Применение ЛП

Моделирование: основа применения ЛП

Виктор Васильевич Лепин

Цель занятия

Цель — научить студентов пользоваться комплексом программных инструментов и технологий, позволяющих в полной мере реализовать разработку и применение оптимизационных моделей.

AMPL

AMPL (A Mathematical Programming Language) — это язык высокого уровня для описания задач математического программирования, использующий декларативно-алгебраический стиль представления моделей математического программирования, близкий к традиционной математической нотации.

Сайт: https://ampl.com/

- Загрузите demo версию
  https://ampl.com/try-ampl/download-a-free-demo/
- Загрузите примеры
  https://ampl.com/resources/the-ampl-book/example-files/

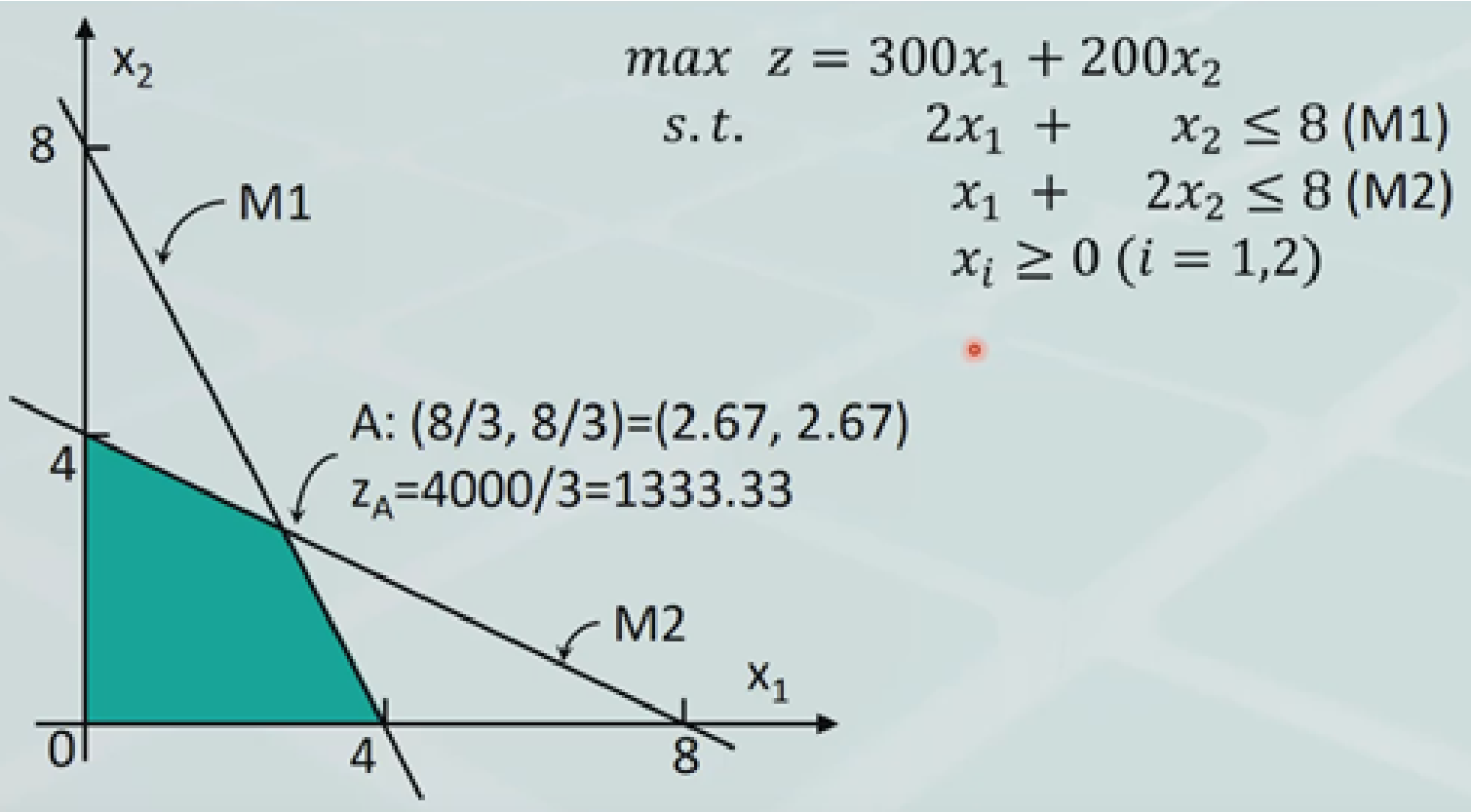
Как решить задачу ЛП

- Можно графичиским методом.

Как решить задачу ЛП

- Можно графичиским методом.
- Можно используя комплекс программных инструментов и технологий, например AMPL.

Графическим методом



AMPL

- Преимущество использования языка алгебраического моделирования AMPL (или его свободно распространяемого аналога MathProg) в том, что для решения задач оптимизации и анализа результатов не требуется программирование.

AMPL

- Преимущество использования языка алгебраического моделирования AMPL (или его свободно распространяемого аналога MathProg) в том, что для решения задач оптимизации и анализа результатов не требуется программирование.
- Языки алгебраического моделирования являются декларативными, то есть, требуют только описания ключевых зависимостей и ограничений в модели, а не алгоритмов ее решения.

AMPL

- Преимущество использования языка алгебраического моделирования AMPL (или его свободно распространяемого аналога MathProg) в том, что для решения задач оптимизации и анализа результатов не требуется программирование.
- Языки алгебраического моделирования являются декларативными, то есть, требуют только описания ключевых зависимостей и ограничений в модели, а не алгоритмов ее решения.
- Модель автоматически преобразуется в стандартный формат представления задач математического программирования, который затем передается решающему алгоритму.

AMPL

- Преимущество использования языка алгебраического моделирования AMPL (или его свободно распространяемого аналога MathProg) в том, что для решения задач оптимизации и анализа результатов не требуется программирование.
- Языки алгебраического моделирования являются декларативными, то есть, требуют только описания ключевых зависимостей и ограничений в модели, а не алгоритмов ее решения.
- Модель автоматически преобразуется в стандартный формат представления задач математического программирования, который затем передается решающему алгоритму.
- После решения задачи все результаты становятся доступными для вывода и анализа средствами языка моделирования.

AMPL

- Все это происходит совершенно прозрачно для пользователя.

- Все это происходит совершенно прозрачно для пользователя.
- Форма записи модели на языке моделирования очень близка к традиционной математической записи в общем виде (т.е. с использованием операторов суммирования, кванторов, индексов и множеств), с которой студенты хорошо знакомы.

AMPL

- Все это происходит совершенно прозрачно для пользователя.
- Форма записи модели на языке моделирования очень близка к традиционной математической записи в общем виде (т.е. с использованием операторов суммирования, кванторов, индексов и множеств), с которой студенты хорошо знакомы.
- AMPL наряду с GAMS и AIMMS является лидирующим инструментом моделирования, который широко применяется во всем мире для быстрой разработки математических моделей.

AMPL

- Все это происходит совершенно прозрачно для пользователя.
- Форма записи модели на языке моделирования очень близка к традиционной математической записи в общем виде (т.е. с использованием операторов суммирования, кванторов, индексов и множеств), с которой студенты хорошо знакомы.
- AMPL наряду с GAMS и AIMMS является лидирующим инструментом моделирования, который широко применяется во всем мире для быстрой разработки математических моделей.
- Для решения моделей могут использоваться коммерческие и свободно распространяемые решатели задач смешанного целочисленного линейного программирования — CPLEX, Gurobi, Xpress-MP, LP_SOLVE, GLPSOL, SCIP;

AMPL

- Все это происходит совершенно прозрачно для пользователя.
- Форма записи модели на языке моделирования очень близка к традиционной математической записи в общем виде (т.е. с использованием операторов суммирования, кванторов, индексов и множеств), с которой студенты хорошо знакомы.
- AMPL наряду с GAMS и AIMMS является лидирующим инструментом моделирования, который широко применяется во всем мире для быстрой разработки математических моделей.
- Для решения моделей могут использоваться коммерческие и свободно распространяемые решатели задач смешанного целочисленного линейного программирования — CPLEX, Gurobi, Xpress-MP, LP_SOLVE, GLPSOL, SCIP;
- нелинейного программирования — Knitro, Minos, Conopt, IP-Opt, SNOPT, BARON;

- программирования в ограничениях – CP Optimizer;

AMPL

- программирования в ограничениях – CP Optimizer;
- локального поиска — LocalSolver.

AMPL

- программирования в ограничениях – CP Optimizer;
- локального поиска — LocalSolver.
- Также доступны ресурсы вычислительного сервера NEOS.

AMPL выбран в качестве базового инструмента по следующим причинам:

- простой язык для разработки моделей и манипулирования ими;

AMPL

- программирования в ограничениях – CP Optimizer;
- локального поиска — LocalSolver.
- Также доступны ресурсы вычислительного сервера NEOS.

AMPL выбран в качестве базового инструмента по следующим причинам:

- простой язык для разработки моделей и манипулирования ими;
- простой доступ к источникам данных для моделей (электронным таблицам и базам данных);

AMPL

- программирования в ограничениях – CP Optimizer;
- локального поиска — LocalSolver.
- Также доступны ресурсы вычислительного сервера NEOS.

AMPL выбран в качестве базового инструмента по следующим причинам:

- простой язык для разработки моделей и манипулирования ими;
- простой доступ к источникам данных для моделей (электронным таблицам и базам данных);
- программное обеспечение не требует установки и доступно на всех платформах (Windows, Mac OS X, Linux);

AMPL

- существует бесплатный, свободно распространяемый, совместимый аналог - GLPK (GNU Linear Programming Kit), позволяющий решать задачи не только в учебных, но и в коммерческих целях;

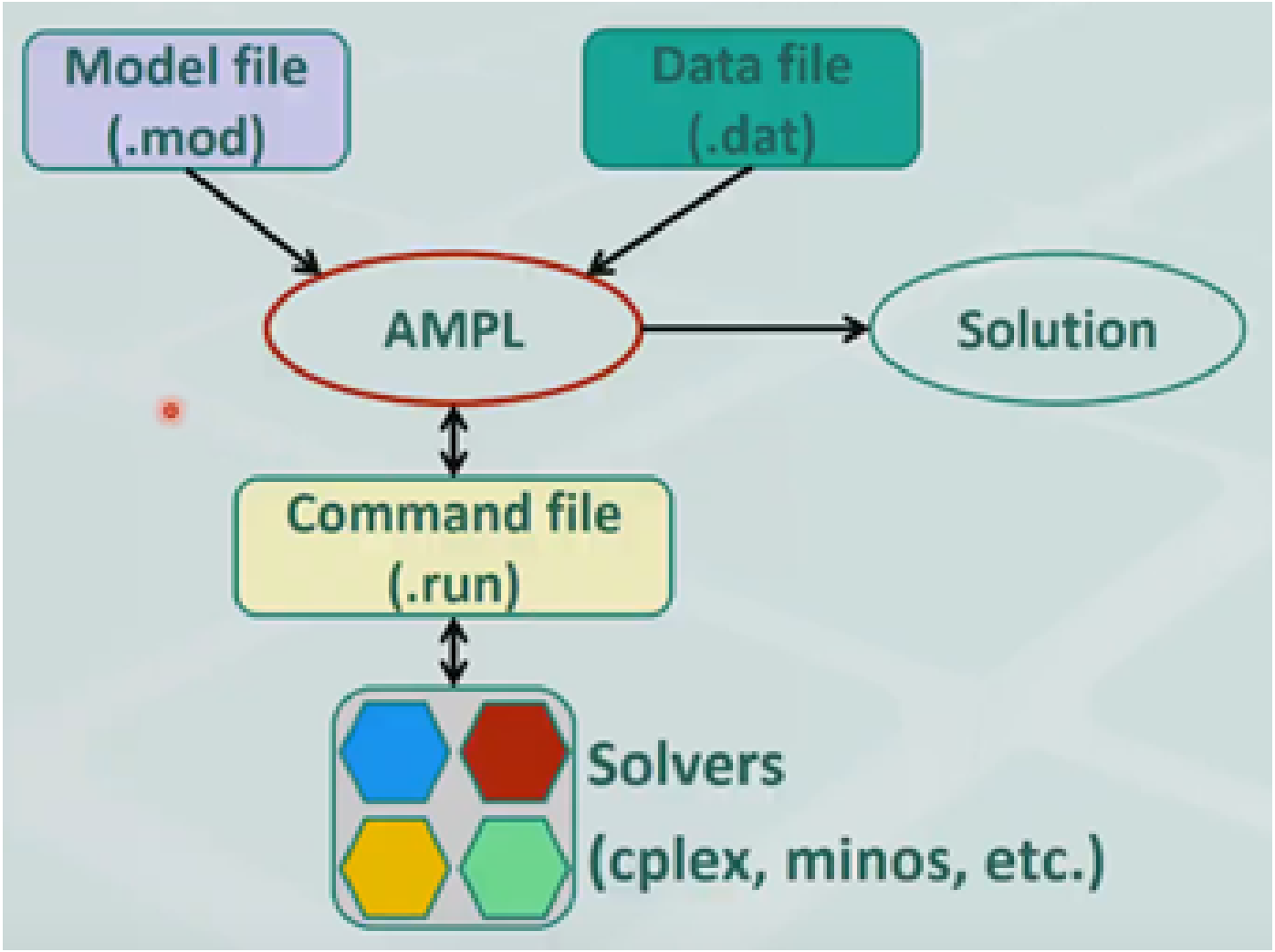
AMPL

- существует бесплатный, свободно распространяемый, совместимый аналог - GLPK (GNU Linear Programming Kit), позволяющий решать задачи не только в учебных, но и в коммерческих целях;
- наличие веб-сервисов (NEOS, MathProg Web IDE), позволяющих решать задачи непосредственно в браузере, без установки какого-либо программного обеспечения, в том числе, на мобильных платформах;

AMPL

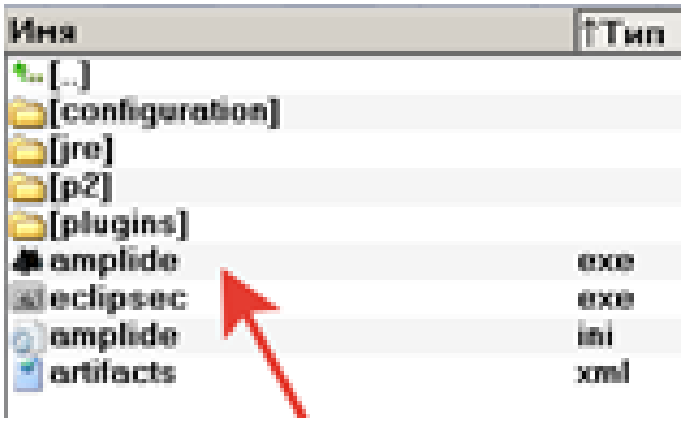
- существует бесплатный, свободно распространяемый, совместимый аналог - GLPK (GNU Linear Programming Kit), позволяющий решать задачи не только в учебных, но и в коммерческих целях;
- наличие веб-сервисов (NEOS, MathProg Web IDE), позволяющих решать задачи непосредственно в браузере, без установки какого-либо программного обеспечения, в том числе, на мобильных платформах;
- наличие превосходно написанных пособий по языку и применению моделирования в целом (книга AMPL: A Modeling Language for Mathematical Programming, доступная на сайте разработчиков).

AMPL файлы


Model file
(.mod)
Data file
(.dat)
AMPL
Solution
Command file
(.run)
Solvers
(cplex, minos, etc.)

AMPL файлы

Войти в прорамму: amplide.exe



AMPL файлы

- .mod — используется для объявления элементов модели: переменных, цели, ограничений и данных (множества и параметры).

AMPL файлы

- .mod — используется для объявления элементов модели: переменных, цели, ограничений и данных (множества и параметры).
- .dat — используется для определения данных для модели.

AMPL файлы

- .mod — используется для объявления элементов модели: переменных, цели, ограничений и данных (множества и параметры).
- .dat — используется для определения данных для модели.
- .run — где определены конфигурации переменных, скриптовые конструкции, такие как чтение таблиц или баз данных.

AMPL файлы

Синтаксис:

- Переменная: var VariableName;

AMPL файлы

Синтаксис:

- Переменная: var VariableName;
- Цель: minimize or maximize ObjectiveName: ...;

AMPL файлы

Синтаксис:

- Переменная: var VariableName;
- Цель: minimize or maximize ObjectiveName: ...;
- Ограничение: subject to RestrictionName: . . . ;

Замечание:

- Каждая строка инструкции должна заканчиваться ";".

AMPL файлы

Синтаксис:

- Переменная: var VariableName;
- Цель: minimize or maximize ObjectiveName: ...;
- Ограничение: subject to RestrictionName: . . . ;

Замечание:

- Каждая строка инструкции должна заканчиваться ";".
- Строка комментария должна начинаться символом #.

AMPL файлы

Синтаксис:

- Переменная: var VariableName;
- Цель: minimize or maximize ObjectiveName: ...;
- Ограничение: subject to RestrictionName: . . . ;

Замечание:

- Каждая строка инструкции должна заканчиваться ";".
- Строка комментария должна начинаться символом #.
- AMPL чувствителен к регистру.

AMPL файлы

Синтаксис:

- Переменная: var VariableName;
- Цель: minimize or maximize ObjectiveName: ...;
- Ограничение: subject to RestrictionName: . . . ;

Замечание:

- Каждая строка инструкции должна заканчиваться ";".
- Строка комментария должна начинаться символом #.
- AMPL чувствителен к регистру.
- Имена переменных должны быть уникальными.

Пример

- Paint Deals производит краски двух цветов: синюю и черную.

Пример

- 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.

Пример

- 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.

Пример

- 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.

Пример

- 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.

Пример

- 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.

Пример

- 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, чтобы максимизировать недельный доход.

Пример

Формальная модель:

$$\max \quad 10 \cdot BluePaint + 15 \cdot BlackPaint \tag{1}$$

$$\text{s.t.} : (\frac{1}{40}) \cdot BluePaint + (\frac{1}{30}) \cdot BlackPaint \leq 40 \tag{2}$$

$$0 \leq BluePaint \leq 1000 \tag{3}$$

$$0 \leq BlackPaint \leq 860 \tag{4}$$

Пример

Файл модели .mod

```
# Part 1: Variable Declaration (var, set, param, etc)
var BluePaint;
var BlackPaint;
# Part 2: Objective Function
maximize Revenue: 10*BluePaint + 15*BlackPaint;
# Part 3: Constraints
subject to Time: (1/40)*BluePaint + (1/30)*BlackPaint
<= 40;
subject to BlueLimit: 0 <= BluePaint <= 1000;
subject to BlackLimit: 0 <= BlackPaint <= 860;
```

Пример

Файл .run

```
# Reset Memory
reset ;
# Load Model
model example1.mod;
# Change Configuration (optional)
option solver cplex;
# Solve Problem
solve;
# Show Results
display BluePaint, BlackPaint;
display Revenue;
expand Time;
```

Пример

Результат решения

```
ampl: include example1.run;
CPLEX 12.2.0.0: No LP presolve or aggregator reductions.
optimal solution; objective 17433.33333
1 dual simplex iterations (0 in phase I)
BluePaint = 453.333
BlackPaint = 860

Revenue = 17433.3

subject to Time:
        0.025*BluePaint + 0.0333333*BlackPaint <= 40;
ampl:
```

Работа с AMPL

- Модели записываются в виде текстового файла <имя файла>.mod.

Работа с AMPL

- Модели записываются в виде текстового файла <имя файла>.mod.
- При написании модели на языке AMPL можно использовать любой текстовый редактор.

Работа с AMPL

- Модели записываются в виде текстового файла <имя файла>.mod.
- При написании модели на языке AMPL можно использовать любой текстовый редактор.
- При написании моделей используются основные команды AMPL.

Работа с AMPL

- Модели записываются в виде текстового файла <имя файла>.mod.
- При написании модели на языке AMPL можно использовать любой текстовый редактор.
- При написании моделей используются основные команды AMPL.
- AMPL модель содержит описания объектов модели, т.е. множеств, переменных, параметров, целевой функции и ограничений.

Работа с AMPL

- Модели записываются в виде текстового файла <имя файла>.mod.
- При написании модели на языке AMPL можно использовать любой текстовый редактор.
- При написании моделей используются основные команды AMPL.
- AMPL модель содержит описания объектов модели, т.е. множеств, переменных, параметров, целевой функции и ограничений.
- Для описания объектов используются служебные слова set,var,param,minimize/maximize,subjectto (или кратко s.t.).

Работа с AMPL

AMPL-модель содержит несколько типов элементов, подробнее описываемых ниже: декларации с ключевыми словами:

- set(множество индексов),
- param(параметр),
- var(переменная),
- arc(дуга — для описания сетевых моделей);
- целевых функций вида maximize minimize;
- ограничений subject to (при ограничениях),
- node (вершина — для описания сетевых моделей).

Работа с AMPL

- Во многом синтаксис команд AMPL очень подобен C.

Работа с AMPL

- Во многом синтаксис команд AMPL очень подобен C.
- AMPL поддерживает такие функции, как abs(), cos(), sin(), log(), sqrt(), exp() с использованием основных операций +, -, *, /, ^или **.

Работа с AMPL

- Во многом синтаксис команд AMPL очень подобен C.
- AMPL поддерживает такие функции, как abs(), cos(), sin(), log(), sqrt(), exp() с использованием основных операций +, -, *, /, ^или **.
- Все команды оканчиваются точкой с запятой «;».

Работа с AMPL

- Во многом синтаксис команд AMPL очень подобен C.
- AMPL поддерживает такие функции, как abs(), cos(), sin(), log(), sqrt(), exp() с использованием основных операций +, -, *, /, ^или **.
- Все команды оканчиваются точкой с запятой «;».
- К командам вывода относятся display, а также команды write и print.

Работа с AMPL

- Во многом синтаксис команд AMPL очень подобен C.
- AMPL поддерживает такие функции, как abs(), cos(), sin(), log(), sqrt(), exp() с использованием основных операций +, -, *, /, ^или **.
- Все команды оканчиваются точкой с запятой «;».
- К командам вывода относятся display, а также команды write и print.
- Имена (идентификаторы) состоят из латинских букв (прописных и строчных), цифр и знаков подчеркивания.

Работа с AMPL

- Во многом синтаксис команд AMPL очень подобен C.
- AMPL поддерживает такие функции, как abs(), cos(), sin(), log(), sqrt(), exp() с использованием основных операций +, -, *, /, ^или **.
- Все команды оканчиваются точкой с запятой «;».
- К командам вывода относятся display, а также команды write и print.
- Имена (идентификаторы) состоят из латинских букв (прописных и строчных), цифр и знаков подчеркивания.
- Символ # означает начало комментария. Все, что находится за этим символом, игнорируется AMPL.

Работа с AMPL

- Во многом синтаксис команд AMPL очень подобен C.
- AMPL поддерживает такие функции, как abs(), cos(), sin(), log(), sqrt(), exp() с использованием основных операций +, -, *, /, ^или **.
- Все команды оканчиваются точкой с запятой «;».
- К командам вывода относятся display, а также команды write и print.
- Имена (идентификаторы) состоят из латинских букв (прописных и строчных), цифр и знаков подчеркивания.
- Символ # означает начало комментария. Все, что находится за этим символом, игнорируется AMPL.
- Комментарии могут быть также ограничены символами /* и */, причем они могут отделены друг от друга несколькими строками.

Работа с AMPL

Команды AMPL используют простой синтаксис:

- Переменные описываются с использованием служебного слова var.

Работа с AMPL

Команды AMPL используют простой синтаксис:

- Переменные описываются с использованием служебного слова var.
- Параметры описываются с использованием служебного слова param.

Работа с AMPL

Команды AMPL используют простой синтаксис:

- Переменные описываются с использованием служебного слова var.
- Параметры описываются с использованием служебного слова param.
- Суммирование $\sum_{i=1}^{n}$ записывается так: sum{i in 1..n}.

Работа с AMPL

Команды AMPL используют простой синтаксис:

- Переменные описываются с использованием служебного слова var.
- Параметры описываются с использованием служебного слова param.
- Суммирование $\sum_{i=1}^{n}$ записывается так: sum{i in 1..n}.
- Служебные слова AMPL (такие как var, param, solve, maximizeи др.), а также имена функций (например, sum,log, sin) зарезервированы и не могут использоваться для имен объектов. К служебным зарезервированным словам относятся также for, if, elseif, else, while, file, system.

Работа с AMPL

Команды AMPL используют простой синтаксис:

- Переменные описываются с использованием служебного слова var.
- Параметры описываются с использованием служебного слова param.
- Суммирование $\sum_{i=1}^{n}$ записывается так: sum{i in 1..n}.
- Служебные слова AMPL (такие как var, param, solve, maximizeи др.), а также имена функций (например, sum,log, sin) зарезервированы и не могут использоваться для имен объектов. К служебным зарезервированным словам относятся также for, if, elseif, else, while, file, system.
- Индексы переменных и ограничений заключаются в квадратные скобки (например, a[i]).

Работа с AMPL

- Числа могут записываться в разных форматах. Так, 0.0123, 1.23D-2, 1.23e-2, 1.23E-2 это эквивалентные записи одного и того же числа 0,0123.

Работа с AMPL

- Числа могут записываться в разных форматах. Так, 0.0123, 1.23D-2, 1.23e-2, 1.23E-2 это эквивалентные записи одного и того же числа 0,0123.
- Литералы — это строки, заключенные в кавычки (одинарные или двойные). Например, ‘abc’, ‘x’, ‘y’, “ABC”.

Работа с AMPL

- Числа могут записываться в разных форматах. Так, 0.0123, 1.23D-2, 1.23e-2, 1.23E-2 это эквивалентные записи одного и того же числа 0,0123.
- Литералы — это строки, заключенные в кавычки (одинарные или двойные). Например, ‘abc’, ‘x’, ‘y’, “ABC”.
- В AMPL модель и данные разделены.

Работа с AMPL

- Числа могут записываться в разных форматах. Так, 0.0123, 1.23D-2, 1.23e-2, 1.23E-2 это эквивалентные записи одного и того же числа 0,0123.
- Литералы — это строки, заключенные в кавычки (одинарные или двойные). Например, ‘abc’, ‘x’, ‘y’, “ABC”.
- В AMPL модель и данные разделены.
- Множества, описанные в модели, не имеют размеров или заданных элементов.

Работа с AMPL

- Числа могут записываться в разных форматах. Так, 0.0123, 1.23D-2, 1.23e-2, 1.23E-2 это эквивалентные записи одного и того же числа 0,0123.
- Литералы — это строки, заключенные в кавычки (одинарные или двойные). Например, ‘abc’, ‘x’, ‘y’, “ABC”.
- В AMPL модель и данные разделены.
- Множества, описанные в модели, не имеют размеров или заданных элементов.
- Использование в модели множества индексов означает лишь то, что используются элементы этого множества, причем неважно, сколько их имеется.

Работа с AMPL

- Числа могут записываться в разных форматах. Так, 0.0123, 1.23D-2, 1.23e-2, 1.23E-2 это эквивалентные записи одного и того же числа 0,0123.
- Литералы — это строки, заключенные в кавычки (одинарные или двойные). Например, ‘abc’, ‘x’, ‘y’, “ABC”.
- В AMPL модель и данные разделены.
- Множества, описанные в модели, не имеют размеров или заданных элементов.
- Использование в модели множества индексов означает лишь то, что используются элементы этого множества, причем неважно, сколько их имеется.
- Задание величин делается только в файле данных, который имеет вид <имя файла>.dat.

Работа с AMPL

- Числа могут записываться в разных форматах. Так, 0.0123, 1.23D-2, 1.23e-2, 1.23E-2 это эквивалентные записи одного и того же числа 0,0123.
- Литералы — это строки, заключенные в кавычки (одинарные или двойные). Например, ‘abc’, ‘x’, ‘y’, “ABC”.
- В AMPL модель и данные разделены.
- Множества, описанные в модели, не имеют размеров или заданных элементов.
- Использование в модели множества индексов означает лишь то, что используются элементы этого множества, причем неважно, сколько их имеется.
- Задание величин делается только в файле данных, который имеет вид <имя файла>.dat.
- Здесь элементы множеств и параметры определяются явно, повторно записывая служебное слово, имя объекта и перечисляя значения после ":=".

Пример решения задачи ЦЛП

Запишем на AMPL модель ЦЛП вида:

$$\sum_{j=1}^{n} c_j x_j \to \max$$

при ограничениях

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i \quad i = 1, \ldots, m,$$

$$x_j = 0, 1, \quad j = 1, \ldots, n.$$

Описание параметров m и n в AMPL выглядит так:

param m;

param n;

Опишем множества индексов $i = 1, \ldots, m$ и $j = 1, \ldots, n$:

set I=1..m;

set J=1..n;

Для описания параметров c_j, b_j, a_{ij} используем запись:

param c{J}; param b{I}; param a{I, J};

Для описания переменных используем запись:

var x{J} binary;

Пример решения задачи ЦЛП

Описание целевой функции $\sum_{j=1}^{n} c_j x_j \to \max$ выглядит следующим образом:
maximize z: sum j in 1..n c[j] * x[j];
Ограничения $\sum_{j=1}^{n} a_{ij} x_j \le b_i \quad i = 1, \ldots, m$, запишутся так:
coni in 1..m: sum j in 1..n a[i,j] * x[j] <= b[i];
Таким образом, модель ЦЛП записывается на AMPL в виде ip.mod:
param m;param n;set I:=1..m;set J:=1..n;param c J;param b I;param a I, J; var x J binary;maximize obj: sumj in Jc[j]*x[j];subject to res i in I: sumj in Ja[i,j]*x[j]<= b[i];
Для решения конкретной задачи ЦЛП вида
$z = 2x_1 + 3x_2 + x_3 \to \max$
при ограничениях
$x_1 + 2x_2 + x_3 \le 2;$
$2x_1 + 3x_2 + 2x_3 \le 4,$
$x_1, x_2, x_3 \in \{0, 1\}.$

Пример решения задачи ЦЛП

Можно использовать описанный выше файл модели ip.mod вместе с файлом данных ip.dat следующего вида

```
param m:=2;param n:=3;param c [1] 2 [2] 3 [3] 1;param b [1] 2 [2] 4;param a:
1 2 3:= 1 1 2 1 2 2 3 2;
```

В окне AMPL набираем операторы:

```
model ip.mod;
data ip.mod;
option solver lpsolve;
solve;
display x, obj;
```

В результате получим:

```
LP_SOLVE 4.0.1.0: optimal, objective 3
5 simplex iterations
3 branch & bound nodes: depth 2
x [*] :=
1 1
2 0
3 1
;
obj = 3
```

Задача о диете

Пример. ***Задача о смесях (о диете).***

Задана: дневная потребность в веществах (белках, жирах, углеводах, витаминах и т.д.) и содержание веществ в единице имеющихся продуктов и цена каждого продукта.

Определить наиболее дешевый рацион, который обеспечил бы дневную потребность в необходимых веществах.

Пусть в дневную норму питания должно входить не менее чем a_1 – первого вещества и т.д., имеется n видов продуктов и в единице j-го продукта содержится a_{ij} единиц i-го вещества, а также, что цена j-го продукта равна c_j;

x_j количество единиц j-го продукта, который планируется вести в рацион

Задача о диете

Все $x_j \geq 0$ и при плане $x_1,\ldots,x_n$ дневное содержание i-го вещества составит $a_{i1}x_1 + \ldots + a_{in}x_n$ единиц, оно должно удовлетворять минимальную дневную потребность

$$a_{i1}x_1 + \ldots + a_{in}x_n \geq a_i \quad i = 1,\ldots,m.$$

Целевая функция $f = c_1x_1 + \ldots + c_nx_n \to \min$

Если запас продуктов ограничен, то на переменные накладываются ограничения

$x_1 \leq b_1,\ldots,x_n \leq b_n$, где b_j - величина запаса j-го продукта.

Задача о диете

В модель можно вводить дополнительные условия, например, соотношение (пропорции) между разными видами продуктов. Задача о диете используется при составлении дешевого рациона для откорма скота.

К задаче сводится задачи о смесях, составление оптимальных в определенном смысле смесей из имеющихся веществ (задача составления смесей нефтепродуктов, которые удовлетворяют соответствующим техническим требованиям и являются наиболее дешевыми).

Задача о диете

Рассмотрим задачу о диете: мы хотим минимизировать стоимость пищи, которую мы едим, при этом должны учитывать некоторые ограничения в питании (калории, уровень сахара и жира). Список продуктов питания: 1. шоколадные конфеты, 2. шоколадное мороженое, 3. кока-кола, 4. пицца.

$$\min_x 50x_1 + 20x_2 + 30x_3 + 80x_4 \qquad (\text{each } x_i \geq 0 \text{ is weighted with the cost})$$

$$\begin{array}{rcrcrcrcll}
400x_1 & + & 200x_2 & + & 150x_3 & + & 500x_4 & \geq & 500 & \text{calories} \\
3x_1 & + & 2x_2 & & & & & \geq & 6 & \text{chocolate} \\
2x_1 & + & 2x_2 & + & 4x_3 & + & 4x_4 & \geq & 10 & \text{sugar} \\
2x_1 & + & 4x_2 & + & x_3 & + & 5x_4 & \geq & 8 & \text{fat}
\end{array}$$

Задача о диете

The file diet.mod

```
# parameters and variables
param N, integer, >0; # число продуктов
param c {1..N}, >0; # вектор стоимости каждого продукта
param a {1..N,1..N+1}; # матрица ограничений
var x {1..N}, >= 0; # переменные решения
# указываем критерий
minimize totalcost : sum{i in 1..N} c[i]*x[i];
# ограничения
subject to nutrition {i in 1..N} : sum{j in 1..N} a[i,j]*x[j] >= a[i,N+1];
```

Задача о диете

Файл diet.dat

```
param N := 4;
param : c :=
1 50
2 20
3 30
4 80
;
param : a :=
1 1 400
1 2 200
1 3 150
1 4 500
1 5 500
2 1 3
2 2 2
2 3 0
2 4 0
2 5 6
3 1 2
3 2 2
3 3 4
3 4 4
3 5 10
4 1 2
4 2 4
4 3 1
4 4 5
4 5 8

;
```

Задача о диете

Файл diet.run

```
model diet.mod;
data diet.dat;
option solver cplexamp;
solve;
display totalcost;
display x;
```

Задача о диете

Результат

```
ampl diet.run
CPLEX 11.0.0: optimal solution; objective 90
2 dual simplex iterations (0 in phase I)
totalcost = 90
x [*] :=
1 0
2 3
3 1
4 0
;
```