ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ В ЗАДАЧАХ ДИНАМИЧЕСКОГО ПРОГРАММИРОВАНИЯ

Кольчатов А.М.,
кандидат технических наук, доцент
Емельянова Т.В.,
кандидат технических наук, доцент
Арзамасский политехнический институт (филиал)
Нижегородского государственного технического университета им. Р.Е. Алексеева (Арзамас)
emelyanova@apingtu.edu.ru

Аннотация. *Данная статья посвящена имитационному моделированию и его применению в задачах динамического программирования. В статье подробно описано решение задачи динамического программирования – выбор кратчайшего пути. Приводится программа моделирования движения автомобиля по заданной сети дорог. Программа написана на языке имитационного моделирования СИМУЛА-67. Статья заинтересует научных, инженерно-технических работников, и студентов технических вузов.*

Ключевые слова: *имитационное моделирование, динамическое программирование, параллельные процессы, язык имитационное моделирование СИМУЛА-67.*

SIMULATION MODELING DYNAMIC PROGRAMMING PROBLEMS

A. Kolchatov, T. Emelyanova
Arzamas Polytechnic Institute (branch) of Nizhny Novgorod State Technical University named after R.E. Alekseyev (Arzamas)

Abstract. *This article focuses on simulation and its application in dynamic programming problems. The article described in detail the solution of dynamic programming - the shortest path selection. Simulation program provides the vehicle on a given road network. The program is written in simulation SIMULA-67. Article interested in scientific, engineering and technical staff, and students of technical universities.*

Key words: *simulation, dynamic programming, parallel processes, simulation language SIMULA-67.*

Задачи динамического программирования, то есть многошаговые (многоэтапные) задачи оптимизации, представляют собой задачи математического программирования с аддитивными или мультипликативными целевыми функциями или сводятся к таковым, см., например, [1,2].

Метод динамического программирования позволяет заменить решение одной задачи со многими переменными последовательным решением ряда задач с меньшим числом переменных.

Принципом, на котором основывается оптимизация многошагового динамического процесса, а также особенности вычислительного процесса, реализующего метод динамического программирования, является принцип оптимальности Р. Беллмана.

Принцип оптимальности. Оптимальное решение обладает тем свойством, что каково бы ни было текущее состояние и “траектория”, приводящая в это состояние, последующие решения должны быть оптимальными относительно текущего состояния и не зависеть от способа как это состояние было достигнуто.

В качестве примера типовой задачи динамического программирования рассмотрим задачу поиска маршрута минимальной длины между двумя заданными городами, если задана сеть дорог, связывающая эти города (рис.1).

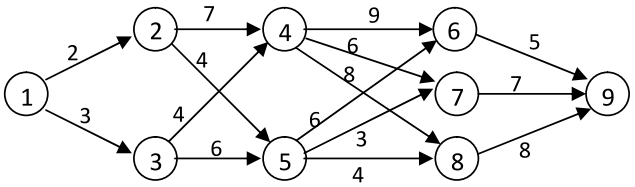

Рис. 1. Граф сети дорог

Процесс решения разбивается на шаги, нумерация которых осуществляется от конца к началу. В данном примере всего 4 шага (этапа). При n=1 выделяется подмножество городов $G_1 = \{6,7,8\}$, из которых конечный пункт (город 9) достигается за один этап (непосредственно). При n = 2 выделяется подмножество городов $G_2 = \{4,5\}$, из которых непосредственной дороги в конечный город нет, а есть маршруты, связывающие их с конечным городом через города множества G_1, то есть за два этапа. Продолжая дальше, получим множества $G_3 = \{2,3\}$ и $G_4 = \{1\}$, связывающие города этих подмножеств с конечным городом соответственно за 3 и 4 этапа. В последнем подмножестве должен находится всего один исходный город. Обозначим через $G_0 = \{9\}$ подмножество, содержащее всего один конечный город.

Любой маршрут из города 1 в город 9 содержит ровно 4 дуги графа, каждая из которых связывает вершины соответствующих подмножеств.

Рассмотрим последний шаг (n=1) и вычислим для него значение целевой функции. В город 9 можно доехать из городов 6,7 и 8. Вычислим длину пути до города 9 из этих городов:

$f_1(6) = L_{6,9} = 5;$

$f_1(7) = L_{7,9} = 7;$

$f_1(8) = L_{8,9} = 8;$

Условно оптимальное управление - это оптимальное управление, при условии что на данном этапе мы находимся в данном городе.

Условно оптимальные управления из каждого города подмножества G_1 будут $U_1(6) = 9$; $U_1(7) = 9$; $U_1(8) = 9$.

Условно оптимальные управления из городов подмножества G_2 находятся из условий вычисления минимального пути из каждого города до конечного города 9:

$f_2(4) = \min\{L_{4,6} + f_1(6), L_{4,7} + f_1(7), L_{4,8} + f_1(8)\} = \min\{9+5, 6+7, 8+8\} = 13.$

Значит условно оптимальное управление из города 4 в конечный город 9 $U_2(4) = 7$, так как из города 4 маршрут 4-7-9 в город 9 имеет минимальную длину, равную 13.

Аналогично

$f_2(5) = \min\{L_{5,6} + f_1(6), L_{5,7} + f_1(7), L_{5,8} + f_1(8)\} = \min\{6+5, 3+7, 4+8\} = 10$ и $U_2(5) = 7$.

Продолжая процесс для n = 3 и n =4, получим

$f_3(2) = \min\{L_{2,4} + f_2(4), L_{2,5} + f_2(5)\} = \min\{7 + 13, 4 + 10\} = 14$, $U_3(2) = 5;$

$f_3(3) = \min\{L_{3,4} + f_2(4), L_{3,5} + f_2(5)\} = \min\{4 + 13, 6 + 10\} = 16$, $U_3(3) = 5;$

$f_4(1) = \min\{L_{1,2} + f_3(2), L_{1,3} + f_3(3)\} = \min\{2 + 14, 3 + 16\} = 16$, $U_4(1) = 2.$

Итак, минимальная длина пути составляет 16 единиц, оптимальный маршрут: 1-2-5-7-9.

Аналогично можно решать задачи, когда требуется найти не минимальное значение целевой функции, а максимальное. В этом случае на каждом шаге операцию min{ }нужно заменить на операцию max{ }.

Заметим, что задачи с мультипликативной целевой функцией в случае положительных мультипликаторов сводятся к задаче с аддитивной целевой функцией путем замены целевой функции на ее логарифм.

Заметим также, что в качестве расстояния между городами может выступать любая неотрицательная функция, которая будет мерой “расстояния” между городами, например, стоимость перевозимого груза по маршруту.

Здесь рассмотрен пример, когда подмножества G_j выделяются однозначно. Однако может быть такой случай, когда из текущего города до конечного могут существовать маршруты с разным количеством шагов (этапов) достижимости, что накладывает некоторые трудности на компьютерную реализацию приведенного алгоритма.

Предлагается решать такого рода задачи с помощью имитационного моделирования движения “автомобилей” по сети дорог, начиная с исходного города. Процесс начинается с одновременного запуска автомобилей с постоянными одинаковыми скоростями по каждой из дорог, выходящей из города. Время движения до города, куда ведет дорога, пропорционально “длине” дороги, по которой движется соответствующий автомобиль. Как только какой-то автомобиль достигает нового города, запускаются в движение новые автомобили по всем дорогам, выходящим из этого города. Разумеется запуск автомобилей в каком-то городе происходит только при первом посещении этого города, тем самым будут отсекаться неконкурентоспособные маршруты. Процесс заканчивается, когда первый автомобиль достигнет конечного города.

Ниже приводится программа такой имитационной модели, написанная на языке СИМУЛА-67. Этот язык был разработан специально для имитационного моделирования дискретных систем (его название происходит от слова simulation - моделирование). СИМУЛА-67, к сожалению незаслуженно “забытый”, дает богатый набор языковых средств, максимально упрощающих программирование имитационных моделей дискретных систем. Описание языка и как работать с ним можно найти, например, в [3].

```
external class terminal;
terminal begin
 /*  декларация глобальных параметров */
 integer N,nn,nk;
 real dlway;

/* N - количество городов; nn - номер города, откуда начинается путь;nk -номер города, где заканчивается путь; dlway - минимальная длина пути(искомый параметр) */
  open (25, 80);
  outtext( “Vvedite obshee kolichestvo gorodov=”);
N:= inint(3);outtext (“->”);outint(N,3);outline(“<-”);

simulation
begin

/* автомобиль  */
process class auto(w);ref(way)w;
 begin
    hold(w.distance);
    w.kuda.pribyl:-w;
    activate w.kuda;
end;

 /* автомобиль  */
 /* дорога */
link class way;
      begin
       ref(city)otkuda,kuda;
       real distance;
      end;

/* дорога  */
/* город */
process class city(numer);integer numer;
 begin
     ref(head) ways;
     ref(way)w,pribyl,wp;
     boolean nebyl;
       nebyl:= true;
       ways:- new head;
  while true do
     begin
       passivate;
       if numer = nk then
        begin wp:-pribyl;
            dlway:= time;
            reactivate MP;
        end else
       if  nebyl then
        begin
          wp:- pribyl;
          nebyl:= false;
            for w:-ways.first,w.suc while w =/= none
do
          activate new auto(w);
        end;
```

```
        end;
    end;

    /* город */
  ref(process) MP;
  integer i,j,m,kd;
  real h;
  ref(city)array citys(1:N);
  ref(city)c;
  ref(way)w;
  ref(head)put;
  boolean b;
     MP:-current;
     put:-new head;

  /* генерация городов */
     for i:= 1 step 1 until N do
       begin
        citys(i) :- new city(i);
        activate citys(i);
       end;

  /* для каждого города генерация и ввод данных
о дорогах */
       for i:= 1 step 1 until N do
          begin
            outtext("vvedite chislo dorog iz goroda");
            outint(i,3);outtext("<-");
            kd:=inint(3);outint(kd,3);
            for j:= 1 step 1 until kd do
              begin
                w:-new way;
                w.otkuda:-citys(i);
                outtext("nomer goroda,kuda doroga=");
                m:=inint(3);
                w.kuda:-citys(m);
                outtext("dlina dorogi=");
                w.distance:=inreal(10);
                w.into(citys(i).ways);
              end;
          end;

  /*генерация сети закончена */
          outtext("vvedite nomer isxodnogo goroda=");
          nn:=inint(3);outline ("<-");
          outtext("vvedite nomer konechnogo goroda=");
          nk:=inint(3);outline ("<-");
           outtext(" vvedite verxnuu ocenku minimalnogo
puty=");
          h:= inreal(10);

    /* Начали движение из начального города */
     activate citys(nn);
     hold(h);

  /* моделирование закончено, вывод результатов
моделирования */
     outtext("min dlina puty=");
     outfix(dlway,2,8);
     outline ("<-");

     /* Вывод пути */
     outtext("Marshrut=");
     b:=true;
     c:-citys(nk);
     while b do
     begin
      c.into(put);
      if c.numer = nn then b:=false else c:-c.wp.otkuda;
     end;
     for c:-put.last,c.pred while c=/=none do
       outint(c.numer,5);
     outline ("Hit a key: ");
     inchar;
     end;
      /* simulation */
     end;
     /* terminal */
```

Приведенная программа для сети (рис.1) выдала следующий результат:

min dlina puty = 16

Marshrut = 1 2 5 7 9

Поясним структуру приведенной программы. В программе имеются объекты трех классов - ав-

томобили (класс auto), дороги (класс way) и города (класс city). Объекты классов auto и city объявлены как процессы (process), то есть их "жизнь" привязана к системному времени, а объекты класса way объявлены как звенья (link) - могут быть членами двунаправленных списков.

Объекты класса way имеют атрибуты otkuda, kuda и distance. Первые два - это ссылки на объекты класса city, показывают, соответственно, откуда начинается дорога и куда она ведет. Третий атрибут указывает на длину дороги. Объекты этого класса в программе используются только как хранилище данных о дорогах сети - никакого сценария жизни у них нет.

Объект класса auto имеет атрибуты w - ссылку на объект класса way, показывающую по какой дороге поедет данный автомобиль. Сценарий жизни объекта класса auto следующий: первый оператор hold(w.distance) планирует следующую активную фазу объекта на момент системного времени time (текущий момент системного времени) + w.distance (длина дороги w). Так как автомобили едут с постоянными одинаковыми скоростями, то в качестве системного времени в данной модели используется длина пройденного пути. Следующим оператором городу, в который едет автомобиль указывается дорога, по которой он едет. Последний оператор активирует город, в который едет автомобиль.

Объект класса city имеет атрибуты numer, ways, w, pribyl, wp и nebyl. Атрибут numer - это номер города в сети, переменная целого типа. Атрибут ways двунаправленный список (объект класса head) - членами которого являются объекты класса way (дороги, выходящие из данного города). Атрибуты w, pribyl, wp - это ссылки на объекты класса way: w - рабочая ссылка, ссылка pribyl указывает на дорогу, по которой прибывает в город очередной автомобиль, а ссылка wp указывает на дорогу, по которой прибыл в город первый автомобиль. Булевская переменная nebyl указывает был или нет в городе автомобиль (nebyl = true автомобилей еще не было: после приезда в город первого автомобиля ее значение становится false). Посредством этой переменной отсекаются заведомо неконкурентоспособные маршруты. Сценарий жизни объекта класса city следующий: переменной nebyl присваивается значение true, генерируется список ways, затем бесконечный цикл, внутри которого первый оператор (passivate) приостанавливает "жизнь" города (делает себя пассивным) и ждет, когда его активирует очередной автомобиль, который едет в этот город. После активации он убеждается является ли этот город конечным городом: если да, то запоминается дорога (в переменной wp), по которой прибыл автомобиль, фиксируется (в переменной dlway) минимальная длина пути (значение системного времени time прибытия первого автомобиля в конечный пункт), в противном случае, если это первый автомобиль (значение nebyl true), запускаются в движение новые автомобили (activate new auto(w)) по всем выходящим из этого города дорогам (цикл по всем дорогам из списка дорог данного города ways). Переменной nebyl присваивается значение false.

В модели присутствует еще один процесс - главная программа. Ссылка на нее MP нужна для того, чтобы закончить моделирование по достижении конечного пункта (оператор reactivate(MP) в теле класса city). В главной программе декларируются ссылка MP, рабочие переменные i, j, m, kd, h, b, рабочие ссылки c, w, put, а также массив ссылок на города сети (ref(city)array citys(1:N);). Сценарий жизни главной программы заключается в следующим: заготавливается ссылка на себя (MP), производится генерация городов, затем для каждого города проводится генерация сети дорог, вводятся номера начального (nn) и конечного (nk) пунктов, активизируется начальный город (activate citys(nn);) и главная программа планирует свою следующую активную фазу (hold(h);) на время time+h, где вводимая перед этим величина h- любое число, большее искомого минимального пути - верхняя его оценка. После окончания моделирования производится вывод результатов - величины минимального пути (dlway) и маршрута следования.

Заметим, что моделировалась система с параллельными процессами, для реализации этой

параллельности в языке СИМУЛА-67 используется так называемая квазипараллельная система исполнения. Конечно, в компьютере все вычисления происходят последовательно, по мере возрастания системного времени происходящих в системе событий, сортировка и вставка уведомлений о событиях в системный список по возрастанию системного времени скрыто от пользователя и реализовано в системе достаточно эффективно.

Таким образом, подробное рассмотрение предложенной программы имитационной модели показывает ее простоту, наглядность, ясность и прозрачность. Структура программы аналогична структуре моделируемой системы, что уменьшает вероятность ошибки при написании такого рода программ имитационных моделей. Мы выражаем уверенность, что в дальнейшем имитационный подход окажется плодотворным для решения и других математических задач.

Список литературы

1. А.В. Кузнецов, В.А. Сакович, Н.И. Холод. Высшая математика. Математическое программирование/Под общей редакцией профессора А.В. Кузнецова. - Минск, Вышэйшая школа, 2001.
2. А.В. Кузнецов, Н.И. Холод, Л.С. Костевич. Руководство к решению задач по математическому программированию/Под общей редакцией А.В. Кузнецова. - Минск, Вышэйшая школа", 2001.
3. А.Н. Андрианов, С.П. Бычков, А.И. Хорошилов. Программирование на языке симула-67/Под редакцией А.Н. Мямлина. - М.,Наука, Главная редакция физико-математической литературы, 1985.