# IO543: Scheduling

## Lecture 21

Brian Denton
Associate Professor, IOE

IOE Bld 2893
btdenton@umich.edu

# Project Schedule



**April 12:**

Semiconductor Manufacturing: Yixing Lu, Yuzhi Wan, Xue Huang

Scheduling Commercials on TV: David Sachs, Bevin Mathew

Learning Effects in Scheduling: Ling Ken, Chao Xia, Wensi Wang and Chengwei Zhai

Ship Production Scheduling: Lauren Claus, Daniel Kilcullen, Seng-Hsiang Lin

Patient Scheduling in Clinics: Armando Bernal, Victor Fuentes, Seok Joo Kwak, Jamie Yap

Job Shop Problems with Earliness and Lateness Objective: Wenbo Sun, Qi He

# Project Schedule



**April 14**

Uber Dispatching: Zhengyang Wun, Ji Wu, Xiaoying Qing

Scheduling in the Olympics: Christine Abney, Amada Bayagich, Eric Buchsbaum, Lauren Fitzpatrick

Nurse Scheduling:  Kelly Ogiesoba, Matt Rouhana, Joe Scherping

Baseball Scheduling: Maria Lopez, Anthony Sciuto, Alejandro Vigo

Medical Procedure Scheduling: Esmaeil Keyvanshokooh, Ece Sanci, Karmel Shehadeh

The South U Dilemma: Olivia Melendez, Jacob Villareal, Mitch McKinstry

# Class Schedule

See the complete schedule on Ctools:

| | | | |
|---|---|---|---|
| 13 | Mar 29, Mar 31 | Stochastic Models | Chapter 9 – selected topics |
| 14 | Apr 5, 7 | Stochastic Models | Chapter 10 – selected topics |
| 15 | Apr 12, 14 | Class Presentations on Applications | |
| 15 | Apr 27 | Final Exam, 10:30-12:30 | |

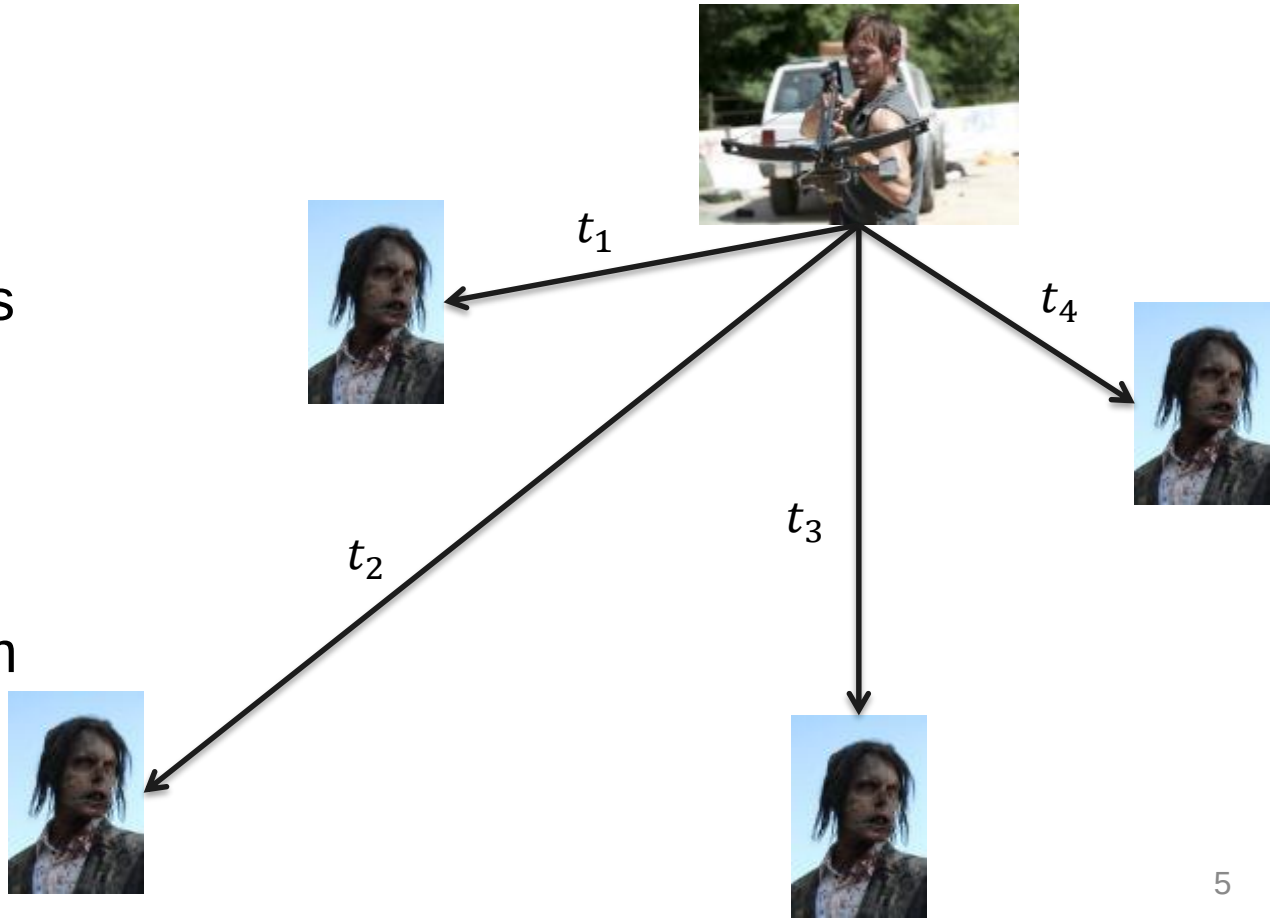
# Zombie Scheduling

Consider the following instance of a zombie scheduling problem:

Zombies move at different speeds

It takes 15 seconds to reload and fire the crossbow

Classify this as a scheduling problem



# $1\,|appt, seq\,|(c^w \sum w_j + c^s \sum s_j)$



The problem $1\,|appt, seq\,|(c^w \sum w_j + c^s \sum s_j)$ is an extension of $1\,|appt\,|(c^w \sum w_j + c^s \sum s_j)$ that considers the additional decision of **how to sequence customers**

Two perspectives on sequencing

- Customers perspective is to find the ideal position in the sequence to minimize waiting
- Managers perspective is to find the best sequence to minimize the cost of <u>waiting and idling</u>

# $1\,|appt, seq\,|(c^w \sum w_j + c^s \sum s_j)$



$1\,|appt, seq\,|(c^w \sum w_j + c^s \sum s_j)$ extremely difficult to solve exactly. Following is a common heuristic:

**Smallest Variance First (SVF) Rule:**

**Step 1:** Compute $Var(Z_j)$ for all customers $= 1, \ldots, n$.

**Step 2:** Sequence customers in nondecreasing order of $Var(Z_j)$.

**Step 3:** Solve $1\,|appt\,|(c^w \sum w_j + c^s \sum s_j)$ to find the appointment times conditional on SVF sequence.

# New Topic!



## Stochastic Parallel Machine Scheduling:

## 3 Different Environments:

- Advance job-to-machine allocations
  - Jobs allocated to machines before processing begins
- Static list <u>policy</u>
  - Jobs sequenced in a list before processing begins, and allocated to the first available machine
- Dynamic list <u>policy</u>
  - Every time the machine is freed the decision maker decides which job to start

# *Advance* Allocation: $Pm \mid \mid \sum C_j$

If job-to-machine allocation decisions are made before the first job starts processing then $Pm \mid \mid \sum C_j$ can be solved easily:

**Theorem:** If job-to-machine allocation decisions are made before random processing times are observed then the SEPT rule is optimal for the stochastic counterpart of $Pm \mid \mid \sum C_j$

**Proof:** The proof is identical to deterministic $Pm \mid \mid \sum C_j$ since the objective is the expectation of the sum of random processing times.

# Advance allocation: $Pm \mid \mid C_{max}$

If job-to-machine allocation decisions are made in advance then $Pm \mid \mid C_{max}$ can be formulated as a **stochastic program**

$$x_{ij} = \begin{cases} 1 \text{ if job j is assigned to machine i} \\ 0 \text{ otherwise} \end{cases}$$

$C_i\ (\omega) =$ completion time of jobs on machine i under scenario $\omega$

$C_{max}\ (\omega) =$ max completion time of jobs on machine i under scenario $\omega$

$$\min E_\omega[\, C_{max}\ (\omega)]$$

$$\text{s.t.} \quad \sum_i x_{ij} = 1, \qquad for\ all\ j$$

$$C_{max}(\omega) \geq C_i(\omega), \qquad for\ all\ i\ and\ \omega$$

$$C_i(\omega) = \sum_j Z_j(\omega) x_{ij}\,, \quad forall\ i\ and\ \omega$$

$$C_i(\omega) \geq 0,\ x_{ij}\ binary,\ \ for\ all\ i, j, \omega$$

# Static list policy: $Pm \mid \mid C_{max}$

For the special case of $P2 \mid \mid C_{max}$ the following can be proven:

**Theorem:** If there are two machines in parallel and the processing times are exponentially distributed, then the longest expected processing time first (LEPT) rule minimizes the expected makespan.

**Proof**: See Pinedo, pages 322 – 326.

<u>Note:</u> the stochastic counterpart of $P2 \mid \mid C_{max}$ with exponential processing times is <u>not NP-Hard</u>.

# Example: Static List Policy $Pm \mid \mid C_{max}$

**Example:** Find the static list policy for the following instance of stochastic $P3 \mid \mid C_{max}$ with independent exponentially distributed processing times:

| Job | $\lambda_j$ |
|---|---|
| 1 | 2.5 |
| 2 | 4.0 |
| 3 | 1.0 |
| 4 | 1.5 |
| 5 | 5.0 |
| 6 | 2.0 |

?

Note: This is not necessarily optimal for P3, but it often works well as a heuristic

# Static list policy: $Pm \mid \mid \sum C_j$

The following <u>static list policy</u> is optimal for $Pm \mid \mid \sum C_j$:

**Theorem:** Given n jobs with processing times such $E[Z_1] \leq E[Z_2] \leq \cdots \leq E[Z_n]$ then shortest expected processing time first (SEPT) rule minimizes total expected completion time under a static list policy.

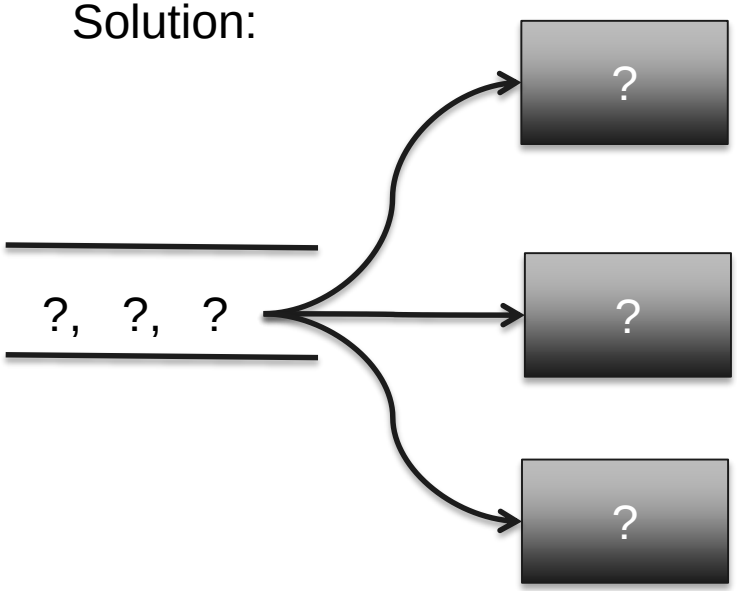
Proof: follows from the deterministic case since $\sum C_j$ is the sum of random processing times.

# Example: $Pm \mid \mid \sum C_j$

**Example:** Find the optimal static list policy for the following instance of stochastic $P3 \mid \mid \sum C_j$ with independent exponentially distributed processing times:

| Job | $\lambda_j$ |
|---|---|
| 1 | 3.5 |
| 2 | 2.0 |
| 3 | 4.0 |
| 4 | 1.5 |
| 5 | 3.0 |
| 6 | 6.0 |

Solution:

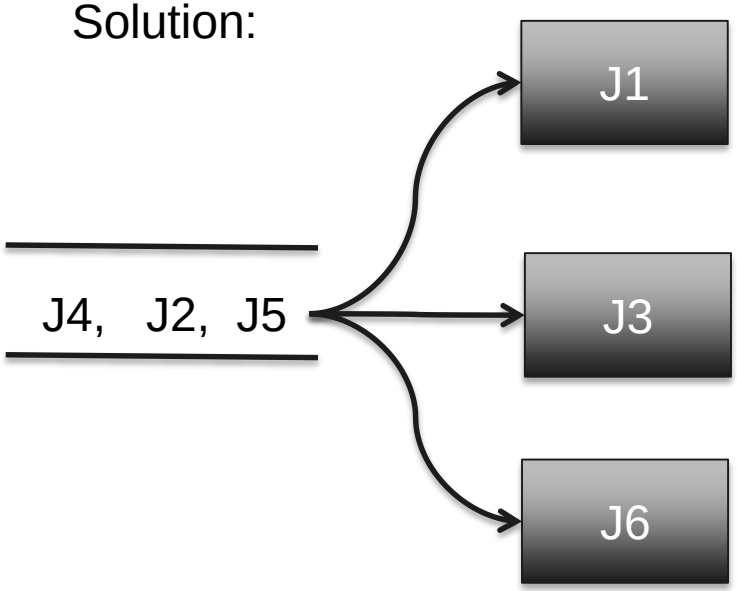


# Example: $Pm \mid \mid \sum C_j$

**Example:** Find the optimal static list policy for the following instance of stochastic $P3 \mid \mid \sum C_j$ with independent exponentially distributed processing times:

| Job | $\lambda_j$ |
|---|---|
| 1 | 3.5 |
| 2 | 2.0 |
| 3 | 4.0 |
| 4 | 1.5 |
| 5 | 3.0 |
| 6 | 6.0 |

Solution:



# Policy Implications

The class of static list policies is a **restriction** of the class of dynamic list policies. Based on this fact, answer what can you say about the following?

1. The difference between the objective function values for $Pm \mid \mid \sum C_j$ under static and dynamic policies?

2. The difference between the objective function values for $P2 \mid \mid \sum C_j$ with 3 jobs under static and dynamic policies?

# Heuristics



Many stochastic scheduling problems are difficult or impossible to solve exactly

Heuristics provide a means for fast approximation **<u>but</u> some measure of accuracy is needed**

**Definition**: The *performance ratio* (PR) of a heuristic for instance, *I*, is the ratio of the heuristic objective value *H(I)* to the optimal objective value *Opt (I).*

**Definition:** The worst case PR is:

$$PR^{WC} = \max_{I} \left\{ \frac{H(I)}{Opt(I)} \right\}$$

# Example: Bin-Packing Problem



The bin-packing problem asks the question: how many (1-dimensional) bins are needed to contain a set of items?

**Example:**

Bin Size = 10

Item Sizes: {5,8,3,4,6,1,7,5}

?

# Example: Bin-Packing Problem

IP Formulation for the bin-packing problem:

$$x_i = \begin{cases} 1 & if\ bin\ i\ selected \\ 0 & otherwise \end{cases} \qquad y_{ij} = \begin{cases} 1 & if\ item\ \ j\ assigned\ to\ bin\ i \\ 0 & Otherwise \end{cases}$$

$$\min \sum_{i=1}^{m} x_i$$ ← # of bins selected

$$s.t. \quad y_{ij} \le x_i \quad i = 1,\ldots,m,\ j = 1,\ldots,n$$ ← Items only allocated to bins that were selected

$$\sum_{i=1}^{m} y_{ij} = 1 \quad j = 1,\ldots,n$$ ← Every item goes in one bin

$$\sum_{j=1}^{n} z_j y_{ij} \le T \quad i = 1,\ldots,m$$ ← Bin size limit (z = item size, T = bin size)

$$y_{ij}, x_i\ binary, i = 1,\ldots,m,\ j = 1,\ldots,n$$

# Example: Bin-Packing Problem



Many bin-packing heuristics have been studied:

**First-fit (FF)**: Start with one empty bin; place each item in the first least full bin in which the item fits. If none then open another bin.

**Best-fit (BF)**: Start with on empty bin; place each item in the bin with the smallest space that the item can fit in. If none, open another bin.

It has been proven no **list heuristic for bin packing** can have a **worst case performance ratio** less than 1.5

Johnson, D.S. et al., 1974, Worst-Case Performance Bounds for Simple One-Dimensional Packing Algorithms, SIAM Journal on Computing

# Parallel Machines with Time Limits

Consider the following problem that is similar to bin packing and parallel machine scheduling

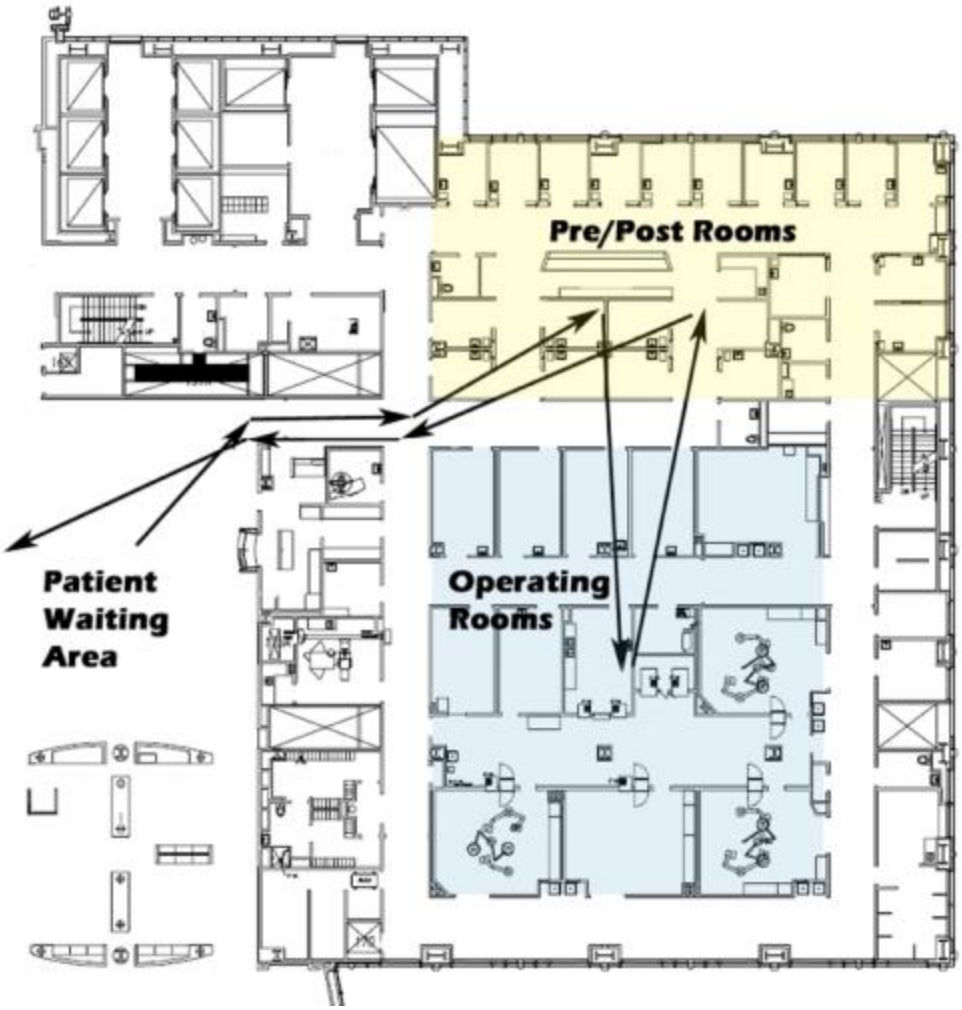
- The **number of machines may be varied** and there is a cost, $c^f$, for each machine used

- Machines are available for a **limited time** during the day, after which there is an **overtime cost** per unit time, $c^v$

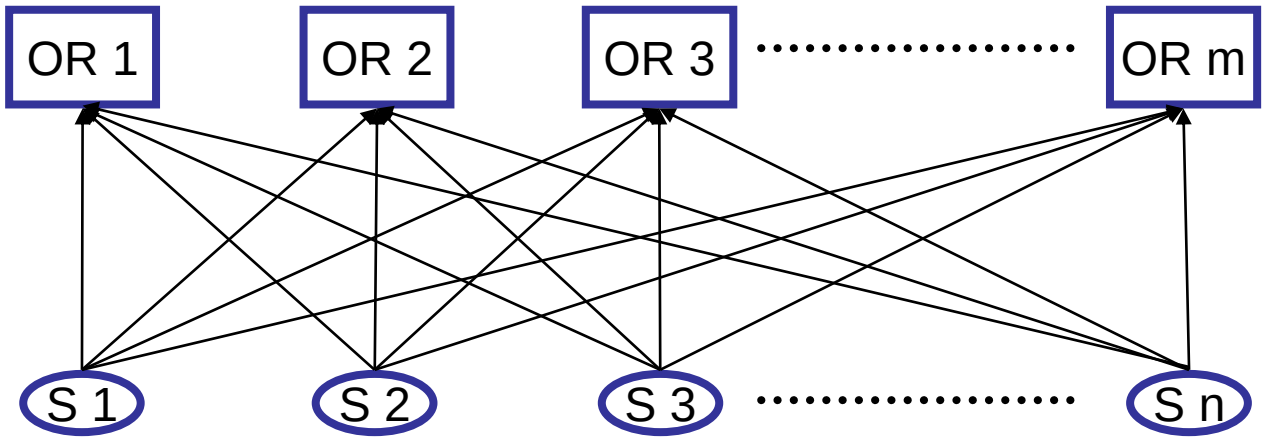
# Application: Surgery Scheduling

Surgery-to-OR scheduling

- Surgeries vary in duration
- Planned OR time is typically 8-10 hour per day
- Supporting staff are needed for every OR opened



# Application: Surgery Scheduling




**Decisions:** How many operating rooms (ORs) to open? Which OR to schedule each surgery block in?

**Criteria:** Cost of ORs opened, Overtime costs for operating rooms

# Extensible Bin-Packing

$$x_i = \begin{cases} 1 & if\ OR\ i\ active \\ 0 & otherwise \end{cases} \qquad y_{ij} = \begin{cases} 1 & if\ surgery\ \ j\ assigned\ to\ OR\ i \\ 0 & Otherwise \end{cases}$$

$$Z = \min\{\sum_{i=1}^{m} c^f x_i + c^v o_i\}$$ ← Cost of ORs + Overtime

$$s.t. \qquad y_{ij} \le x_i \quad i = 1,\ldots,m,\ j = 1,\ldots,n$$ ← Surgeries only scheduled in ORs that are active

$$\sum_{i=1}^{m} y_{ij} = 1 \quad j = 1,\ldots,n$$ ← Every surgery goes in one OR

$$\sum_{j=1}^{n} z_j y_{ij} - o_i \le T \quad i = 1,\ldots,m$$ ← Overtime

$$y_{ij}, x_i\ binary, \quad o_i \ge 0$$

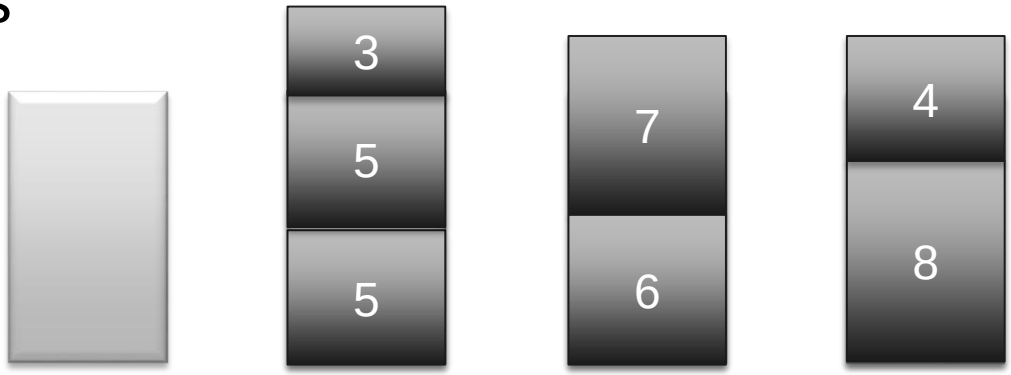
# Example: Extensible Bin-Packing

Extensible bin-packing allows overfilling bins, at a cost

**Example:**

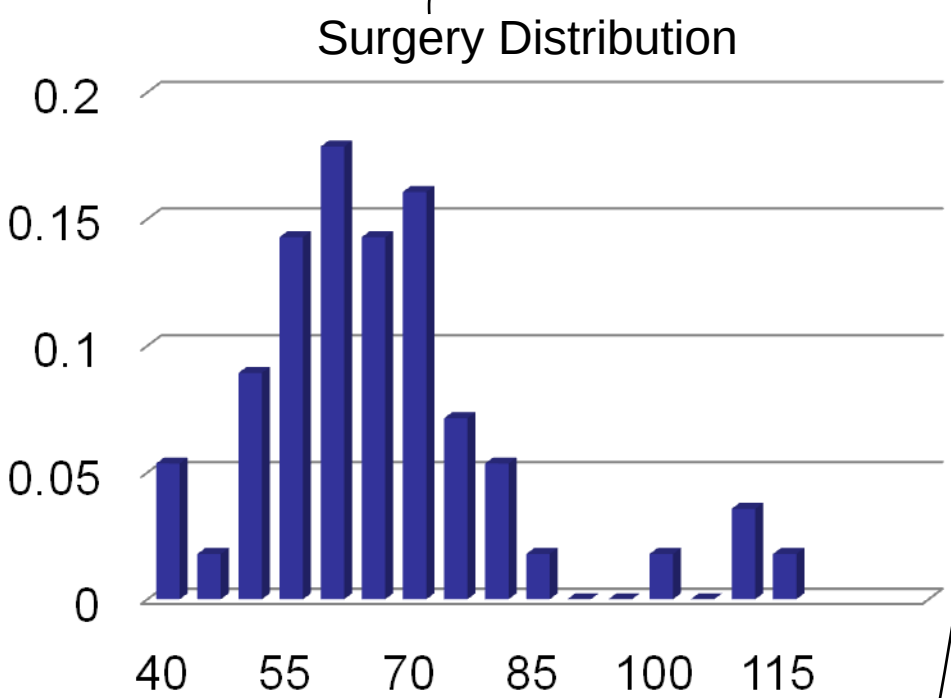
Bin Size = 10
Cost per bin 1:
Cost per unit 0.1:
Item Sizes: {5,8,3,4,6,1,7,5}

Optimal Solution: 3 Bins



# Stochastic Extensible Bin-Packing

The stochastic extensible bin-packing problem can be formulated as a stochastic program




$$Q(\mathbf{x}) = \min\{\sum_{i=1}^{m} c^f x_i + c^v E_\omega[o_i(\omega)]\}$$

$$s.t. \quad y_{ij} \le x_i \quad i = 1,\ldots,, j = 1,\ldots,n$$

$$\sum_{i=1}^{m} y_{ij} = 1 \quad j = 1,\ldots,n$$

$$\sum_{j=1}^{n} Z_j(\omega) y_{ij} - o_i(\omega) \le T x_i \qquad i = 1,..,m, \omega = 1,\ldots,K$$

$$y_{ij}, x_i \text{ binary, for all } i, j, \quad o_i(\omega) \ge 0, \text{ for all } i, \omega$$

# ELPT-Based Heuristic

Expected Longest Processing Time First (ELPT) Heuristic:

Compute mean surgery durations, $E[Z_j]$

$$Step\ 0:\ m = LB$$

$$LB = \left\lceil \frac{\sum_{j=1}^{n} E[Z_j]}{T(1+\frac{c^f}{c^v T})} \right\rceil$$

$$Step\ 1: Compute\ LPT(m)$$

$$Step\ 2:\quad if\ o_i = 0\ for\ all\ m\ STOP;$$
$$Else\ m = m+1\ and\ return\ to\ Step\ 1.$$

- Sort surgeries from longest to shortest
- Sequentially apply surgeries to emptiest room

# Average Performance

- Numerical experiments based on 20 problem instances generated using data from Mayo Clinic
- Solved the stochastic program (SP) with 10,000 scenarios to find optimal solution for every instance
- Following are the **ratios** (obtained from MC sampling) of the optimal SP solution to the solution from MV problem and ELPT heuristic

| Performance Measure | High Overtime Cost | | Low Overtime Cost | |
|---|---|---|---|---|
| | MV Problem | ELPT | MV Problem | ELPT |
| Mean | 0.89 | 0.92 | 0.99 | 0.99 |
| StDev | 0.05 | 0.06 | 0.01 | 0.01 |
| Max | 0.96 | 0.97 | 0.99 | 1.0 |
| Min | 0.81 | 0.81 | 0.97 | 0.97 |

# In-Class Assignment

Apply the LPT-based heuristic to the following instance of the **EBP** problem

- A Thoracic Surgery Unit has 9 surgeries with mean durations:
  - Surgery 1, Lobectomy, 75 minutes
  - Surgery 2, Biopsy, 150 minutes
  - Surgery 3, Thoracoscopy, 80 minutes
  - Surgery 4, Esophagogastrectomy: 330 minutes
  - Surgery 5, Bronchoscopy, 50 minutes
  - Surgery 6, Bronchoscopy, 50 minutes
  - Surgery 7, Thymectomy, 280 minutes
  - Surgery 8, Biopsy, 120 minutes
  - Surgery 9 Lobectomy, 175 minutes
- 6 ORs are available for 480 minutes each day (Regular Time)
- Overtime cost is $30/minute
- The cost of opening an OR/day is $4400