Министерство образования и науки
Российской Федерации

Федеральное государственное бюджетное образовательное
учреждение высшего профессионального образования
«Ивановский государственный энергетический университет
имени В.И. Ленина»

Кафедра высокопроизводительных вычислительных систем

2236

АЛГОРИТМ МУРАВЬЯ

Методические указания

Иваново 2015

Составитель С.Г. СИДОРОВ

Редактор И.Ф. ЯСИНСКИЙ

Методические указания предназначены для использования в качестве справочного материала при выполнении студентами, изучающими дисциплину «Системы искусственного интеллекта», лабораторной работы по теме «Применение алгоритмов муравья при решении задач». Рассматриваются алгоритм муравья и его реализация для решения задачи коммивояжера на языке программирования Pascal.

Утверждены цикловой методической комиссией ИВТФ.

Рецензент
кафедра высокопроизводительных вычислительных систем ФГБОУВПО «Ивановский государственный энергетический университет имени В.И. Ленина»

Содержание

Описание алгоритма

Алгоритмы муравья (Ant algorithms), или оптимизация по принципу муравьиной колонии, были придуманы Марко Дориго (Marco Dorigo) и обладают специфическими свойствами, присущими муравьям, использующим их для ориентации в физическом пространстве. Такие алгоритмы можно использовать для решения не только статичных, но и динамических проблем, например маршрутизации в меняющихся сетях.

Естественная мотивация

Хотя муравьи и слепы, они умеют перемещаться по сложной местности, находить пищу на большом расстоянии от муравейника и успешно возвращаются домой. Выделяя ферменты во время перемещения, муравьи изменяют окружающую среду, обеспечивают коммуникацию, а также отыскивают обратный путь в муравейник.

Чем больше муравьев используют один и тот же путь, тем выше концентрация ферментов на этом пути. Чем ближе внешняя точка к муравейнику, тем чаще к ней перемещались муравьи. Более удаленных точек муравьи достигают реже, поэтому по дороге к ним они применяют более сильные ферменты. Чем выше концентрация ферментов на пути, тем предпочтительнее он для муравьев по сравнению с другими доступными путями. Так «муравьиная логика» позволяет выбирать более короткий путь между конечными точками.

Рассмотрим пример, показанный на рис.1. Два муравья из муравейника должны добраться до пищи, которая находится за препятствием. Во время перемещения каждый муравей выделяет немного фермента, используя его в качестве маркера.

При прочих равных условиях каждый муравей выберет свой путь. Первый муравей выбирает верхний путь, а второй – нижний. Так как нижний путь в два раза короче верхнего, второй муравей достигнет цели за время T_1. Первый муравей в этот момент пройдет только половину пути.

Когда один муравей достигнет пищи, он берет один из объектов и возвращается к муравейнику по тому же пути. За время T_2 второй муравей вернулся в муравейник с пищей, а первый муравей достиг пищи.

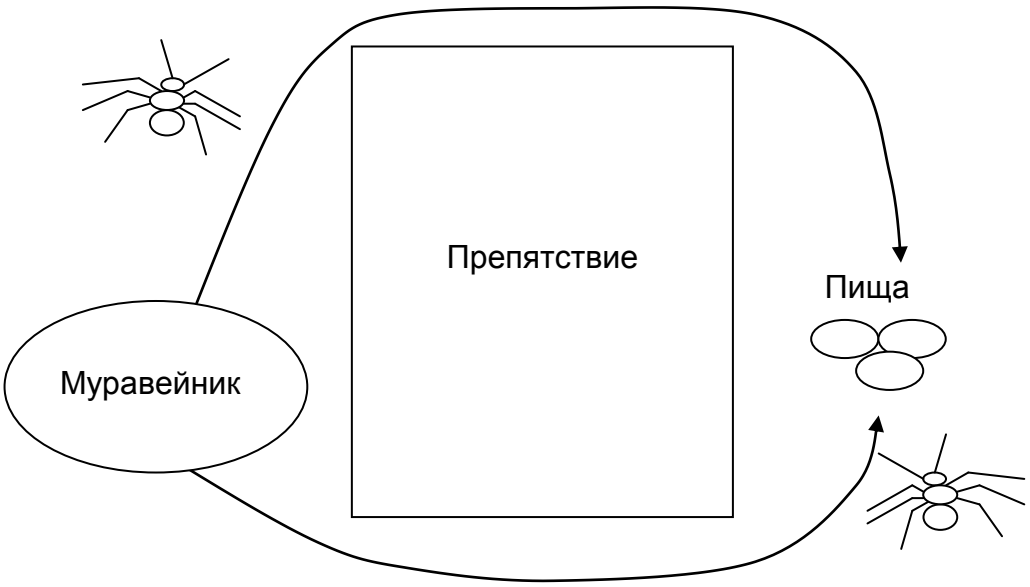


Рис. 1. Маршруты муравьев

При перемещении каждого муравья на пути остается немного фермента. Для второго муравья за время $T_0 – T_2$ путь был покрыт ферментом дважды. За время T_3 второй муравей вновь вернулся к пище, а первый муравей находился на полпути к муравейнику. За время T_4 первый муравей вернулся в муравейник, а второй муравей уже успел дважды достичь места, где находилась пища, и вернуться. При этом концентрация фермента на нижнем пути будет в два раза выше, чем на верхнем. Поэтому первый муравей в следующий раз выберет нижний путь, поскольку там концентрация фермента выше.

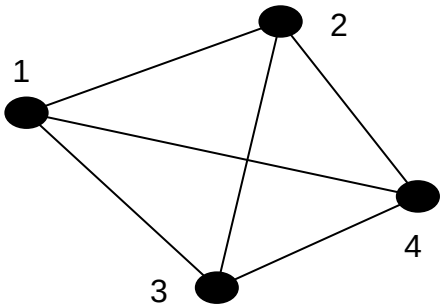
В этом и состоит базовая идея алгоритма муравья – оптимизация путем непрямой связи между автономными агентами.

Алгоритм муравья

Рассмотрим алгоритм муравья при решении конкретной проблемы.

Граф

Предположим, что окружающая среда для муравьев представляет собой закрытую двумерную сеть. Сеть – это группа узлов, соединенных посредством граней. Каждая грань имеет вес, который мы обозначим как расстояние между двумя узлами, соединенными ею. Граф двунаправленный, поэтому муравей может путешествовать по грани в любом направлении (рис.2).



Граф с вершинами V = {1,2,3,4}
Грани E = {{1,2}, {1,4}, {1,3}, {2,3}, {2,4}, {3,4}}

Рис. 2. Граф

Муравей

Муравей – это программный агент, который является членом большой колонии и используется для решения какой-либо проблемы. Муравей снабжается набором простых правил, которые позволяют ему выбирать путь в графе. Он поддерживает список табу (Tabu list), то есть список узлов, которые он уже посетил. Таким образом, муравей должен проходить через каждый узел только один раз. Путь между двумя узлами графа, по которому муравей посетил каждый узел только один раз, называется путем Гамильтона (Hamiltonian path), по имени математика сэра Уильяма Гамильтона (Sir William Hamilton).

Узлы в списке «текущего путешествия» располагаются в том порядке, в котором муравей посещал их. Позже список используется для определения протяженности пути между узлами.

Настоящий муравей во время перемещения по пути будет оставлять за собой фермент. В алгоритме муравья агент оставляет фермент на гранях сети после завершения путешествия.

Начальная популяция

После создания начальная популяция муравьев поровну распределяется по узлам сети. Необходимо равное разделение муравьев между узлами, чтобы все узлы имели одинаковые шансы стать отправной точкой. Если все муравьи начнут движение из одной точки, это будет означать, что данная точка является оптимальной для старта, а на самом деле это не известно.

Движение муравьев

Движение муравьев основывается на одном и очень простом вероятностном уравнении (1). Если муравей еще не закончил путь (path), то есть не посетил все узлы сети, для определения следующей грани пути используется уравнение

$$P = \frac{\tau(r,u)^{\alpha} \times \eta(r,u)^{\beta}}{\sum_{k} \tau(r,u)^{\alpha} \times \eta(r,u)^{\beta}}. \qquad (1)$$

Здесь τ(r,u) – интенсивность фермента на грани между узлами r и u; η(r,u) – функция, которая представляет измерение обратного расстояния для грани; α – вес фермента; β – коэффициент эвристики. Параметры α и β определяют относительную значимость двух параметров, а также их влияние на уравнение.

Муравей путешествует только по узлам, которые еще не были посещены (как указано списком табу). Поэтому вероятность рассчитывается только для граней, которые ведут к еще не посещенным узлам. Переменная k представляет грани, которые еще не были посещены.

Путешествие муравья

Пройденный муравьем путь отображается, когда муравей посетит все узлы диаграммы. Следует обратить внимание на то, что циклы запрещены, поскольку в алгоритм включен список табу. После завершения длина пути может быть подсчитана – она равна сумме всех граней, по которым путешествовал муравей. Уравнение (2) показывает количество фермента, который был оставлен на каждой грани пути для муравья k. Переменная Q является константой.

$$\Delta\tau_{ij}^{k}(t) = \frac{Q}{L^{k}(t)}. \qquad (2)$$

Результат уравнения является средством измерения пути. Короткий путь характеризуется высокой концентрацией фермента, более длинный путь – более низкой. Полученный результат используется в уравнении (3), чтобы увеличить количество фермента вдоль каждой грани пройденного пути.

$$\tau_{ij}(t) = \Delta\tau_{ij}(t) + \left(\tau_{ij}^{k}(t) \times \rho\right). \quad (3)$$

Данное уравнение применяется ко всему пути, при этом каждая грань помечается ферментом пропорционально длине пути. Поэтому следует дождаться, пока муравей закончит путешествие, и только потом обновить уровни фермента, в противном случае истинная длина пути останется неизвестной. Константа ρ – значение между 0 и 1.

Испарение фермента

В начале пути у каждой грани есть шанс быть выбранной. Чтобы постепенно удалить грани, которые входят в худшие пути в сети, ко всем граням применяется процедура испарения фермента (Pheromone evaporation). Используя константу ρ из уравнения (3), применяем для этого уравнение

$$\tau_{ij}(t) = \tau_{ij}(t) \times (1 - \rho). \quad (4)$$

Поэтому для испарения фермента используется обратный коэффициент обновления пути.

Повторный запуск

После того как путь муравья завершен, грани обновлены в соответствии с длиной пути и произошло испарение фермента на всех гранях, алгоритм запускается повторно. Список табу очищается, и длина пути обнуляется. Муравьям разрешается перемещаться по сети, основывая выбор грани на уравнении (1). Этот процесс может выполняться для постоянного количества путей или до момента, когда на протяжении нескольких запусков не было отмечено повторных изменений. Затем определяется лучший путь, который и является решением.

Пример итерации

Разберем алгоритм на примере с муравьями, выбирающими разные пути для достижения одной цели (рис.1). Обозначим узел «муравейник» V_0, а узел «пища» – V_1. Первого муравья обозначим через A_0, второго – A_1. Муравей A_0 пошел по верхнему пути и сделал 20 шагов, A_1 – по нижнему и сделал 10 шагов. По формуле (2) рассчитываем количество «нанесенного» фермента.

ρ=0,6; α=3,0; β=1,0	A_0	A_1
Пройденное расстояние	20	10
Уровень фермента Q / пройденное расстояние	0,5	1,0

По уравнению (3) рассчитываем количество фермента, которое будет применено. Для муравья A_0 результат составляет 0,1+(0,5x0,6) = 0,4. Для муравья A_1 результат составляет 0,1+(1,0x0,6) = 0,7.

Далее с помощью уравнения (4) определяется, какая часть фермента испарится и, соответственно, сколько останется. Результаты (для каждого пути) составляют: 0,4 x (1,0 – 0,6) = 0,16 и 0,7 x (1,0 – 0,6) = 0,28.

Эти значения представляют новое количество фермента для каждого пути (верхнего и нижнего соответственно). Теперь переместим муравьев обратно в узел V_0 и воспользуемся вероятностным уравнением выбора пути (1), чтобы определить, какой путь должны выбрать муравьи.

Вероятность того, что муравей выберет верхний путь (представленный количеством фермента 0,16), составляет (0,16)3,0x(0,5)1,0/((0,16)3,0x(0,5)1,0)+((0,28)3,0x(1,0)1,0) = P(0,085).

Вероятность того, что муравей выберет нижний путь (представленный количеством фермента 0,28), составляет (0,28)3,0x(1,0)1,0 / ((0,16)3,0x(0,5)1,0)+((0,28)3,0x(1,0)1,0) = P(0,915).

При сопоставлении двух вероятностей оба муравья выберут нижний путь, который является наиболее оптимальным.

Код программы решения задачи коммивояжера

```
(************************************************)
(*          Решение задачи коммивояжера         *)
(*                Алгоритм муравья              *)
(*          (c) 2006, ИГЭУ, Сидоров С.Г.        *)
(*           e-mail: sgsidorov@mail.ru          *)
(************************************************)

Program AntKommy;

const
      MaxTowns = 30;                (*число городов*)
      MaxAnts  = 30;                (*число муравьев*)
      Alfa     = 1.0;               (*вес фермента*)
      Beta     = 5.0;               (*эвристика*)
      Rho      = 0.5;               (*испарение*)
      Q        = 100;               (*константа*)
      InitOdor = 1/MaxTowns;        (*начальный запах*)
      MaxWay   = 100;               (*предел координат*)
      MaxTour  = MaxTowns * MaxWay; (*предел пути*)
      MaxTime  = 20 * MaxTowns;     (*предел итераций*)

type
      TVector = array[1..MaxTowns] of byte;

      TMatrix = array[1..MaxTowns,1..MaxTowns] of double;

      TTown = record    (*город*)
        x       :double;     (*абсцисса города*)
        y       :double;     (*ордината города*)
      end;

      TAnt = record     (*муравей*)
        TekTown :word;       (*текущий город*)
        Tabu    :TVector;    (*список табу*)
        Path    :TVector;    (*маршрут муравья*)
        NumTown :word;       (*число городов в маршруте*)
        Len     :double;     (*общая длина пути*)
      end;


var  Towns    :array[1..MaxTowns] of TTown; (*города*)
     Ants     :array[1..MaxAnts] of TAnt;   (*муравьи*)
     DistMap  :TMatrix;                     (*карта расстояний*)
     OdorMap  :TMatrix;                     (*карта запахов*)
     Best     :TAnt;                        (*лучший путь*)
     CurTime  :longint;                     (*текущее время*)
```

```
procedure MakeTowns;   (*создание городов*)
var  i,j    :word;
     xd,yd  :double;
begin
     (*создание городов*)
     for i:=1 to MaxTowns do begin
         Towns[i].x:=random(MaxWay)+1;
         Towns[i].y:=random(MaxWay)+1;
         for j:=1 to MaxTowns do OdorMap[i,j]:=InitOdor;
     end;
     (*вычисление расстояний между городами*)
     for i:=1 to MaxTowns do begin
         DistMap[i,i]:=0;
         for j:=i+1 to MaxTowns do begin
             xd:=Towns[i].x-Towns[j].x;
             yd:=Towns[i].y-Towns[j].y;
             DistMap[i,j]:=sqrt(xd*xd+yd*yd);
             DistMap[j,i]:=DistMap[i,j];
         end;
     end;
end;


procedure MakeAnts(r:word); (*0/1-создание/повторное
муравьев*)
var  i,j,k  :word;
begin
     k:=1;  (*текущий город*)
     for i:=1 to MaxAnts do begin
         if (r>0)and(Ants[i].Len<Best.Len) then Best:=Ants[i];
         Ants[i].TekTown:=k; k:=k+1; if k>MaxTowns then k:=1;
         for j:=1 to MaxTowns do begin
             Ants[i].Tabu[j]:=0;
             Ants[i].Path[j]:=0;
         end;
         Ants[i].Tabu[Ants[i].TekTown]:=1;
         Ants[i].Path[1]:=Ants[i].TekTown;
         Ants[i].NumTown:=1;
         Ants[i].Len:=0;
     end;
end;


function Chance(i,j:word):double; (*вероятность грани*)
begin
     Chance:=Exp(Alfa*Ln(OdorMap[i,j]))*
             Exp(Beta*Ln(1/DistMap[i,j]));
end;
```

```
function NextTown(k:integer):word; (*выбор следующего города*)
var  i,j :word;   (*текущий,следующий город*)
     d,p :double; (*суммарная,текущая вероятность*)
begin
     d:=0; i:=Ants[k].TekTown;
     for j:=1 to MaxTowns do
         if Ants[k].Tabu[j]=0 then d:=d+Chance(i,j);
     j:=MaxTowns;
     if d<>0 then
        Repeat
           j:=j+1; if j>MaxTowns then j:=1;
           if i<>j then p:=Chance(i,j)/d;
        Until (Ants[k].Tabu[j]=0) and (random<p);
     NextTown:=j;
end;


function AntsMoving:boolean; (*перемещение муравьев*)
var  k    :word;    (*номер муравья*)
     m    :boolean; (*флаг перемещений*)
     Next :word;    (*следующий город*)
begin
     m:=false;
     for k:=1 to MaxAnts do begin
         if Ants[k].NumTown<MaxTowns then begin
            Next:=NextTown(k);
            Ants[k].NumTown:=Ants[k].NumTown+1;
            Ants[k].Path[Ants[k].NumTown]:=Next;
            Ants[k].Tabu[Next]:=1;
            Ants[k].Len:=Ants[k].Len+
                         DistMap[Ants[k].TekTown,Next];
            if Ants[k].NumTown=MaxTowns then
               Ants[k].Len:=Ants[k].Len+
            DistMap[Ants[k].Path[MaxTowns],Ants[k].Path[1]];
            Ants[k].TekTown:=Next;  m:=true;
         end;
     end;
     AntsMoving:=m;
end;
```

```
procedure UpdateOdors; (*испарение и нанесение фермента*)
var  i,j,k,ant :integer;
begin
     for i:=1 to MaxTowns do          (*испарение фермента*)
         for j:=1 to MaxTowns do
             if i<>j then begin
                OdorMap[i,j]:=OdorMap[i,j]*(1-Rho);
                if OdorMap[i,j]<0 then OdorMap[i,j]:=InitOdor;
             end;
     for ant:=1 to MaxAnts do begin  (*нанесение фермента*)
         for k:=1 to MaxTowns do begin
             i:=Ants[ant].Path[k];
             if k<MaxTowns then j:=Ants[ant].Path[k+1]
                           else j:=Ants[ant].Path[1];
             OdorMap[i,j]:=OdorMap[i,j]+Q/Ants[ant].Len;
             OdorMap[j,i]:=OdorMap[i,j];
         end;
     end;
     for i:=1 to MaxTowns do
         for j:=1 to MaxTowns do
             OdorMap[i,j]:=OdorMap[i,j]*Rho;
end;


begin
     Randomize; CurTime:=0; Best.Len:=MaxTour;
     MakeTowns; MakeAnts(0);
     While CurTime<MaxTime do begin
           if not AntsMoving then begin
              UpdateOdors;  MakeAnts(1);
              writeln('Время=',CurTime,' Путь=',Best.Len:6:2);
           end;
           CurTime:=CurTime+1;
     end;
     writeln('Оптимальный путь = ',Best.Len:6:2); readln;
end.
```

Обсуждение алгоритма муравья

Пример запуска для 30 городов

```
Время=29 Путь=545.63
Время=59 Путь=512.80
Время=89 Путь=507.48
Время=119 Путь=505.44
Время=149 Путь=505.44
Время=179 Путь=495.47
Время=209 Путь=495.47
Время=239 Путь=495.47
Время=269 Путь=490.20
Время=299 Путь=490.20
Время=329 Путь=490.20
Время=359 Путь=490.20
Время=389 Путь=490.20
Время=419 Путь=490.20
Время=449 Путь=490.20
Время=479 Путь=490.20
Время=509 Путь=490.20
Время=539 Путь=490.20
Время=569 Путь=490.20
Время=599 Путь=490.20
Оптимальный путь = 490.20
```

Изменение параметров алгоритма

Был открыт ряд комбинаций α/β, которые позволяют находить хорошие решения за небольшое время. Эти комбинации приведены в таблице.

α	β
0,5	5,0
1,0	1,0
1,0	2,0
1,0	5,0

Параметр α (Alfa) ассоциируется с количеством фермента, а параметр β (Beta) с видимостью (длиной грани). Чем больше значение параметра, тем он важнее для вероятностного уравнения, которое используется при выборе грани.

Параметр ρ (Rho) представляет коэффициент, который применяется к распыленному на пути ферменту, а (1,0 – ρ) представляет коэффициент испарения для существующего фермента. Были проведены тесты при ρ>0,5, и все они показали интересные результаты. При установке значения ρ<0,5 результаты были неудовлетворительными. В первую очередь этот параметр определяет концентрацию фермента, которая хранится на гранях.

Наилучший результат достигается в том случае, если количество муравьев равно количеству городов.

Области применения

Алгоритм муравья может применяться для решения многих задач распределения ресурсов и работы, требующих оптимизации.

Например:

- прокладка маршрутов для автомобилей;
- расчет цветов для графиков;
- маршрутизации в сетях;
- создание пути;
- реконструкция изображения;
- назначение задач и планирование;
- размещение сети;
- глобальная маршрутизация;
- обнаружение и распознавание визуальных объектов;
- разработка специальных цифровых фильтров.

Более подробно способы применения описаны в книге Марко Дориго «Алгоритмы муравья для абстрактной оптимизации» (Ant Algorithms for Discrete Optimization), 1999.

Задания для самостоятельного выполнения

Вариант задания определяется по номеру персонального компьютера в компьютерном классе либо по согласованию с преподавателем.

Допускается реализация задания на любом из изученных языков программирования.

№	Задание
1	Разработать программу решения задачи о размещении ферзей. Разместить N ферзей на шахматной доске размером NxN так, чтобы ни один ферзь не угрожал другому.
2	Разработать программу решения задачи раскроя. Найти оптимальный вариант раскроя некоторого количества рулонов бумаги фиксированной ширины для различных заказчиков (которым нужны различные количества рулонов различной ширины), минимизировав при этом отходы.
3	Разработать программу решения транспортной задачи. Составить оптимальный план перевозок между N складами и K магазинами, при котором стоимость перевозок будет минимальна. Известна потребность в товаре каждым магазином, наличие товара на складах и стоимость перевозки единицы продукции с каждого склада до каждого магазина.
4	Разработать программу поиска минимального остовного дерева. Есть несколько городов, которые необходимо соединить дорогами так, чтобы можно было добраться из любого города в любой другой (напрямую или через другие города). Разрешается строить дороги между заданными парами городов, и известна стоимость строительства каждой такой дороги. Требуется решить, какие именно дороги нужно строить, чтобы минимизировать общую стоимость строительства.
5	Разработать программу решения задачи о максимальном потоке. Как (т.е. по каким маршрутам) послать максимально возможное количество грузов из начального пункта в конечный пункт, если пропускная способность путей между пунктами ограничена?

6	Разработать программу решения задачи о назначениях. Имеется некоторое число работ и некоторое число исполнителей. Любой исполнитель может быть назначен на выполнение любой (но только одной) работы, но с неодинаковыми затратами. Нужно распределить работы так, чтобы выполнить их с минимальными затратами.
7	Разработать программу о назначении целей. Найти оптимальное распределение комплекта различного вооружения для поражения целей для нанесения максимального поражения противнику.
8	Разработать программу решения задачи о загрузке (ранце). Из заданного множества предметов со свойствами «стоимость» и «вес» требуется отобрать некое число предметов таким образом, чтобы получить максимальную суммарную стоимость при одновременном соблюдении ограничения на суммарный вес.
9	Разработать программу оптимизации выбора оборудования. На приобретение оборудования для цеха заданной площади выделена некоторая сумма. Имеется возможность приобрести станки типов А и Б. Известна площадь, занимаемая станками, их стоимость и производительность за сутки. Найти оптимальный вариант закупок, обеспечивающий максимум общей производительности.
10	Разработать программу планирования номенклатуры и объемов выпуска продукции. Предприятие может выпускать кастрюли, кофеварки и самовары. Известны данные о производственных мощностях, имеющихся на предприятии (штамповка, отделка, сборка в штуках изделий за сутки) и удельная прибыль на одно изделие. При этом штамповка и отделка проводятся на одном и том же оборудовании, а сборка проводится на отдельных участках. Найти оптимальный план производства для максимизации прибыли.
11	Разработать программу решения производственной задачи. Цех может производить стулья и столы. На производство стула идет 5 единиц материала, на производство стола – 20 единиц. Производство стула требует 10 человеко-часов, стола – 15. Имеется 400 единиц материала и 450 человеко-часов. Прибыль при производстве стула – 45 у.е., при производстве стола – 80 у.е. Сколько надо сделать стульев и столов, чтобы получить максимальную прибыль?

12	Разработать программу размещения локальной сети. Найти оптимальную конфигурацию прокладки сетевого кабеля и коммутационного оборудования.
13	Разработать программу оптимизации диеты. Необходимо составить самый дешевый рацион питания цыплят, содержащий необходимое количество определенных питательных веществ (тиамина и ниацина). Известна необходимая пищевая ценность рациона (в калориях), продукты входящие в смесь, содержание тиамина и ниацина в этих продуктах, а также их питательная ценность (в калориях). Сколько надо взять каждого из продуктов для одной порции куриного корма, чтобы цыплята получили необходимую им дозу питательных веществ и калорий, а стоимость порции была минимальна?
14	Разработать программу решения задачи о кратчайшем пути. Как кратчайшим путем (с наименьшим расходом топлива и времени, т.е. дешевле) попасть из пункта А в пункт Б?
15	Разработать программу реконструкции изображения. Имеется множество отсканированных фрагментов изображения. Требуется восстановить исходное изображение по данным фрагментам. Определить также недостающие фрагменты изображения.
16	Разработать программу аппроксимации заданной функции рядом Фурье. Подобрать оптимальное количество и значения коэффициентов Фурье заданной функции.

Контрольные вопросы

1. Опишите суть алгоритма муравья.
2. Какие классы задач можно решать с помощью алгоритма муравья?
3. Назовите основные этапы алгоритма муравья.
4. Как формируется начальная популяция?
5. Как описывается уравнение движения муравьев?
6. Как вычислить количество фермента, оставленного на каждой грани пути?
7. Как описывается постепенное испарение фермента?
8. Как отобрать наилучшее решение?
9. Приведите примеры решения практических задач, используя алгоритм муравья.

АЛГОРИТМ МУРАВЬЯ
Методические указания

Составитель СИДОРОВ Сергей Георгиевич

Редактор Н.С. Работаева

Подписано в печать Формат 60х84 1/16.
Печать плоская. Усл.печ.л. 1,16. Тираж 30 экз. Заказ №

ФГБОУВПО «Ивановский государственный энергетический университет имени В.И.Ленина».
Отпечатано в УИУНЛ ИГЭУ
153003, г.Иваново, ул.Рабфаковская, 34.