                        > Предок[Исток] отсутствует
4  Дист ← []
5  Дист[Исток] ←0
6  ХорСерые ← []
7  ЧислоХорд ←0
8  ДугиСтока ←[]
9  Очередь ← []
10 Поиск_Хорд(Исток)
11 АктДуги ← Ток ← УзТок ← []    >Создание массивов
12 УзТок[Исток] ← 1                >Исток – токовый узел
13 ТокиСтока()
14 ДугиКонтура(и, v)
15 ТокиКонтуров()

Поиск_Хорд(Исток)                  >Поиск в ширину токовых узлов
1  Занести(Очередь, Исток)
2  ЦиклПока(Очередь ≠ ПУСТО)
3          и ← Извлечь(Очередь)
4          ЦиклДляВсех( v ∈ СписСмеж [и])
5          (Цвет[v] = СЕРЫЙ) И
6             ЧислоХорд ← ЧислоХорд +1
```

```
7            ХорСерые[(u,v)] ← []
8          (v = Сток) И
9            ДугиСтока[u]=u
10         (Цвет[u] =БЕЛЫЙ) И
11           Цвет[u] ] ← СЕРЫЙ
12           Дист[v]  ← Дист [u] +1
13           Предок[v]  ← u
14         Занести(Очередь,v)
15         Цвет[u] =ЧЕРНЫЙ
```

ТокиСтока()

```
1  ЦиклДляВсех( u ∈ ДугиСтока)
2     v ← 0                          >Узел 0 — это Сток
3     ЦиклВып_Пока(УзТок[u] = ПУСТО)
4        УзТок[u] ← 1                >Узел u — токовый
5        Ток[(u,v)] ← 1              >Наличие тока в дуге (u,v)
6        Занести(АктДуги,u)          >Пополнение очереди
                                     активных дуг
7        v ← u
8        u ← Предок[u]
9        (УзТок[u] ≠  ПУСТО) И
10          Ток[(u,v)] ← 1
```

ДугиКонтура(*u,v*)

```
1  ЦиклДляВсех((u,v) ∈ ХорСерые)
2     ЦиклВып_Пока(u ≠ v)
3        (Дист[v]  = Дист[u]) И
4           ХорСерые[(u,v)][u] ← u
5           u ← Предок[u]
6        ХорСерые[(u,v)][v] ← v
7        v←Предок[v]
```

ТокиКонтуров()

```
1  ЦиклПока(АктДуги ≠ ПУСТО)
2    vАкт ← Извлечь(АктДуги)          >Извлечение активной
                                      дуги
3      ЦиклДляВсех((u,v) ∈ ХорСерые)
4        (ХорСерые[(u,v)][vАкт] ≠ ПУСТО) И >Найден контур с актив-
                                      ной дугой
5           Ток[(u,v)] ← 1            >Наличие тока в серой
                                      хорде (u,v)
```

```
6        ЦиклДляВсех( v ∈ ХорСерые[(u,v)])
7          vA ← (ХорСерые[(u,v)][v]      > Узел vA конца дуги
                                           контура
8          Индекс ← (Предок[vA],vA)
9          Ток[Индекс] =  ПУСТО) И       >Активные дуги про-
                                           пускаем
10           Ток[Индекс] ← 1
11           Занести(АктДуги,vA)          >Пополнение списка
                                           активных дуг
12       ХорСерые[(u,v)] ←  []            >Удаление активного
                                           контура
```

Алгоритм метода. Определение активных ребер КС, т.е. ребер, по которым протекает ток без определения его направления и значения, реализуется на основе алгоритма поиска в ширину узлов, достижимых из узла ИСТОК. Процедура поиска токов выделением контуров в неориентированном графе $G = (V, E)$ обеспечивает его представление с помощью списков смежности (*СписСмеж*).

Главная процедура **Модель_Токи_Контур**(*СписСмеж*) формирует массивы данных для каждой вершины графа.

Поиск_Хорд(*Исток*) — это поиск в ширину из вершины ИСТОК доступных вершин графа G и запоминание в массиве дуг *ХорСерые*, связывающих текущую серую вершину u с серыми вершинами v. Цвет каждой вершины $u \in V$ хранится в переменной *Цвет*[u], а предки — в переменной *Предок*[u]. Если предка нет (например, если u = *Исток*), то происходит очистка элемента массива *Предок*[u] = ПУСТО.

Расстояние от ИСТОКА до вершины u хранится в элементе массива *Дист*[u]. Используется массив *Очередь* для хранения и извлечения серых вершин.

Работа алгоритма **Поиск_Хорд**(*Исток*) аналогична процедуре поиска в ширину, описанной в [1], но имеет два дополнения.

1. При появлении дуги (u, v), где *Цвет*[v] = СЕРЫЙ, создается пустой массив *ХорСерые*[(u, v)] для хранения дуг контура, связанных с данной серой хордой (строка 7).

2. При достижении узла СТОКА (появление дуги (u, v), где v = *Сток*) дуга (u, v) запоминается в массиве *ДугиСтока*[u] (строка 9).

Таким образом в результате работы процедуры **Поиск_Хорд**(*Исток*) выявляются все серые хорды и для каждой из них создается массив, в котором хранятся дуги контура и массив *ДугиСтока*, в котором хранятся дуги, направленные к СТОКУ.

Затем на основе дуг из массива *ДугиСтока* с помощью процедуры **ТокиСтока**() (строка 13) строятся по предкам токовые пути к ИСТОКУ. Дуги этих путей хранятся в массиве *АктДуги*, токи — в массиве *Ток*. Узлы, через которые проходят токовые пути (токовые узлы), отмечаются в массиве *УзТок*[*u*]. Массив *АктДуги* используется в дальнейшем как очередь при просмотре контуров в процедуре **ТокиКонтуров**().

Процедура **ДугиКонтура**(*u*, *v*) заполняет созданные массивы для каждой из серых хорд *ХордСерые*[(*u*, *v*)] дугами, принадлежащими этому контуру. Для каждой серой хорды (*u*, *v*) выполняется просмотр по предкам двух ребер дерева — из ее начала *u* и конца *v*. Подъем по ребрам продолжается до тех пор, пока узлы предки различны ($u \neq v$, строка 2).

ТокиКонтуров(). В основу процедуры положено следующее утверждение: если активное ребро принадлежит некоторому из контуров, то все ребра контура являются активными, т.е. существуют простые пути от вершины СТОК до вершины ИСТОК, которые содержат все ребра контура.

Данное утверждение следует из того факта, что между двумя узлами простого цикла существует два простых пути. Поэтому в процедуре **ТокиКонтуров**() проверяется принадлежность к контурам графа каждого из найденных активных ребер. Если контур найден, то все его ребра становятся активными и включаются в очередь списка активных ребер. При этом серые хорды в такой список не включаются, так как они принадлежат только одному контуру, который уже проанализирован.

Процедура состоит из трех циклов. В первом цикле (строка 1) извлекается очередная активная дуга (*vАкт*) из очереди *АктДуги* (строка 2), в которой представлены только ребра остовного дерева, построенного с помощью алгоритма поиска в ширину.

Во втором цикле проверяется оставшаяся часть серых хорд, а именно: нет ли в контуре, который связан с определенной хордой, активной дуги *vАкт*. Если найден контур, включающий дугу с узлом *vАкт*, то в серой хорде и всех неактивных дугах ее контура в третьем цикле устанавливаются токи (строки 5—10). Новые активные дуги заносим в очередь *АктДуги*. Массив просмотренного активного контура делаем пустым (строка 12).

Оценка вычислительной сложности метода поиска активных ребер. Общая структура алгоритма метода представлена в главной процедуре **Модель_Токи_Контур**, которая содержит ряд последовательно выполняемых процедур: **Поиск_Хорд**(*Исток*), **ТокиСтока**(), **ДугиКонтура** (*u*, *v*) и **ТокиКонтуров**(). Поэтому для оценки вычислительной сложности метода достаточно оценить вычислительную сложность каждой процедуры в отдельности.

В процедуре **Поиск_Хорд**(*Исток*) реализован метод поиска в ширину, вычислительная сложность которого оценивается величиной $O(V+E)$, где V — число узлов графа, а E — число его ребер.

В процедуре **ТокиСтока**() строятся пути от узла СТОК до узла ИСТОК для всех дуг из массива *ДугиСтока*. Число элементов массива *ДугиСтока* не превышает общего числа хорд, которое для произвольного связного графа равно $E-V+1$. При этом длина (число ребер) произвольного пути меньше диаметра графа, который не превышает числа узлов. Поэтому верхняя оценка вычислительной сложности алгоритма процедуры **ТокиСтока**() не превышает величины $O(V(E-V+1))$.

С помощью процедуры **ДугиКонтура**(u, v) выстраивается часть системы фундаментальных циклов (контуров), каждый из которых содержит только одну из серых хорд. Каждый из контуров строится из серой хорды по узлам-предкам до их совпадения. Поэтому длина цикла не превышает числа узлов. Поскольку число серых хорд также не превышает общего числа хорд, которое для произвольного связного графа равно $E-V+1$, как и для процедуры **ТокиСтока**(), вычислительная сложность процедуры **ДугиКонтура**(u, v) также не превышает величины $O(V(E-V+1))$.

Наиболее сложной для оценки представляется процедура **ТокиКонтуров**(). Согласно алгоритму она содержит три вложенных цикла. В таких случаях при оценке вычислительной сложности число операций должно быть, как минимум, пропорционально произведению числа значений параметра цикла.

Следовательно, можно предположить, что при числе активных дуг в очереди *АктДуги*, равном числу узлов, при числе контуров, равном $E-V+1$, а также при длине контура, равном числу узлов, вычислительная сложность алгоритма процедуры **ТокиКонтуров**() составляет величину $O(V^2(E-V+1))$. На самом деле ее вычислительная сложность составляет $O(V(E-V+1))$. Докажем это.

Для каждого из контуров представим условно информацию о принадлежащих ему ребрах строкой из нулей и единиц, где единица является признаком принадлежности ребра контуру. Такая информация, построенная для всех контуров, образует сигнальную матрицу с числом строк, равным числу серых хорд (не превышает $E-V+1$), и числом столбцов, не превышающим числа ребер остовного дерева (не более V). Проанализируем работу алгоритма процедуры **ТокиКонтуров**() на сигнальной матрице.

Во внешнем (первом) цикле процедуры извлекается активная дуга (*vАкт*) из очереди *АктДуги*, что для сигнальной матрицы означает выбор ее столбца, где число операций по определению номера столбца имеет порядок $O(1)$.

Во втором цикле анализируются элементы столбца, число которых не превышает числа серых хорд. Заметим, что такой столбец анализируется единственный раз. Если среди элементов столбца окажутся единицы, то определен активный контур (строка сигнальной матрицы), которому принадлежит анализируемая активная дуга.

Тогда включается в работу третий цикл, в ходе которого все ребра такого контура объявляются активными, после чего контур (строка матрицы) исключается из дальнейшего анализа.

В ходе работы трех циклов элементы сигнальной матрицы обрабатываются не более двух раз: первый раз — в столбце (второй цикл), а второй раз — в строке (третий цикл)). При этом общее число элементов сигнальной матрицы равно $V(E - V + 1)$. Поэтому число операций в алгоритме пропорционально $V(E - V + 1)$.

Следовательно, вычислительная сложность алгоритма метода поиска активных ребер выявлением и просмотром всех контуров в КС составляет величину $O(V(E - V + 1))$.

Выводы

Метод определения токовых ребер в КС моделей электроподстанций на основе анализа фундаментальной системы циклов обеспечивает выделение минимально необходимого числа токовых узлов и токовых ребер для построения СЛАУ $AX + B = 0$ с матрицей Максвелла.

Для спецификации алгоритма можно использовать модификацию известного представления в виде псевдокода, ориентированную на русскоязычного пользователя и технологию модельного (формульно- функционального) программирования.

The authors propose a method for determining ribs in the graph of commutation structure of electric substation, the indeterminate direction current passing along them (current ribs) on the basis of analysis of the fundamental system of cycles (circuits). The described algorithm of the method implementation it is proposed to be used at the stage of preparation of SLAE $AX+B=0$ with the Maxwell matrix of minimum order. The pseudocode modification which does not depend on the programming language is used in the algorithm. Computation complicacy of the algorithm has been estimated.

СПИСОК ЛИТЕРАТУРЫ

1. *Самойлов В.Д.* Определение напряжений и токов коммутационной структуры поиском на графе // Зб. наук. праць ІПМЕ ім. Г.Є. Пухова. Спец. вип. «Моделювання та інформаційні технології: Матеріали конференції «Моделювання — 2010». Т. 3. — Київ, 2010. — С. 131—139.

2. *Самойлов В.Д., Абрамович Р.П.* Поиск токов в коммутационных структурах решением СЛАУ // Электрон. моделирование. — 2013. — **35**, № 1. — С. 95 — 107.
3. *Кормен Т.Х., Лейзерсон Ч.И., Ривест Р.Л. и др.* Алгоритмы: построение и анализ. 2-е изд. : Пер. с англ. — М. : Изд. дом «Вильямс», 2007. — 1296 с.

Поступила 17.02.14

ВИННИЧУК Степан Дмитриевич, д-р техн. наук, ст. науч. сотр, и.о. зав. отделом Ин-та проблем моделирования в энергетике им. Г.Е. Пухова НАН Украины. В 1977 г. окончил Черновицкий государственный университет. Область научных исследований — моделирование тепловых и гидравлических процессов в системах кондиционирования воздуха и процессов динамического изменения частоты в электроэнергетических системах, теория алгоритмов.

САМОЙЛОВ Виктор Дмитриевич, д-р техн. наук, профессор, главный научный сотрудник Ин-та проблем моделирования в энергетике им. Г.Е. Пухова НАН Украины. В 1960 г. окончил Украинскую академию сельскохозяйственных наук. Область научных исследований — компьютерные технологии моделирования, тренажеры, диагностика профессиональной компетентности в энергетике.