электронный научно-технический журнал
ИНЖЕНЕРНЫЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51036. ISSN 2307-0595

# Программный инструментарий моделирования потоков в сетях в Visual C++

# 10, октябрь 2015

**Русакова З. Н.[1,*], Семин В. М.[1]**

УДК: 519.682

[1]Россия, МГТУ им. Н.Э. Баумана
* z.n.rusakova@mail.ru

## Введение

Работа посвящена вопросам дальнейшего развития системы принятия решений МАТ-ПРОГ, описанной в работе [1]. Рассматриваются вопросы автоматизации процесса моделирования потоков в сетях, определения пропускной способности сетей, моделирования дерева разрезов сети. Абстракция потоков в сетях - модель решения задачи, применяемая к широкому кругу приложений. Это такие задачи как определение максимального количества информации, которое может быть передано по каналам связи, задача о максимальной пропускной способности газопроводов, нефтепроводов, задачи рациональной перевозки грузов, задачи об оптимальных назначениях и многое другое [2,3,4,5].

Для решения задачи применяется объектно-ориентированный подход на основе паттернов проектирования в среде Visual C++. Разработанный алгоритмический и программный инструментарий может использоваться как шаблон для решения указанных задач моделирования сетей

## 1. Формальные модели систем и основные понятия

В качестве формальных моделей реальных систем важное место занимают графы [1,2]. Для описания сети используется представление взвешенным конечным графом без циклов и петель, ориентированным в одном направлении от вершины, являющейся истоком (входом сети) к вершине, являющейся стоком (выходом графа): g(v,e) - где v-это множество вершин сети, e - это множество ребер сети.

Для описания ребер сети используются следующие основные параметры:

i, j – номера узлов, инцидентных ребру,

r - пропускная способность ребра между узлами i, j: максимальное количество ресурса сети, которое может быть пропущено через него в единицу времени,

f - поток по ребру: количество ресурса сети, проходящего через него в единицу времени,

c = r-f - остаточная пропускная способность ребра.

Остаточная сеть состоит из тех ребер, по которым можно пропустить поток cij =rij−fij, который определяет остаточную пропускную способность.

Насыщенное ребро - это ребро, поток по которому равен его пропускной способности f=r. Пропускная способность пути ограничивается минимальной пропускной способностью ребра этого пути. Дополняющий (или аугментальный) путь в остаточной сети - простой путь, по которому можно переслать дополнительный поток меньший или равный остаточной пропускной способности: $f \leq \min c_{ij}$ этого пути. После этого одно из ребер станет насыщенным.

Разрезом в сети G(V,E) называется разбиение множества V на 2 части A и B = V/A, причем исток принадлежит A, сток - B. В результате будем иметь ребра, концы которых принадлежат разным множествам. Эти ребра образуют разрез. Пропускной способностью разреза называют сумму пропускных способностей этих ребер.

## 2. Алгоритмы решения оптимизационных задач моделирования потоков

Решение задачи построения минимального разреза и поиска максимального потоков в сетях основывается на методе Форда-Фалкерсона (ФФ) [1,2]: на любой сети максимальная величина потока из истока в сток равна минимальной пропускной способности разреза, отделяющего сток от истока. Метод ФФ добавляет последовательно потоки по дополняющим путям, пока существует дополнительный путь. Т.е. до построения максимального потока, величина которого определяется минимальной пропускной способности разреза, отделяющего сток от истока. На каждом шаге алгоритма определяется дополняющий путь, по которому можно пропустить поток, который используется для увеличения потока, если пути нет - поток максимален.

В основе алгоритмов, реализующих этот метод, лежит алгоритм поиска дополняющих путей. Алгоритм выбора дополняющего пути определяет время решения. В случае неудачного выбора алгоритм может не выйти из цикла или не получить решение за приемлемое время.

**Алгоритм поиска дополняющих путей.** Для решения рассматриваемого класса задач актуальным является создание эффективных алгоритмов поиска в графах. На основе фундаментальных поисковых алгоритмов в ширину и в глубину разработаны оптимизированные алгоритмы поиска дополняющих путей, в основе которых используются динамические структуры данных – списки. Так как сети, как правило, разрежены для описания сетей и дополняющих путей используется преставление в виде списка смежных вершин.

В основе алгоритмов поиска аугментальных путей для определения минимального сечения сети, являются методы поиска в ширину и в глубину. Их применение вытекает из свойства: никакой аугментальный путь не может уменьшить длину кратчайшего пути из вершины истока в вершину стока в остаточной сети. В процессе поиска пути этими алгоритмами неявно определяется подграф дерева поиска: лес деревьев поиска в ширину и в глубину.

При решении задач поиска в графах вводится понятие открытых и закрытых вершин или белые и черные вершины. Вершина – открытая, если не порождены ее потомки. Вершина –закрытая, если в процессе поиска для нее определены ее потомки.

После раскрытия вершины она становится закрытой или черной. Эти вершины хранятся и обрабатываются в двух списках поиска: список открытых вершин и список закрытых вершин. Вершины графа в процессе поиска из списка открытых или белых вершин переписываются в список закрытых или черных. Этот список закрытых вершин содержит в себе дерево решения задачи, из которого выделяется результат: либо путь, либо покрытие графа.

Методы поиска в ширину и в глубину отличаются стратегией формирования списков закрытых и открытых вершин. Стратегия метода поиска в ширину: вершина для раскрытия выбирается из головы списка открытых вершин, а потомки записываются в хвост списка открытых. Стратегия метода поиска в глубину: вершина для раскрытия выбирается из головы списка открытых вершин, но и потомки записываются в голову (механизм стека).

## 3. Описание иерархии классов

Программная реализация моделирования рассмотренных задач осуществляется средствами Visual C++ [4,5,6].

Для решения задачи применяется объектно-ориентированный подход с использованием шаблонов проектирования: паттернов контейнер и итератор [7, 8].

Для описания ребра графа используется структура данных классового типа. Этот класс представляет звено динамического списка и включает поля, описывающие ребро графа: nn - номер начальной вершины, nk - номер конечной вершины, nw, pw, rw, f - параметры, связанные с характеристикой ребер: пропускная способность, остаточная пропускная способность, поток по ребру.

**Описание класса Zveno**

```
class Zveno{
public:
int pw, rw, f ;  //, вычисляемые параметры потока
int nw, nn, nk; // номер ветви, начальной и конечной вершины ребра
Zveno (){}  // конструкторы и деструктор
Zveno ( int wn, int wk,int j){
        nn=wn;    nk=wk;    nw=j;}
~Zveno (){}
};
```

Основной структурой моделирования является контейнерный параметризированный класс Spisok, описывающий контейнер на основе шаблона, методами которого выполняется построение списков открытых и закрытых вершин (или – для раскрашенных вершин -

черных или белых). В основе иерархии классов – параметризированный класс Elem, используемый для организации двунаправленного списка, и параметризированный класс Spisok.

Шаблон для представления звена списка параметризованный классом Elem.:

```
template < class T >
class Elem {
   public:
    T  data; // параметр шаблона
    Elem * next, * prev; // ссылки на следующее и предыдущее звено
    Elem(){} // конструкторы
    Elem(  T  d ){ data= d; next=0; prev=0;}
    };
```

В поля класса Spisok включены указатели, хранящие начало и конец списка - first,last, текущий указатель cur, который перемещается по списку, и вспомогательный - tec. В методах класса осуществляется реализация основных алгоритмов создания и обработки списков: добавления элемента в голову списка void add_head (T temp), в хвост списка void add (T temp), удаления элемента из головы списка void del_head ( ) и из хвоста списка void del().

**Описание шаблонного класса**

```
template < class T >   //   class T  параметр шаблона
class Spisok  {
public:
    Elem <T> * first,* last,* cur, *tec;
    public:
    Spisok ()        {first=0;last=0;cur=0; }
// Методы класса
// Метод – добавить элемент в хвост списка
   void add (T temp ) {
       cur =new Elem <T>  ( temp); // создание звена – вызов конструктора
          if(  first ){   // если звено –  не заглавное
          last->next=cur; //подключение звена
         cur->prev=last;
         last=cur;
          cur->next=0;        }
    else      {  // если звено –  заглавное
                first=cur;
                 last =cur;         }
}
// Метод – добавить элемент в голову  списка
```

```
    void add_head ( T temp ) {
    cur =new Elem  <T> (temp); ); // создание звена – вызов конструктора
          if( first == 0){ {  // если звено –  заглавное
              first=cur;
     last =cur;
         }
       else {     ){   // если звено –  не заглавное
           cur->next=first; //подключение звена
               first->prev=cur;
          first=cur;         }
}
// Метод – удалить звено из головы  списка
     void del_head ( ) {
          Elem <T>  *tmp;
    if(first){
             tmp=first;
             first=first->next;
             first->prev=0;
             delete tmp;  }
          else           cout <<"  first=0  "<<endl;
   }
// метод удаления звена из хвоста списка
     int  del_xwost( ){
    if ( last== 0)    return 0;
    else
              if ( first->next ){ //если есть следующее звено
             tec=first;
            while (tec->next->next!=0)
                 tec=tec->next;
              last=tec->next;
             last->next=0;
             delete tec->next;
              return 1;
            }
            else {
                  tec=first; first=0; delete tec; last=0; return 1;
                }
            }
}; // конец описания шаблона класса.
```

## 4. Вычисление максимального потока и минимального разреза

Списки открытых и закрытых вершин и список, описывающий граф сети, создаются на основе методов класса шаблонного класса Spisok. В качестве параметров шаблона используются классы, описывающие звено списка, вершины и другие вспомогательные классы. Построение дополняющих путей и вычисление максимального потока дерева разрезов и построение минимального сечения осуществляется в другом классе Poisk. Полями этого класса являются указатели на списки открытых и закрытых вершин и указатель на дерево решения, методами класса void form_open_spisok(), void form_close_spisok() строятся двунаправленные списки открытых и закрытых вершин. Для формирования списков примеряются методы шаблонного класса Spisok. Параметром шаблона является класс Zveno;

Описание шаблонного класса:

```
template < class T >   //   class T  параметр шаблона
class Poisk  {
public:
    int stok, ostok;
   Elem <T> * open_spisok,* close_spisok , * razrez;
    public:
    Poisk ()          { }
    Poisk (int istokp, int stokp) { stok= stokp , istok= istokp ;}
  // Методы класса – формирования списков открытых и закрытых вершин
    void form_open_spisok();
   void form_close_spisok ();
   void form_ razrez ();
};
```

Используя принципы формирования списков вершин в методах обхода графов в ширину и в глубину, выполняется вычисление дополняющего потока на каждом шаге итерационного поиска максимального потока. Начальная вершина – вершина истока, целевая вершина – сток. Эти вершины передаются в конструктор класса Poisk.

Для реализации алгоритма используется список вершин сети, в котором для каждой вершины хранится информация о ее цвете. Вершина описывается структурой:

```
struct ver{
        int num_ver;  //номер вершины
         int metka; //цвет вершины или метка – 1или 0
};
```

Основные переменные и методы класса формирования дополняющего пути и дополняющего потока: имена истока, стока, число ветвей и узлов сети; максимальный поток; список вершин сети, список списков для хранения, полученных в процессе поиска дополняющих путей; список для хранения вершин дополняющих путей. Текущий элемент хра-

нит число вершин i-го дополняющего пути, множество вершин разреза, лежащих по разные стороны от истока и стока, ребра разреза, вершины которых принадлежат этим множествам.

В методе поиска выбор вершины для раскрытия (или потомка текущей вершины) определяется в результате выполнения двух условий: вершины не помечена (т.е. она белая) и инцидентна ребру, одна из вершин которого совпадает с вершиной для раскрытия. Выход из цикла в двух случаях: нашли решение (l- флаг решения) или список открытых вершин пуст (flagn - флаг решения), что означает завершение обхода графа и формирование покрытия.

Метод решения, формирующий список закрытых вершин на основе метода поиска в ширину, рассматривается как метод класса Poisk - void form_ razrez ();

Минимальный разрез – поле класса Poisk указатель * razrez) для переменных, объявленных как поля формы в среде Visual C++.

Разработанный программный инструментарий для систем моделирования потоков в сетях позволяет решить задачу автоматизации расчета максимальных потоков и разрезов в сетях.

Результаты моделирования потоков выводятся в текстовом описании и графическом описании. Ниже приводится графическое представление результата решения: ветви, входящие в минимальный разрез, выделяются при выводе красным цветом, вершина истока – синим, стока - красным цветом, что позволяет визуально определить множества вершин, относящихся к истоку и стоку (A и B):

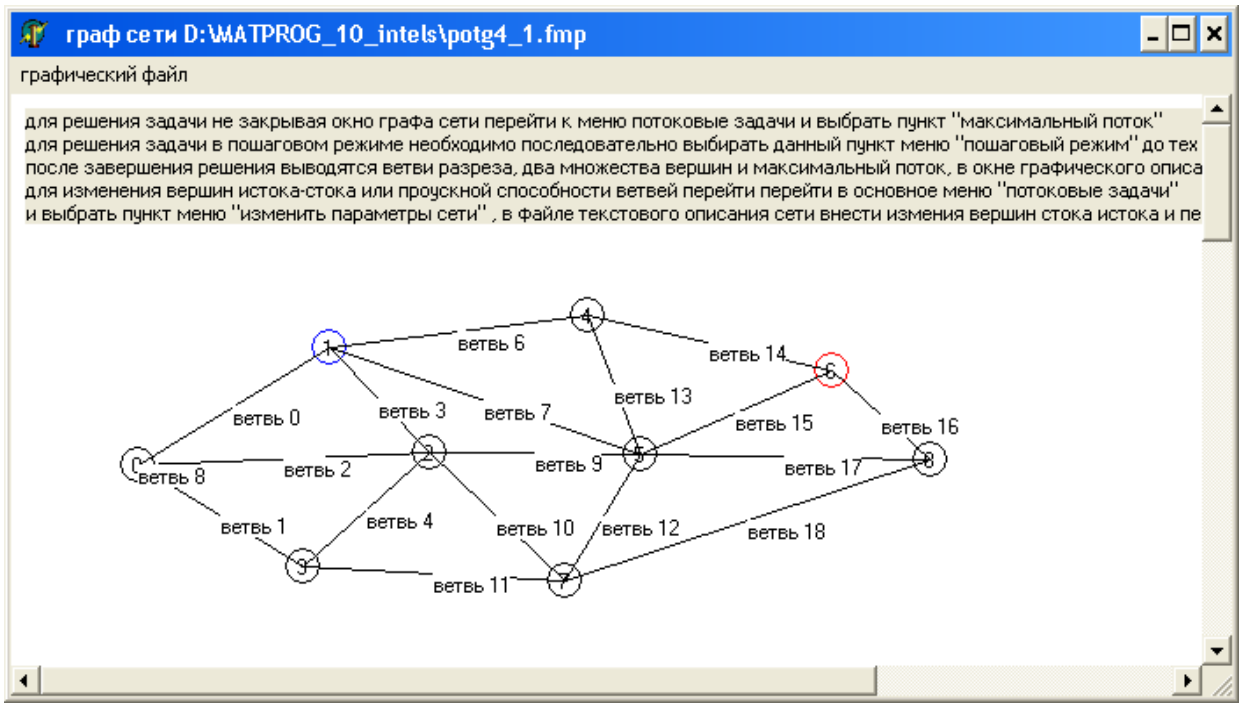


**Рис. 1.** Исходное описание сети, задан исток и сток

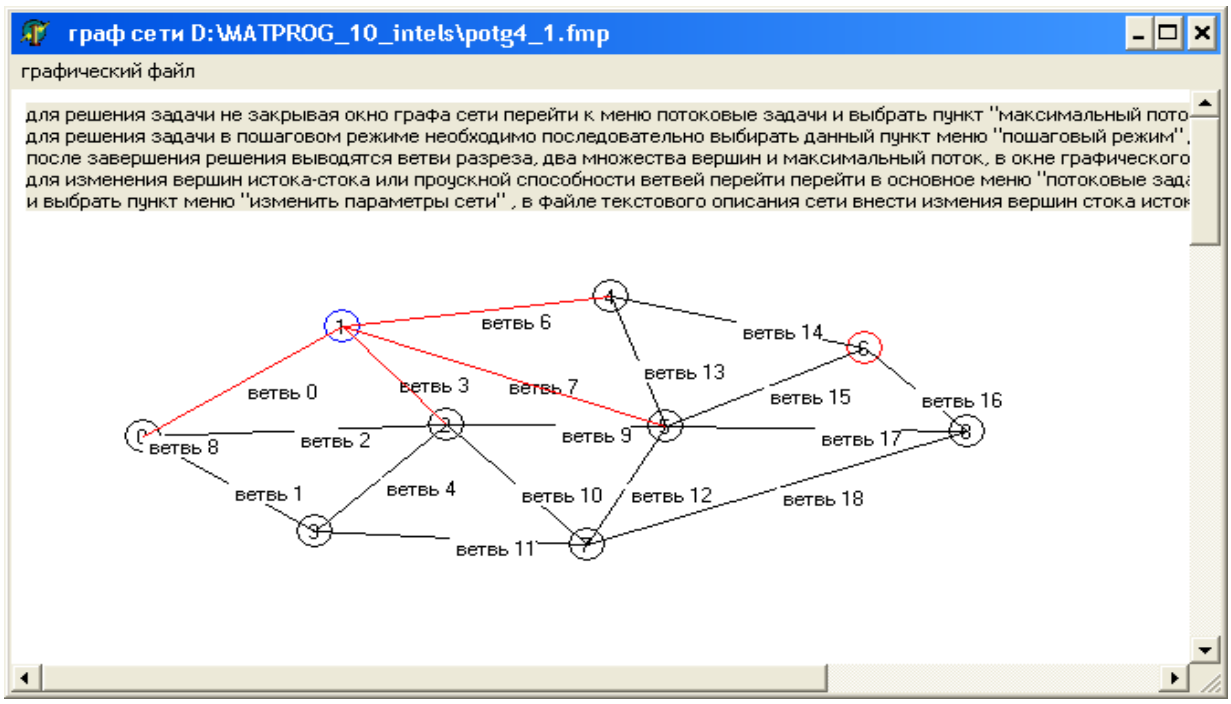


**Рис .2.** Результаты моделирования графическое окно

## Заключение.

На основе метода потенциалов разработан модифицированный алгоритм поиска дополняющих путей, использующий стратегию поиска в ширину, в условиях ограничения на выбор потомков. Программный инструментарий реализован в среде Visual C++: на основе технологии параметризованных контейнерных классов. В качестве параметра шаблона используются классы, описывающие различные объекты предметной области.

## Список литературы

[1]. Русакова З.Н. Система моделирования и интеллектуализации задач принятия решений. // Инженерный журнал: наука и инновации. 2013. №2 (14). 9 с. Режим доступа: http://engjournal.ru/catalog/it/hidden/537.html (дата обращения: 15.09.15) **DOI:** 10.18698/2308-6033-2013-2-537

[2]. Русакова З.Н. Динамические структуры данных и вычислительные алгоритмы. Visual C++. СПб.: Образовательные проекты. 2013. 272 с.

[3]. Русакова З.Н., Семин В.М. Программный инструментарий системы поддержки принятия решений «МАТПРОГ». // Десятый международный симпозиум «Интеллектуальные системы» INTELS'2012. (Россия, Вологда, Вологодский государственный технический университет 25 – 29 июня 2012 г.) / Доклад. М.: МГТУ им. Н.Э. Баумана. 2012. 4 с.

[4]. Ахо А., Хопкрофт Дж., Ульман Дж. Структуры данных и алгоритмы. М.: Издательский дом «Вильямс». 2003. 382 с.

[5]. Грешилов А.А. Математические методы принятия решений. 2-е изд., испр. и доп. : учеб. пособие (с рассчетными программами на оптическом диске). М.: МГТУ им. Н.Э. Баумана. 2014. 647 с.

[6]. Хэзфилд Р., Кирби Л. и др. Искусство программирования на С. Фундаментальные алгоритмы, структуры данных и примеры приложений. Энциклопедия программиста. Пер. с англ. М.:, СПб.:, Киев: ДиаСофт. 2001. 736 с. [Richard Heathfield, Lawrence Kirby, Ian Woods, Steve Summit, Ian Kelly, Jack Klein, Peter Seebach, Scott Fluhrer, Ben Pfaff, & 8 more. C++ Unleashed. Series: Unleashed. Publisher: Sams. 2000. 1392 p.]

[7]. Страуструп Б. Язык программирования С++. 3-е изд. / Пер. с англ. С. Анисимова, М. Кононова под ред. Ф. Андреева, А. Ушакова. СПб.: Невский диалект, М.: Бином. 2007. 1054 с. [Bjarne Stroustrup. The C++ Programming Language. Third Edition. AT&T Labs. Murray Hill, New Jersey]

[8]. Иванова Г.С., Ничушкина Т. Н. Объектно-ориентированное программирование: учебник / под общ. ред. Г. С. Ивановой. М.: Изд-во МГТУ им. Н. Э. Баумана. 2014. 455 с.

[9]. Саттер Г. Решение сложных задач на С++. Серия С++ In-Depth. / Пер. с англ. И. В. Красикова. М.: Вильямс. 2015. 395 с. [Herb Sutter. Exceptional C++. Style. Addison-Wesley, 2004.]

[10]. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. СПб.: Питер. ДМК Пресс. 2010. 366 с.

[11]. Павловская Т.А. С/С++. Программирование на языке высокого уровня. СПб.: Питер, 2003. — 461 с.