Экономико-математическое моделирование с использованием языка AMPL

Е.А. Нурминский

1 октября 2001 г.

Аннотация

Учебное введение в систему AMPL описания моделей линейного программирования.

Содержание

1 Язык моделирования AMPL

В настоящее время для подготовки сложных моделей линейного программирования используют специальные высокоразвитые языки моделирования: GAMS, MGG, MaGen, DATAFORM, PAM, MIMI/LP, LINGO и др. Значительную популярность приобрела и система AMPL, привлекательной особенностью которой является наличие свободно распространяемой "студенческой" версии.

Эта система использует декларативно-алгебраический стиль представления моделей линейного программирования, близкий к традиционной математической нотации. Вместе с тем AMPL дает возможность описать и сложные модели с множеством различных типов логических условий, сложных систем индексации переменных и ограничений. AMPL позволяет использовать содержательную мнемонику переменных и ограничений, предоставляет средства генерации отчетов, совместима с промышленными программами решения задач линейного программирования, имеет возможность вывода подготовленной модели в других общепринятых форматах. Большой значение имеют встроенные в AMPL средства проверки корректности подготовленных моделей, что позволяет избежать чувствительных ошибок моделирования на самых ранних этапах.

Описание всех возможностей AMPL находится за пределами возможностей этого текста, для этого существует книга [1]. Здесь будут приведены лишь базовые сведения о системе, которые в первую очередь могут понадобится для описания моделей и выполнения практических и лабораторных работ этого курса. Такое представление удобней всего дать на конкретном примере модели линейного программирования и в качестве таковой выбрана наверное простейшая из них — транспортная, которая уже была рассмотрена выше.

Транспортную модель прежде всего характеризуют множества производителей, которое мы будем обозначать через S и потребителей, которое будет обозначаться D. Обьявить о существовании этих множеств в AMPL-версии транспортной задачи можно с помощью двух деклараций

```
set S; # производители
set D; # потребители
```

Знак #, как можно догадаться, начинает комментарий, продолжающийся до конца строки. В нашем примере обозначения S и D не слишком мнемоничны, поэтому для целей последующей подготовки отчетов и пр. их не мешает снабдить и более походящими синонимами. Скажем, если речь идет о доставках угля с шахт на электростанции, то множества S и D можно задекларировать как

Таблица 1: Теоретико-множественные операции AMPL.

AMPL-запись	Традиционное математическое обозначение
`A union B`	$A \cup B = \{c: c \in A$ или $c \in B\}$
`A intsct B`	$A \cap B = \{c: c \in A$ и $c \in B\}$
`A diff B`	$A \backslash B = \{c: c \in A$ и $c \notin B\}$

```
set S 'mines'; # производители
set D 'power plants'; # потребители
```

Из S и D можно организовать новое множество — множество возможных маршрутов перевозок производитель $\rightarrow$ потребитель. Это множество (обозначим его R) с формальной точки зрения представляет набор пар (s, d),где $s \in S$ и $d \in D$, т.е. является декартовым произведением $S \times D$. AMPL-декларация имитирует математическую:

```
set R := S cross D; # маршруты перевозок
```

Помимо декартова произведения AMPL поддерживает и другие теоретико-множественные операции, приведеные в табл. 1

Для такой простейшей модели множеств S, D и R достаточно и можно перейти к следующему элементу описания задачи — числовым данным, которые на жаргоне AMPL называютя параметрами (`param`). Для транспортной задачи таковыми являются обьемы спроса и производства, а также стоимости (тарифы) перевозок единицы продукта, скажем тонны угля, от производителей к потребителям.

Соответствующие данные декларируются как

```
param supply{S} >= 0;
param demand{D} >= 0;
param cost{R} >= 0;
```

Выражения в фигурных скобках представляют собой так называемые *индексные выражения* и AMPL предоставлят множество вариантов их написания. В частности `cost` можно было бы определить как

```
param cost{(s,d) s in S d in D} >= 0;
```

Неравенства (с очевидным смыслом) в определениях `supply, demand, cost` служат целям контроля правильности подготовки данных. В этом случае ограничения по сути дела тривиальны, но в принципе они могут быть ужесточены с целью более тщательной проверки данных на корректность:

```
param zero := sum {s in S} supply[s] -
   sum { d in D } demand[d] = 0;
```

которое сразу проверяет совместимость данных по производству и потреблению.

Целью решения задачи линейного программирования является поиск оптимальных значений некоторых "свободных" величин, называемых переменными. В транспортной задаче — это обьемы перевозок от производителей к потребителям. В синтаксе AMPL это будет указано как

```
var shipmnts {R} >= 0;
```

что обьявляет набор переменных `shipmnts` , проиндексированых маршрутами (каждому маршруту соответствует своя переменная — обьем перевозки по этому маршруту) из множества R.

В нашем случае R — это просто декартово произведение $S \times D$, что означает возможность любому производителю отправить груз любому поребителю. В реальности R может быть собственным подмножеством $S \times D$, что будет отражать, например, специфику сложившихся связей. Поэтому и разумно отличать R от $S \times D$ и ввести для множества возможных маршрутов свое обозначение.

Переменные в линейном программировании должны удовлетворять ограничениям — линейным функциям от переменных. В транспортной задаче — это балансы по производителям

```
sbalance 'supply balance' { s in S }:
  sum { d in D } shipmnts [ s, d ] = supply [s];
```

и по потребителям:

```
dbalance 'demand balance' { d in D }:
  sum { s in S } shipmnts [ s, d ] = demand [d];
```

В этих определениях `sbalance, dbalance` — символические имена ограничений, `'supply balance'`, `'demand balance'` — их синонимы, `sum` — операции суммирования по последующим индексныи выражениям значений переменных `shipmnts`.

После ограничений можно описать и целевую функцию:

```
minimize tcost : sum { (s,d) in R }
  shipmnts [ s,d ] * cost [ s, d ];
```

что в общем-то завершает структурное описание задачи.

Заметим, что нигде собственно говоря не были указаны конкретные числовые значения стоимостных коеффициентов, обьемы производства–потребления, конкретные списки производителей и потребителей. В общем-то это отделение структурного описания задачи от конкретных числовых данных и составляет одну из отличительных особенностей AMPL. В этом смысле он (она ?) напоминает традиционную практику языков программирования — отделять программы от данных.

Однако в конце-концов программы вссе же нуждаются в данных и AMPL предоставляет для этого соответствующие средства. Для этого в AMPL-описании модели существует специальный (последний) раздел, начинающийся командой `data`, в котором задаются конкретные числовые и строковые данные — множества и параметры.

Простые множества можно определить непосредственно перечислением элементов:

```
set S := Лучегорск Артем Райчихинск;
set D := ПримЭС АртемЭС ВладЭС;
```

Множество маршрутов R определится при этом соответствующей операцией.

Данные о проиводстве–потреблении представляют собой одномерные массивы и могут быть заданы списками пар *индекс–значение*:

```
param supply :=
     Лучегорск         1500
     Артем              200
     Райчихинск        2000;
```

Такая табличная форма для AMPL ничего не значит и с тем же успехом `supply` можно было бы задать и так:

```
param supply := Лучегорск 1500 Артем 200 Райчихинск 2000;
```

Массив `supply` можно было бы определить и по-элементно:

```
supply[ Лучегорск ]  = 1500;
supply[ Артем ] = 200;
        ...
```

Аналогичным образом можно определить и потребление `demand`, чего мы делать не будем.

Оставшиеся числовые данные представляют собой двумерный массив стоимостей `cost`. При небольшом количестве производителей–потребителей его можно задать непосредственно таблицей:

```
param cost :
                    ПримЭС  АртемЭС  ВладЭС :=
     Лучегорск      300.    400.     500.
     Артем          200.    100.     150.
     Райчихинск     600.    800.     850;
```

Если таблица слишком велика, можно использовать *срезки* массива `cost`:

```
param cost [ Лучегорск , * ] :=
   ПримЭС 300.
   АртемЭС 400.
   ВладЭС 500.;
```

```
param cost [ Артем , * ] :=
   ПримЭС 200.
   АртемЭС 100.
   ВладЭС 150.;
param cost [ Райчихинск , * ] :=
   ПримЭС 600.
   АртемЭС 800.
   ВладЭС 850.;
```

Срезки многомерных массивов можно делать по любой комбинации индексов, так что с таким же успехом можно определить и `cost [*, ПримЭС ]` , `cost [*, АртемЭС ]` , `cost [*, ВладЭС ]` .

Сводя все элементы описания задачи вместе и добавляя некоторые комментарии получаем итоговое представление модели

```
# Опции AMPL

      option auxfiles rc;

# Декларации

      set S 'mines';            # производители
      set D 'power plants';     # потребители
      set R := S cross D;       # маршруты перевозок
      param supply{S} >= 0;
      param demand{D} >= 0;
      param cost{R} >= 0;
      param zero := sum {s in S} supply[s] - sum { d in D } demand[d] = 0;
      var shipmnts {R} >= 0;

# Ограничения ( балансы ) и целевая функция ( расходы )

      sbalance 'supply balance' { s in S }:
        sum { d in D } shipmnts [ s, d ] = supply [s];
      dbalance 'demand balance' { d in D }:
        sum { s in S } shipmnts [ s, d ] = demand [d];
      minimize tcost : sum { (s,d) in R }
        shipmnts [ s,d ] * cost [ s, d ];

# Данные

      data;
      set S := Лучегорск Артем Райчихинск;
      set D := ПримЭС АртемЭС ВладЭС;
      param supply :=
           Лучегорск         1500.
           Артем              200.
           Райчихинск        2000.;
      param demand :=
           ПримЭС            2200.
           АртемЭС            300.
           ВладЭС            1200. ;
      param cost :
                          ПримЭС АртемЭС ВладЭС :=
           Лучегорск      300.   400.    500.
           Артем          200.   100.    150.
           Райчихинск     600.   800.    850;
```

Записав это определение в файл `model.ampl` можно затем вызвать AMPL-компилятор командой

```
  ampl model.ampl
```

Таблица 2: Опции печати AMPL.

Опция вывода	Результат
`-o0`	Печать отсутствует
`-o!`	Печать отсутствует и не выводится служебный файл `genmod`. Опция несовместима с AMPL-командой `write`
`-obstub`	Бинарный формат AMPL, однако первые 10 строк текстовые.
`-ogstub`	Текстовый формат AMPL: без MPS-вывода, нет файла спецификаций, в остальном совпадает с опцией `-omstub`
`-om`	MPS-файл выводится по каналу стандартного вывода. Вся остальная печать отсутствует.
`-omstub`	MPS-файл выводится плд именем `stub.mps`, имена строк–столбцов в файлы `stub.row` и `stub.col`, фиксированые переменные в `stub.fix`, неиспользуемые переменные в `stub.unv` излишние ограничения в `stub.slc` коррекция целевой функции в `stub.adj`, файл срецификации программы MINOS в `stub.spc`.
`-onstub`	MPS-файл выводится по каналу стандартного вывода, в остальном совпадает с `-omstub`

на что последует сообщение

```
No option outopt (-o flag) given.
```

Это сообщение предупреждает о том, что мы не определили формат и тип результатов, которые хотелось бы получить от AMPL-процессора.

Возможные варианты выдачи результатов приведены в табл. 1. При использовании опции `-om` результирующий MPS файл имеет вид:

```
NAME          stub              |    C0005     R0005     1
ROWS                            |    C0005     R0007     100
 E  R0001                       |    C0006     R0002     1
 E  R0002                       |    C0006     R0006     1
 E  R0003                       |    C0006     R0007     150
 E  R0004                       |    C0007     R0003     1
 E  R0005                       |    C0007     R0004     1
 E  R0006                       |    C0007     R0007     600
 N  R0007                       |    C0008     R0003     1
COLUMNS                         |    C0008     R0005     1
    C0001     R0001     1       |    C0008     R0007     800
    C0001     R0004     1       |    C0009     R0003     1
    C0001     R0007     300     |    C0009     R0006     1
    C0002     R0001     1       |    C0009     R0007     850
    C0002     R0005     1       |RHS
    C0002     R0007     400     |    B         R0001     1500
    C0003     R0001     1       |    B         R0002     200
    C0003     R0006     1       |    B         R0003     2000
    C0003     R0007     500     |    B         R0004     2200
    C0004     R0002     1       |    B         R0005     300
    C0004     R0004     1       |    B         R0006     1200
    C0004     R0007     200     |ENDATA
    C0005     R0002     1       |
```

Для более компактного вывода MPS-файл "сложен" в две колонки с обозначеной символом | линией раздела.

Заметим, что строки и столбцы в MPS-файле носят условные имена. Их сответствие с реальными устанавливается в дополнительных файлах `stub.row` и `stub.col`. В первом из них находятся "настоящие имена" ограничений в том порядке, как они идут в секции `ROWS` MPS-файла задачи:

```
sbalance['Лучегорск']
sbalance['Артем']
sbalance['Райчихинск']
dbalance['ПримЭС']
dbalance['АртемЭС']
dbalance['ВладЭС']
tcost
```

Во втором — "настоящие имена" переменных

```
shipmnts['Лучегорск','ПримЭС']
shipmnts['Лучегорск','АртемЭС']
shipmnts['Лучегорск','ВладЭС']
shipmnts['Артем','ПримЭС']
shipmnts['Артем','АртемЭС']
shipmnts['Артем','ВладЭС']
shipmnts['Райчихинск','ПримЭС']
shipmnts['Райчихинск','АртемЭС']
shipmnts['Райчихинск','ВладЭС']
```

в том порядке, как они идут в секции `COLUMNS` MPS-файла задачи. Для того, чтобы эти вспомогательные файлы создались, потребовалось включить в AMPL-файл соответствующую команду `option auxfiles rc`.

2 Контрольные вопросы

1. Заданы множества

   ```
   set A := А Б В Г и Д;
   set B := С и Е;
   ```

   Эти множества

   (a) имеют пустое пересечение ?

   (b) имеют непустое пересечение ?

2. Как в AMPL определить число элементов в множестве ?

3. Пусть множества A и B определены в вопросе 2. Сколько элементов в множествах

   (a) `set C := B diff A;`

   (b) `set C := A cross B;`

   (c) `set C := A union B;`

4. Чему будет равен массив `val` после выполнения следующих команд:

   ```
   set ind := 1 ... 15;
   param val {i in ind } := if )i > 7) then i - (i/2)*2;
      else i+1;
   ```

5. Чему будет равена матрица `A` после выполнения следующих команд:

   ```
   param A {1 ... 5, 1 ... 5 };
   ```

   (a) `param A {i in 1 ... 5, j in i ... 5 } := 2 default 1;`

   (b) `param A {i in 1 ... 4, j in 1 ... i+1 } := 1;`

Таблица 3: Типовые данные для задача КОРМА

	силос	кукуруза	зерно	свекла	площ	тр. расходы
Грязино	300	400	500	600	300	14.4
Хлябино	300	400	500	600	300	14.4
Болотино	300	400	500	600	300	14.4
Гнилушки	300	400	500	600	300	14.4
Корм.единиц	20	30	50	30		

6. Написать на AMPL вычисление значения полинома

$$p(x) = a_0 + a_1 t + a_2 t^2 + \ldots + a_m t^m$$

m-ой степени в заданой точке t как линейной функции его коеффициентов.

7. Используя результаты предыдущей задачи описать процедуру определения коеффициентов интерполяционного полинома Лежандра [1] как задачу линейного программирования.

3 Самостоятельная работа

1. Ознакомиться с содержанием сайта `www.ampl.org`, скопировать систему AMPL под свою платформу и документацию к ней.
2. Инсталировать AMPL и AMPL-совместимую программу решения задач линейного программирования.
3. Подготовить с помощью AMPL и решить тестовые задачи.
4. Подготовить с помощью AMPL и решить индивидуальную задачу.
5. Подготовить отчет.

4 Задачи для индивидуального выполнения

Корма В кормовых отделения Грязино, Хлябино, Болотино и Гнилушки фермерского хозяйства "Золотой телец" готовят силос, кукурузу, кормовое зерно, свеклу для скармливания скоту, находящемуся в центральном откормочном отделении фермы Зимовье. Каждое из отделений располагает своими посевными площадями, различающимися по типам почв и, следовательно, имеющими по данным многолетней статистики различную урожайность для разных культур. Пищевая ценность культур определяется в условных кормовых единицах (у.к.е) на единицу веса кормов и известны потребности единицы поголовья скота в корме в терминах у.к.е.

Хозяйства расположены на различном расстоянии от центральной усадьбы и стоимость доставки единицы веса кормов в усадьбу существенно влияет на экономику хозяйства.

Учитывая то, что по прогнозам рыночных аналитиков средняя сдаточная стоимость одной единицы скота будет составлять 2440 руб, а для выращивания этой единицы требуется 180 у.к.е., определить оптимальный план производства кормов и выращивания скота. Все данные приведены в табл. 4

Лес Лесное хозяйство имеет некоторое число участков леса, предназначеного для рубки. С i-го участка леса в год t добывается $c_i = A_i + B_i t$ кубометров леса. Лес вывозится на склады, расположенные на железнодорожных станциях, емкость складов ограничена. В конце каждого года весь лес вывозится. Транспортные расходы на доставку 1 кубометра древесины на склады известны. Все необходимые данные приведены в типовой таблице 4.

Необходимо: зная предполагаемую динамику цен на древесину $p_1, p_2, \ldots, p_T$ составить на период T лет планы рубки и вывоза древесины по участкам с учетом ограниченной пропускной способности складов.

[1] см. курс математического ананлиза

Таблица 4: Типовые данные для задача ЛЕС

Участок	Площадь	A_i	B_i	Склад А	Склад Б
А	200	2000	100	20.80	30.90
А	200	2000	100	20.80	30.90
А	200	2000	100	20.80	30.90
А	200	2000	100	20.80	30.90
А	200	2000	100	20.80	30.90
А	200	2000	100	20.80	30.90
Емкость складов				800.4	630.8

Таблица 5: Прогноз изменений максимальной дневной температуры.

Изменение t	Вероятность
+3.0	0.15
0.0	0.75
-2.0	0.10

Перспективное и оперативное планирование Компания *Сладкая жизнь* производит и продает мороженое, дневной спрос на которое зависит от погоды и по исследованиям аналитического отдела компании описывается формулой

$$D = A + Bt$$

где t — максимальная дневная температура (^{o} C).

В начале периода планирования $t = 24^o$ и на ближайшие дни синоптики прогнозируют достаточно устойчивую погоду с вероятностями изменения температуры от одного дня к следующему, приведеными в табл. 4. Компания составляет перспективный план производства на 3 дня вперед и имеет возможность оперативно корректировать его в зависимости от реализовавшегося спроса. При выпуске 1 т. мороженого по перспективному плану производственнные издержки составляют C_1 у.е. Оперативная коррекция требует дополнительных затрат и выпуск 1 т. мороженого в режиме оперативной коррекции обходится в $C_2 > C_1$ у.е. Перспективный план и коррекции выпускаются на одном и том же оборудовании ограниченной производительности K т/день.

Компания имеет склады большой емкости, на которых можно создавать запасы мороженного и эти запасы можно использовать для покрытия дневного спроса. Стоимость хранения 1 т. мороженого на складе — C_3 у.е. В конце планового периода (3 дня) невостребованый запас отправляется на переработку с затратами C_4.

Необходимо: определить перспективный план производства и варианты коррекции, которые минимизируют *средние* (в смысле математического ожидания) расходы компании.

Список литературы

[1] Kernighan B., AMPL O'Reiley