Network Models II

Shortest Path

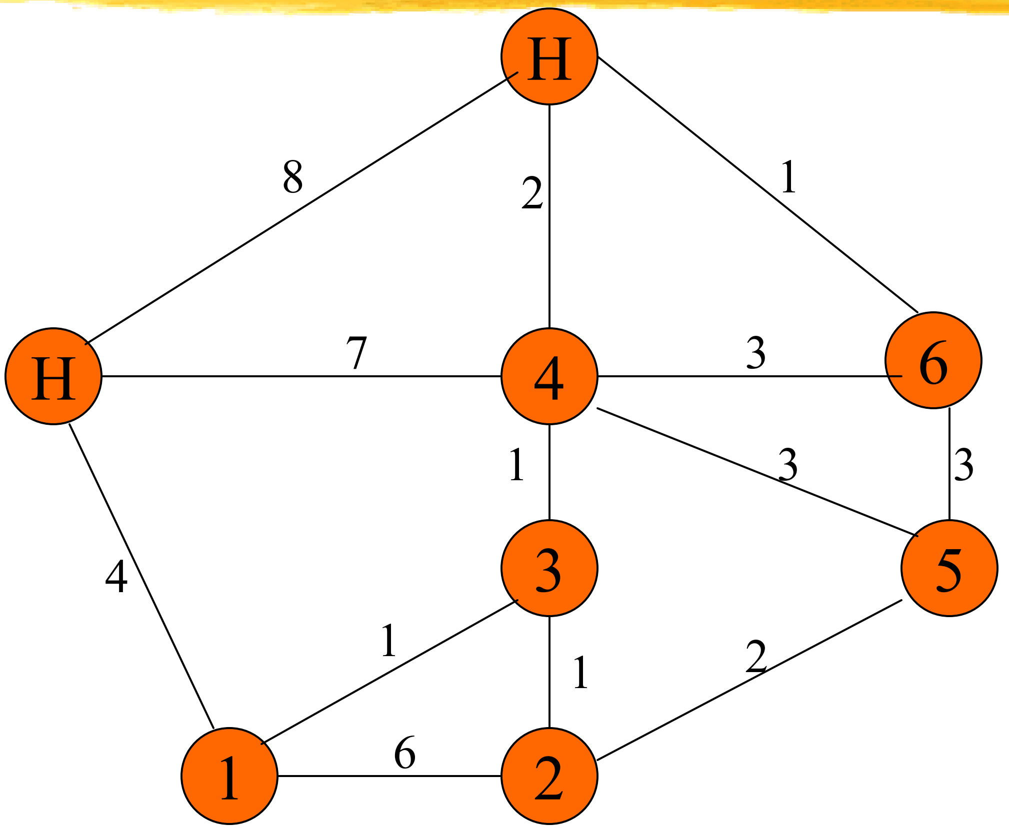
Cross Docking

Enhance Modeling Skills

Modeling with AMPL

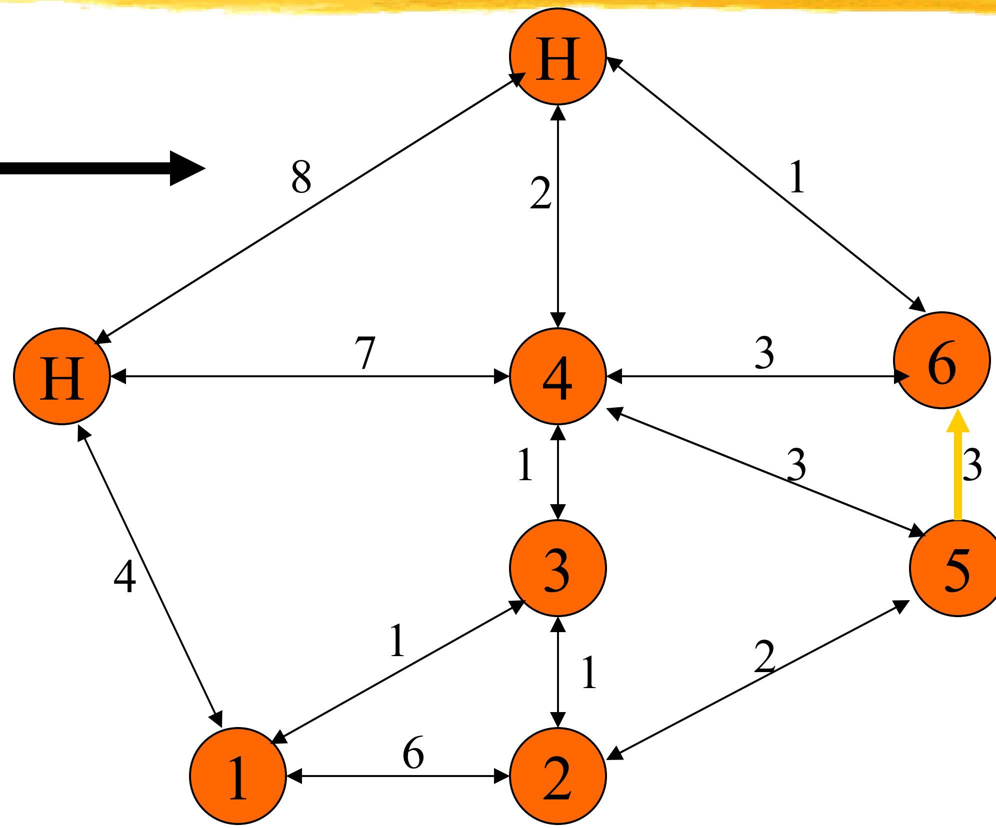
The Shortest Path Model

- Find the shortest path from Home to 5



Direction

- Two-way streets
- One-way streets



03ShortestPathModel.xls

Shortest Path Model

Connectivity

From\To	Home	Site 1	Site 2	Site 3	Site 4	Site 5	Site 6	Site 7
Home		1			1			1
Site 1	1		1	1				
Site 2		1		1		1		
Site 3		1	1		1			
Site 4	1			1		1		1
Site 5			1		1		1	
Site 6					1	1		1
Site 7	1				1		1	

Distance

From\To	Home	Site 1	Site 2	Site 3	Site 4	Site 5	Site 6	Site 7
Home		4			7			8
Site 1	4		6	1				
Site 2		6		1		2		
Site 3		1	1		1			
Site 4	7			1		3		2
Site 5			2		3		3	
Site 6					3	3		1
Site 7	8				2		1	

Route

From\To	Home	Site 1	Site 2	Site 3	Site 4	Site 5	Site 6	Site 7	Total From
Home									0
Site 1									0
Site 2									0
Site 3									0
Site 4									0
Site 5									0
Site 6									0
Site 7									0
Total To	0	0	0	0	0	0	0	0	
Total From - Total To	0	0	0	0	0	0	0	0	
Net Required	1	0	0	0	0	-1	0	0	

Total Distance	Home	Site 1	Site 2	Site 3	Site 4	Site 5	Site 6	Site 7	Total From
Home	0	0	0	0	0	0	0	0	0
Site 1	0	0	0	0	0	0	0	0	0
Site 2	0	0	0	0	0	0	0	0	0
Site 3	0	0	0	0	0	0	0	0	0
Site 4	0	0	0	0	0	0	0	0	0
Site 5	0	0	0	0	0	0	0	0	0
Site 6	0	0	0	0	0	0	0	0	0
Site 7	0	0	0	0	0	0	0	0	0
Total To	0	0	0	0	0	0	0	0	0

Challenge

- Build a Solver model

A Solver Model

- The Objective: Minimize $U$21
- The Variables: $C$13:$J$20
- The Constraints:
  - Only travel on existing edges
    - $C$13:$J$20 <= $C$3:$J$10
  - Number From - Number To = Net Required
    - $C$22:$J$22 = $C$23:$J$23

Flow Conservation

- Number From - Number To = Net Required
- Number of times we leave - Number of times we enter = ?
- +1 at Home (we leave once)
- -1 at Site 5 (we arrive once)
- 0 everywhere else
  - each time we arrive (if ever), we leave

Compare with Assignment Model

- Assignment Model
  - Sum across each row = 1
  - Sum down each column = 1
  - Each variable appears in 2 constraints
- Shortest Path Model
  - Sum across a row - Sum down the column = 0
  - Trips out of a site - Trips into the site
  - Each variable appears in ? constraints

Network Flow Problems

- Each variable appears in at most two constraints
  - At most one constraint as - the variable
  - At most one constraint at + the variable
- Assignment
  - Sum across each row = 1
  - Sum down each column = 1
- Shortest Path
  - Sum across the a row - sum down the col = #

Bounds

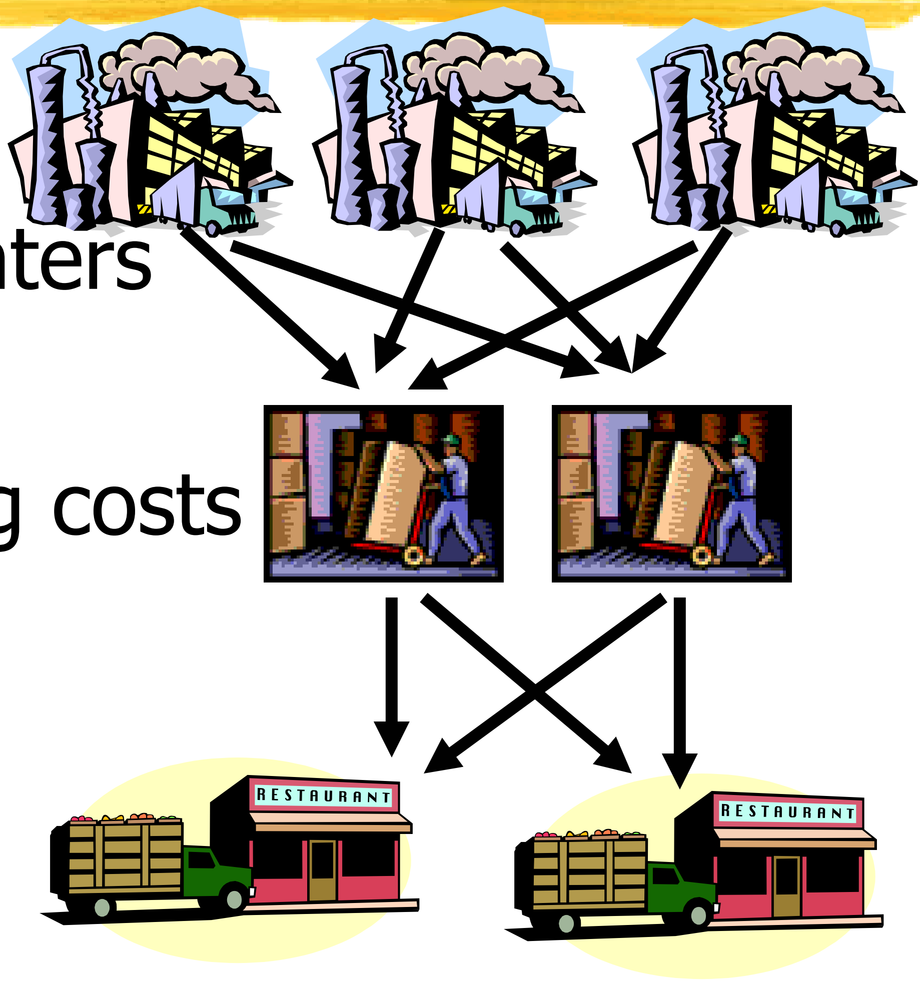
- Variables can also have bounds
  - e.g., in the Shortest Path Model:
    - Number of times we use each variable
      - Lower bound: >= 0
      - Upper bound: <= 1 if it is an edge, 0 otherwise

Properties of Network Flows

- If the bounds and RHS are integral, the solution will be integral
- It the costs are integral, the reduced costs and marginal values will be integral
- Can be solved very quickly
- Limited demands on memory

Crossdocking

- 3 plants
- 2 distribution centers
- 2 customers
- Minimize shipping costs



A Network Model

Minimum Cost Network Flow Problem

Unit Shipping Costs

Transportation Costs ($ 000/Ton)

Plant to DC	DC 1	DC 2
Plant 1	$ 5.0	$ 5.0
Plant 2	$ 1.0	$ 1.0
Plant 3	$ 1.0	$ 0.5

DC to Customer	DC 1	DC 2
Customer 1	$ 2.0	$ 2.0
Customer 2	$ 12.0	$ 12.0

Capacities
Flows
Payments

Arc Capacities

Transportation Capacities

Costs
Flows
Payments

Plant to DC	DC 1	DC 2
Plant 1	200	200
Plant 2	200	200
Plant 3	200	200

DC to Customer	DC 1	DC 2
Customer 1	200	200
Customer 2	200	200

Shipments

Plant to DC	DC 1	DC 2	Total Out	**Supply**
Plant 1	-	-	-	200
Plant 2	-	-	-	300
Plant 3	-	-	-	100
Total In	-	-		

Costs
Capacities
Payments

DC to Customer	DC 1	DC 2	Total In	**Demand**
Customer 1	-	-	-	400
Customer 2	-	-	-	180
Total Out	-	-		

Net Flow	DC 1	DC 2
	-	-

Payments

Costs
Capacities
Flows

Plant to DC	DC 1	DC 2	Total Out
Plant 1	$ -	$ -	$ -
Plant 2	$ -	$ -	$ -
Plant 3	$ -	$ -	$ -
Total In	$ -	$ -	$ -

DC to Customer	DC 1	DC 2	Total Out
Customer 1	$ -	$ -	$ -
Customer 2	$ -	$ -	$ -
Total In	$ -	$ -	$ -

Total Shipping Cost $ -

Challenge

⌘Build a Solver Model

A Solver Model

- Objective: Minimize $K$28
- Variables: $C$17:$D$19, $C$23:$D$24
- Constraints:
  - Do not exceed supply at the plants
    - $E$17:$E$19 <= $F$17:$F$19
  - Meet customer demand
    - $E$23:$E$24 >= $F$23:$F$24
  - Do not exceed shipping capacity
    - $C$17:$D$19 <= $K$6:$L$8 and
    - $C$23:$D$24 <= $K$11:$L$12

And…

- Flow conservation at the DCs
  - \$C\$28:\$D\$28 = 0

- Supply and Demand like Autopower
- Flow conservation at DCs like Shortest Path

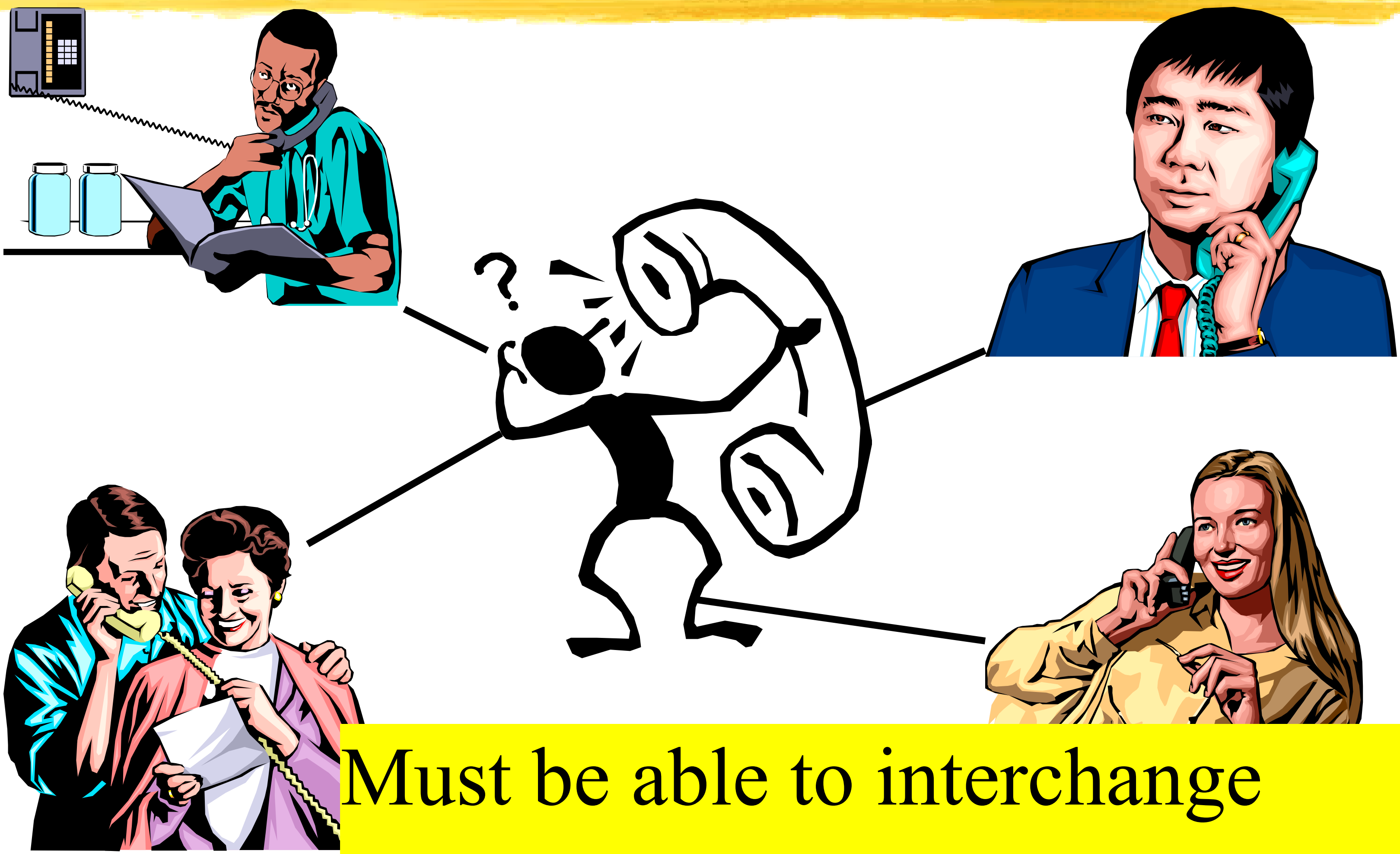
Network Flows: Good News

- Lots of applications
- Simple Models
- Optimal Solutions Quickly
- Integral Data, Integral Answers

Network Flows: Bad News

- Underlying Assumptions
  - Single Homogenous Product
  - Linear Costs
  - No conversions or losses
  - ...

Homogenous Product



Must be able to interchange positions of product anywhere

Linear Costs

- No Fixed Charges
- No Volume Discounts
- No Economies of Scale

Summary

- Network Flows
  - Simple Formulation
    - Flow Out (sum across a row) <= Capacity
    - Flow In (sum down a column) >= Demand
    - Flow In - Flow Out = Constant
  - Limited by
    - Homogenous Product
    - Linear Costs
    - etc.
  - Integer Data give Integral Solutions

Modeling with AMPL

- Problems with Excel Solver
  - Integration of "Model" and Data
    - Example:
      - Change the time horizon of our Inventory Model
  - Excel is a limited database tool
- Algebraic Modeling Languages
  - Separate the "Model" from the Data
  - Keep the data in databases

How they work

Conceptual Model

Data

Algebraic Modeling Language
AMPL/OPL/GAMS/XPress/...

Optimizer
CPLEX
OSL
XPress

Why AMPL

- Established in US
- Very good book
- Lower barrier to entry
- Free "student" version
- Industrial strength tool

Our Use of AMPL

- Pseudo AMPL to discuss models
  - In class
  - In exams
- Need to be precise about
  - What's a parameter, variable, …
  - Indexing: relationships between variables, data, constraints
- Challenges and Project

Is this necessary/valuable?

- AMPL is very detailed
  - Expect 1 or 2 per team to master
  - Rest to read and understand
- Brings out the real issues
  - Practical implementation -- you can oversee
  - Data issues -- the real challenge
- Valuable tool

The Transportation Model

- set ORIG;
- set DEST;
- param supply {ORIG};
- param demand {DEST};
- param cost {ORIG, DEST};
- var Trans {ORIG, DEST} >= 0;

Transportation Model

```
minimize Total_Cost:
   sum{o in ORIG, d in DEST}
         cost[o,d]*Trans[o,d];
s.t. Supply {o in ORIG}:
   sum{d in DEST} Trans[o,d] <= supply[o];
s.t. Demand {d in DEST}:
   sum{o in ORIG} Trans[o,d] >= demand[d];
```

The Data

- An Access Database called TransportationData.mdb
- Tables in the database
  - Origins: Supply information

Origin	Supply
Amsterdam	500
Antwerp	700
The Hague	800

  - Destinations: Demand information

Destination	Demand
Leipzig	400
Liege	200
Nancy	900
Tilburg	500

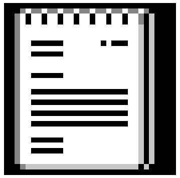
The Costs

- Cost: Unit transportation costs

origin	destination	cost
Amsterdam	Leipzig	120
Amsterdam	Liege	41
Amsterdam	Nancy	130
Amsterdam	Tilburg	59.5
Antwerp	Leipzig	61
Antwerp	Liege	100
Antwerp	Nancy	40
Antwerp	Tilburg	110
The Hague	Leipzig	102.5
The Hague	Liege	122
The Hague	Nancy	90
The Hague	Tilburg	42

AMPL’s Output

- AMPL reads the model and the data, combines the two and produces (in human readable form) …



AMPLOutput.txt

Produced by the command:

expand >AMPLOutput.txt

Reading Data

table OriginTable IN "ODBC"

"D:\Personal\15057\TransportationData.mdb"

"Origins":

ORIG <- [Origin], supply~Supply;

Explanation:

'table' is a keyword that says we will read or write data

'OriginTable' is a name we made up. No other AMPL model entity can have this name

'IN' is a key word that says we are reading data.

"ODBC" says we are using ODBC to read the data

Explanation

- "D:\Personal\15057\TransportationData.mdb” is the path to the database. Alternatively you can create a DSN (data source name) for this file, say TransportData, and use the command “DSN=TransportData”.
- "Origins” is the name of the table in the database. Alternatively we can use an SQL command like “SQL=SELECT * FROM Origins”
- The : is syntax. What follows is the mapping of the data we read to AMPL objects that will hold it.
- The brackets [] around Origin mean that this field in the database indexes the data, e.g., 500 is the supply for Amsterdam.

Explanation Continued

- ORIG <- [Origin] says that the values of the field Origin will define the set ORIG of origins
- supply~Supply says that the values of the parameter supply should hold the values read from the field Supply in the database
- read table OriginTable; reads the data.

Reading Data

```
table DestinationTable IN "ODBC"
"D:\Personal\15057\TransportationData.mdb"
"Destinations":
DEST <- [Destination], demand~Demand;
```

Explanation:

- ‘table’ is a keyword that says we will read or write data
- ‘DestinationTable’ is a name we made up. No other AMPL model entity can have this name
- ‘IN’ is a key word that says we are reading data.
- “ODBC” says we are using ODBC to read the data

Explanation

- "D:\Personal\15057\TransportationData.mdb" is the path to the database. Alternatively you can create a DSN (data source name) for this file, say TransportData, and use the command "DSN=TransportData".
- "Destinations" is the name of the table in the database. Alternatively we can use an SQL command like "SQL=SELECT * FROM Destinations"
- The : is syntax. What follows is the mapping of the data we read to AMPL objects that will hold it.
- The brackets [] around Destination mean that this field in the database indexes the data, e.g., 400 is the demand for Leipzig.

Explanation Continued

- DEST <- [Destination] says that the values of the field Destination will define the set DEST of destinations
- demand~Demand says that the values of the parameter demand should hold the values read from the field Demand in the database

Reading Cost

table CostTable IN "ODBC"
"D:\Personal\15057\TransportationData.mdb"
"Cost":
[origin, destination], cost;

Explanation:

- 'table' is a keyword that says we will read or write data
- 'CostTable' is a name we made up. No other AMPL model entity can have this name
- 'IN' is a key word that says we are reading data.
- "ODBC" says we are using ODBC to read the data

Explanation

- "D:\Personal\15057\TransportationData.mdb" is the path to the database. Alternatively you can create a DSN (data source name) for this file, say TransportData, and use the command "DSN=TransportData".
- "Cost" is the name of the table in the database. Alternatively we can use an SQL command like "SQL=SELECT * FROM Cost"
- The : is syntax. What follows is the mapping of the data we read to AMPL objects that will hold it.
- The brackets [] around origin and destination mean that these two fields in the database index the data, e.g., 120 is the unit transportation cost from Amsterdam to Leipzig.

Explanation Continued

- We don't have an <- here, because we are not defining the members of a set.
- We read the values of the field cost in the database into the parameter cost. Note that since these two names are identical, we don't need the ~.

- read table CostTable; reads the data.

Running AMPL

- model d:\15057\TransportationModel.mod;
- option solver cplex; # use cplex to solve
- solve;
- display Trans;

Writing Output

```
table TransOutTable OUT "ODBC"
"D:\Personal\15057\TransportationData.mdb"
"TransOut":
{origin in ORIG, destination in DEST:
        Trans[origin, destination] > 0}
   -> [origin, destination], Trans[origin,destination]~Trans;
write table TransOutTable;
```

Explanation:

- 'table' is a keyword that says we will read or write data
- 'TransOutTable' is a name we made up. No other AMPL model entity can have this name

Explanation

- 'OUT' is a key word that says we are writing data.
- "ODBC" says we are using ODBC to write the data
- "D:\Personal\15057\TransportationData.mdb" is the path to the database. Or you can use "DSN=..."
- "TransOut" is the name of the table to create. AMPL drops and writes this table. Any data currently in the table is lost.
- : is syntax. It separates the description of the destination from the definition of the data and the mapping of the columns

More Explanation

- {origin in ORIG, destination in DEST:
  Trans[origin, destination] > 0} defines the index set that will control the data to write out. This says to only report on origin-destination pairs where we actually send a positive flow.
- -> is syntax. It separates the indexing from the data definition and mapping to fields of the output table.
- [origin, destination] indicates that the records of the output table are indexed by the origin-destination pairs. AMPL will write a new record for each pair.
- Trans[origin,destination]~Trans says to create a field called Trans in the table and to populate it with the values of the Trans variable.

Explanation completed

- write table TransOutTable; actually writes the data.
- The output is:

origin	destination	Trans
Amsterdam	Leipzig	300
Amsterdam	Liege	200
The Hague	Leipzig	100
The Hague	Tilburg	500
The Hague	Nancy	200
Antwerp	Nancy	700

- More details available at:
- http://www.ampl.com/cm/cs/what/ampl/NEW/tables.html