Optimizing Work Schedules

The case of Shaikh Zayed Hospital

Raiya Momin - 24110005 Rija Abdul Rabb - 24110183 Eesha Yousaf - 24110333 Syed Haris Shah - 24110131

PROBLEM

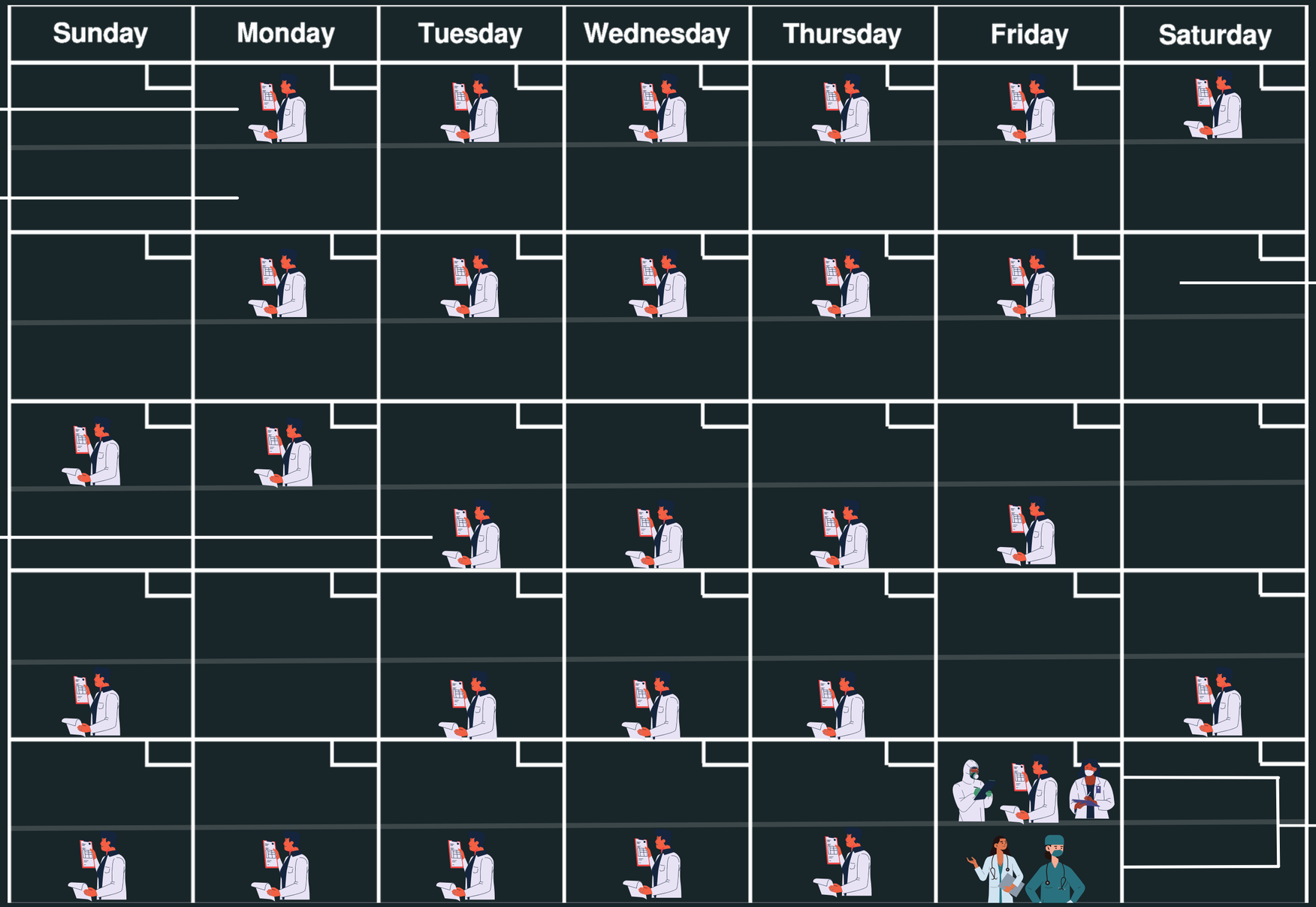
Shaikh Zayed Hospital realised the lack of time-effective clear plan on allocating shifts. Currently, they have to manually prepare the schedule for each month, which leads to overlapping shifts, under or over allocation of doctors leading resulting in inefficiency in terms of planning and scheduling rosters

Scale of the problem

- Increases with multiple doctors, wards
- Complexity increases with rotating shifts
- Improper allocation of doctors - reduced productivity
- Miscommunication due to changes in duties and rotation

- Time inefficient method
- Scheduling has to be redone for each month
- Issue of replacing duties to avoid clashes
- Monetary Costs associated with scheduling

CONSTRAINTS CONSIDERED


Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
one shift per day
rotating shifts biweekly
at least 1 leave per week
minimum requirements for morning and evening shifts

SCHEDULING PROGRAM

```
from pulp import *

no_of_days = 30
no_of_doctors = 20
minimum_doctors_required_morning = 8
minimum_doctors_required_evening = 6

# Decision variables
morning_shift = LpVariable.dicts('M', (range(no_of_days), range(no_of_doctors)),
0, 1, 'Binary')
evening_shift = LpVariable.dicts('E', (range(no_of_days), range(no_of_doctors)),
0, 1, 'Binary')

# Objective Function
objective = None
for day in range(no_of_days):
    for doctor in range(no_of_doctors):
        objective += (morning_shift[day][doctor]) + (evening_shift[day][doctor])
problem = LpProblem('shifts', LpMaximize)
problem += objective

# Ensuring minimum doctor requirement of ward in each shift is fulfilled
for day in range(0, no_of_days):
    morning_shifts = None
    evening_shifts = None
    for doctor in range(no_of_doctors):
        morning_shifts += morning_shift[day][doctor]
        evening_shifts += evening_shift[day][doctor]
    problem += morning_shifts >= minimum_doctors_required_morning
    problem += evening_shifts >= minimum_doctors_required_evening

# Ensuring each doctor only works one shift in a day
for day in range(no_of_days):
    for doctor in range(no_of_doctors):
        shifts = None
        shifts += morning_shift[day][doctor] + evening_shift[day][doctor]
        problem += shifts <= 1

# Each doctor must have at least one day off per week
shifts_covered_in_a_week = 6
for doctor in range(no_of_doctors):
    for day in range(no_of_days-7):
        shifts = 0
        for following_day in range(7):
            checking_day = day+following_day
```

```
            shifts += morning_shift[checking_day][doctor] +
evening_shift[checking_day][doctor]
        problem += shifts <= shifts_covered_in_a_week

# Ensuring doctors work the same shift consequetively
for doctor in range(no_of_doctors):
    for day in range(no_of_days - 1):
        shifts = None
        shifts += morning_shift[day][doctor] + evening_shift[day+1][doctor]
    problem += shifts <= 1

for doctor in range(no_of_doctors):
    for day in range(no_of_days - 1):
        if ((day + 1) % 7 != 0) and ((day + 2) % 7 != 0):
            shifts = None
            shifts += evening_shift[day][doctor] + morning_shift[day+1][doctor]
        problem += shifts <= 1

# Change doctor shifts after two weeks
for doctor in range(no_of_doctors):
    for day in range(0, no_of_days - 14, 2):
        shifts = None
        shifts += morning_shift[day][doctor] + morning_shift[day+7][doctor] +
morning_shift[day+14][doctor]
        problem += shifts <= 2

print("Current Status: ", LpStatus[problem.status])

problem.solve()


print("Shift covered = 1, Shift not covered = 0")

for day in range(no_of_days):
    for doctor in range(no_of_doctors):
        print(morning_shift[day][doctor], " - shift covered = ",
int(morning_shift[day][doctor].varValue))
        print(evening_shift[day][doctor], " - shift covered = ",
int(evening_shift[day][doctor].varValue))
```