Session 5. Integer Programming

* Integer Programming (IP)

An *integer programming* problem is a linear programming (LP) in which some or all of the variables are required to be nonnegative integers.

- Pure integer programming problem
  General integer numbers
- Mixed integer programming problem
  Some of the variables are integer numbers
- 0-1 integer programming problem
  Zero-one binary numbers

* LP Relaxation of the IP

- The *feasible region* for any IP must be contained in the *feasible region* for the corresponding LP relaxation.
- Integrality gap
  $= LP(z^*)/IP(z^*)$ for a *maximization* problem
  $= IP(z^*)/LP(z^*)$ for a *minimization* problem
- Why not rounding off to the nearest integer?
  It may lead to the non-optimal or infeasible solution!

* IP Solution Algorithm

- Branch-and-bound method
- Cutting plane algorithm

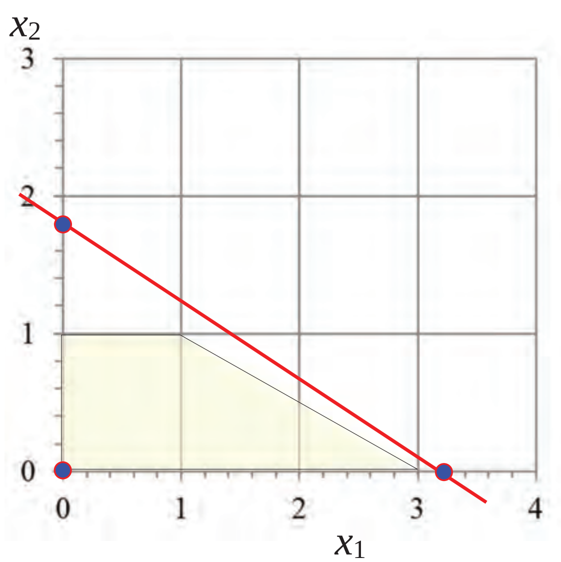
Much more *difficult* to find the optimal solution than LP problems!

© Young H. Chun

A. LP Relaxation

Ex 1] Sub-optimal solution

Max $z=11x_1+20x_2 \quad 180x_1+320x_2 \le 576;\ x_i \ge 0$ and integer



▪ LP solution and round-off

(x_1, x_2)	z
(0, 0)	0
(3.2, 0)	35.2
(0, 1.8)	**36**

(x_1=0, x_2 = 1) with z^*=20?

▪ Integer solution

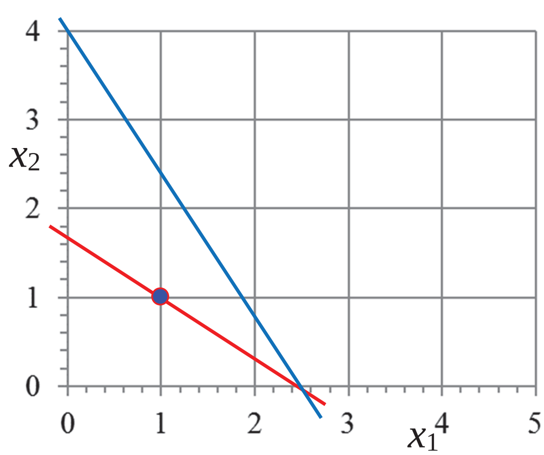
(x_1, x_2)	z
(0, 0)	0
(3, 0)	**33**
(1, 1)	31
(0, 1)	20

(x_1=3, x_2=0) with z^*=33!

Ex 2] Infeasible solution

Max $z = 4\,x_1 + 1\,x_2$

$1.6\,x_1 + 1\,x_2 \le 4;\ \ 2\,x_1 + 3\,x_2 = 5;\ \ x_i \ge 0$ and integer



▪ LP solution and round-off

(x_1, x_2)	z
(2.5, 0)	**10**
(0, 1.66)	1.66

(x_1=2, x_2=0) is *infeasible*!

▪ Integer solution

(x_1, x_2)	z
(1, 1)	5

(x_1=1, x_2=1) with z^*=5!

Ex 3] LP Relaxation

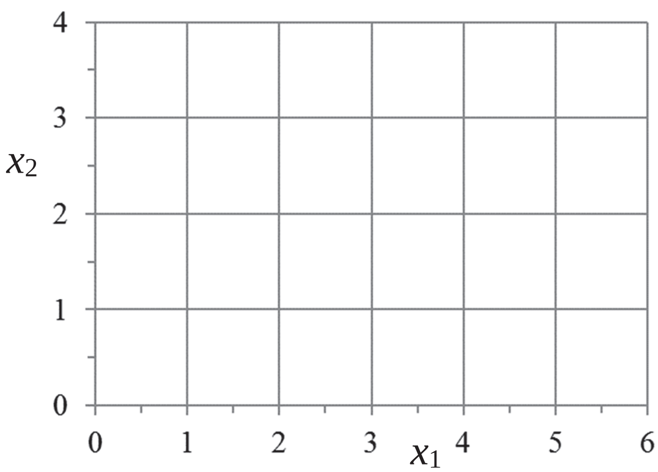
A furniture maker in Houma has 5 units of wood and 28 hours of free time, in which he will make decorative screens. He estimates that model I requires 1 unit of wood and 8 hours of time, while model II requires 2 units of wood and 7 hours of time. The prices of the models are \$80 and \$120, respectively. How many screens of each model should the furniture maker assemble if he wishes to *maximize* his sales revenue?

- Variables

  z =

  x_1 =

  x_2 =



- Objective function

- Constraints

(a) LP solution

(x_1, x_2)	z

(b) IP solution

(x_1, x_2)	z

Round-off: (2, 1) with z=280

Optimal: (1, 2) with z=320

B. IP Formulations and Applications

Ex 1] 0-1 Knapsack Problem

Josh Camper is going on an overnight hike. There are four items Josh is considering taking along on the trip. The weight of each item and the benefit Josh feels he would obtain from each item are listed as follows:

Item, i	Benefit	Weight (kg)	Benefit/Weight
1	40	20	2.0
2	45	30	1.5
3	10	10	1.0
4	20	40	0.5

Suppose Josh's knapsack can hold up to 45 kg of items. Determine which items Josh should take to *maximize* the total benefit.

- Variables (zero-one binary variables)

  $x_i =$

- Objective function

- Constraints

- Solution: $x_i^* = (0, 1, 1, 0)$ and $z^* = 55$

Multi-dimensional knapsack problem (bin packing problem)?

© Young H. Chun

Ex 2] Work Scheduling Problem

The Pontchartrain Tollway Authority has the following minimal daily requirements for toll keepers:

Shift	1	2	3	4	5	6
Period	6~10	10~14	14~18	18~22	22~2	2~6
Minimum number Of toll keepers	8	6	8	7	5	3

Toll keepers report to the toll booths at the beginning of each period and work for 8 consecutive hours. The Authority wants to determine the *minimum* number of toll keepers to employ so that there will be sufficient number of personnel available for each period.

Formulate this problem as an IP.

- Variables

- Objective function

- Constraints

Ex 3] Set-Covering Problem

A *dink* (double income no kid) couple, Amy and Brad want to divide their main household chores between them so that each has *two* tasks but the total time they spend on household duties is kept to a *minimum*. Their efficiencies on these tasks differ, where the time each would need to perform the task is given by the following table.

Hours per week needed

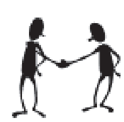
c_{ij}	Marketing	Cooking	Dishwashing	Laundering
Amy	3.2	7.4	4.1	2.5
Brad	3.9	6.8	4.5	2.7

Formulate an integer programming model for this problem.

- Variables

- Objective function

- Constraints



Ex 4] Location Analysis

Fire companies should be located in a least cost fashion, but subject to each demand point being within five minutes of a fire company. There are 5 demand points denoted by 1, 2, ..., 5 and 3 potential fire company sites denoted by A, B, and C.

For each demand point, we have tabulated below which sites are within 5 minutes of the demand point. A "o" in row *i*, column *j* indicates that site *j* is within 5 minutes of demand point *i*.

		Potential Fire Company Sites, *i*		
		A	B	C
Demand Points	1	o		o
	2		o	o
	3	o	o	
	4			o
	5	o	o	
Cost ($)		23	42	67

Formulate this problem as an integer program.

- Variables

- Objective function

- Constraints

© Young H. Chun

C. IP Formulations with Binary Variables

* Big *M* Method

Max $z = 1\,x_1 + 2\,x_2$

subject to

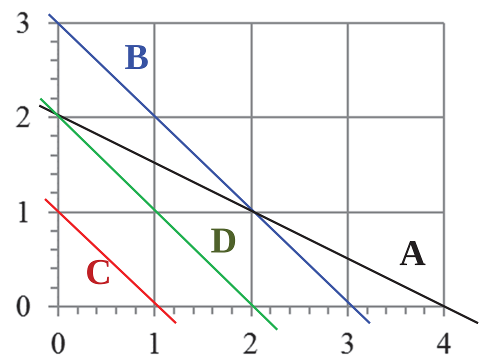
(**A**) $1\,x_1 + 2\,x_2 \le 4$

(**B**) $1\,x_1 + 1\,x_2 \le 3$

(**C**) $1\,x_1 + 1\,x_2 \ge 1$

(**D**) $1\,x_1 + 1\,x_2 = 2$

$x_1, x_2 \ge 0$



Let $\boldsymbol{y}$ = (0, 1) and $\boldsymbol{M}$ is a *very large* positive number.

- How to remove Constraint **B**?

(B) $1\,x_1 + 1\,x_2 \le 3 + M\,(1\text{-}y)$

If $y = 1$, then $1\,x_1 + 1\,x_2 \le 3$ => Keep it

If $y = 0$, then $1\,x_1 + 1\,x_2 \le 3 + M$ => Removed

- How to remove Constraint **C**?

(C) $1\,x_1 + 1\,x_2 + M\,(1\text{-}y) \ge 1$

If $y = 1$, then $1\,x_1 + 1\,x_2 \ge 1$ => Keep it

If $y = 0$, then $1\,x_1 + 1\,x_2 + M \ge 1$ => Removed

- How to remove Constraint **D**?

(D) $1\,x_1 + 1\,x_2 + M\,(1\text{-}y) \ge 2$

$1\,x_1 + 1\,x_2 \le 2 + M\,(1\text{-}y)$

If $y = 1$, then $1\,x_1 + 1\,x_2 \ge 2$ => Keep it

$1\,x_1 + 1\,x_2 \le 2$ => Keep it

If $y = 0$, then $1\,x_1 + 1\,x_2 + M \ge 2$ => Removed

$1\,x_1 + 1\,x_2 \le 2 + M$ => Removed

* Fixed-Charge Problem

If $x_i \leq 0$ then cost = 0; If $x_i > 0$, then cost = $b_i + a_i x_i$.

=> cost = $b_i \boldsymbol{y}_i + a_i x_i$

$x_i \leq \boldsymbol{M} \boldsymbol{y}_i$

* Either-Or Constraints

Choose either $f(x_1, x_2, \ldots, x_n) \leq b_1$ or $g(x_1, x_2, \ldots, x_n) \leq b_2$.

=> $f(x_1, x_2, \ldots, x_n) \leq b_1 + \boldsymbol{M}(1-\boldsymbol{y}_1)$

$g(x_1, x_2, \ldots, x_n) \leq b_2 + \boldsymbol{M}(1-\boldsymbol{y}_2)$

$\boldsymbol{y}_1 + \boldsymbol{y}_2 = 1$

=> $f(x_1, x_2, \ldots, x_n) \leq b_1 + \boldsymbol{M}\boldsymbol{y}$

$g(x_1, x_2, \ldots, x_n) \leq b_2 + \boldsymbol{M}(1-\boldsymbol{y})$

* *k* of Three Constraints

Choose k of the three constraints:

$f(x_1, x_2, \ldots, x_n) \leq b_1$, $g(x_1, x_2, \ldots, x_n) \leq b_2$, and

$h(x_1, x_2, \ldots, x_n) \leq b_3$

=> $f(x_1, x_2, \ldots, x_n) \leq b_1 + \boldsymbol{M}(1-\boldsymbol{y}_1)$

$g(x_1, x_2, \ldots, x_n) \leq b_2 + \boldsymbol{M}(1-\boldsymbol{y}_2)$

$h(x_1, x_2, \ldots, x_n) \leq b_3 + \boldsymbol{M}(1-\boldsymbol{y}_3)$

$\boldsymbol{y}_1 + \boldsymbol{y}_2 + \boldsymbol{y}_3 = k$

* If-Then Constraints

If $f(x_1, x_2, \ldots, x_n) \leq b_1$ is imposed, then $g(x_1, x_2, \ldots, x_n) \leq b_2$.

=> $f(x_1, x_2, \ldots, x_n) \leq b_1 + \boldsymbol{M}(1-\boldsymbol{y}_1)$

$g(x_1, x_2, \ldots, x_n) \leq b_2 + \boldsymbol{M}(1-\boldsymbol{y}_2)$

$\boldsymbol{y}_1 \leq \boldsymbol{y}_2$

© Young H. Chun

Ex 1] Fixed-Charge Problem

Vanka Cloth Company is capable of manufacturing three types of clothing: shirts, shorts, and pants. The manufacture of each type of clothing requires that Vanka has the appropriate type of machinery available. The machinery needed to manufacture each type of clothing must be rented at the following rates: Shirt machinery, $200 per week; shorts machinery, $150 per week; pants machinery, $100 per week.

The manufacture of each type of clothing also requires the amounts of *cloth* and *labor* given in Table. Each week, 160 square yards of cloth and 150 hours of labor are available. The variable unit cost and selling price for each type of clothing are also shown in Table. Formulate an IP whose solution will maximize Vanka's weekly profits.

	Labor (hours)	Cloth (sq yd)	Variable cost	Sales price
Shirt	3	4	$6	$12
Shorts	2	3	$4	$8
Pants	6	4	$8	$15

- Variables

- Objective function

- Constraints

Ex 2] Either-Or Constraints

Hyundai Auto is considering manufacturing three types of autos: compact, midsize, and large. The resources required for, and the profits yielded by, each type of car are shown in Table. At present, 6,000 tons of steel and 60,000 hours of labor are available. In order for production of a type of car to be economically feasible, at least 1,000 cars of that type must be produced.

Formulate an IP to *maximize* Hyundai's profit.

	Compact	Midsize	Large	Resource
Steel required	1.5 tons	3 tons	5 tons	6,000
Labor required	30 hours	25 hours	40 hours	60,000
Profit yielded	\$2,000	\$3,000	\$4,000	

- Variables

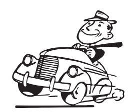
- Objective function

- Constraints

D. Integer Optimization for Network Models*

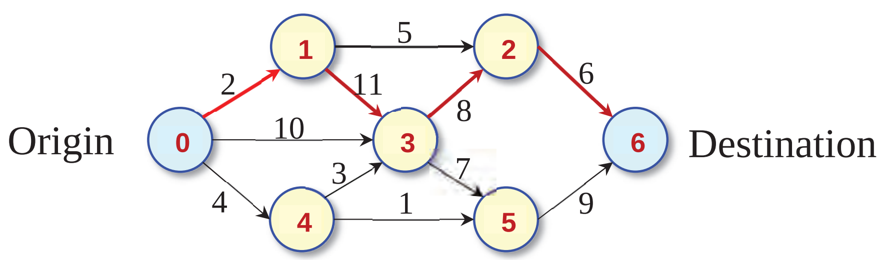
* Network Models

- Network models are applicable to a variety of decision problems that can be modeled as *network optimization* problems and solved efficiently and effectively.
- Some of these problems are really *physical* problems such as transportation or flow of commodities. Other problems are more of an abstract representation of processes or activities.
- The family of network optimization problems includes: assignment, critical path, max flow, shortest path, transportation, and traveling salesman problems.



* Graph Theory, $G = \{V, E\}$

- **Node** (*vertex*): In a transportation network, these might be locations or cities on a map.
- **Arc** (*edge*): It may be either *directed* or *undirected*. In a transportation network, the *arcs* might be roads, or navigation channels in rivers.



- A network with n nodes has as many as $n(n\text{-}1)/2$ arcs. If *directed*, this number might be doubled. This large number of possible arcs is one of the reasons why there are special solution algorithms for special types of network problems.

* Case 1. Transportation Network

Tiger Airlines must decide on the amounts of jet fuel to purchase from the 3 oil companies, which can furnish up to the following amounts of fuel during the coming month:

Oil company, i	1	2	3
Maximum supply, s_i	1,500	2,500	6,000

The airline refuels aircraft at the 4 airports it serves. The minimum required amounts of jet fuel at each airport are:

Airport, j	1	2	3	4
Minimum demand, d_j	1,800	3,100	1,200	3,900

When transportation costs are added to the price per gallon supplied, the combined cost per gallon is

c_{ij}		Airport			
		1	2	3	4
Oil company	1	12	11	11	11
	2	9	10	8	13
	3	15	12	12	14

Formulate a LP to determine a supply plan for Tiger Airlines.

More efficient algorithms are available for a *balanced* transportation problem!

* Case 2. Assignment Problem

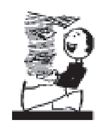
c_{ij}		Client, j		
		1	2	3
Project leader, i	1	10	16	32
	2	14	22	40
	3	22	24	34

Chris and Associates, Inc., is an accounting firm that has new clients. Project leaders will be assigned to the three clients. Based on the different backgrounds and experiences of the leaders, the various leader-client assignments differ in terms of projected completion times. The possible assignments and the estimated completion times in days are given in the table.

Formulate the problem as an integer program and solve it.

- Variables

- Objective function

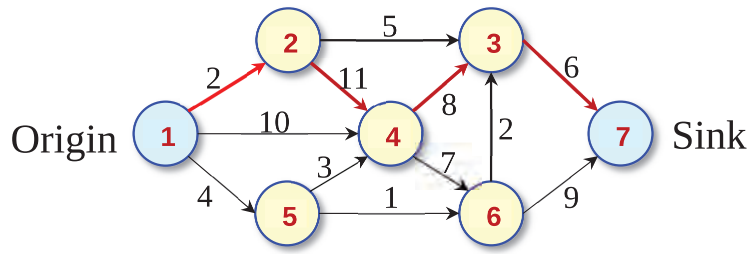


- Constraints

The *Hungarian method* is more efficient!

* Case 3. Shortest-Path Problem

Find the shortest path from node 1 to node 6 in the network:

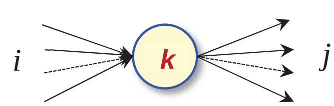


Given a shortest path problem with cost c_{ij}, node 0 is the origin and node 6 is the sink (or destination).

- Variables

- Objective function

- Constraints



Arrive from *i* Leave for *j*

Dynamic programing is more efficient computationally.

* Case 4. Traveling Salesperson Problem (TSP)

Given a list of n cities and the distances c_{ij} between each pair of cities, what is the shortest possible route that visits each city and *returns* to the origin city?

- Decision variables

x_{ij} = 1 if city j is visited immediately after city i
0 otherwise

- Objective function: Minimize the total distance traveled

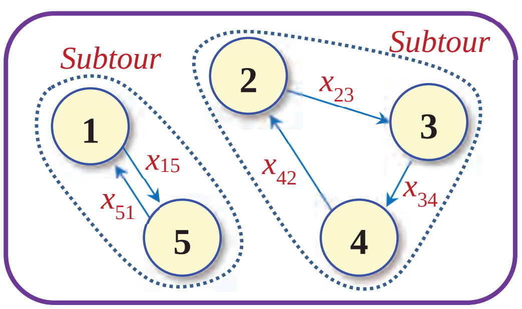
$$Min\ \ z = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \text{ for } i \neq j$$

- Assignment constraints

$\sum_{j=1}^{n} x_{ij} = 1$ for i = 1, 2, .., n (row sum = 1)

$\sum_{i=1}^{n} x_{ij} = 1$ for j = 1, 2, .., n (column sum = 1)

- In addition, we need constraints for sub-tour prevention!



x_{ij}	1	2	3	4	5	Total
1					1	1
2			1			1
3				1		1
4		1				1
5	1					1
Total	1	1	1	1	1	n=5

There should be only a single tour covering all cities, and not two or more *disjointed* sub-tours that only collectively cover all cities.

* Constraints for Sub-tour Elimination in TSP

- **Method 1**: *Miller-Tucker-Zemlin* formulation

Add the following sub-tour prevention constraints:

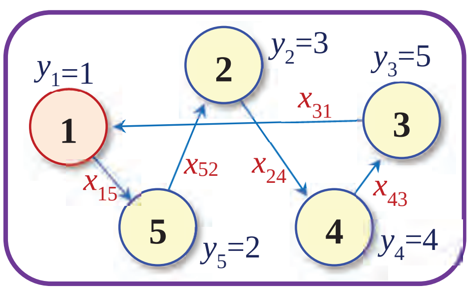
$$y_j \geq y_i + 1 + n\,(x_{ij} - 1) \qquad \text{for } 2 \leq i \text{ and } j \leq n \text{ and } y_1 = 1$$

where the additional *decision variable* y_i is the "position" of city i in the tour. (i.e., $y_1=1$ and $2 \leq y_i \leq n$ for $i \geq 2$)

If $x_{ij} = 1$, then $y_j \geq y_i + 1$ — y_j must be greater than y_i

If $x_{ij} = 0$, then $y_j \geq y_i - (n-1)$ — No restriction on y_j

Ex 1] Sub-tour prevention constraints for TSP with n=5



y_i	1	3	5	4	2	
x_{ij}	1	2	3	4	5	Total
1					1	1
2				1		1
3	1					1
4			1			1
5		1				1
Total	1	1	1	1	1	

$y_1=1$

$$y_2 - y_2 + 1 + 5\,(x_{22} - 1) \leq 0 \qquad y_3 - y_2 + 1 + 5\,(x_{32} - 1) \leq 0$$
$$y_2 - y_3 + 1 + 5\,(x_{23} - 1) \leq 0 \qquad y_3 - y_3 + 1 + 5\,(x_{33} - 1) \leq 0$$
$$y_2 - y_4 + 1 + 5\,(x_{24} - 1) \leq 0 \qquad y_3 - y_4 + 1 + 5\,(x_{34} - 1) \leq 0$$
$$y_2 - y_5 + 1 + 5\,(x_{25} - 1) \leq 0 \qquad y_3 - y_5 + 1 + 5\,(x_{35} - 1) \leq 0$$

$$y_4 - y_2 + 1 + 5\,(x_{42} - 1) \leq 0 \qquad y_5 - y_2 + 1 + 5\,(x_{52} - 1) \leq 0$$
$$y_4 - y_3 + 1 + 5\,(x_{43} - 1) \leq 0 \qquad y_5 - y_3 + 1 + 5\,(x_{53} - 1) \leq 0$$
$$y_4 - y_4 + 1 + 5\,(x_{44} - 1) \leq 0 \qquad y_5 - y_4 + 1 + 5\,(x_{54} - 1) \leq 0$$
$$y_4 - y_5 + 1 + 5\,(x_{45} - 1) \leq 0 \qquad y_5 - y_5 + 1 + 5\,(x_{55} - 1) \leq 0$$

and $2 \leq y_i \leq 5$ for i = 2, 3, 4, and 5.

© Young H. Chun

▪ **Method 2**: *Dantzig-Fulkerson-Johnson* formulation

- Step 1: Solve the IP with the assignment constraints only.

- Step 2: If there is no sub-tour in the solution, then that is the optimal solution. Stop!

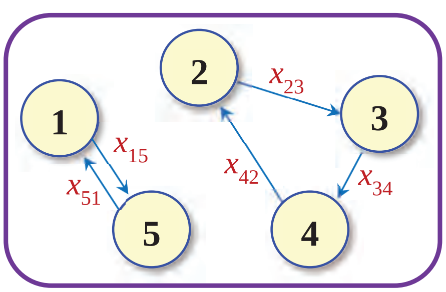
- Step 3: If we find a sub-tour Q which is the subset of m cities where $2 \le m \le n$-1, then add the following constraints to eliminate the sub-tour Q:

$$\sum_{i \in Q} \sum_{j \in Q} x_{ij} \le m - 1$$

- Step 4: Solve the IP with the sub-tour breaking constraint as well as the assignment constraints. Go to Step 2.

Note that the possible number of sub-tours Q is 2^n-2, which is too many for a large n. That is why we add the sub-tour breaking constraints as *lazy constraints*. That is, we generate and add these constraints to our IP model, in a *lazy* fashion, as needed.

Ex 2] Suppose that the *initial* IP solution has two sub-tours, Q_1={1, 5} and Q_2={2, 3, 4}, as shown below. Generate the corresponding *lazy constraints* to break each sub-tour.



- Break the subtour Q_1={1, 5}

- Break the subtour Q_2={2, 3, 4}

We only need to add *one* of the two lazy constraints to the IP model and re-run it. (If Q_1 is broken, Q_2 is also broken!)

* Exact Algorithms for TSP

▪ Complete enumeration (Brute-force search)

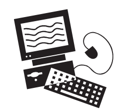
All possible paths are considered and the path of *least* cost is the optimal solution (n! possible paths!).

▪ Integer programming, dynamic programming, branch-and-bound, cutting-plane method

* Heuristic Algorithms for TSP

▪ Nearest neighbor (*NN*) algorithm

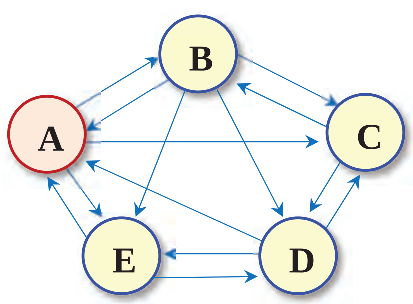
Starting from city 1, the next city is simply the closest city that has not yet been visited.



▪ Greedy algorithm

A tour is constructed by repeatedly selecting the *shortest edge* that does not create a sub-tour.

Ex 3] Traveling salesperson problem with n=5



c_{ij}	A	B	C	D	E	Total
A	99	3	9	99	13	
B	3	99	4	5	9	
C	99	1	99	7	99	
D	4	99	7	99	3	
E	12	99	99	2	99	
Total						

▪ *Nearest neighbor* algorithm:

▪ *Greedy* algorithm:

▪ *Optimal* solution:

Appendix: SAS/OR for Integer Programming

* OPTMODEL

The OPTMODEL procedure provides a framework for specifying and solving mixed integer linear programs (MILPs).

Ex 1] General integer problem

$$\text{Min } z = 2x_1 - 3x_2 - 4x_3$$

Subject to

$$-2x_2 - 3x_3 \geq -5 \quad \text{(R1)}$$
$$x_1 + x_2 + 2x_3 \leq 4 \quad \text{(R2)}$$
$$x_1 + 2x_2 + 3x_3 \leq 7 \quad \text{(R3)}$$
$$x_i \geq 0 \text{ and integers}$$

- SAS input

```
proc optmodel;
   var x{1..3} >= 0 integer;

   min f = 2*x[1] - 3*x[2] - 4*x[3];

   con r1: -2*x[2] - 3*x[3] >= -5;
   con r2: x[1] + x[2] + 2*x[3] <= 4;
   con r3: x[1] + 2*x[2] + 3*x[3] <= 7;

   solve with milp / presolver = automatic
         heuristics = automatic;
   print x;
quit;
```

- SAS solutions

$x_1 = 0, \ x_2 = 1, \ x_3 = 1$

Ex 2] Transportation problem:

From \ To	Boston	New York	Supply
Detroit	$30	$20	200
Pittsburgh	$40	$20	100
Demand	150	150	

- **LP model**

- **SAS Input**

```
proc optmodel;

   /* specify parameters */
   set O={'Detroit','Pittsburgh'};
   set D={'Boston','New York'};
   number c{O,D}=[30 20
                  40 10];
   number a{O}=[200 100];
   number b{D}=[150 150];

   /* model description */
   var x{O,D} >= 0;
   min total_cost = sum{i in O, j in D}c[i,j]*x[i,j];
   constraint supply{i in O}: sum{j in D}x[i,j]=a[i];
   constraint demand{j in D}: sum{i in O}x[i,j]=b[j];

   /* solve and output */
   solve;
   print x;
```

- **SAS Output**

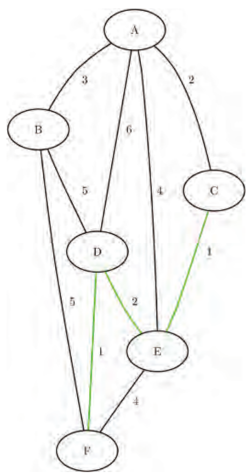
Solution Summary	
Solver	Dual Simplex
Objective Function	total_cost
Solution Status	Optimal
Objective Value	6500
Iterations	0
Primal Infeasibility	0
Dual Infeasibility	0
Bound Infeasibility	0

x		
	Boston	New York
Detroit	150	50
Pittsburgh	0	100

* OPTGRAPH

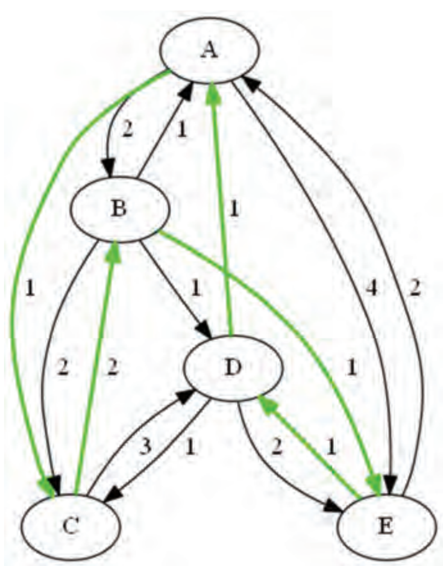
The OPTGRAPH procedure includes a number of graph theory, combinatorial optimization, and network analysis algorithms.

Ex 3] Shortest path problem




```
data LinkSetIn;
   input from $ to $ weight @@;
   datalines;
A B 3  A C 2  A D 6  A E 4
B D 5  B F 5
C E 1
D E 2  D F 1
E F 4
;
proc optgraph
   data_links     = LinkSetIn;
   shortpath
      source = C
      sink = F
      out_weights = ShortPathW
      out_paths   = ShortPathP;
run;
```

Ex 4] Traveling salesman problem of a *directed* graph




```
data LinkSetIn;
   input from $ to $ weight @@;
   datalines;
A B 2.0 A C 1.0 A E 4.0
B A 1.0 B C 2.0 B D 1.0 B E 1.0
C B 2.0 C D 3.0
D A 1.0 D C 1.0 D E 2.0
E A 2.0 E D 1.0
;
proc optgraph
   direction   = directed
   loglevel    = moderate
   data_links = LinkSetIn
   out_nodes   = NodeSetOut;
   tsp
      out      = TSPTour;
run;
%put &_OPTGRAPH_;
%put &_OPTGRAPH_TSP_;
```