Introduction to CPLEX OPL

Decision-aid Methodologies in Transportation: Computer Lab 7

Iliya Markov

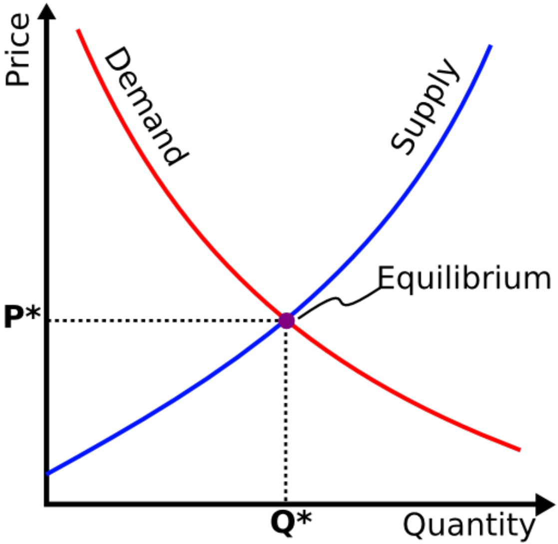
Transport and Mobility Laboratory
School of Architecture, Civil and Environmental Engineering
École Polytechnique Fédérale de Lausanne

March 31, 2015

Overview

Overview

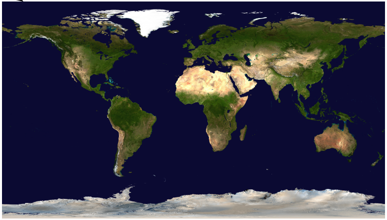
Demand – Supply



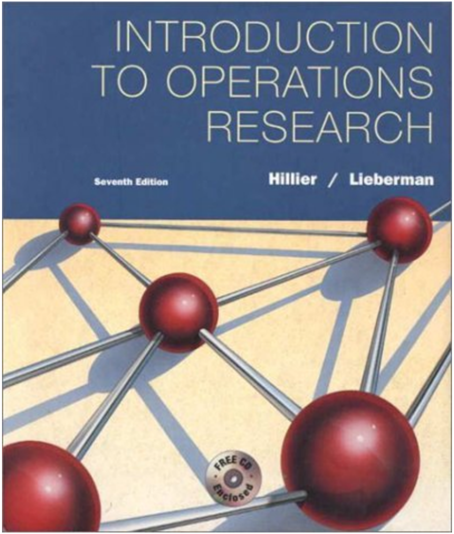
Supply – Allocation of Resources

Boeing 777 - Cost $300 M

?

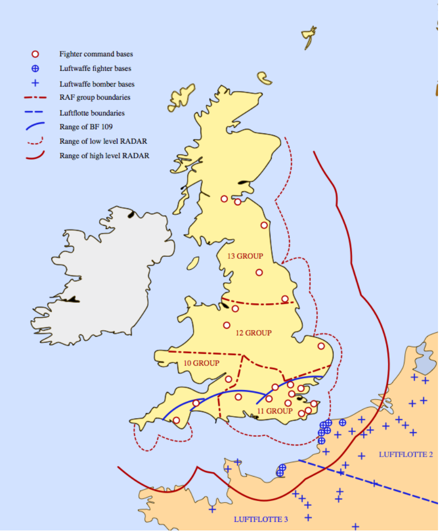


How to solve the problem?

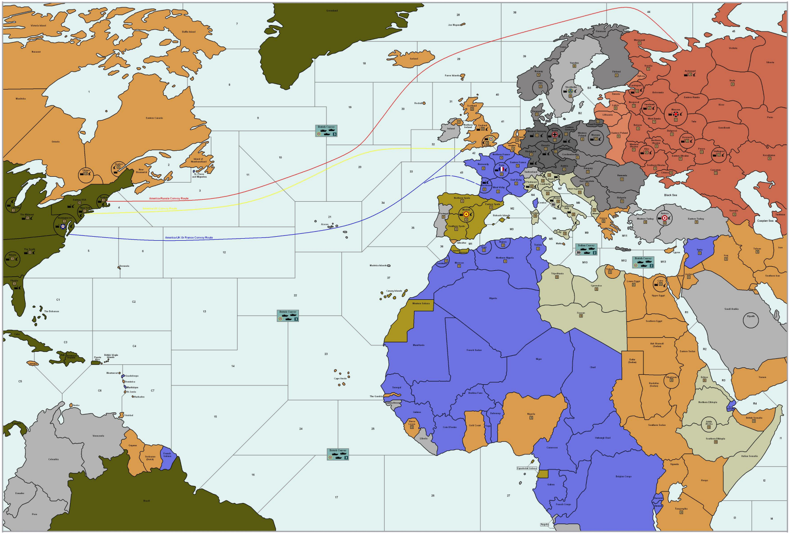


Overview

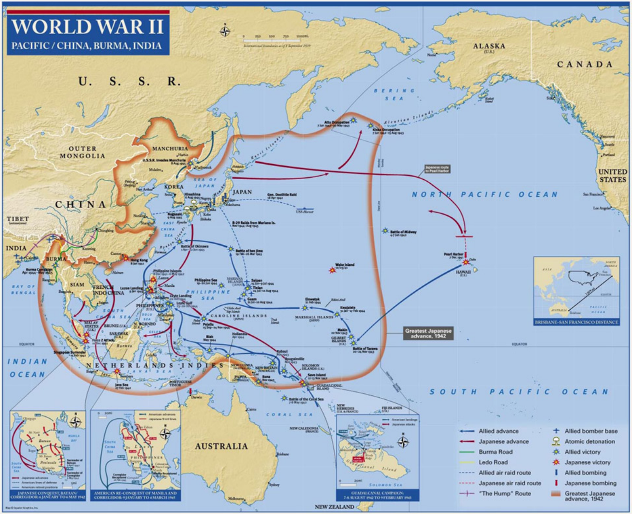
The Battle of Britain



The Battle of the North Atlantic



Campaign in the Pacific



©Committee, Guilford County Veterans Memorial. Maps ©International Mapping Associates.

Invention of the Simplex Method – 1947

Figure: George Dantzig

- November 8, 1914 – May 13, 2005.
- Professor Emeritus of Transportation Sciences and Professor of Operations Research and of Computer Science at Stanford.
- Developed the Simplex method, which is used for solving linear programs.
- The journal Computing in Science and Engineering listed it as one of the top 10 algorithms of the twentieth century.

Overview

Introduction

- A linear program is an optimization problem that seeks to minimize/maximize a linear function subject to linear equality and inequality constraints.
- Every linear program can be written in canonical form as:

$$\text{minimize} \quad \mathbf{c}^{\mathrm{T}}\mathbf{x} \tag{1}$$
$$\text{subject to} \quad A\mathbf{x} \leq \mathbf{b} \tag{2}$$
$$\mathbf{x} \geq \mathbf{0} \tag{3}$$

- You will learn a lot about linear programming during the lectures of this course.
- We use linear programming because:
  - of the high level of methodological maturity of the approach
  - it is convex, i.e. we can guarantee a global optimum.

Components of a linear program

- Parameters: $\mathbf{c}$, $\mathbf{b}$ and A
  - This is the known data at the time of optimization.
- Decision variables: $\mathbf{x}$
  - Always non-negative.
  - We are looking for their optimal values
  - Depending on the domains of the decision variables we can have:
    - LP – linear programming
    - ILP – integer linear programming
    - MILP – mixed integer linear programming
- Objective function: $\mathbf{c}^{\mathrm{T}}\mathbf{x}$
  - A linear combination of the decision variables that we are trying to minimize.
- Constraints: $A\mathbf{x} \leq \mathbf{b}$
  - Linear inequalities that define the feasible set over which the optimization is to take place. The values of the decision variables respect the constraints.

Example 1

- At a refinery, the refining process requires the production of at least 2 liters of gasoline for 1 liter of heating oil.
- To meet heating demands, the refinery must produce at least 3 million liters of heating oil daily.
- The demand for gasoline does not exceed 6 million liters daily.
- Gasoline sells at CHF 1.45/L and heating oil at CHF 1.20/L.
- Formulate an LP with the objective of maximizing total daily revenue.

Example 2

- Suppose there are 3 staple foods available - corn, milk and bread, with their attributes given in the table below:

Food	Cost per serving	Vitamin A	Calories	Max servings
Corn	CHF 1.80	107	72	10
Milk	CHF 2.30	500	121	10
Bread	CHF 0.50	0	65	10

- The maximum number of servings per day for each food is 10
- The number of calories per day is restricted between 2'000 and 2'250.
- The amount of vitamin A per day is restricted between 5'000 and 50'000.
- Formulate an LP with the objective of minimizing the cost of the diet.

Solving the problems

- How do we actually solve these problems?
- In this course you will learn the theory of the Simplex method, which is used to solve linear programs.
- Examples like the ones you have seen are very simple, and can even be solved by hand.
- For realistic problem sizes, however, we need powerful software.
- In the remaining labs of this course, you will learn how to use CPLEX, with a specific application on modeling complex transportation systems.

Overview

IBM ILOG CPLEX Optimization Studio

Shopping cart items

Quantity	Part number		*IBM price excluding tax	Line total
1	D0CV0LL		8,740.00	8,740.00
Authorized User	**Description**	IBM ILOG CPLEX Optimization Studio Developer Edition Authorized User License + SW Subscription & Support 12 Months		

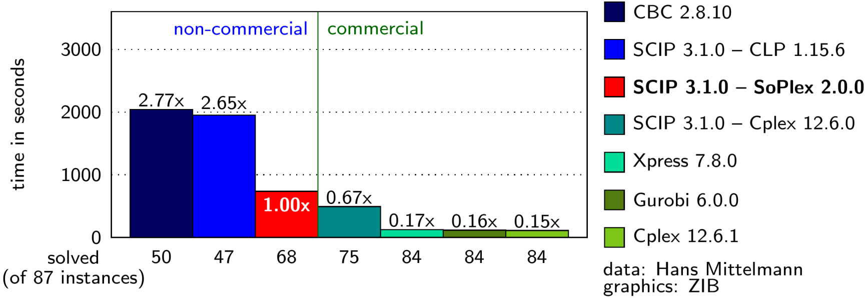
- 90 Days Trial
- The distribution bundles OPL with CPLEX into an Integrated Development Environment (IDE) called IBM ILOG CPLEX Optimization Studio.

What is OPL? What is CPLEX?

- OPL, the Optimization Programming Language, is a modeling language used to formulate mathematical models.
- It provides a syntax that is very close to the mathematical formulation, thus making the computer implementation very easy.
- It enables a clean separation between the model and the accompanying data. Thus, the same model can be solved for different input data with little extra effort.
- The mathematical model is solved by the CPLEX solver and the result is reported in the IDE.
- CPLEX can solve linear, quadratic and quadratically constrained programs, but in this course we will only solve linear programs.
- FYI, there are other modeling languages, such as GAMS, AMPL, Mosel, and other solvers, for example Gurobi, GLPK, CBC...

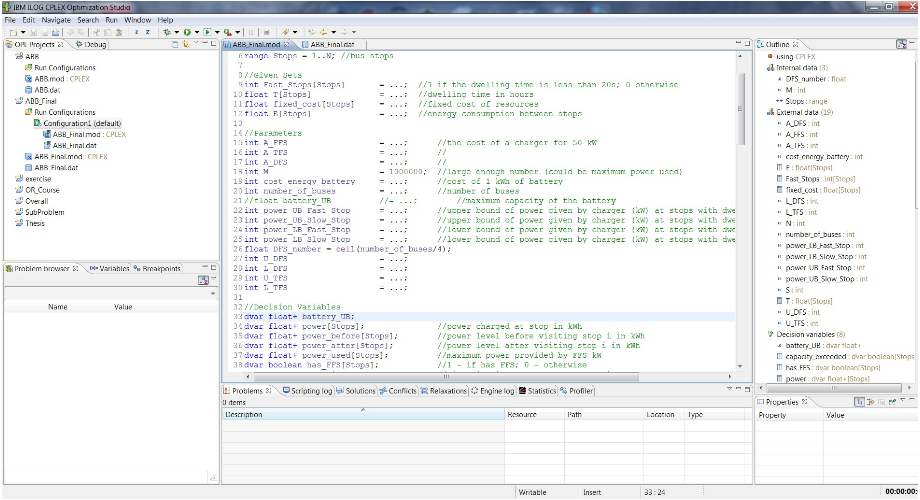
Why do we use CPLEX?

- CPLEX is a state-of-the-art commercial solver
- It is in active development and is continuously improved
- It is widely used in both academia and industry



Source: SCIP website

Layout


IBM ILOG CPLEX Optimization Studio
File Edit Navigate Search Run Window Help
OPL Projects
Debug
ABB
Run Configurations
ABB.mod : CPLEX
ABB.dat
ABB_Final
Run Configurations
Configuration1 (default)
ABB_Final.mod : CPLEX
ABB_Final.dat
ABB_Final.mod : CPLEX
ABB_Final.dat
exercise
OR_Course
Overall
SubProblem
Thesis
ABB_Final.mod
ABB_Final.dat
range Stops = 1..N; //bus stops
//Given Sets
int Fast_Stops[Stops] = ...; //1 if the dwelling time is less than 20s; 0 otherwise
float T[Stops] = ...; //dwelling time in hours
float fixed_cost[Stops] = ...; //fixed cost of resources
float E[Stops] = ...; //energy consumption between stops
//Parameters
int A_FFS = ...; //the cost of a charger for 50 kW
int A_TFS = ...; //
int A_DFS = ...; //
int M = 1000000; //large enough number (could be maximum power used)
int cost_energy_battery = ...; //cost of 1 kWh of battery
int number_of_buses = ...; //number of buses
//float battery_UB //= ...; //maximum capacity of the battery
int power_UB_Fast_Stop = ...; //upper bound of power given by charger (kW) at stops with dwe
int power_UB_Slow_Stop = ...; //upper bound of power given by charger (kW) at stops with dwe
int power_LB_Fast_Stop = ...; //lower bound of power given by charger (kW) at stops with dwe
int power_LB_Slow_Stop = ...; //lower bound of power given by charger (kW) at stops with dwe
float DFS_number = ceil(number_of_buses/4);
int U_DFS = ...;
int L_DFS = ...;
int U_TFS = ...;
int L_TFS = ...;
//Decision Variables
dvar float+ battery_UB;
dvar float+ power[Stops]; //power charged at stop in kWh
dvar float+ power_before[Stops]; //power level before visiting stop i in kWh
dvar float+ power_after[Stops]; //power level after visiting stop i in kWh
dvar float+ power_used[Stops]; //maximum power provided by FFS kW
dvar boolean has_FFS[Stops]; //1 - if has FFS; 0 - otherwise
Outline
using CPLEX
Internal data (3)
DFS_number : float
M : int
Stops : range
External data (19)
A_DFS : int
A_FFS : int
A_TFS : int
cost_energy_battery : int
E : float[Stops]
Fast_Stops : int[Stops]
fixed_cost : float[Stops]
L_DFS : int
L_TFS : int
N : int
number_of_buses : int
power_LB_Fast_Stop : int
power_LB_Slow_Stop : int
power_UB_Fast_Stop : int
power_UB_Slow_Stop : int
S : int
T : float[Stops]
U_DFS : int
U_TFS : int
Decision variables (8)
battery_UB : dvar float+
capacity_exceeded : dvar boolean[Stops
has_FFS : dvar boolean[Stops]
power : dvar float+[Stops]
Problem browser
Variables
Breakpoints
Name
Value
Problems
Scripting log
Solutions
Conflicts
Relaxations
Engine log
Statistics
Profiler
0 items
Description
Resource
Path
Location
Type
Properties
Property
Value
Writable
Insert
33 : 24
00:00:00:

Syntax

- Data declarations - known parameters
  - simple

    ```
    int c = 8; float b = 3.2; string s = "EPFL, TRANSP-OR";
    ```
  - range/set

    ```
    range Days = 1..7;
    {string} season = {"spring","summer","autumn","winter"};
    ```
  - array:

    ```
    float a[season] = [1.0, 2.0, 3.0, 5.0];
    a["winter"]; // for accessing
    ```
  - from data file:
    - in the model file:

      ```
      {string} countries = ...;
      ```
    - in the data file:

      ```
      countries = {"Switzerland", "France", "Italy"};
      ```

Syntax

- In OPL, we do no define each variable and parameter separately, rather we define them over sets for the purpose of indexing.
- Let's learn by example. In the diet problem, we need these sets:

```
{string} foods = {"corn", "milk", "bread"};
{string} subs = {"vitamin A", "calories"};
```

- We can define parameter arrays (single and multi-dimensional):

```
float cost[foods] = [1.80, 2.30, 0.50];
int maxserving[foods] = [10, 10, 10];
int contents[foods][subs] = [[107,72],[500,121],[0,65]];
int mincontent[subs] = [5000, 2000];
int maxcontent[subs] = [50000, 2250];
```

- We can define a continuous decision variable as follows:

```
dvar float+ amount[foods];
```

Syntax

- We always need an objective function

```
minimize, maximize

minimize sum(f in foods) cost[f] * amount[f];
```

- And of course constraints:

```
subject to {
 forall(f in foods)
  amount[f] <= maxserving[f];

 forall(s in subs) {
  sum(f in foods) contents[f,s] * amount[f] >= mincontent[s];
  sum(f in foods) contents[f,s] * amount[f] <= maxcontent[s];
 }
};
```

Syntax

- There are many other particularities that we will cover during the labs.
- For example tuples, filters, etc.
- Types of decision variables – usually "float+" for continuous, "int+" for integer, "boolean" for binary.
- A good starting point to learn by yourself is the course webpage: `http://transp-or.epfl.ch/courses/decisionAid2015/`.
- There are several introductory resources under the labs section.

Small Example

A company produces two products:

- doors
- windows

It has three production facilities with limited production time available:

- Facility 1 produces the metal frame
- Facility 2 produces the wooden frame
- Facility 3 produces glass and mounts the parts

Each product generates a revenue, and requires a given amount of time of each facility's capacity. Find the number of products of each type to produce in order to maximize the revenue.

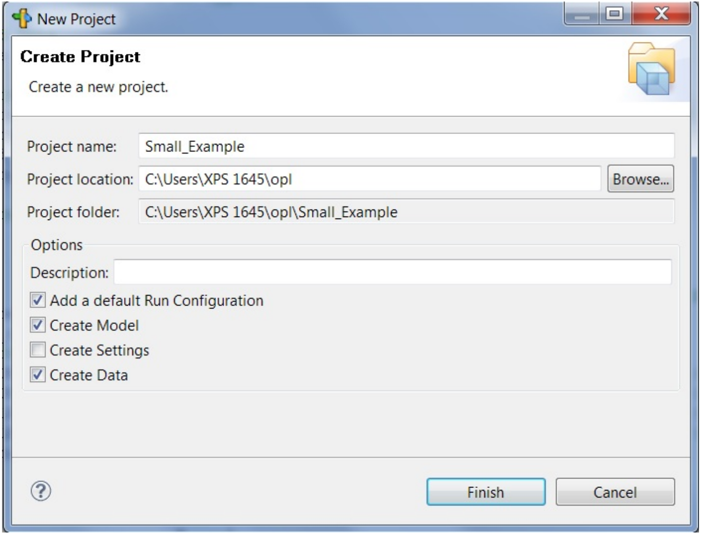
Small Example

	Hours per product		Hours at Disposal
	Door	Window	
Factory 1	1	0	4
Factory 2	0	3	12
Factory 3	3	2	18
Revenue per product	3 000	5 000	–

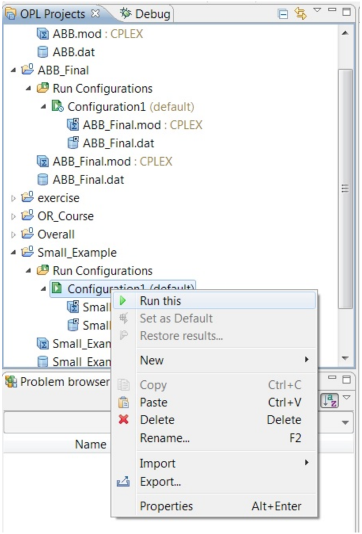
Small Example

- Formulate the problem mathematically:
    - input parameters
    - decision variable(s)
    - objective function
    - constraints
- Model the problem in OPL
    - Start by thinking what sets you need to index your parameters and variables.
    - Then define the parameters with the given input data.
    - Declare the decision variables.
    - Write the objective and constraints.
    - OPL is installed on the machines in this lab – you should run as administrator.
- Run the model and check your results in the "Solutions" tab

Create New Project in OPL



How to Run the Model



Results

- Decision is integer:
  - revenue: 29 000
  - $x = [3\ 4]$
- Decision is float:
  - revenue: 30 000
  - $x = [3.3333\ 4]$

References

- The presentation has been based on:
- http://folk.uio.no/trulsf/opl/opl_tutorial.pdf
- The lab slides of Tomáš Robenek from 2013