Пример. Составление оптимального плана производства

Основные этапы моделирования

(план выполнения задания).

1. Формализация линейной оптимизационной задачи;
2. Построение математической модели;
3. Реализация математической модели в форме файла .mod;
4. Создание файла .dat;
5. Решение оптимизационных задач средствами AMPL;
6. Итоговый отчет (формулировка проблемы, математическая модель ЛП, содержание файлов .mod, .dat, .run, результат).

1. Описание задачи

Производственная компания Paint Deals производит краски двух цветов: синюю и черную. Синяя краска продается по цене 10 руб. за литр, а черная краска продается по цене 15 руб. за литр. Компания владеет технологическим заводом, который может производить краску одного цвета за раз. Однако скорость производства синей краски составляет 40 литров в час, а скорость производства черной краски составляет 30 литров в час. Кроме того, по оценке отдела маркетинга, на рынке можно продать не более 860 литров черной краски и 1000 литров синей краски. В течение недели установка может работать 40 часов, а краску можно хранить в течение следующей недели.
Определите, сколько литров каждой краски необходимо произвести за неделю, чтобы максимизировать недельный доход.

2. Построение математической модели

Для решения задачи необходимо построить математическую модель. Процесс построения модели можно начать с ответа на следующие вопросы:

1. Какие величины необходимо определить в модели (т.е. каковы переменные модели)?
2. Какова цель оптимизации?
3. Как численно описать эту цель?
4. Каким ограничениям должны удовлетворять переменные?

Производственному менеджеру компании необходимо спланировать объем производства красок так, чтобы максимизировать прибыль от их продажи. Переменными модели являются:
x_1 — еженедельный объем выпуска краски 1 типа, л.,
x_2 — еженедельный объем выпуска краски 2 типа, л.,

Суммарная еженедельная прибыль при реализации красок всех типов составляет:
$Z=10x_1+15x_2$.
Целью компании является определение среди всех допустимых значений x_1, x_2, таких, которые максимизируют суммарную прибыль Z (целевую функцию).

Перейдем к ограничениям, которым должны удовлетворять переменные x_1, x_2. Объем производства ни одного вида продукции не может быть отрицательным, поэтому:
$x_1, x_2 \geq 0$.

Руководство компании ограничивает объемы, из-за величины спроса на краски:

$$x_1 \leq 1000,\ x_2 \leq 860.$$

Время работы оборудования ограничено:

$$\frac{x_1}{40} + \frac{x_2}{30} \leq 40.$$

Получаем модель:

$$\max(10x_1 + 15x_2)$$
$$\frac{x_1}{40} + \frac{x_2}{30} \leq 40;$$
$$x_1 \geq 0;$$
$$x_2 \geq 0;$$
$$x_1 \leq 1000;$$
$$x_2 \leq 860.$$

Запускаем amplide.exe
Создаем файл модели e1.mod

```
var x1; # количество краски 1
var x2; # количество краски 1
maximize z: 10*x1 + 15*x2;
subject to time: (1/40)*x1 + (1/30)*x2 <= 40;
subject to 1_limit: 0 <= x1 <= 1000;
subject to 2_limit: 0 <= x2 <= 860;
```

Все данные в модели и ее можно решить.
Создаем файл e1.run

```
reset;

model e1.mod;
option solver cplex;
solve;
display z;
display x1;
display x2;
```

В окне файла e1.run щелкаем правой клавишей мыши. Выбираем Send To AMPL. В окне Console получаем ответ:

```
ampl: include 'd:\Lab1\e1.run';
CPLEX 12.6.3.0: optimal solution; objective 17433.33333
1 dual simplex iterations (0 in phase I)
z = 17433.3
x1 = 453.333
x2 = 860
```

3. Более общая модель

Преобразовать задачу ЛП в формат, понятный AMPL, было легко, ясно, что если бы задача имела больше деталей или часто менялась, то подготовить ее к решению в AMPL намного сложнее.

По этой причине мы обычно используем более общий алгебраический способ формулирования моделей линейного программирования. Проведем следующую формализацию:

Дано:

n – число красок;

t – время работы оборудования;

p_i – прибыль с литра краски;

r_i – литров производит оборудование краски цвета i за час;

m_i – объем спроса на краску i;
Переменные:
x_i — еженедельный объем выпуска краски i, л.,
Максимизировать:

$$\sum_{i=1}^{n} p_i x_i$$

Условия:

$$\sum_{i=1}^{n} \frac{1}{r_i} x_i \leq t$$

$$0 \leq x_i \leq m_i, \quad i = 1, \ldots, n.$$

Если $n = 2$, $t = 40$, $p_1 = 10$, $p_2 = 15$, $r_1 = 40$, $r_2 = 30$, $m_1 = 1000$, и $m_2 = 860$, то получаем задачу из раздела 2. На самом деле, запись модели заняло даже больше времени! Однако вы можете представить, что по мере увеличения количества красок алгебраическая формулировка становится более эффективной.
Если у нас есть способ определить параметры (n; t; p_i; r_i; m_i), модель останется неизменной независимо от их значений. Таким образом, мы видим, что первый пример на самом деле является экземпляром массовой задачи о производстве красок, а не моделью.
Это именно то, что заложено в AMPL. У нас есть возможность выразить алгебраическое представление модели и эти значения для параметров по отдельности. Для этого используются два отдельных файла, файл модели и файл данных. AMPL считывает модель из файла .mod, данные из файла .dat и объединяет их в формат, понятный для решающей программы. Затем он передает этот экземпляр задачи решающей программе, которая, в свою очередь, решает экземпляр задачи и возвращает решение.
Посмотрите на новую модель e2.mod:

```
param n;
param t;
param p{i in 1..n};
param r{i in 1..n};
param m{i in 1..n};
var paint{i in 1..n};
maximize z: sum{i in 1..n} p[i]*paint[i];
subject to time: sum{i in 1..n} (1/r[i])*paint[i] <= t;
subject to capacity{i in 1..n}: 0 <= paint[i] <= m[i];
```

Сравните эту модель с моделью из предыдущего раздела. Легко видеть, что это более общая модель. Обратите внимание на следующее:
Объявление каждого параметра начинается с ключевого слова param;
Индексированные параметры объявляются с использованием синтаксической переменной name_variable in range. Например, для параметра, который мы алгебраически объявили как p_i, i = 1..n, мы можем использовать i как индексную переменную и 1..n как диапазон, и, таким образом, мы получаем p{i in 1..n}.
Индексированные переменные объявляются таким же образом, начиная с ключевого слова var.
Аналогично записывается суммирование: sum{i in 1..n}
Иногда мы представляем наборы ограничений в одной строке. В формулировке в начале этого раздела мы написали:
$0 \leq paint_i \leq m_i$ for each i; $i = 1,\ldots,n$
для одного из ограничений.
Фактически, это набор из n ограничений, по одному для каждого цвета краски. В AMPL это выражается в фигурных скобках после имени ограничения. Таким образом, это ограничение становится:
subject to paint {i in 1..n}: 0 <= paint[i] <= m[i];

Помимо указания модели, мы также должны указать данные. Это данные для той же проблемы, которую мы решили в предыдущем разделе, которую мы сохраняем в файле e2.dat.

```
param n:= 2;
param t:= 40;
param p:= 1 10 2 15;
param r:= 1 40 2 30;
param m:= 1 1000 2 860;
```

Как видите, один из способов указать данные параметра – это использовать ключевое слово param, двоеточие, знак равенства и значение. Если параметр имеет более одного компонента, просто укажите индекс параметра (в данном случае 1 или 2), за которым следует значение.

Обратите внимание: поскольку AMPL игнорирует символы возврата каретки в файлах и вместо этого ищет точки с запятой для завершения строк, вы также можете записать последние три параметра как:

```
param p:= 1 10
2 15;
param r:= 1 40
2 30;
param m:= 1 1000
2 860;
```

Если модель и данные были введены правильно, вы должны получить точно такой же результат и целевое значение, как и раньше.

Файл e2.run

```
reset;

model e2.mod;
data e2.dat;
option solver cplex;
solve;
display z;
display paint;
```

Результат:

```
CPLEX 12.6.3.0: optimal solution; objective 17433.33333
1 dual simplex iterations (0 in phase I)
z = 17433.3

paint [*] :=
1  453.333
2  860
;
```

Изменяя только файл данных, теперь вы можете решить эту модель с другими параметрами стоимости и различным количеством возможных красок.

4.1. Другой способ ввода данных

В AMPL есть несколько способов задания данных, некоторые из которых более краткие, чем способ, который мы видели в предыдущем разделе. Хотя многие из них применимы скорее к более сложным моделям, данные для простого примера, показанного выше, можно записать так:

```
param n:= 2;
param t:= 40;
param: p r m:=
1 10 40 1000
2 15 30 860;
```

4.2. Множества

В предыдущих примерах мы должны были помнить, что каждый параметр и переменная номер 1 были для краски 1, а 2 - для 2-ой краски. Вместо этого AMPL позволяет нам определять множество красок и использовать эти имена напрямую для индексации параметров и переменных. Взгляните на следующий пример e4.mod:

```
set P;
param t;
param p{i in P};
param r{i in P};
param m{i in P};
var paint{i in P};
maximize z: sum{i in P} p[i]*paint[i];
subject to time: sum{i in P} (1/r[i])*paint[i] <= t;
subject to capacity{i in P}: 0 <= paint[i] <= m[i];
```

Если вы сравните эту версию второй модели с исходной версией, вы увидите, что они очень похожи. Однако есть несколько важных отличий:

Вместо того, чтобы определять параметр n для представления количества красок, мы определили набор P, который фактически содержит краски. Это станет яснее, когда мы определим данные ниже.

Вместо того, чтобы индексировать параметры по индексам 1..n, мы проиндексировали их по элементам множества P

Новый вариант файла данных e4.dat:

```
set P:= blue gold;
param t:= 40;
param p:= blue 10
gold 15;
param r:= blue 40
gold 30;
param m:= blue 1000
gold 860;
```

Обратите внимание, что после определения набора каждый элемент параметра был указан с использованием имени элемента набора, а не номера индекса. Как видите, это намного легче читать и следовать. Фактически, мы можем комбинировать обозначение набора с более компактным обозначением данных, представленным в предыдущем разделе, и мы получаем еще вариант файла данных e4.dat:

```
set P:= blue gold;
param t:= 40;
param: p r m:=
blue 10 40 1000
gold 15 30 860;
```

Результат:

```
CPLEX 12.6.3.0: optimal solution; objective 17433.33333
1 dual simplex iterations (0 in phase I)
z = 17433.3

paint [*] :=
blue  453.333
gold  860
;
```

5. Переменные и параметры с двумя измерениями

Часто модель математического программирования имеет переменные и параметры с двумя измерениями. Мы можем записывать данные параметров в табличной форме, и AMPL позволяет нам вводить эту информацию в табличной форме в файл данных. Рассмотрим следующий пример. Компания Лида расширилась и теперь имеет три склада, все хранят краску. В конкретную неделю краска должна быть отправлена четырем

покупателям. Для каждого склада и каждого клиента стоимость доставки кг. краски разная. Стоимость доставки указана в следующей таблице. Склады указаны в первом столбце, клиенты - в первой строке, а стоимость доставки от склада до клиента можно прочитать из таблицы.

	Магазин 1	Магазин 2	Магазин 3	Магазин 4
Склад 1	1	2	1	3
Склад 2	3	5	1	4
Склад 3	2	2	2	2

Кроме того, запас краски на каждом из складов составляет:

Склад 1	250
Склад 2	800
Склад 3	760

Запрос из магазинов:

Магазин 1	300
Магазин 2	320
Магазин 3	800
Магазин 4	390

Обратите внимание, что общее предложение равно общему спросу. Следующая модель AMPL настроена для минимизации общей стоимости при условии удовлетворения спроса.
Посмотрите на новую модель e5.mod:

```
param warehouse; # number of warehouses
param customer; # number of customers
#transportation cost from warehouse i
#to customer j
param cost{i in 1..warehouse, j in 1..customer};
param supply{i in 1..warehouse}; #supply at warehouse i
param demand{i in 1..customer}; #demand at customer j
var amount{i in 1..warehouse, j in 1..customer};
minimize Cost:
sum{i in 1..warehouse, j in 1..customer} cost[i,j]*amount[i,j];
subject to Supply {i in 1..warehouse}:
sum{j in 1..customer} amount[i,j] = supply[i];
subject to Demand {j in 1..customer}:
sum{i in 1..warehouse} amount[i,j] = demand[j];
subject to positive{i in 1..warehouse, j in 1..customer}:
amount[i,j]>=0;
```

Соответствующий файл данных e5.dat.:

```
param warehouse:= 3;
param customer:= 4;
param cost: 1 2 3 4 :=
1 1 2 1 3
2 3 5 1 4
3 2 2 2 2;
param supply:= 1 250 2 800 3 760;
param demand:= 1 300 2 320 3 800 4 390;
```

Файл e5.run:

```
reset;

model e5.mod;
data e5.dat;
option solver cplex;
solve;
```

```
display Cost;
display amount;
```
Результат:
```
ampl: model 'd:\Lab1\e5.mod';
ampl: include 'd:\Lab1\e5.run';
CPLEX 12.6.3.0: optimal solution; objective 2570
5 dual simplex iterations (0 in phase I)
Cost = 2570

amount :=
1 1   250
1 2     0
1 3     0
1 4     0
2 1     0
2 2     0
2 3   800
2 4     0
3 1    50
3 2   320
3 3     0
3 4   390
;
```

Приведем модель, в которой используются множества e52.mod:
```
set Warehouses;
set Customers;
#transportation cost from warehouse i
#to customer j
param cost{i in Warehouses, j in Customers};
param supply{i in Warehouses}; #supply at warehouse i
param demand{j in Customers}; #demand at customer j
var amount{i in Warehouses, j in Customers};
minimize Cost:
sum{i in Warehouses, j in Customers} cost[i,j]*amount[i,j];
subject to Supply {i in Warehouses}:
sum{j in Customers} amount[i,j] = supply[i];
subject to Demand {j in Customers}:
sum{i in Warehouses} amount[i,j] = demand[j];
subject to positive{i in Warehouses, j in Customers}:
amount[i,j]>=0;
```
Файл данных может выглядеть так e52.dat:
```
set Warehouses:= Oakland San_Jose Albany;
set Customers:= Home_Depot K_mart Wal_mart Ace;
param cost: Home_Depot K_mart Wal_mart Ace:=
Oakland 1 2 1 3
San_Jose 3 5 1 4
Albany 2 2 2 2;
param supply:= Oakland 250
San_Jose 800
Albany 760;
param demand:= Home_Depot 300
K_mart 320
Wal_mart 800
Ace 390;
```
Файл e52.run:
```
reset;

model e52.mod;
data e52.dat;
option solver cplex;
```

```
solve;
display Cost;
display amount;
```
Результат:
```
ampl: include 'd:\Lab1\e52.run';
CPLEX 12.6.3.0: optimal solution; objective 2570
5 dual simplex iterations (0 in phase I)
Cost = 2570

amount :=
Albany   Ace          390
Albany   Home_Depot    50
Albany   K_mart       320
Albany   Wal_mart       0
Oakland  Ace            0
Oakland  Home_Depot   250
Oakland  K_mart         0
Oakland  Wal_mart       0
San_Jose Ace            0
San_Jose Home_Depot     0
San_Jose K_mart         0
San_Jose Wal_mart     800
;
```

6. ПОСТРОЕНИЕ МАТЕМАТИЧЕСКИХ МОДЕЛЕЙ ЦЕЛОЧИСЛЕННОГО ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ.

Часто модель математического программирования требует, чтобы некоторые (или все) переменные принимали только целые значения. К счастью, AMPL позволяет нам включить это с небольшим изменением модели. Добавив ключевое слово integer в объявление var, мы можем ограничить объявленную переменную целочисленными значениями. Кроме того, добавив ключевое слово binary в объявление var, мы можем ограничить объявленную переменную значениями 0 и 1.

В качестве иллюстрации рассмотрим следующий пример. Компания Лида теперь расширила производственные мощности на каждом из своих складов. Однако за это расширение пришлось заплатить. Помимо затрат на доставку, теперь компания должна платить фиксированную стоимость открытия склада.

Запас краски на каждом из складов составляет:

Склад 1 – 550 Склад 2 – 1100 Склад 3 – 1060.

Стоимость открытия каждого из складов составляет:

Склад 1 – 500 Склад 2 – 500 Склад 3 – 500

Обратите внимание, что доступное предложение теперь превышает общий спрос. Следующая модель AMPL настроена для минимизации общей стоимости при условии удовлетворения спроса.

Мы используем обозначения множества, введенные в разделе 4, для пояснения модели и данных. Это особенно полезно, если у клиентов и складов есть имена или названия местоположений.

Пример модели для задачи размещения склада e6.mod:
```
set Warehouses;
set Customers;
#transportation cost from warehouse i
#to customer j
param cost{i in Warehouses, j in Customers};
param supply{i in Warehouses}; #supply capacity at warehouse i
param demand{j in Customers}; #demand at customer j
param fixed_charge{i in Warehouses}; #cost of opening warehouse j
var amount{i in Warehouses, j in Customers};
```

```
var open{i in Warehouses} binary; # = 1 if warehouse i is opened, 0 otherwise
minimize Cost:
sum{i in Warehouses, j in Customers} cost[i,j]*amount[i,j]
+ sum{i in Warehouses} fixed_charge[i]*open[i];
subject to Supply {i in Warehouses}:
sum{j in Customers} amount[i,j] <= supply[i]*open[i];
subject to Demand {j in Customers}:
sum{i in Warehouses} amount[i,j] = demand[j];
subject to positive{i in Warehouses, j in Customers}:
amount[i,j]>=0;
```

Обратите внимание на следующие изменения в этой модели:
Мы ввели переменную open, чтобы указать, открыт ли склад, и определили ее как binary (0,1).
Параметр fixed_charge сохраняет стоимость открытия склада.
Общая стоимость теперь включает как стоимость доставки, так и фиксированную плату за открытие склада.
Ограничение Supply изменено, чтобы склад поставлял краску, только если он открыт.
Для завершения модели нам необходимо указать соответствующие данные. Файл данных:
Файл e6.dat:

```
set Warehouses:= Oakland San_Jose Albany;
set Customers:= Home_Depot K_mart Wal_mart Ace;
param cost: Home_Depot K_mart Wal_mart Ace:=
Oakland 1 2 1 3
San_Jose 3 5 1 4
Albany 2 2 2 2;
param supply:= Oakland 550
San_Jose 1100
Albany 1060;
param demand:= Home_Depot 300
K_mart 320
Wal_mart 800
Ace 390;
param fixed_charge:= Oakland 500
San_Jose 500
Albany 500;
```

Файл e6.run:

```
reset;

model e6.mod;
data e6.dat;
option solver cplex;
solve;
display Cost;
display amount;
display open;
```

Результат:

```
ampl: include 'd:\Lab1\e6.run';
CPLEX 12.6.3.0: optimal integer solution; objective 3820
13 MIP simplex iterations
0 branch-and-bound nodes
Cost = 3820

amount :=
Albany   Ace          390
Albany   Home_Depot   300
Albany   K_mart       320
Albany   Wal_mart       0
```

```
Oakland  Ace            0
Oakland  Home_Depot     0
Oakland  K_mart         0
Oakland  Wal_mart       0
San_Jose Ace            0
San_Jose Home_Depot     0
San_Jose K_mart         0
San_Jose Wal_mart     800
;

open [*] :=
  Albany  1
 Oakland  0
San_Jose  1
;
```