MediFlow

A Decision-Support System for Optimising Patient Flow and Staff Scheduling in Healthcare Facilities

Project Report

MSL304 – Operations Management

Indian Institute of Technology Delhi

Submitted by:

Abhikrit Bhardwaj (2022ME21333)

Sakhare Yash Balram (2022CH71496)

Contents

1 Executive Summary

Healthcare facilities routinely struggle with long queues, uneven workload distribution, and subjective scheduling decisions. The **MediFlow Optimiser** integrates two analytical tools—a discrete-event **Patient Flow Simulator** and an optimisation-based **Staff Rota Generator**—to provide quantitative, data-backed decision support.

The simulator analyses waiting times, queue lengths, and staff utilisation using an M/M/c queue model implemented with SimPy. The optimiser produces minimum-cost weekly staff schedules using integer programming via PuLP, accounting for availability, hourly cost, and working-hour constraints.

The combined system assists managers in evaluating what-if scenarios, identifying bottlenecks, improving service levels, and reducing operational cost.

2 Operational Problems

2.1 Patient Flow and Bottleneck Identification

Clinics operate as multi-stage queues where delays accumulate unpredictably. Without quantitative tools, managers struggle to identify the real bottleneck. The simulator provides clarity by modelling arrivals, service processes, and queue behaviour under varying load and capacity.

2.2 Staff Scheduling and Labour Cost

Staff scheduling involves balancing coverage, cost, working limits, and availability. Manual schedules often lead to inefficiencies such as understaffing (long waits) or overstaffing (high cost). The MediFlow optimiser ensures constraints are respected while minimising labour expenditure.

3 The MediFlow Optimiser System

3.1 Patient Flow Simulator (Queueing Analysis)

Objective: Compute average waiting time, queue length, utilisation, and identify bottlenecks.

Inputs

- Patient arrival rate (patients/hr)
- Service rate (patients/hr per staff)
- Number of staff on duty
- Simulation duration

Outputs

- Average waiting time (W_q)
- Average queue length ($L_q = \lambda W_q$)
- Staff utilisation
- Bottleneck warnings

3.2 Staff Rota Optimiser (Integer Programming)

Objective: Generate the lowest-cost feasible rota covering all shifts.

Inputs

- Staff list and hourly cost
- Weekly maximum hours
- Availability by shift
- Required staffing levels

Outputs

- Optimal weekly shift assignments
- Total cost
- Staff workload distribution

4 Technical Methodology

4.1 Queue Simulation Model

The system models an M/M/c queue with exponential inter-arrival and service times:

$$\text{InterArrival} \sim \text{Exp}(\lambda), \qquad \text{ServiceTime} \sim \text{Exp}(\mu) \tag{1}$$

Key computed metrics:

$$W_q = \text{mean waiting time}, \qquad L_q = \lambda W_q \tag{2}$$

$$\rho = \frac{\text{Total Busy Time}}{cT} \tag{3}$$

SimPy manages event scheduling, server utilisation, and queue progression.

4.2 Rota Optimisation Model

Binary decision variable:

$$x_{s,h} = \begin{cases} 1 & \text{if staff } s \text{ works shift } h \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Objective Function

$$\min \sum_{s,h} x_{s,h} \cdot \text{Cost}(s) \cdot 8 \tag{5}$$

Constraints

Shift Coverage:

$$\sum_{s} x_{s,h} \geq R(h) \tag{6}$$

Maximum Hours:

$$8 \sum_{h} x_{s,h} \leq H_{\max}(s) \tag{7}$$

Availability:

$$x_{s,h} = 0 \quad \text{if } s \text{ unavailable for } h \tag{8}$$

One Shift Per Day:

$$x_{s,(d,AM)} + x_{s,(d,PM)} \leq 1 \tag{9}$$

5 Model Assumptions

5.1 Simulation Assumptions

- Poisson arrival process.
- Exponential service time distribution.
- Identical staff servers.
- Infinite queue capacity.
- No patient abandonment.
- FCFS queue discipline.

5.2 Optimisation Assumptions

- All shifts are 8 hours.
- No overtime allowed.
- Availability strictly enforced.

- Costs linear with respect to hours.
- One shift per staff per day.

6 Implementation

The system is implemented in Python with three modules:

- **main.py** – user interface using Inquirer
- **simulator.py** – SimPy discrete-event simulation
- **optimiser.py** – PuLP integer programming model

The program runs in a terminal and supports scenario testing.

7 Results

7.1 Patient Flow Simulation

A baseline simulation with:

$$\lambda = 10, \quad \mu = 4, \quad c = 3, \quad T = 50$$

produced:

- $W_q = 0.247$ hours
- $L_q = 2.47$ patients
- Utilisation $\rho = 0.83$
- System stable

7.2 Staff Rota Optimisation

After adjusting Wednesday morning staffing requirements, the optimiser produced a feasible minimum-cost rota:

$$\text{Total Cost} = 2552$$

Staff	Assigned Shifts
Nurse_A	Mon_AM, Tue_AM, Wed_AM, Thu_AM, Fri_AM
Nurse_B	Mon_PM, Tue_PM, Wed_PM, Thu_PM, Fri_PM
Nurse_C	Tue_AM, Thu_AM
Tech_D	Mon_AM, Tue_PM, Wed_AM, Thu_PM, Fri_AM

Table 1: Optimal Weekly Rota Generated by MediFlow

8 Discussion

The simulation results confirm expected queueing behaviour: waiting times increase sharply when utilisation approaches 1, while the system remains stable for utilisation levels below 0.85. The optimiser prefers assigning shifts to lower-cost staff (notably Tech_D), provided coverage and availability constraints are satisfied. This behaviour is consistent with cost-minimising objectives.

The stable utilisation level ($\rho = 0.83$) observed in the baseline scenario indicates a well-balanced clinic configuration. Managerial decisions such as increasing service rate or reallocating staff can be tested easily by modifying input parameters, making MediFlow a flexible decision-support tool.

9 Conclusion

The MediFlow Optimiser demonstrates that simulation and optimisation techniques can significantly improve healthcare operational performance. The simulator identifies bottlenecks and predicts waiting times, while the optimiser generates cost-efficient staff schedules.

The tool is modular, extendable, and applicable to real clinics with minor modifications.

10 References

References

[1] SimPy Development Team, *SimPy Documentation*, 2023.

[2] Stuart Mitchell, “PuLP: A Linear Programming Toolkit for Python,” 2009.

[3] D. Gross, J. Shortle, J. Thompson, and C. Harris, *Fundamentals of Queueing Theory*, Wiley, 2018.

[4] S. Nahmias and T. Olsen, *Production and Operations Analysis*, Waveland Press, 2015.

Appendix: Selected Code Snippets

Main Menu Structure (main.py)

```
options = [
    inquirer.List(
        "choice",
        message="Choose␣an␣option:",
        choices=[
            "Patient␣Flow␣Simulator",
            "Staff␣Rota␣Optimiser",
            "Exit",
        ],
    )
]
answer = inquirer.prompt(options)
```

Patient Simulation Process (simulator.py)

```
def patient_process(self, name):
    arrival = self.env.now
    with self.staff.request() as req:
        yield req
        wait = self.env.now - arrival
        service = random.expovariate(self.service_rate)
        yield self.env.timeout(service)
```

Rota Optimisation Objective (optimiser.py)

```
model += pulp.lpSum(
    x[s][sh] * STAFF_COST[s] * SHIFT_DURATIONS[sh]
    for s, sh in allowed
)
```

Shift Coverage Constraint

```
for sh in SHIFTS:
    model += pulp.lpSum(
        x[s][sh] for s in STAFF if (s, sh) in allowed
    ) >= SHIFT_REQUIREMENTS[sh]
```