IO543: Scheduling

Lecture 7

Brian Denton
Associate Professor, IOE

IOE Bld 2893
btdenton@umich.edu

Announcements

- Assignment 3 is up on ctools (due Feb 11)
- One more assignment before the midterm (due Feb 18)
- Midterm take home exam:
  - Pick up at start of class on Feb 23
  - Return no later than start of class Feb 25
  - You may use **Pinedo textbook and lecture slides and notes** for the exam, but **no other resources or communication** with anyone about the midterm

Today

- Wrap up single machine problems
  - $1||\Sigma\, w_j T_j$
  - $1\,|s_{jk}\,|\,C_{max}$

Total Weighted Tardiness $1||\Sigma\, w_j T_j$

Last lecture we talked about $1||\Sigma\, T_j$ which is a special case of $1||\Sigma\, w_j T_j$

<u>Theorem (3.6.2 Pinedo):</u> The problem $1||\Sigma\, w_j T_j$ is <u>strongly</u> NP-hard

<u>Proof:</u> By reduction of 3 partition to $1||\Sigma\, w_j T_j$

There is no known dynamic programming method to efficiently solve $1||\Sigma\, w_j T_j$

Branch and Bound for $1||\Sigma\, w_j T_j$

Branch-and-bound can be used to solve $1||\Sigma\, w_j T_j$. The tree is constructed starting with the <u>last job in the sequence</u>.

- At the jth level, branching occurs on n - j + 1 remaining jobs that are put in the $(n-j+1)^{th}$ position

- The following property can be used to **eliminate nodes**

<u>Lemma (Pinedo 3.6.1):</u> If there are two jobs *j* and *k* with $d_j \leq d_k, \mathrm{p_j} \leq p_k$ and $w_j \geq w_k$, then there is an optimal sequence in which job *j* appears before job *k*.

Branch and Bound for $1||\Sigma\, w_j T_j$

A bounding method for fathoming nodes in the binary tree is based on relaxing $1||\Sigma\, w_j T_j$ to a **transportation problem**

x_{jk} is a **binary decision variable** that is 1 if one unit of job j is processed in time slot k, and 0 otherwise

Define cost coefficients c_{jk} such that

$$c_{jk} = \begin{cases} 0, & if\ k \leq d_j \\ w_j, & if\ k > d_j \end{cases}$$

For $j = 1, \ldots, n$ and $\mathrm{k} = 1, \ldots, \mathrm{C}_{\max}$.

B&B for $1||\Sigma\, w_j T_j$

Transportation Problem Formulation:

$$\min \sum_{j=1}^{n} \sum_{k=1}^{C_{max}} c_{jk} x_{jk}$$

s.t. $\sum_{k=1}^{C_{max}} x_{jk} = p_j\,,$ $\quad j = 1, \dots, n$

$\sum_{j=1}^{n} x_{jk} = 1,$ $\quad k = 1, \dots, C_{max}$

$x_{jk} \geq 0,$ $\quad j = 1, \dots, n, k = 1, \dots, C_{max}$

This problem can be solved very efficiently via *transportation simplex* method.

Q. Why does this give a lower bound on $1 \mid \mid \sum w_j T_j$?

Example

- Read Section 3.6 of Pinedo
- Work through Example 3.6.3 of Pinedo

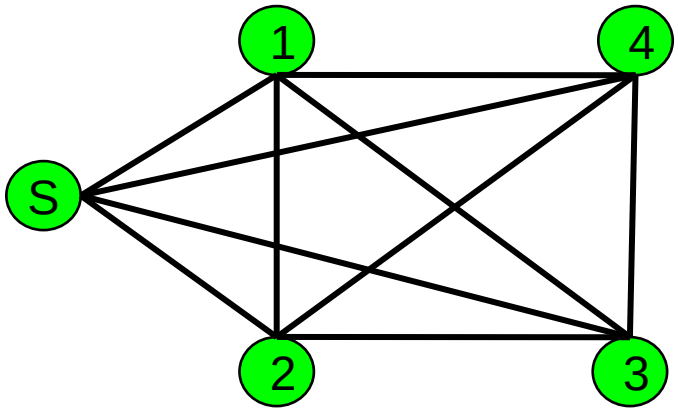
Sequence Dependent Setups

- Many processes require a **setup** between jobs
  - Retooling
  - Preperation
  - Cleaning

- $1 \left| s_{jk} \right| C_{max}$: minimize makespan with setups

Travelling Salesperson Problem

$1 \mid s_{jk} \mid C_{max}$ is equivalent to the travelling salesperson problem (TSP) which can be described as follows: find the tour through a set of locations that minimizes the total distance travelled



$$x_{ij} = \begin{cases} 1 & \text{if salesperson leaves city } i \text{ and travels next to city } j \\ 0 & \text{otherwise} \end{cases}$$

$$c_{ij} = \text{distance between cities } i \text{ and } j \text{ if } i \neq j$$

$$c_{ii} = M, \text{ where } M \text{ is a large positive number}$$

Greedy Heuristic

Large problems often cannot be solved exactly

A simple **greedy heuristic:**

Step 1: Find the shortest distance between any two nodes and arbitrarily select one as the starting node.

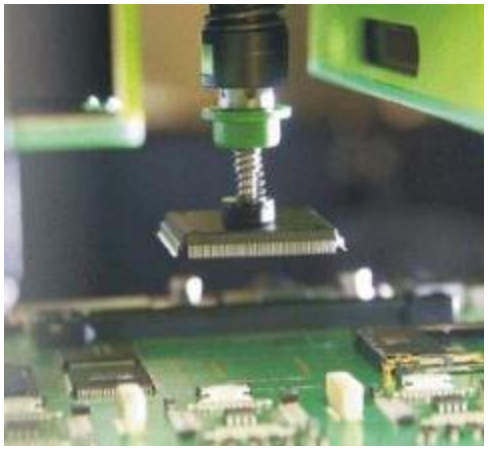
Step 2: Choose the nearest node from the list of unvisited nodes.

Step 3: If all nodes have been visited STOP; Otherwise go to Step 2.

Example: Automation

A manufacturer of printed circuit boards uses programmable drill machines to drill six holes in each 10x5 cm board. The x and y coordinates of each hole are given below. The time it takes to move from one hole to the next is proportional to the distance between holes. What drilling sequence minimizes the total time?

x	y	Location
1	2	1
3	1	2
5	3	3
7	2	4
8	3	5



Exercise: Use the greedy heuristic to find a solution to this problem.

TSP Formulation

The assignment model can be used to solve a TSP:

$$\text{minimize:} \quad \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} x_{ij}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} x_{ij} = 1 \quad 1 \le i \le n$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad 1 \le j \le n$$

c_{ij} = distance from city i to city j

x_{ij} = 1 if tour visits i then j, and 0 otherwise (binary)

Example: Automation

Distance Matrix:

	1	2	3	4	5
1	0.00	2.24	4.12	6.00	7.07
2	2.24	0.00	2.83	4.12	5.39
3	4.12	2.83	0.00	2.24	3.00
4	6.00	4.12	2.24	0.00	1.41
5	7.07	5.39	3.00	1.41	0.00

Formulation:

min 1000x11 + 2.24x12 + 4.12x13 + 6x14 + 7.07x15 +2.24x21 + 1000x22 + 2.83x23 + 4.12x24 + 5.38x25 +4.12x31 + 2.83x32 + 1000x33 + 2.23x34 + 3x35 +6x41 + 4.12x42 + 2.24x43 + 1000x44 + 1.41x45 +7.07x51 + 5.39x52 + 3x53 + 1.41x54 + 1000x55

s.t.

x11 + x12 + x13 + x14 + x15 = 1
x21 + x22 + x23 + x24 + x25 = 1
x31 + x32 + x33 + x34 + x35 = 1
x41 + x42 + x43 + x44 + x45 = 1
x51 + x52 + x53 + x54 + x55 = 1
x11 + x21 + x31 + x41 + x51 = 1
x12 + x22 + x32 + x42 + x52 = 1
x13 + x23 + x33 + x43 + x53 = 1
x14 + x24 + x34 + x44 + x54 = 1
x15 + x25 + x35 + x45 + x55 = 1
Xij binary for all i and all j

AMPL Code

tsp.mod

```
# The assignment problem.
param N integer > 2; # Number of nodes
set Nodes ordered := {1..N};
set Arcs := {i in Nodes, j in Nodes: i<>j}; # Complete graph except degenerate arcs.
param xcoord {Nodes};
param ycoord {Nodes};

# Use this definition for Euclidean models, where the data gives point coordinates.
param length {(i,j) in Arcs} := sqrt((xcoord[i] - xcoord[j])^2 + (ycoord[i] - ycoord[j])^2);

# Use this definition for asymmetric models, where the data gives length explicitly.
# param length {(i,j) in Arcs};
var x {(i,j) in Arcs} >= 0;
minimize Tourlength: sum {(i,j) in Arcs} length[i,j] * x[i,j];
subject to DegreesIn {i in Nodes}: sum {(i,j) in Arcs} x[i,j] = 1;
subject to DegreesOut {j in Nodes}: sum {(i,j) in Arcs} x[i,j] = 1;
```

AMPL Code

tsp.dat

```
data;

param N := 5;

#set Nodes := pos1, pos2, pos3, pos4, pos5;

param:    xcoord   ycoord  :=
  1       1    2
  2       3    1
  3       5    3
  4       7    2
  5       8    3 ;
```

Example: Automation

Optimal Solution from AMPL →

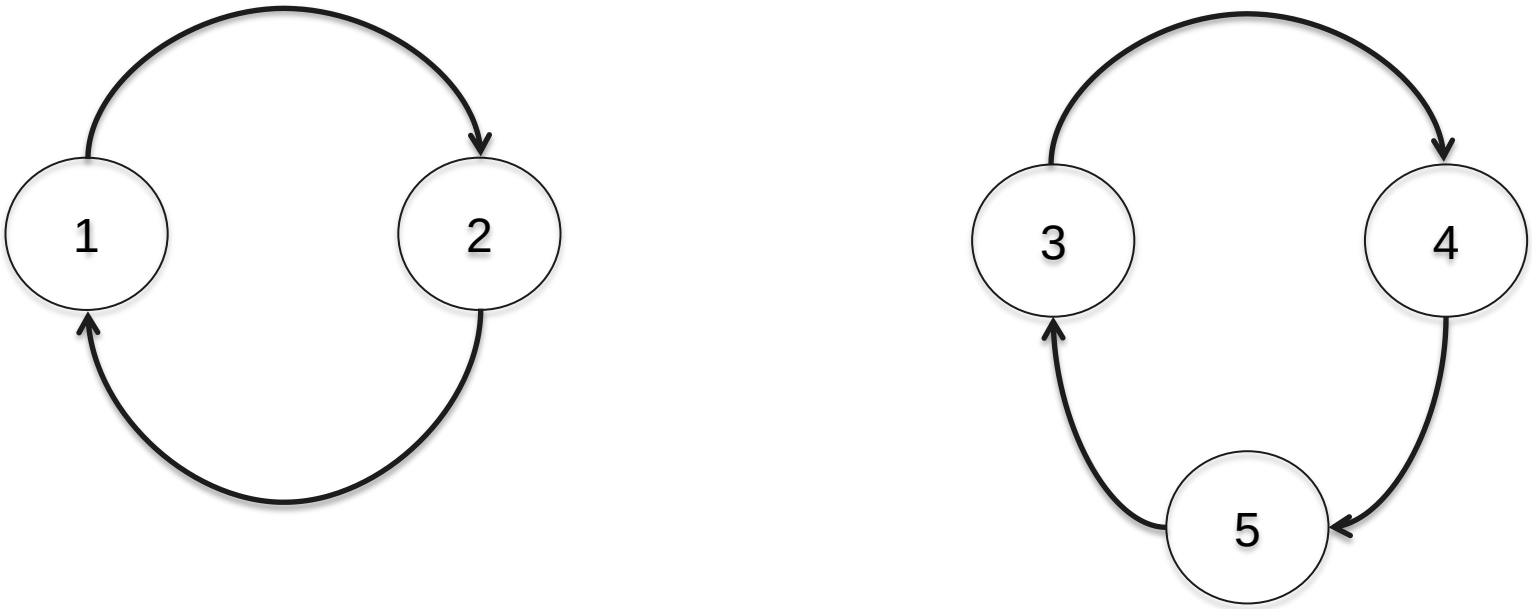
x12 = 1
x13 = 0
x14 = 0
x15 = 0
x21 = 1
x23 = 0
x24 = 0
x25 = 0
x31 = 0
x32 = 0
x34 = 1
x35 = 0
x41 = 0
x42 = 0
x43 = 0
x45 = 1
x51 = 0
x52 = 0
x53 = 1
x54 = 0

Construct a figure illustrating the sequence of positions for the drilling process

Objective = 11.224

Example: Automation

Problem: Optimal Solution from AMPL gives **subtours**



Solution: Add **Subtour Elimination** Constraints:

$$x_{12} + x_{21} \leq 1$$

$$x_{34} + x_{45} + x_{53} \leq 2$$

Example: Automation

Iteration 2:

Add constraints:

$$x_{12} + x_{21} \leq 1$$

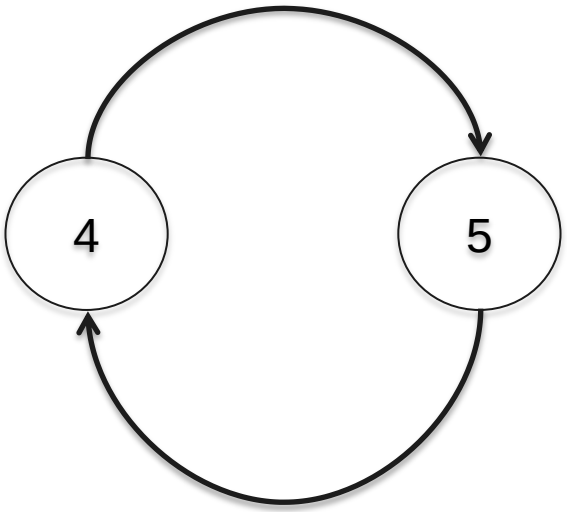
$$x_{34} + x_{45} + x_{53} \leq 2$$

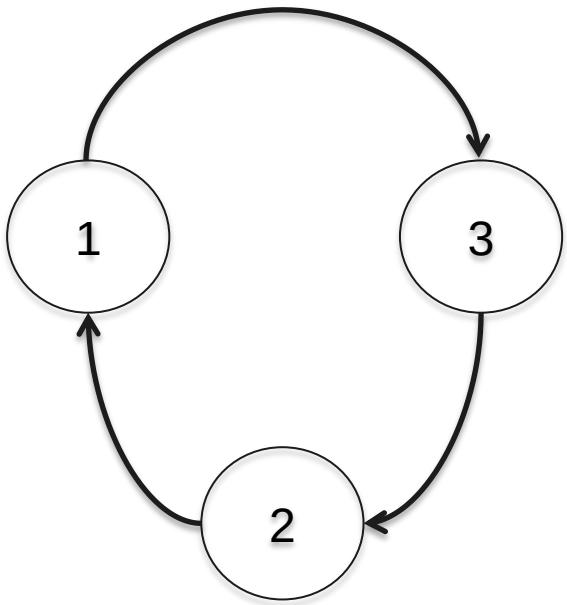
Objective = 12.02

x12 = 0
x13 = 1
x14 = 0
x15 = 0
x21 = 1
x23 = 0
x24 = 0
x25 = 0
x31 = 0
x32 = 1
x34 = 0
x35 = 0
x41 = 0
x42 = 0
x43 = 0
x45 = 1
x51 = 0
x52 = 0
x53 = 0
x54 = 1

Example: Automation

Optimal Solution from AMPL still gives **subtours**





Add more Subtour Elimination Constraints:

$$x_{45} + x_{54} \leq 1$$

$$x_{13} + x_{32} + x_{21} \leq 2$$

Example: Automation

Iteration 2:

Add constraints:

$$x_{34} + x_{43} \leq 1$$

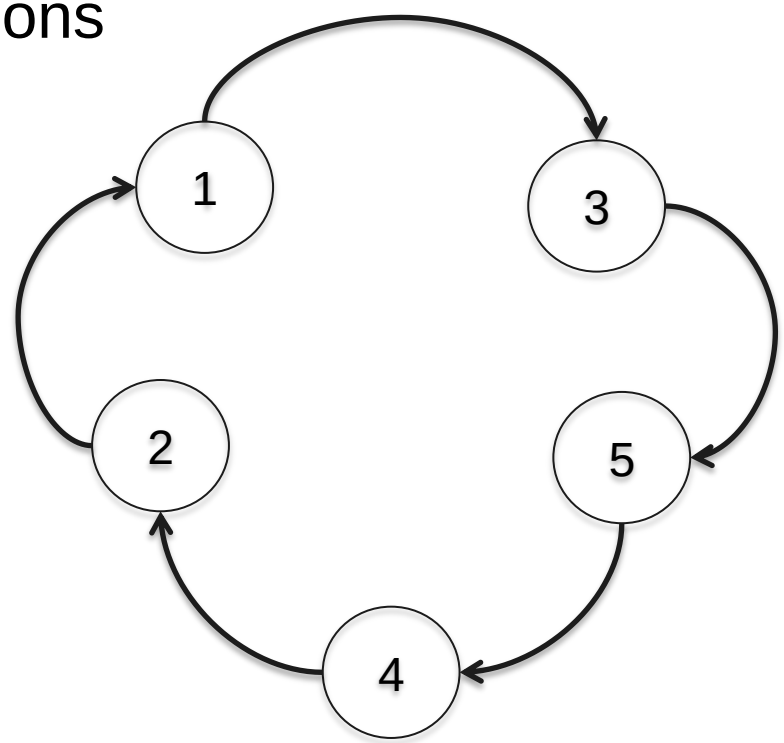
$$x_{15} + x_{52} + x_{21} \leq 2$$

Objective = 14.90

x12 = 0
x13 = 1
x14 = 0
x15 = 0
x21 = 1
x23 = 0
x24 = 0
x25 = 0
x31 = 0
x32 = 0
x34 = 0
x35 = 1
x41 = 0
x42 = 1
x43 = 0
x45 = 0
x51 = 0
x52 = 0
x53 = 0
x54 = 1

Example: Automation

Optimal Solution from AMPL (finally) gives a feasibly tour for the drill positions



Total distance travelled is 14.90cm.

Example: Automation

The formulation can be fixed by adding all possible **subtour elimination** constraints:

$$\sum_{i,j \in S} x_{ij} \leq |S| - 1, \text{ for every subset } S.$$

Unfortunately there are 2^n of these

Fortunately adding these constraints incrementally often results in a feasible solution in much less than 2^n iterations

TSP Alternative Formulation

Variables: x_{ij} = 1 if we move from node i to node j, else 0

u_i = number of nodes visited at city i

Min total cost: $\min \sum_i \sum_j c_{ij} x_{ij}$,

Enter each node once: $\sum_i x_{ij} = 1$ for all j.

Leave each node once: $\sum_j x_{ij} = 1$ for all i.

Subtour breaking: $u_i + 1 \leq u_j + n(1 - x_{ij})$, for $i = 2, \ldots, n$, $i \neq j$, $j = 2,\ldots,n$,

$x_{ij} \in \{0, 1\}$ for all i, j; $u_i \geq 0$ for all i.

This does not work will for large problem instances due to poor LP relaxation

Next Time

Next class we will cover:

Problems with multiple objectives: Sections 4.1-4.3 of Pinedo

Introduction to Parallel Machine Models: Sections 5.1 and 5.2 of Pinedo