Consulting Project: NurseNext Optimization

Management Science - Healthcare Workforce Management

Client Briefing: NurseNext Hospital

Meet Your Client

Industry: Healthcare Client Contact: Dr. Robin Torres, Chief Operating Officer Facility: Medium-sized hospital with 3 departments Staff: 21 nurses, 50 beds, 24/7 operations

The COO's Challenge

Dr. Robin Torres, COO of NurseNext Hospital

"I became a doctor to help people, but now I spend 8 hours every week manually creating nurse schedules and they're still terrible!

The situation is unsustainable:

- Massive overtime costs
- 25% sick leave rate because nurses are exhausted
- High turnover, we lose 3-4 experienced nurses annually
- Constant complaints about unfair weekend distribution
- Last month, 2 nurses quit citing 'burnout and unfair scheduling'

The manual process is killing us. I try to be fair, but with 21 nurses, 3 departments, different skill levels, shift preferences, and labor laws... I just can't optimize it all in my head.

We need an automated scheduling system that:

1. Cuts overtime massively, if possible
2. Distributes weekends fairly (the biggest complaint)
3. Respects labor laws (consecutive shifts, rest periods)
4. Maintains quality of care (right skills at right times)
5. Is robust to sick calls (we always have 1-2 nurses out)

Can you build this for us? I have data from last month's schedules and our staffing requirements." :::

The Business Context

Current Situation
Departments:

- Emergency Department (ED): High-intensity, requires experienced nurses

- Medical-Surgical (Med-Surg): General care, moderate intensity
- Intensive Care Unit (ICU): Critical patients, highest skill requirements

Staffing Model:

NurseNext uses a modern hybrid staffing model:

- Department Specialists (13 nurses): Core staff assigned to specific departments
  - 4 ED specialists, 5 Med-Surg specialists, 4 ICU specialists
  - Deep expertise in their department's specialized care
  - Only work in their assigned department
- Flexible Float Pool (8 nurses): Cross-trained resource nurses
  - Can work in any department (ED, Med-Surg, or ICU)
  - Provide flexibility for workload balancing across departments
  - Essential for covering peak demand and sick calls

This creates an interesting resource allocation challenge: How should flexible nurses be distributed across departments to minimize cost and maximize fairness?

Shifts (Each 8 hours):

- Morning (7:00-15:00): Highest patient activity
- Evening (15:00-23:00): Moderate activity
- Night (23:00-7:00): Lower activity but critical monitoring

Scheduling Horizon: 1 week (21 shifts total = 7 days × 3 shifts/day)

Cost Structure
- Regular wage: €25/hour (€200 per 8-hour shift)
- Overtime wage: €37.50/hour (€300 per overtime shift, 1.5x multiplier)
- Weekend premium: +€50 per weekend shift
- Night premium: +€30 per night shift
- Understaffing penalty: €500 per understaffed shift (patient safety risk)

Labor Law Constraints

> ⚠ Legal Requirements (Must Comply!)
>
> 1. Maximum consecutive shifts: 5 days in a row
> 2. Minimum rest period: 11 hours between shifts
> 3. Maximum weekly hours: 48 hours per nurse (6 shifts max)
> 4. Weekend work limit: Maximum 2 weekend days per month

Fairness Expectations
Nurses' union has negotiated fairness requirements:

- Weekend shifts distributed equally across nurses
- Night shifts distributed equally
- Workload (total hours) balanced within ±10%
- No nurse should work significantly more undesirable shifts than others

The Data

Nurse Information

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from datetime import datetime, timedelta

# Set random seed for reproducibility
np.random.seed(2025)

# Generate nurse data
def generate_nurse_data():
    nurses = []

    # ED Specialists (4 nurses) - only work in ED
    for i in range(1, 5):
        nurses.append({
            'nurse_id': i,
            'name': f'Nurse ED-{i}',
            'department': 'ED',
            'departments_qualified': ['ED'],
            'nurse_type': 'Specialist',
            'skill_level': np.random.choice(['Junior', 'Senior', 'Senior',
'Expert'], p=[0.2, 0.5, 0.2, 0.1]),
            'experience_years': np.random.randint(1, 15),
            'can_work_night': np.random.choice([True, True, False], p=[0.6,
0.3, 0.1]),
            'weekend_worked_last_month': np.random.randint(1, 4)
        })

    # Med-Surg Specialists (5 nurses) - only work in Med-Surg
    for i in range(5, 10):
        nurses.append({
            'nurse_id': i,
            'name': f'Nurse MS-{i-4}',
            'department': 'Med-Surg',
            'departments_qualified': ['Med-Surg'],
            'nurse_type': 'Specialist',
            'skill_level': np.random.choice(['Junior', 'Senior', 'Senior'],
p=[0.3, 0.5, 0.2]),
            'experience_years': np.random.randint(1, 12),
            'can_work_night': np.random.choice([True, True, False], p=[0.7,
0.2, 0.1]),
            'weekend_worked_last_month': np.random.randint(1, 4)
        })

    # ICU Specialists (4 nurses) - only work in ICU
    for i in range(10, 14):
        nurses.append({
            'nurse_id': i,
            'name': f'Nurse ICU-{i-9}',
            'department': 'ICU',
```

```python
            'departments_qualified': ['ICU'],
            'nurse_type': 'Specialist',
            'skill_level': np.random.choice(['Senior', 'Expert', 'Expert'],
p=[0.4, 0.4, 0.2]),
            'experience_years': np.random.randint(3, 20),
            'can_work_night': True,  # ICU nurses must be able to work
nights
            'weekend_worked_last_month': np.random.randint(1, 4)
        })

    # Flexible Float Pool (8 nurses) - can work in ANY department
    for i in range(14, 22):
        nurses.append({
            'nurse_id': i,
            'name': f'Nurse FLOAT-{i-13}',
            'department': 'Float',
            'departments_qualified': ['ED', 'Med-Surg', 'ICU'],
            'nurse_type': 'Flexible',
            'skill_level': np.random.choice(['Junior', 'Senior', 'Senior',
'Expert'], p=[0.15, 0.5, 0.25, 0.1]),
            'experience_years': np.random.randint(2, 15),
            'can_work_night': np.random.choice([True, True, True, False],
p=[0.75, 0.15, 0.05, 0.05]),  # Most can work nights
            'weekend_worked_last_month': np.random.randint(1, 4)
        })

    return pd.DataFrame(nurses)

nurses_df = generate_nurse_data()

print("NURSENEXT HOSPITAL - NURSE ROSTER")
print("=" * 100)
print(nurses_df.to_string(index=False))
print("\n" + "=" * 100)
print(f"Total nurses: {len(nurses_df)}")
print(f"\nBy type:")
print(f"  Specialists:
{len(nurses_df[nurses_df['nurse_type']=='Specialist'])} nurses")
print(f"    - ED: {len(nurses_df[nurses_df['department']=='ED'])} nurses")
print(f"    - Med-Surg: {len(nurses_df[nurses_df['department']=='Med-
Surg'])} nurses")
print(f"    - ICU: {len(nurses_df[nurses_df['department']=='ICU'])}
nurses")
print(f"  Flexible (Float Pool):
{len(nurses_df[nurses_df['nurse_type']=='Flexible'])} nurses (can work any
department)")
print(f"\nBy skill level:
Junior={len(nurses_df[nurses_df['skill_level']=='Junior'])}, "
      f"Senior={len(nurses_df[nurses_df['skill_level']=='Senior'])}, "
      f"Expert={len(nurses_df[nurses_df['skill_level']=='Expert'])}")
```

```
NURSENEXT HOSPITAL - NURSE ROSTER
====================================================================================================
```

```
 nurse_id           name department departments_qualified nurse_type
skill_level    experience_years    can_work_night   weekend_worked_last_month
      1    Nurse ED-1        ED               [ED] Specialist
Junior               3            True                           1
      2    Nurse ED-2        ED               [ED] Specialist
Senior              13           False                           1
      3    Nurse ED-3        ED               [ED] Specialist
Senior               2           False                           2
      4    Nurse ED-4        ED               [ED] Specialist
Senior               1            True                           1
      5    Nurse MS-1  Med-Surg         [Med-Surg] Specialist
Senior               1            True                           2
      6    Nurse MS-2  Med-Surg         [Med-Surg] Specialist
Senior               2            True                           2
      7    Nurse MS-3  Med-Surg         [Med-Surg] Specialist
Junior              10           False                           1
      8    Nurse MS-4  Med-Surg         [Med-Surg] Specialist
Senior               6           False                           1
      9    Nurse MS-5  Med-Surg         [Med-Surg] Specialist
Senior              10            True                           1
     10   Nurse ICU-1       ICU              [ICU] Specialist
Senior               8            True                           3
     11   Nurse ICU-2       ICU              [ICU] Specialist
Expert              10            True                           3
     12   Nurse ICU-3       ICU              [ICU] Specialist
Expert              18            True                           3
     13   Nurse ICU-4       ICU              [ICU] Specialist
Expert              18            True                           2
     14 Nurse FLOAT-1     Float   [ED, Med-Surg, ICU]   Flexible
Senior               4            True                           3
     15 Nurse FLOAT-2     Float   [ED, Med-Surg, ICU]   Flexible
Senior               2            True                           1
     16 Nurse FLOAT-3     Float   [ED, Med-Surg, ICU]   Flexible
Expert               8            True                           1
     17 Nurse FLOAT-4     Float   [ED, Med-Surg, ICU]   Flexible
Senior               5            True                           3
     18 Nurse FLOAT-5     Float   [ED, Med-Surg, ICU]   Flexible
Senior               2            True                           2
     19 Nurse FLOAT-6     Float   [ED, Med-Surg, ICU]   Flexible
Expert              11            True                           1
     20 Nurse FLOAT-7     Float   [ED, Med-Surg, ICU]   Flexible
Senior               8            True                           2
     21 Nurse FLOAT-8     Float   [ED, Med-Surg, ICU]   Flexible
Senior               9            True                           1

=====================================================================================
Total nurses: 21

By type:
  Specialists: 13 nurses
    - ED: 4 nurses
    - Med-Surg: 5 nurses
    - ICU: 4 nurses
  Flexible (Float Pool): 8 nurses (can work any department)
```

```
By skill level: Junior=2, Senior=14, Expert=5
```

Shift Requirements

```python
# Define shift requirements for each department and shift type
# Format: (min_nurses, min_senior_nurses)

shift_requirements = {
    'ED': {
        'Morning': {'min_nurses': 2, 'min_senior': 1},
        'Evening': {'min_nurses': 2, 'min_senior': 1},
        'Night': {'min_nurses': 1, 'min_senior': 1}
    },
    'Med-Surg': {
        'Morning': {'min_nurses': 3, 'min_senior': 2},
        'Evening': {'min_nurses': 2, 'min_senior': 1},
        'Night': {'min_nurses': 1, 'min_senior': 1}
    },
    'ICU': {
        'Morning': {'min_nurses': 2, 'min_senior': 1},
        'Evening': {'min_nurses': 1, 'min_senior': 1},
        'Night': {'min_nurses': 2, 'min_senior': 2}
    }
}

# Display requirements
print("\nSHIFT STAFFING REQUIREMENTS")
print("=" * 70)
for dept, shifts in shift_requirements.items():
    print(f"\n{dept} Department:")
    for shift, req in shifts.items():
        print(f"  {shift:8} - Min {req['min_nurses']} nurses (at least
{req['min_senior']} Senior/Expert)")

# Calculate total weekly demand
total_shifts_needed = 0
for dept in shift_requirements:
    for shift in shift_requirements[dept]:
        total_shifts_needed += shift_requirements[dept][shift]
['min_nurses'] * 7  # 7 days

print("\n" + "=" * 70)
print(f"Total shift-slots needed per week: {total_shifts_needed}")
print(f"Available capacity (21 nurses × 6 shifts max): {21 * 6} shift-
slots")
print(f"Capacity margin: {21 * 6 - total_shifts_needed} shift-slots")
```

```
SHIFT STAFFING REQUIREMENTS
======================================================================
```

```
ED Department:
  Morning  - Min 2 nurses (at least 1 Senior/Expert)
  Evening  - Min 2 nurses (at least 1 Senior/Expert)
  Night    - Min 1 nurses (at least 1 Senior/Expert)

Med-Surg Department:
  Morning  - Min 3 nurses (at least 2 Senior/Expert)
  Evening  - Min 2 nurses (at least 1 Senior/Expert)
  Night    - Min 1 nurses (at least 1 Senior/Expert)

ICU Department:
  Morning  - Min 2 nurses (at least 1 Senior/Expert)
  Evening  - Min 1 nurses (at least 1 Senior/Expert)
  Night    - Min 2 nurses (at least 2 Senior/Expert)


======================================================================
Total shift-slots needed per week: 112
Available capacity (21 nurses × 6 shifts max): 126 shift-slots
Capacity margin: 14 shift-slots
```

Week Structure

```python
# Define the week structure
days_of_week = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday',
'Saturday', 'Sunday']
shift_types = ['Morning', 'Evening', 'Night']

# Create a schedule template DataFrame
schedule_template = []
for day_num, day in enumerate(days_of_week):
    for shift in shift_types:
        is_weekend = day in ['Saturday', 'Sunday']
        schedule_template.append({
            'day': day,
            'day_num': day_num,
            'shift': shift,
            'is_weekend': is_weekend,
            'shift_code': f"{day[:3]}_{shift[:3]}"  # e.g., "Mon_Mor"
        })

schedule_template_df = pd.DataFrame(schedule_template)

print("\nWEEK SCHEDULE STRUCTURE (21 shifts total)")
print("=" * 60)
print(schedule_template_df.to_string(index=False))
```

```
WEEK SCHEDULE STRUCTURE (21 shifts total)
============================================================
     day  day_num   shift  is_weekend shift_code
  Monday        0 Morning       False    Mon_Mor
  Monday        0 Evening       False    Mon_Eve
```

```
   Monday          0   Night        False     Mon_Nig
  Tuesday          1 Morning        False     Tue_Mor
  Tuesday          1 Evening        False     Tue_Eve
  Tuesday          1   Night        False     Tue_Nig
Wednesday          2 Morning        False     Wed_Mor
Wednesday          2 Evening        False     Wed_Eve
Wednesday          2   Night        False     Wed_Nig
 Thursday          3 Morning        False     Thu_Mor
 Thursday          3 Evening        False     Thu_Eve
 Thursday          3   Night        False     Thu_Nig
   Friday          4 Morning        False     Fri_Mor
   Friday          4 Evening        False     Fri_Eve
   Friday          4   Night        False     Fri_Nig
 Saturday          5 Morning         True     Sat_Mor
 Saturday          5 Evening         True     Sat_Eve
 Saturday          5   Night         True     Sat_Nig
   Sunday          6 Morning         True     Sun_Mor
   Sunday          6 Evening         True     Sun_Eve
   Sunday          6   Night         True     Sun_Nig
```

Minimal Helper Functions

We provide only basic data structures and a validation function. You must implement the scheduling algorithm yourself.

```python
def validate_schedule(schedule, nurses_df, shift_requirements):
    """
    Validate a schedule against all constraints.

    Args:
        schedule: Dict or DataFrame representing nurse assignments
                  Format example: {('Monday', 'Morning', 'ED'): [nurse_id1,
nurse_id2, ...], ...}
        nurses_df: DataFrame with nurse information
        shift_requirements: Dict with staffing requirements

    Returns:
        dict with validation results and violations
    """
    violations = []
    warnings = []

    # This is a TEMPLATE - YOU need to implement the actual validation
logic!
    # Check:
    # 1. All shifts have minimum staff
    # 2. Senior/Expert requirements met
    # 3. No nurse works more than 6 shifts per week
    # 4. No nurse works more than 5 consecutive days
    # 5. Minimum 11 hours rest between shifts
    # 6. Weekend distribution fairness

    print("Validation function is a template - you must implement full
```

```python
    logic!")

    return {
        'is_valid': len(violations) == 0,
        'violations': violations,
        'warnings': warnings
    }

def calculate_schedule_cost(schedule, nurses_df):
    """
    Calculate total cost of a schedule.

    Args:
        schedule: Your schedule representation
        nurses_df: DataFrame with nurse information

    Returns:
        dict with cost breakdown
    """
    # Constants
    REGULAR_SHIFT_COST = 200
    OVERTIME_SHIFT_COST = 300  # 1.5x multiplier
    WEEKEND_PREMIUM = 50
    NIGHT_PREMIUM = 30
    UNDERSTAFFING_PENALTY = 500

    # YOU must implement this!
    # Calculate costs based on:
    # - Regular shifts vs overtime (>40 hours/week)
    # - Weekend premiums
    # - Night premiums
    # - Understaffing penalties

    print("Cost calculation function is a template - you must implement!")

    return {
        'total_cost': 0,
        'regular_cost': 0,
        'overtime_cost': 0,
        'premium_cost': 0,
        'penalty_cost': 0
    }

print("\nHelper function templates loaded")
print("YOU must implement the full logic!")
```

```
Helper function templates loaded
YOU must implement the full logic!
```

Your Task

> **i Deliverables**
>
> You must develop a scheduling solution that creates a one-week schedule for all 21 nurses and provides:
>
> ## 1. Jupyter Notebook with Complete Solution
> Your notebook should include:
>
> - Data exploration: Insights about current staffing patterns and constraints
> - Algorithm implementation: Your scheduling approach (construction + improvement)
> - Results: Complete one-week schedule with all assignments
> - Fairness analysis: Distribution of weekends, nights, workload across nurses
> - Cost analysis: Total cost and breakdown
> - Robustness testing: What-if analysis for 1-2 nurses calling in sick
> - Code quality: Well-commented, clear, and reproducible
>
> ## 2. Presentation (8 minutes maximum)
> - Problem understanding (1 min): Dr. Torres's challenges in your own words
> - Your approach (2 min): Algorithm choice, how you handled constraints and fairness
> - Results (3 min): Schedule visualization, cost savings, fairness metrics
> - Business impact (1 min): Recommendations for NurseNext, implementation plan
> - Q&A preparation (1 min): Trade-offs, limitations, next steps
>
> ## 3. Key Metrics to Report
> - Total weekly cost
> - Overtime cost reduction vs. simple greedy solution
> - Fairness metrics:
>   - Weekend distribution (mean, std, min, max)
>   - Night shift distribution
>   - Total shifts per nurse
> - Constraint compliance (labor law violations: should ideally be ZERO!)
> - Robustness: Impact of 1-2 sick nurses

Constraints and Requirements

> **!** Hard Constraints (Should Satisfy!)
>
> 1. Shift coverage: All shifts meet minimum staffing requirements
> 2. Skill requirements: Each shift has required number of Senior/Expert nurses
> 3. Maximum shifts: No nurse works more than 6 shifts per week (48 hours)
> 4. Maximum consecutive: No nurse works more than 5 consecutive days
> 5. Minimum rest: At least 11 hours between shifts (no Evening→Morning or Night→Morning/Evening)
> 6. Department qualification: Nurses only work in departments they're qualified for
>   - Specialists: Can only work in their assigned department (check `departments_qualified` list)
>   - Flexible nurses: Can work in any department (ED, Med-Surg, or ICU)
> 7. Night shift eligibility: Only nurses with `can_work_night=True` on night shifts
>
> ### Soft Constraints (Optimize for Fairness)
>
> 1. Weekend fairness: Equal distribution of weekend shifts
> 2. Night shift fairness: Equal distribution of night shifts among eligible nurses
> 3. Workload balance: Total shifts per nurse should be within ±1 shift of average
> 4. Undesirable shift balance: Fair distribution of "undesirable" shifts (nights, weekends)

Tips for Success

1. Start with constraints: Get a feasible schedule first, optimize fairness second
2. Use data structures wisely: Consider using:
   - Dict: `{(day, shift, dept): [nurse_ids]}`
   - DataFrame: Rows = nurses, Columns = shifts
   - Your own structure!
3. Fairness is subjective: Define clear metrics for "fair" distribution
4. Visualize the schedule: A Gantt chart or heat map makes patterns obvious
5. Labor laws are non-negotiable: Zero violations if possible. Check after every change!
6. Think like a nurse: What makes a schedule "good" from their perspective?
   - Predictable patterns
   - Fair weekend distribution
   - No excessive consecutive shifts
7. Overtime is expensive: Keeping nurses under 40 hours saves €100/shift!

Common Pitfalls to Avoid

- Ignoring labor laws: Legal violations = instant failure in real world!
- Unfair weekend distribution: This is nurses' #1 complaint, track it carefully!
- Not checking 11-hour rest: Evening→Morning violates rest requirements
- Forgetting skill requirements: ICU night needs 2 Senior/Expert nurses!

- Over-optimization: A good, fair schedule beats a "perfect" but unfair one
- No robustness testing: Real hospitals have sick calls every week!
- Poor visualization: Make the schedule easy to read for nurses!

Data Access

All data is provided in this notebook:

- `nurses_df`: DataFrame with 21 nurses and their attributes
- `shift_requirements`: Dict with staffing requirements by department and shift
- `schedule_template_df`: Week structure (21 shifts)
- Helper templates: `validate_schedule()`, `calculate_schedule_cost()`

You must implement the full scheduling algorithm and constraint checking logic!

Deadline

- Notebook submission & Presentation: Lecture 12
- Good luck, consultants!

Bibliography