# Multi-Location Transshipment

**Group 1:**

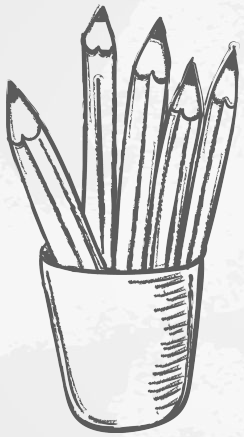
Jacob Andreesen  Jeff Chen
Miao Xu  Yiyi Wang

01
Vision

02
Technical Approach

03
Supporting Software

04
Responsibilities & Timeline



# 01

# Vision

# The Problem

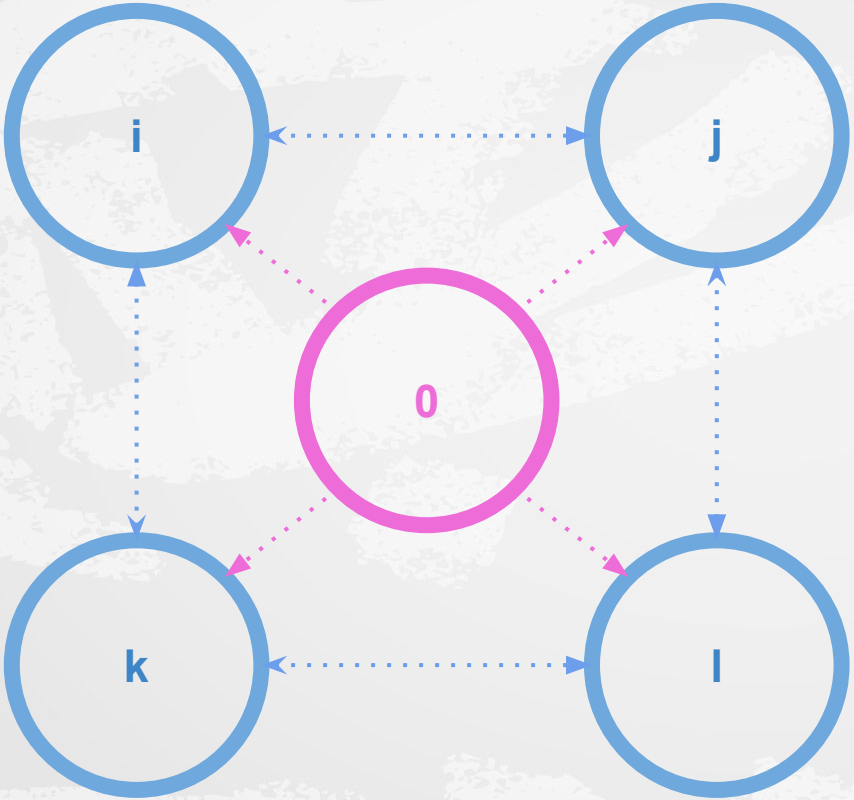
**Transshipments** provide an effective mechanism for correcting **discrepancies** between a locations' **observed demand** and their available **inventory**.

Planned and systematic transshipments can be used to **reduce costs** and **improve customer service** and **decrease infrastructure** requirements.

# Our Mission

**Goal:** Find the transshipment and replenishment **quantities** that **minimize** the expected long-run average **cost**. The cost is the sum of the replenishment, transshipment, holding costs.

# System Configuration

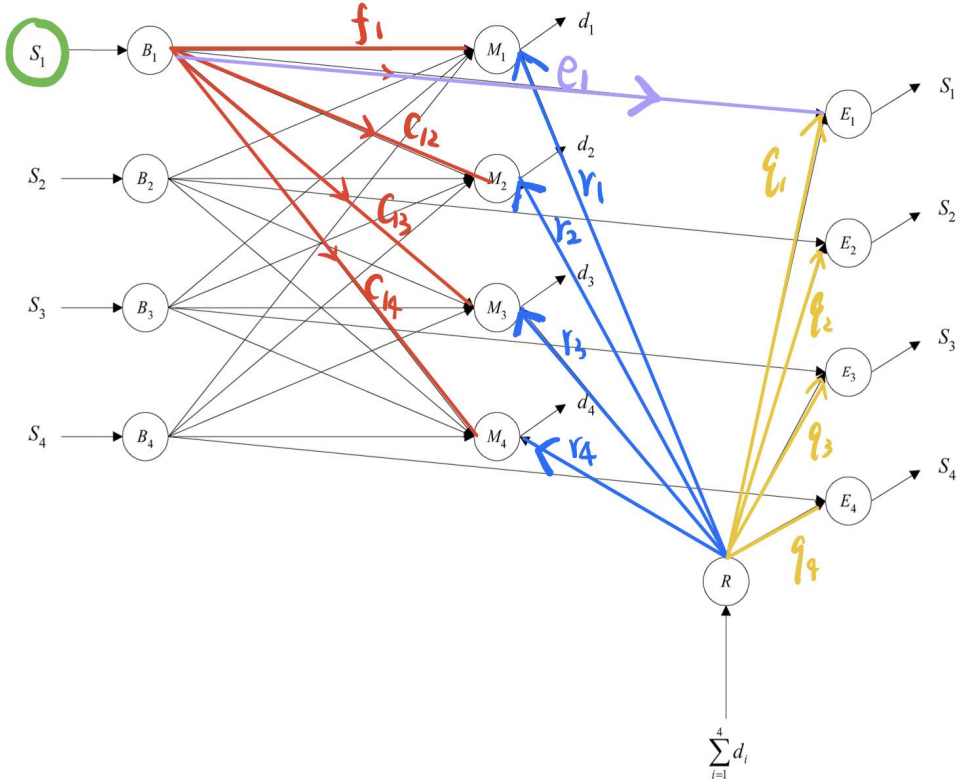



1
Supplier

4
Non-identical
Retailers

5
Demands

# Network Flow with Four Retailers



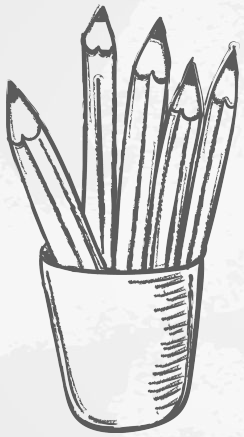

B= beginning of the day

M=start selling inventory

E=end of the day

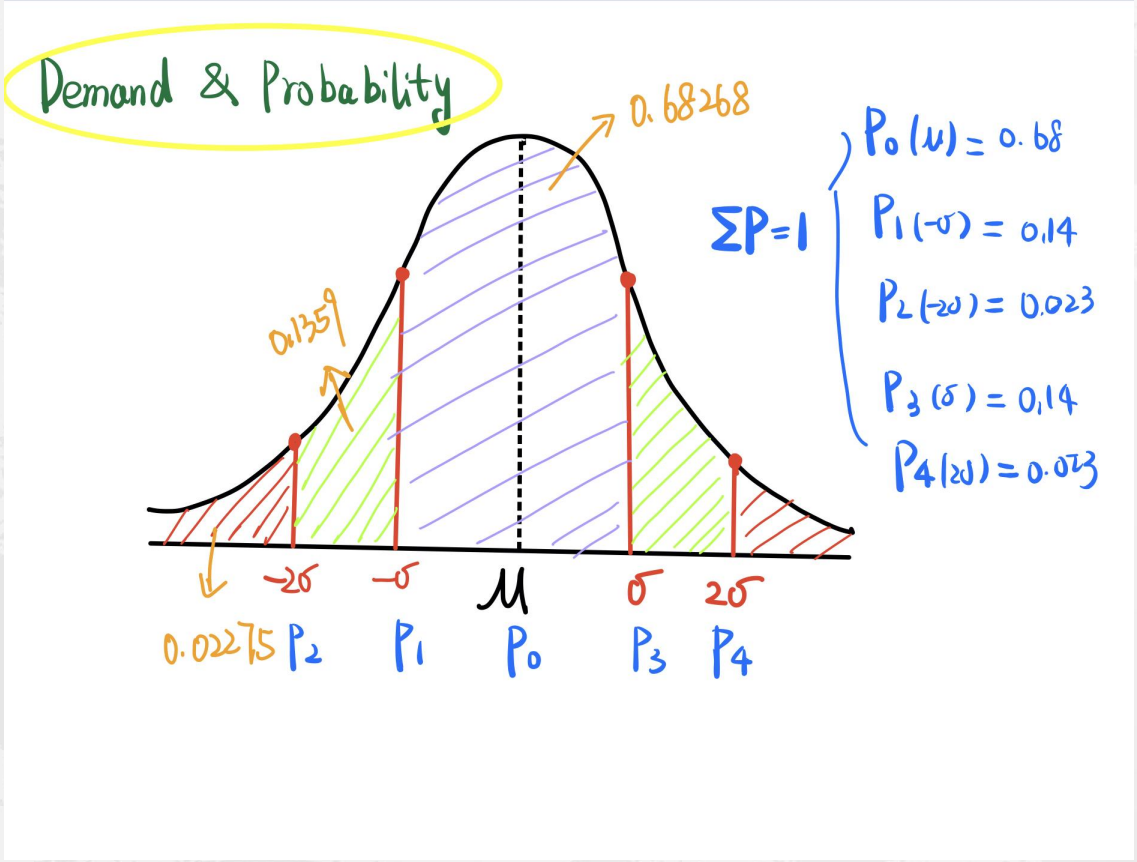
S=order-up-to quantities (beginning of the S = end of the S, after replenishment )

R=replenishment



# 02 Technical Approach

# Random Demand Generation




-Demand discretized around the mean demand value using SD

| Retailer | Normal Demand | Demand SD |
|---|---|---|
| 1 | 100 | 20 |
| 2 | 200 | 50 |
| 3 | 150 | 30 |
| 4 | 170 | 50 |

# Decision Variables

### *First-Stage Variables*

**Si**: order-up-to quantities (i = 1, 2, … , N)

**Di**: random normally distributed demands (i = 1, 2, … , N)

### *Second-Stage Variables*

**ei**: ending inventory held at retailer i.

**fi**: stock at retailer i used to satisfy demand at retailer i.

**qi**: inventory at retailer i increased through replenishment.

**ri**: amount of shortage met after replenishment at retailer i.

**tij**: stock at retailer i transshipped to meet demand at retailer j.

**hi**: unit cost of holding inventory at retailer i.

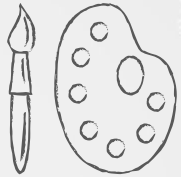
**cij**: unit cost of transshipment from retailer i to j.

**pi**: penalty cost for shortage at retailer i.

# Goals

## First Stage

Finding the optimal order-up-to quantities S(i) (Decision variable)



## Second Stage

Given Si and Di,
Minimize the expected long-run average cost over 5^4 scenarios

# Optimization Setup

**First Stage Objective:**

$$\min_{S \geq 0} E[h(S, \tilde{D})],$$

**Second Stage Objective:**

$$h(S, D) = \min \sum_i h_i e_i + \sum_{i \neq j} c_{ij} t_{ij} + \sum_i p_i r_i.$$

**Constraints:**

$$f_i + \sum_{j \neq i} t_{ij} + e_i = s_i, \quad \forall i$$

$$f_i + \sum_{j \neq i} t_{ji} + r_i = d_i, \quad \forall i$$

$$\sum_i r_i + \sum_i q_i = \sum_i d_i$$

$$e_i + q_i = s_i, \quad \forall i$$

$$e_i, f_i, q_i, r_i, s_i, t_{ij} \geq 0, \quad \forall i, j.$$

# Algorithm

## SGD

$$\min_{s \geq 0} E[h(s, \tilde{D})]$$

$$h(s, D) = \min \sum_i h_i e_i + \sum_{i \neq j} c_{ij} t_{ij} + \sum_i p_i r_i$$

$\min \; d_s^T y$

s.t. $Dy = \xi_s - C_s x$ $\quad \pi$ (dual multiplier)

$y \geq 0$

Given $S, D$, put $v = -c$, for all $s$ and solve LP:

1. Calculate $C_s x$
2. update $r_s = \xi_s - C_s x$
3. Solve $\min d_s^T y$, with $y \geq 0$, $Dy = r_s$
4. get dual multiplier $\pi_s$
5. update $v^+ \longleftarrow v - \sum P_N \pi_N^T C_N$

} iterate $N = 30, 50 \cdots$ (Sample size)

After $N$ scenarios have been solved.

updated $v^+$ is the subgradient of $f(x)$ at $x$

$$x^+ \longleftarrow P_B (x - \alpha_k v)$$

Stop after $m$ iterations

## Benders

$$\min_{s \geq 0} E[h(s, \tilde{D})]$$

$$h(s, D) = \min \sum_i h_i e_i + \sum_{i \neq j} c_{ij} t_{ij} + \sum_i p_i r_i. \quad (2)$$

$\min_s \mu$ (1)

① Solve LP (1)

to get $S_i$ and lowerbounds $l_i$.

② Given $S_i$, solve LP (2) for all $5^4$ scenarios

to get upperbounds $u_i$ and $\alpha_i, \beta_i$. $u_i = \alpha_i + \beta_i S_i$

③ Add $\mu \geq \alpha_i + \beta_i S$ as the constraint of LP (1)

iterate steps 1 to 3 until $u_i - l_i \leq \varepsilon$

stop after $m$ iterations

# Comparison

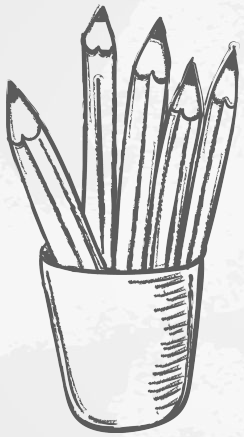
## SGD

- SGD randomly pick *one data point or sample a small number of data points* at each iteration to reduce the computation enormously.
- Sample of 30 -> 50

## Benders Decomposition

- Subproblems are solved to determine the Benders cuts for the master problem. Cuts are then added to the master problem, which is then re-solved until no cuts can be generated.

Compare SGD and Benders Decomposition by the number of iterations and CPU time.



# 03 Supporting Software

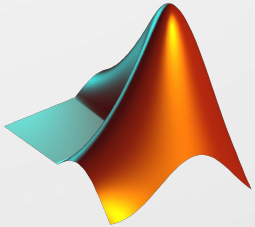
# Supporting Libraries

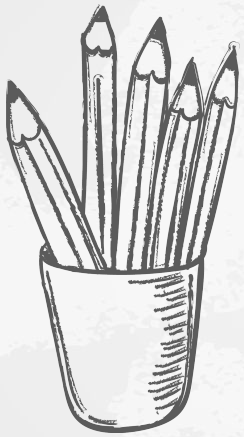
## Pyomo

## PySP


## Pandas


## (Potentially Matlab)





# 04 Responsibilities & Timeline

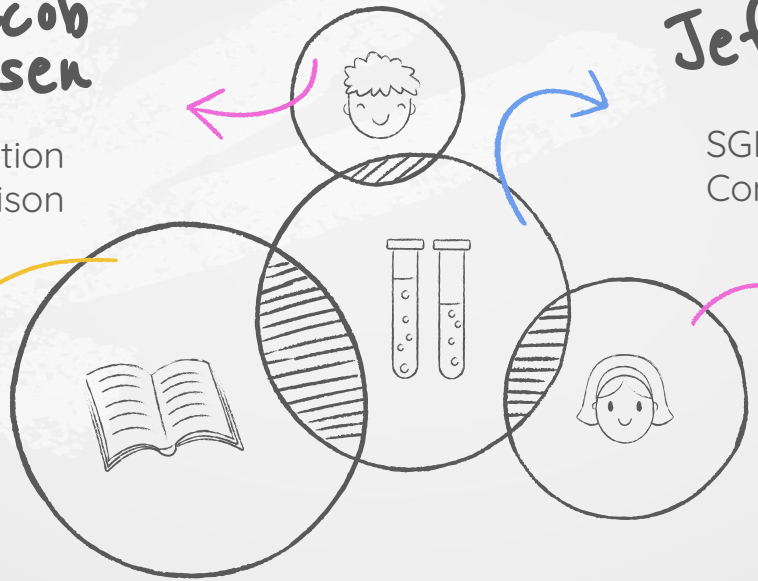
# Responsibilities

## Jacob Andreesen

SGD implementation Comparison

## Jeff Chen

SGD Implementation Comparison



## Miao Xu

Bender Decomposition Comparison

## Yiyi Wang

Bender Decomposition Comparison

# A Timeline Works Well


Problem Definition

SGD & BD Design

Implementation

Comparison & Review

Thanks

Fight on!