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

2237

АЛГОРИТМ ОТЖИГА

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 алгоритма отжига при решении задач». Рассматриваются алгоритм отжига и его реализация для решения задачи о размещении ферзей на языке программирования Pascal.

Утверждены цикловой методической комиссией ИВТФ.

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

АЛГОРИТМ ОТЖИГА
Методические указания

Составитель СИДОРОВ Сергей Георгиевич

Редактор Н.С. Работаева

Подписано в печать Формат 60x84 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.

Содержание

Описание алгоритма

Метод оптимизации, который называется отжигом или симуляцией восстановления (Simulated annealing), моделирует процесс восстановления. Восстановление – это физический процесс, который заключается в нагреве и последующем контролируемом охлаждении субстанции. В результате получается прочная кристаллическая структура, которая отличается от структуры с дефектами, образующейся при быстром беспорядочном охлаждении. Структура здесь представляет собой кодированное решение, а температура используется для того, чтобы указать, когда и как будут приниматься новые решения.

После нагрева субстанции до точки плавления свойства структуры зависят от коэффициента охлаждения. Если структура охлаждается медленно, будут формироваться крупные кристаллы, что очень полезно для строения субстанции. Если субстанция охлаждается скачкообразно, образуется слабая структура.

Чтобы расплавить материал, требуется большое количество энергии. При понижении температуры уменьшается и количество энергии.

Пример

Взбалтываем емкость. В ней находится поверхность сложной формы и шарик, который пытается найти точку равновесия.

При высокой температуре шарик свободно перемещается по поверхности, а при низкой взбалтывание становится менее интенсивным и перемещения шарика сокращаются.

Задача заключается в том, чтобы найти точку минимального перемещения при сильном взбалтывании. При снижении температуры уменьшается вероятность того, что шарик выйдет из точки равновесия.

Алгоритм отжига может быть разделен на пять основных этапов (рис.1).

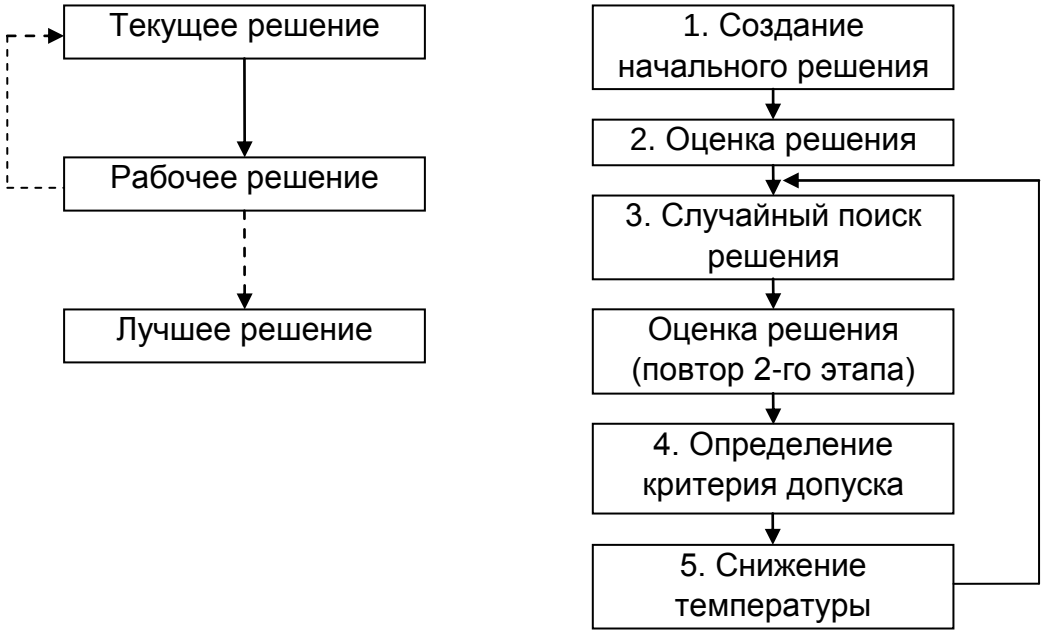

Рис. 1. Блок-схема алгоритма

Создание начального решения

Для большинства проблем начальное решение будет случайным. На первом шаге оно помещается в текущее решение (Current solution). Другая возможность заключается в том, чтобы загрузить в качестве начального решения уже существующее, возможно найденное во время предыдущего поиска.

Оценка решения

Оценка решения состоит из декодировки текущего решения и выполнения нужного действия, позволяющего понять его целесообразность для решения данной проблемы. Такое решение может просто состоять из набора переменных, а эффективность решения будет оценена на основании того, насколько успешно удалось решить данную задачу.

Случайный поиск решения

Поиск решения начинается с копирования текущего решения в рабочее решение (Working solution). Затем рабочее решение произвольно модифицируется. Способ модификации зависит от способа кодирования. Например, при решении задачи коммивояжера, в которой каждый элемент представляет собой город, можно просто переставить местами два элемента. Это позволяет сохранить целостность решения, и при этом не происходит повторения или пропуска города. После выполнения поиска рабочего решения производится его оценка. Поиск нового решения основан на методе Монте-Карло (случайным образом).

Определение критерия допуска

На этом этапе алгоритма имеется два решения. Первое – оригинальное решение, которое называется текущим, второе – найденное решение, называемое рабочим. С каждым решением связана определенная энергия, представляющая собой его эффективность (например, чем ниже энергия, тем более эффективно решение).

Затем рабочее решение сравнивается с текущим решением. Если рабочее решение имеет меньшую энергию, чем текущее (то есть является более предпочтительным), то рабочее решение копируется в текущее решение и происходит переход к этапу снижения температуры.

Если рабочее решение хуже, чем текущее решение, определяется критерий допуска, чтобы выяснить, что следует сделать с текущим рабочим решением. Вероятность допуска основывается на уравнении (базируется на законе термодинамики)

$$P(\delta E) = \exp(-\delta E / T). \qquad (1)$$

Значение этой формулы визуально представлено на графике (рис. 2). При высокой температуре (свыше 60 ^{o}C) плохие решения принимаются чаще, чем отбрасываются. Если энергия меньше, вероятность принятия решения выше. При снижении температуры вероятность принятия худшего решения также снижается. При этом более высокий уровень энергии также способствует уменьшению вероятности принятия худшего решения.

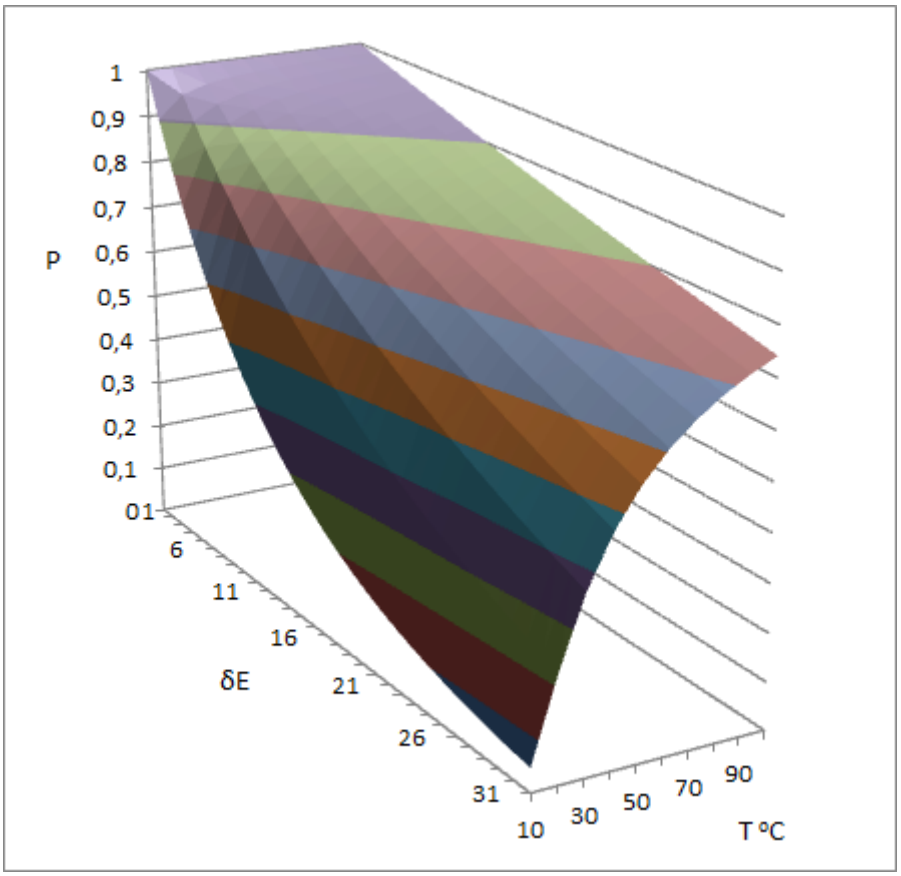

Рис. 2. Изменение вероятности допуска P

При высоких температурах симулированное восстановление позволяет принимать худшие решения для того, чтобы произвести более полный поиск решений. При снижении температуры диапазон поиска также уменьшается, пока не достигается равенство при температуре 0 ^{o}C.

Снижение температуры

После ряда итераций по алгоритму при данной температуре ее ненамного снижают. Существует много вариантов снижения температуры, например: $T_{i+1} = aT_i$. Константа a меньше единицы.

Повтор

При одной температуре выполняется несколько итераций. После их завершения температура будет понижена. Процесс продолжится, пока температура не достигнет нуля.

Пример итерации

Предположим, что температура окружающей среды равна 50 ^{o}C, а энергия текущего решения составляет 10. Текущее решение копируется в рабочее, и осуществляется поиск. После оценки энергии энергия нового рабочего решения равна 20. В этом случае энергия рабочего решения выше, чем энергия начального решения. Поэтому используем критерий допуска:

энергия текущего решении равна 10;

энергия рабочего решения равна 20;

дельта энергии равна 20 – 10 = 10.

Подставив это значение и значение температуры 50 ^{o}C в уравнение (1), получаем вероятность P = exp(–10/50) = 0,818731. Видно, что вероятность принятия худшего решения достаточно велика.

Предположим, что температура окружающей среды равна 2 ^{o}C, а энергия имеет следующие показатели:

энергия текущего решения равна 3;

энергия рабочего решения равна 7;

дельта энергии равна 7 – 3 = 4.

Подставив это значение и значение температуры 2 ^{o}C в уравнение (1), получаем вероятность P = exp(–4/2) = 0,135335. Видно, что вероятность выбора рабочего решения для последующих итераций невелика.

Пример решения задачи о размещении N ферзей (NQP)

Постановка задачи

Разместить N ферзей на шахматной доске размером NxN так, чтобы ни один ферзь не угрожал другому (рис. 3).

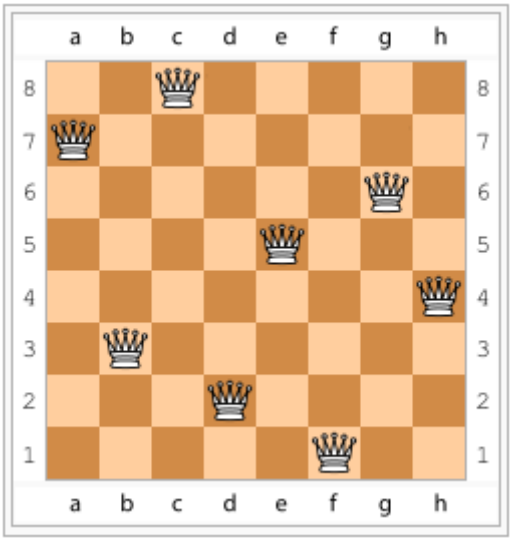

Рис. 3. Пример размещения ферзей на шахматной доске

Историческая справка

Задача о размещении 8 ферзей впервые была решена в 1850 г. Карлом Фридрихом Гаубом (Carl Friedrich Gaub). Применялся метод проб и ошибок. Затем задача решалась методом поиска в глубину (1987), методом «разделяй и властвуй» (1989), генетическими алгоритмами (1992) и др. В 1990 г. Рок Сосик (Rok Sosic) и Цзюн Гу (Jun Gu) решили проблему для 3000000 ферзей с использованием метода локального поиска и минимизации конфликтов.

Представление решения

Представление решения (кодировка) задачи стандартна: для сужения области поиска используется конечное решение. В каждой строке и каждом столбце может располагаться только один ферзь. Это ограничение намного упрощает создание кодировки, которая управляется алгоритмом отжига.

Так как каждый столбец содержит только одного ферзя, для отображения решения будет использоваться массив из N элементов. В элементах этого массива хранятся строчные индексы положения ферзя. Например, для указанного рисунка это

2	6	1	7	4	8	3	5

.

Для создания произвольного решения сначала нужно его инициализировать, позволив каждому ферзю занять строку, соответствующую столбцу. Затем необходимо пройти по каждому столбцу и выбрать произвольное число от 1 до N для каждого столбца. Затем два элемента перемещаются (текущий столбец и произвольно выбранный столбец). Когда алгоритм достигает конца, автоматически формируется решение.

Получив кодировку, видим, что на доске нет конфликтов по горизонтали или диагонали. При оценке решения следует учитывать только конфликты по диагонали.

Энергия

Энергия решения определяется как количество конфликтов, которые возникают в кодировке. Задача заключается в том, чтобы найти кодировку, при которой энергия равна нулю (то есть на доске нет конфликтов).

Температура

Для данной проблемы поиск решения начинается с температуры 30 ^{o}C, и постепенно она снижается до нуля (см. формулу на с.7). При этом значение a=0,98. Как будет видно далее, график температуры показывает сначала быстрое снижение, а потом медленное схождение к конечной температуре – нулю. При каждом изменении температуры выполним 100 итераций. Это позволит в соответствии с алгоритмом осуществить несколько операций поиска на каждом уровне.

Код программы решения задачи о размещении ферзей

```
(************************************************)
(*       Решение задачи о размещении ферзей     *)
(*                 Метод отжига                 *)
(*          (c) 2006, ИГЭУ, Сидоров С.Г.        *)
(*           e-mail: sgsidorov@mail.ru          *)
(************************************************)

Program Otzig;


const N    = 20;   (*размер доски и число ферзей*)
      Tn   = 30.0; (*начальная температура*)
      Tk   = 0.5;  (*конечная температура*)
      Alfa = 0.98; (*скорость охлаждения*)
      ST   = 100;  (*число итераций при смене T*)


type  TMember = record    (*решение*)
         Plan   :array[1..N] of word; (*кодировка*)
         Energy :word;                (*энергия*)
      end;


var  Current     :TMember;      (*текущее решение*)
     Working     :TMember;      (*рабочее решение*)
     Best        :TMember;      (*лучшее решение*)
     T           :double;       (*температура*)
     Delta       :double;       (*разница энергий*)
     P           :double;       (*вероятность допуска*)
     fNew        :boolean;      (*флаг нового решения*)
     fBest       :boolean;      (*флаг лучшего решения*)
     Time        :longint;      (*этап поиска*)
     Step        :longint;      (*шаг на этапе поиска*)
     Accepted    :longint;      (*число новых решений*)


procedure Swap(var M:TMember); (*модификация решения*)
var  x,y,v :word;
begin
     x:=Random(N)+1;
     Repeat
           y:=Random(N)+1;
     Until x<>y;
     v:=M.Plan[x]; M.Plan[x]:=M.Plan[y]; M.Plan[y]:=v;
end;
```

```
procedure New(var M:TMember); (*инициализация решения*)
var  i:word;
begin
     for i:=1 to N do M.Plan[i]:=i;
     for i:=1 to N do Swap(M);
end;

procedure CalcEnergy(var M:TMember); (*расчет энергии*)
const dx:array[1..4] of integer = (-1,1,-1,1);
      dy:array[1..4] of integer = (-1,1,1,-1);
var  j,x,tx,ty :word;
     error     :word;
begin
     error:=0;
     for x:=1 to N do begin
         for j:=1 to 4 do begin
             tx:=x+dx[j]; ty:=M.Plan[x]+dy[j];
             While (tx>0)and(tx<=N)and
                   (ty>0)and(ty<=N) do begin
                   if M.Plan[tx]=ty then inc(error);
                   tx:=tx+dx[j]; ty:=ty+dy[j];
             end;
         end;
     end;
     M.Energy:=error;
end;

procedure Copy(var MD,MS:TMember); (*копирование решения*)
var  i:word;
begin
     for i:=1 to N do MD.Plan[i]:=MS.Plan[i];
     MD.Energy:=MS.Energy;
end;

procedure Show(M:TMember); (*отображение на экране*)
var  x,y :word;
begin
     writeln('Решение:');
     for y:=1 to N do begin
         for x:=1 to N do
             if M.Plan[x]=y then write('Q')
                            else write('.');
         writeln;
     end;
     writeln;
end;
```

```
begin
     Randomize; T:=Tn; fBest:=false;
     Time:=0; Best.Energy:=100;
     New(Current);
     CalcEnergy(Current);
     Copy(Working,Current);
     While T>Tk do begin
           Accepted:=0;
           for Step:=0 to ST do begin
               fNew:=false;
               Swap(Working);
               CalcEnergy(Working);
               if Working.Energy<=Current.Energy then
                  fNew:=true
               else begin
                  Delta:=Working.Energy-Current.Energy;
                  P:=exp(-Delta/T);
                  if P>random then begin
                     Accepted:=Accepted+1;
                     fNew:=true;
                  end;
               end;
               if fNew then begin
                  fNew:=false;  Copy(Current,Working);
                  if Current.Energy<Best.Energy then begin
                     Copy(Best,Current); fBest:=true;
                  end;
               end else begin
                  Copy(Working,Current);
               end;
           end;
           writeln('Temp=',T:6:2,' Energy=',best.Energy:4);
           T:=T*Alfa;
           Time:=Time+1;
     end;
     if fBest then Show(Best);
end.
```

Описание кода реализации алгоритма отжига

Описание подпрограмм

Функция New создает решение, при котором все ферзи помещаются на доску. Для каждого ферзя задаются идентичные индексы строки и столбца. Это обозначает отсутствие конфликтов по горизонтали и вертикали. Затем решение изменяется с помощью процедуры Swap. Позднее Swap используется в алгоритме, чтобы изменить рабочее решение, выведенное из текущего решения.

Оценка решения выполняется с помощью процедуры CalcEnergy. Процедура идентифицирует все конфликты, которые существуют для текущего решения. Диапазоны dx и dy используются для расчета следующего положения на доске для каждого найденного ферзя.

По очереди выбирается каждый ферзь на доске, а затем происходит перемещение по всем четырем диагоналям в поиске конфликтов, то есть ищется другой ферзь. Если ферзь найден, значение переменной конфликта error увеличивается.

Процедура Copy используется для копирования одного решения (MS) в другое (MD). Решение кодируется и сохраняется в структуре TMember.

Процедура Show выдает на экран представление доски из закодированного решения. Символ «Q» соответствует ферзю, а «.» соответствует пустой клетке доски.

Описание программы

После инициализации начальных значений переменных происходит инициализация текущего решения в переменной Current и оценка энергии решения при помощи процедуры CalcEnergy. Текущее решение копируется в рабочее, после чего запускается алгоритм.

Внешний цикл алгоритма выполняется до тех пор, пока текущая температура не станет меньше конечной температуры или не сравняется с ней. Это позволяет избежать использования нулевой температуры в функции расчета вероятности.

Внутренний цикл работает по методу Монте-Карло. Он выполняет ряд итераций при текущей температуре в целях полного изучения возможностей поиска при данной температуре.

Первый шаг – изменение рабочего решения с помощью процедуры Swap. Затем рассчитывается энергия рабочего решения, которая сравнивается с текущим решением. Если энергия нового рабочего решения меньше или равна энергии текущего решения, рабочее решение принимается по умолчанию. В противном случае выполняется уравнение оценки вероятности допуска. Таким образом, определяется, будет ли выбрано худшее решение. Дельта энергии рассчитывается как разница между рабочей энергией и текущей. Это означает, что энергия рабочего решения больше, чем энергия текущего решения. В программе просто генерируется случайное число в интервале от 0 до 1, которое затем сравнивается с результатом уравнения оценки допуска. Если условие допуска выполнено (результат уравнения больше случайного значения), то рабочее решение принимается. Затем рабочее решение необходимо скопировать в текущее, так как переменная Working, в которой на данный момент хранится рабочее решение, будет повторно изменена при следующей итерации внутреннего цикла.

Если рабочее решение не было принято, текущее решение копируется поверх рабочего. При следующей итерации старое рабочее решение удаляется, программа изменяет текущее решение, и далее поиск повторяется.

Выполнив требуемое количество итераций внутреннего цикла, температуру необходимо снизить. Для этого она умножается на константу Alfa, после чего внешний цикл повторяется.

В конце алгоритма выводится решение, хранящееся в переменной Best, которое было найдено (если вообще было найдено). О найденном решении сигнализирует переменная fBest. Она устанавливается во внутреннем цикле после того, как было определено, что обнаружено решение, энергия которого меньше энергии текущего решения Best.

Области применения

Метод отжига может быть эффективным при решении задач различных классов, требующих оптимизации. Например:

- создание пути;
- реконструкция изображения;
- назначение задач и планирование;
- размещение сети;
- глобальная маршрутизация;
- обнаружение и распознавание визуальных объектов;
- разработка специальных цифровых фильтров.

Поскольку метод отжига представляет собой процесс генерации случайных чисел, поиск решения с использованием данного алгоритма может занять значительное время. В некоторых случаях применение алгоритма может вообще не позволить найти решение или выбрать не самое оптимальное.

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 коммивояжера. Найти оптимальный маршрут движения коммивояжера между городами, исключающий повторное посещение городов. Известны координаты городов (расстояния между городами).
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. Для изготовления стула требуется 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 отжига.
2. Какие классы задач можно решать с помощью алгоритма отжига?
3. Назовите основные этапы алгоритма отжига.
4. Как кодируется текущее решение?
5. Как оценить текущее и новое решения?
6. Какие критерии допуска используются в алгоритме?
7. Как уменьшить температуру в процессе работы алгоритма?
8. Как отобрать наилучшее решение?
9. Приведите примеры решения практических задач с помощью алгоритма отжига.