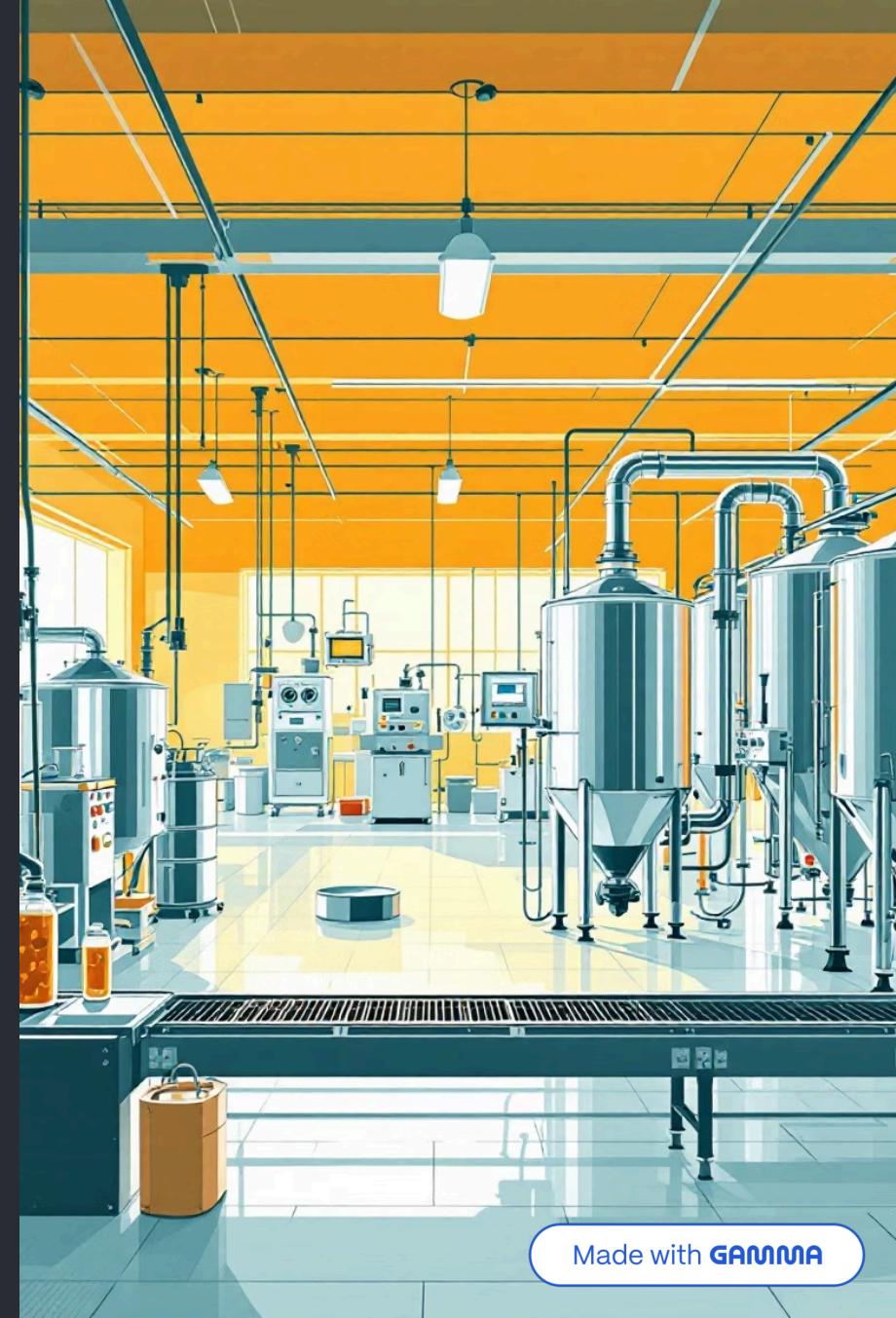


# Production Optimization Using MILP

Optimizing production schedules for three high-demand beverage products:  
Protein Shake, Zero-Sugar Energy Drink, and Functional Water





## The Production Challenge

### Problem Context

Our manufacturing plant is struggling to keep pace with surging demand across all three product lines. Frequent delivery delays, capacity constraints, and inefficient changeovers are creating bottlenecks that impact customer satisfaction and profitability.

**Critical Need:** A data-driven production plan that maximizes throughput while minimizing costs and meeting service level commitments.

### Production Flow

Our three-stage process moves products through:

- **Mixing Tank (M1):** Ingredient blending and formulation
- **Filling Line (M2):** High-speed bottle filling and capping
- **Packaging Line (M3):** Final boxing and palletizing

# Understanding the Business Impact



## Core Question

How can we meet weekly demand at minimum cost while ensuring high service levels and operational efficiency?



## Key Challenges

- Product changeovers consuming valuable production time
- Peak demand creating late order risk
- No systematic scheduling optimization



## Business Impact

- Inventory imbalances across product lines
- Elevated operational costs
- Poor on-time delivery (OTD) performance
- Customer dissatisfaction and risk of lost business

```
if (dayIndex == 0) {
    // Initialize initial inventory levels
    for (String product : products) {
        initialInventory.put(product, 0);
    }
}

// Read daily demand from CSV file
List<Map<String, Integer>> dailyDemand = new ArrayList<Map<String, Integer>>();
try (BufferedReader reader = new BufferedReader(new FileReader("daily_demand.csv"))){
    String line;
    while ((line = reader.readLine()) != null) {
        Map<String, Integer> dailyDemandEntry = new HashMap<String, Integer>();
        String[] values = line.split(",");
        for (int i = 0; i < values.length; i++) {
            String product = products[i];
            int quantity = Integer.parseInt(values[i]);
            dailyDemandEntry.put(product, quantity);
        }
        dailyDemand.add(dailyDemandEntry);
    }
}
}

// Main loop: Process each day
for (int dayIndex = 0; dayIndex < 7; dayIndex++) {
    Map<String, Integer> dailyDemandEntry = dailyDemand.get(dayIndex);
    Map<String, Integer> currentInventory = new HashMap<String, Integer>(initialInventory);
    Map<String, Integer> production = new HashMap<String, Integer>();

    // Process daily demand
    for (String product : products) {
        int demand = dailyDemandEntry.get(product);
        if (currentInventory.getOrDefault(product, 0) < demand) {
            System.out.println("Insufficient inventory for product " + product + " on day " + (dayIndex + 1));
            continue;
        }
        currentInventory.put(product, currentInventory.get(product) - demand);
        production.put(product, demand);
    }

    // Check for overtime
    boolean overtimeRequired = false;
    for (String product : products) {
        if (production.get(product) > machineHours * overtimeRate) {
            overtimeRequired = true;
            break;
        }
    }

    // Calculate overtime cost
    double overtimeCost = 0;
    if (overtimeRequired) {
        overtimeCost = overtimeRate * overtimeHours;
    }

    // Calculate total cost
    double totalCost = production.values().stream().mapToInt(Integer::intValue).sum() * unitProductionCost + overtimeCost;

    // Output results
    System.out.println("Day " + (dayIndex + 1) + " Results:");
    System.out.println("Production: " + production);
    System.out.println("Current Inventory: " + currentInventory);
    System.out.println("Total Cost: " + totalCost);
}
```

# Building the Optimization Model

## Input Parameters

We gathered comprehensive production data to build an accurate model:

- Available machine hours per day
- Processing time per product per machine
- Setup time required for product switching
- Weekly demand targets for all products

## Weekly Demand Targets

- **P1 (Protein Shake):** 1,600 units
- **P2 (Energy Drink):** 1,200 units
- **P3 (Functional Water):** 1,000 units

## MILP Model Framework

**Mixed-Integer Linear Programming (MILP)** provides the mathematical foundation for optimal decision-making.

**Objective:** Minimize total production cost plus late delivery penalties

**Decision Variables:** Quantity produced per product per day across all machines

**Constraints:**

- Machine capacity limits
- Setup time requirements
- Daily working hours
- Inventory balance equations



# Three Scenarios to Test the System

1

## Scenario A: Flexible Plan

**Purpose:** Establish baseline performance

Normal operating system with balanced priorities across cost, throughput, and service level. This scenario tests standard conditions without extreme constraints.

2

## Scenario B: Setup-Minimizing

**Purpose:** Evaluate batching strategy

Heavy penalties applied to product changeovers to test whether larger batch production runs can reduce setup time and improve overall efficiency.

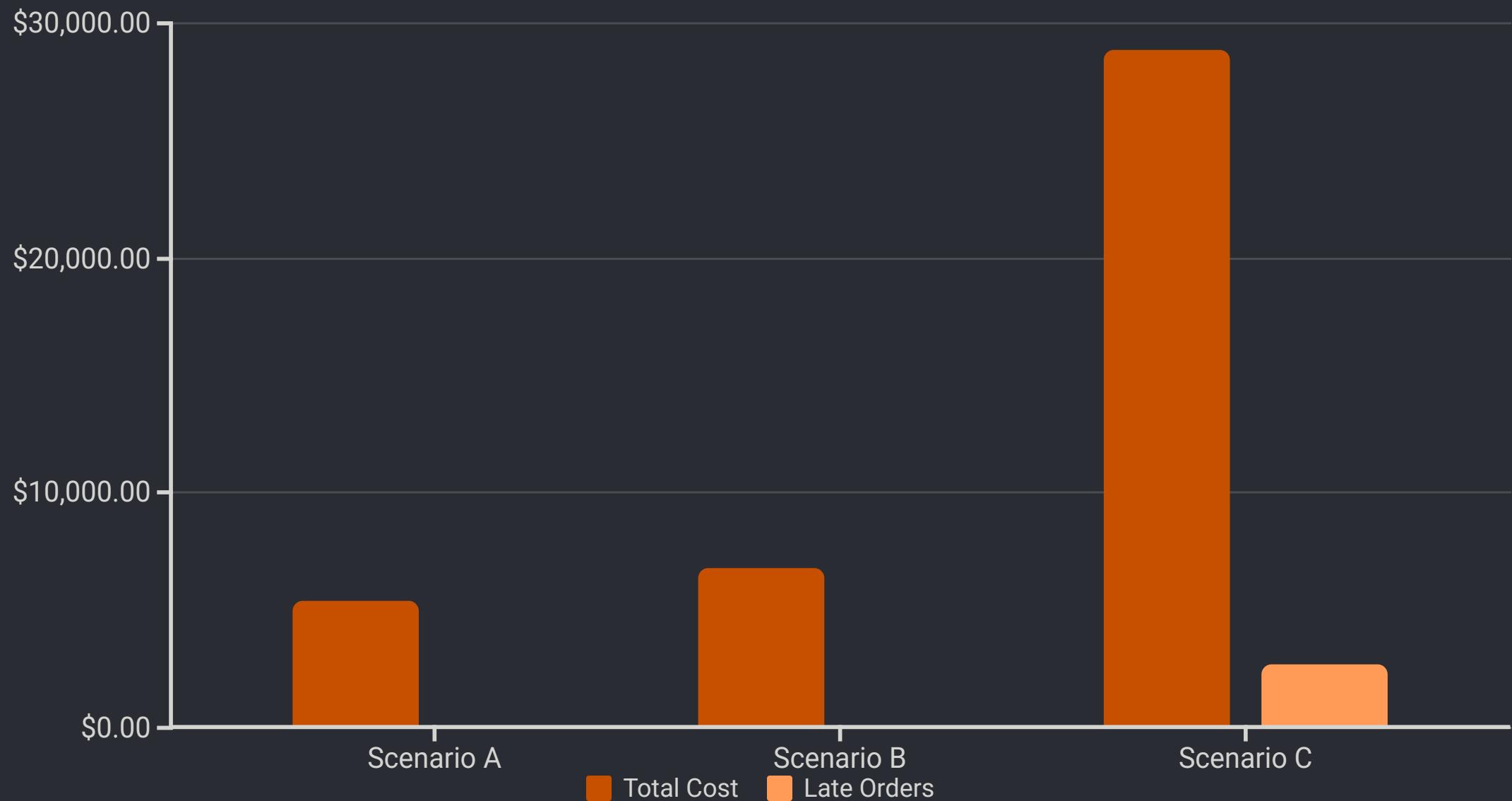
3

## Scenario C: Stress Test

**Purpose:** Identify breaking points

Reduced working hours combined with rush demand for P2 (Energy Drink). This scenario simulates capacity overload to reveal system vulnerabilities.

# Performance Results Across Scenarios



**3,720**

**Total Throughput**

Units produced in Scenarios A & B

**2,400**

**Scenario C Output**

Significant drop under stress conditions

**97.9%**

**OTD Performance**

Scenarios A & B delivery success

**47%**

**Scenario C OTD**

System collapse under overload

# Key Insights from the Analysis

01

**Scenario A delivers optimal balance** of cost efficiency and service level, achieving 97.9% on-time delivery with the lowest total cost.

02

**Scenario B's setup reduction strategy** increases costs by 27% without improving delivery performance, suggesting batching alone isn't the answer.

03

**Scenario C reveals critical vulnerability:** The system completely breaks down under capacity constraints, with late orders jumping to 2,700 units.

04

**Smart sequencing is essential** across all successful scenarios, highlighting the importance of optimized production scheduling.

05

**M1 (Mixing Tank) emerges** as the factory's constraint, operating at 100% utilization and limiting overall throughput capacity.

# Root Cause Analysis: What's Holding Us Back?



## Issue #1: M1 Bottleneck

The Mixing Tank operates at 100% utilization across all scenarios, creating a hard ceiling on factory throughput. This constraint forces production sequencing delays and limits our ability to scale.

## Issue #2: Setup Time Impact

Daily product switching consumes valuable production hours. More frequent changeovers mean less time for actual manufacturing. Scenario B showed that batching helps, but doesn't solve the root problem.

## Issue #3: System Breaking Point

Under tight capacity with rush demand, the entire system collapses. Scenario C produced 2,700 late units (47% OTD) because M1 simply cannot handle the overload without additional capacity.

# Strategic Recommendations

## Recommended Approach

### Implement Scenario A: Flexible Production Plan

This strategy delivers the optimal balance:

- **97.9% on-time delivery** performance
- **Lowest operational cost** at \$5,375
- **Maximum throughput** of 3,720 units
- **Fair distribution** across all three product lines



#### Expand M1 Capacity

Add a second Mixing Tank or extend shift hours to eliminate the bottleneck



#### Reduce Setup Time

Implement SMED (Single-Minute Exchange of Die) methodology to minimize changeover duration



#### Strategic Batching

Introduce batch production during peak demand weeks to reduce setup frequency



#### Automate Planning

Deploy MILP-based software to generate optimal weekly schedules automatically



# 98% Service Level

## Achieved

### Through Data-Driven Optimization

**Key Takeaway:** MILP optimization revealed the Mixing Tank (M1) as the plant's primary constraint and unlocked a 98% service level using an optimal flexible production schedule. By combining mathematical optimization with strategic capacity investments, we can meet growing demand while minimizing costs and maximizing customer satisfaction.

- **Next Steps:** Proceed with Scenario A implementation while initiating capital planning for M1 capacity expansion. Begin SMED training program to reduce setup times across all production lines.