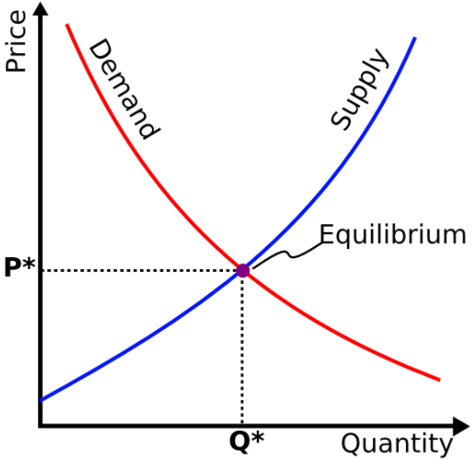
# Decison-Aid Methodologies in Transportation

## Optimization Exercise 1

Transport and Mobility Laboratory
EPFL

April 1, 2014

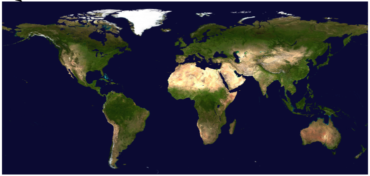
# Demand – Supply



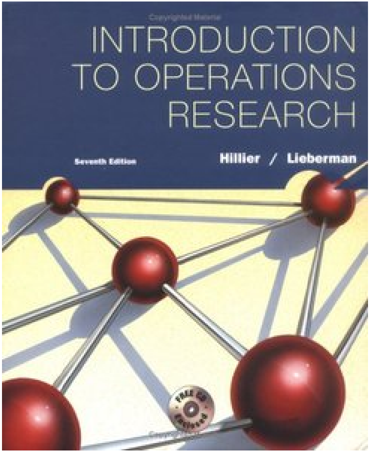
# Supply – Allocation of Resources


Boeing 777 - Cost $300 M

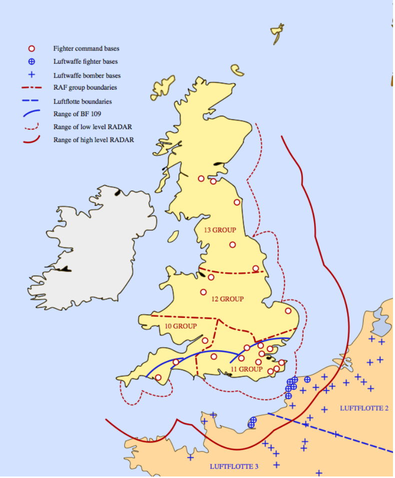
?



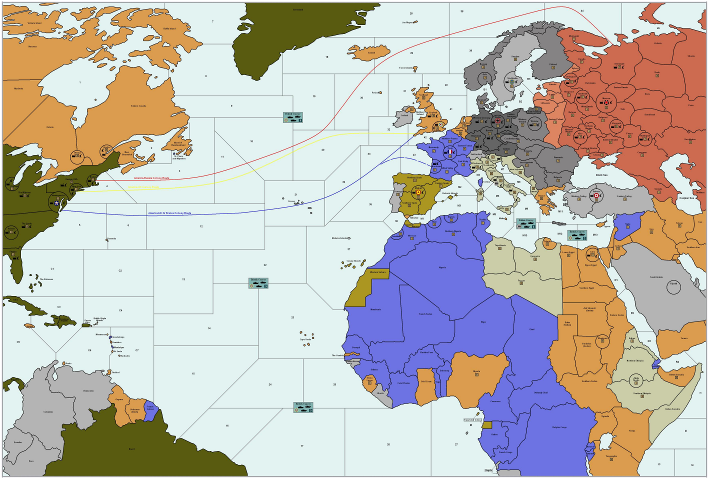
# How to solve the problem?



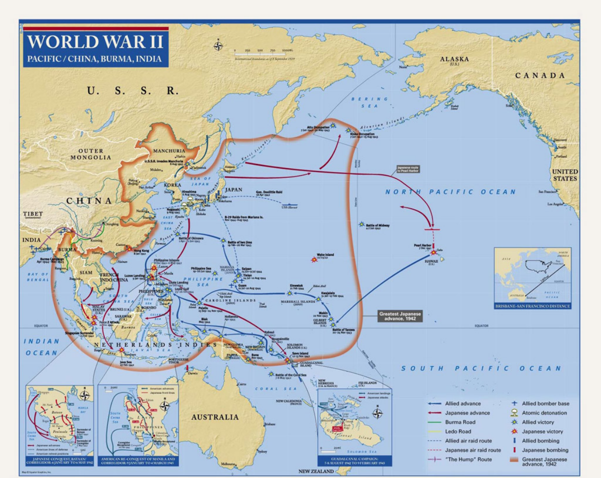
# The Battle of Britain



# The Battle of the North Atlantic



# Campaign in the Pacific



# Invention of 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
- finding the best assignment of 70 people to 70 jobs – the number of possible configurations exceeds the number of particles in the universe
- The journal Computing in Science and Engineering listed it as one of the top 10 algorithms of the twentieth century

# References I

- Integer Programming – Global Impact by George Nemhauser, Euro 2013, Rome, Italy
- `http://euro2013.org/wp-content/uploads/Nemhauser_EuroXXVI.pdf`

# IBM ILOG CPLEX Optimization Studio

**Shopping cart items**

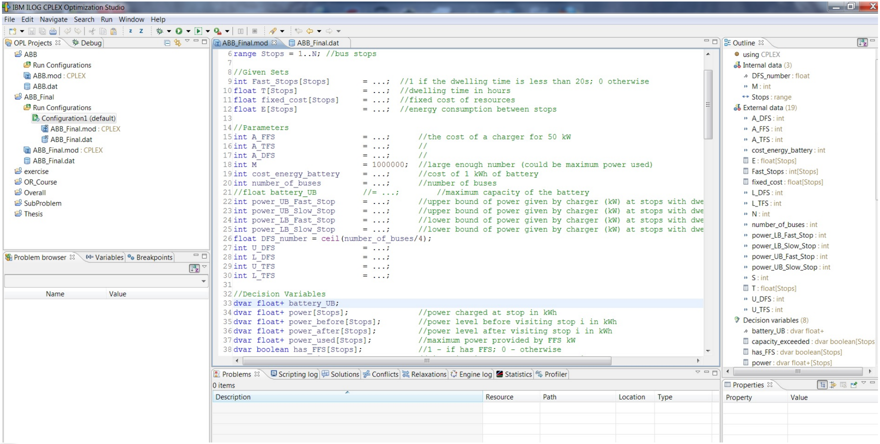
| Quantity | Part number | *IBM price excluding tax | Line total |
|---|---|---|---|
| 1 | D0CV0LL | 8,740.00 | 8,740.00 |
| Authorized User | **Description** | IBM ILOG CPLEX Optimization Studio Developer Edition Authorized User License + SW Subscription & Support 12 Months | |

- 90 Days Trial
- OPL Studio – Part of the distribution

# Why OPL?

- OPL provides syntax that is close to the mathematical formulation, thus making it easier to make the transition from the mathematical formulation to something that can be solved by the computer.
- It enables a clean separation between the model and the accompanying data. The same model can then be solved with different input data with little extra effort.
- Other modeling languages: GAMS, AMPL and Mosek
- Used for linear and integer problems; not for non-linear problems!

# 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
Problem browser
Variables
Breakpoints
Name
Value
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]
Problems
Scripting log
Solutions
Conflicts
Relaxations
Engine log
Statistics
Profiler
0 items
Description
Resource
Path
Location
Type
Properties
Property
Value

# Syntax

- Data Declarations – to define known parameters
  - **simple**: int c = 8; float b = 3.2; string s = "TRANSP-OR";
  - **range/tuple/set**: range Days = 1..7; tuple clock {int h; int m; int s} clock now=<11,23,45>; now.h; {string} season = {"spring","summer","autumn","winter"};
  - **array**: int a[1..4] = [2,0,1,4]; clock a[1..2] = [<1,2,3>,<4,5,6>]; float a[season] = [1.0,2.0,3.0,5.0]; a["winter"];
  - **data initialization from a dat file**:
    - in the model file: {string} days[Days] = ...;
    - in the data file: days = {"Monday", "Tuesday",...,"Sunday"};
- Decision Variables –
  - dvar float+ x; // <=> dvar float x in 0..infinity;
  - dvar int+ y; // <=> dvar int y in 0..maxint;
  - dvar boolean z; // <=> dvar int z in 0..1;

# Syntax

- Objective Function –

```
minimize, maximize
maximize 6 * x;
```

- Constraints –

```
subject to {
    forall(i,j in I : i>j) // ":" conditional filter
        x[i][j] <= 10;
    sum(i,j in I)x[i][j] * 5 == 65;
};
```

- Postprocessing data –

```
execute{writeln("x=", x);}
```

the code within the execute{} is called a script (javascript).

# Code structure

- data structure definition (OPL: Optimization Programming Language, developed by IBM)
- data initialization (OPL, within mod file, read from dat file, Excel, or database)
- data preprocessing (javascript in execute{} block, **optional**)
- decision variable definition (OPL)
- Objective function definition (OPL, only one objective is allowed)
- Constraints definition (OPL)
- data postprocessing (javascript, **optional**)

# Small Example

A company produces two products:

- doors
- windows

It has three production facilities with limited production time available:

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

The products are produced in series of 200 items and each series generates a revenue depending on the product. Each series require a given amount of time of each factory's capacity. The problem is to find the number of series of each product to maximize the revenue.

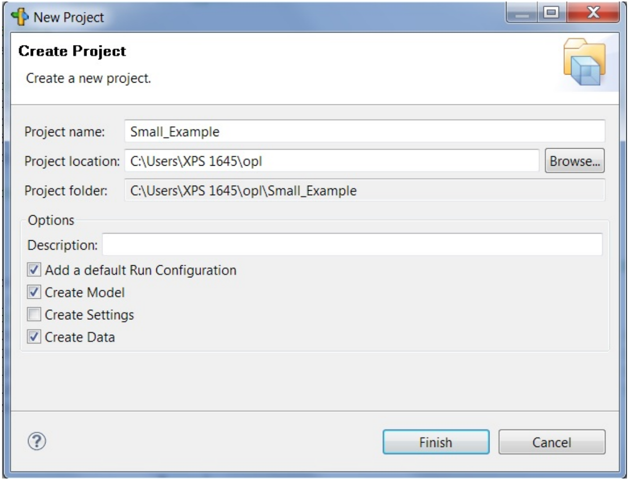
# Small Example

| | Hours/Series | | Hours at Disposal |
|---|---|---|---|
| | Door | Window | |
| Factory 1 | 1 | 0 | 4 |
| Factory 2 | 0 | 3 | 12 |
| Factory 3 | 3 | 2 | 18 |
| Revenue/Series | 3 000 | 5 000 | – |

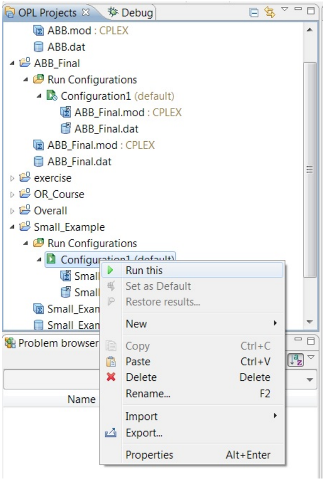
# Small Example

- Formulate the problem mathematically:
  - input parameters
  - decision variable(s)
  - objective function
  - constraints
- Model the problem in OPL
- Run the model and check your results in the "Solutions" tab
- When finished proceed to the instructions 8 file

# 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]

TRANSP-OR

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

TRANSP-OR

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

# References II

- The presentation has been based on the following tutorial:
- `http://folk.uio.no/trulsf/opl/opl_tutorial.pdf`