Competition - Restaurant Staffing Crisis

Management Science - La Étoile Three-Star Restaurant

Client Briefing

La Étoile, our three-Michelin-star restaurant, faces a staffing crisis this weekend.

The Situation

"I need to schedule my 18 servers across 6 shifts this weekend. The problem? My experienced servers cost €75/hour while juniors cost only €25/hour. But it gets worse! Shifts have different lengths (4-6 hours), servers have strong preferences about when they can work, and if I don't have enough experienced servers on busy shifts, we face penalties ranging from €0 to €1,200 per missing experienced server from our parent company!"

The Challenge

You have multiple competing objectives. You need more experienced servers than available, shifts have different lengths affecting labor costs, and servers have varying preferences for different shifts. Critically, some servers cannot work certain shifts due to personal commitments (shown as preference = 0). Your job is to find the best compromise:

- Minimize total cost (labor + experience penalties + preference penalties)
- Respect hard availability constraints (preference = 0 means CANNOT work that shift)
- Balance multiple factors: shift coverage, server happiness, and costs
- Strategic decisions: which expensive penalties to accept?
- Navigate interdependencies created by availability constraints

> Tip
>
> Fortunately, La Étoile has already clearly quantified the weight of all objectives for you, so you don't need to involve a pareto frontier in the optimization process.

Your Resources

- 6 Experienced Servers (€75/hour)
- 12 Junior Servers (€25/hour)
- 6 Shifts to cover (3 servers each)
- Everyone works exactly one shift

The Data

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import random
from typing import Dict, List, Tuple

# Set seeds
np.random.seed(2025)
random.seed(2025)

# Server data
servers = pd.DataFrame({
    'Server_ID': [f'E{i:02d}' for i in range(1, 7)] + [f'J{i:02d}' for i in
range(1, 13)],
    'Type': ['Experienced'] * 6 + ['Junior'] * 12,
    'Hourly_Rate': [75] * 6 + [25] * 12,
    'Name': ['François', 'Amélie', 'Baptiste', 'Céline', 'Damien', 'Élise']
+
            ['Jules', 'Margot', 'Théo', 'Camille', 'Léa', 'Hugo',
             'Emma', 'Louis', 'Chloé', 'Lucas', 'Manon', 'Ethan']
})

# Shift data with varying lengths - Weekend coverage
shifts = pd.DataFrame({
    'Shift_ID': ['FD', 'FL', 'SL', 'SD', 'UL', 'UD'],
    'Shift_Name': ['Friday Dinner', 'Friday Late', 'Saturday Lunch',
                   'Saturday Dinner', 'Sunday Lunch', 'Sunday Dinner'],
    'Day': ['Friday', 'Friday', 'Saturday', 'Saturday', 'Sunday',
'Sunday'],
    'Time': ['18:00-00:00', '23:00-03:00', '11:00-15:00',
             '17:00-21:00', '11:00-16:00', '17:00-22:00'],
    'Hours': [6, 4, 4, 4, 5, 5],
    'Required_Servers': [3, 3, 3, 3, 3, 3],
    'Min_Experienced': [2, 0, 1, 2, 1, 2],  # 8 needed, 6 available =
shortage of 2!
    'Expected_Covers': [120, 40, 80, 140, 90, 130],  # Number of guests
    'Penalty_Per_Missing': [800, 0, 500, 1200, 600, 1000]  # Penalties!
})

# Server preferences (0 = CANNOT WORK, 1-10 = preference level)
np.random.seed(2025)
preferences = {}

# Hard availability constraints (0 = cannot work this shift)
unavailable = {
    # Experienced server constraints
    'E01': ['FD', 'UL', 'SL', 'FL'],  # François: Can work SD, UD only
    'E02': ['UD', 'UL', 'SL', 'FL'],  # Amélie: Can work SD, FD, FL only
    'E03': ['SD', 'FD', 'SL', 'FL'],  # Baptiste: Can work UD, UL, FL only
    'E04': ['SD', 'SL', 'UL', 'FL'],  # Céline: Can work FD, UD only
    'E05': ['SD', 'SL', 'UL', 'FL'],  # Damien: Can work FD, UD only
    'E06': ['FD', 'UL', 'SL', 'FL'],  # Élise: Can work SD, UD, FL only
    # Junior constraints
```

```
    'J02': ['SD', 'UD', 'FD'],
    'J05': ['SD', 'UD', 'FD'],
    'J08': ['SD', 'UD', 'FD'],
}

for _, server in servers.iterrows():
    server_prefs = []
    for _, shift in shifts.iterrows():
        # Check hard constraints first
        if server['Server_ID'] in unavailable and shift['Shift_ID'] in
unavailable[server['Server_ID']]:
            pref = 0  # CANNOT WORK
        else:
            # Generate preferences based on server type
            if server['Type'] == 'Experienced':
                if 'Dinner' in shift['Shift_Name']:
                    pref = np.random.choice([5, 7, 8, 9, 10], p=[0.1, 0.2,
0.3, 0.3, 0.1])
                elif 'Late' in shift['Shift_Name']:
                    pref = np.random.choice([1, 2, 3, 4, 5], p=[0.2, 0.3,
0.2, 0.2, 0.1])
                else:  # Lunch
                    pref = np.random.choice([3, 4, 5, 6, 7], p=[0.1, 0.2,
0.3, 0.3, 0.1])
            else:  # Junior servers
                if 'Late' in shift['Shift_Name']:
                    pref = np.random.choice([4, 6, 7, 8, 9], p=[0.1, 0.2,
0.2, 0.3, 0.2])
                elif 'Lunch' in shift['Shift_Name']:
                    pref = np.random.choice([3, 5, 6, 7, 8], p=[0.1, 0.2,
0.3, 0.3, 0.1])
                else:  # Dinner
                    pref = np.random.choice([2, 4, 5, 6, 7], p=[0.1, 0.2,
0.3, 0.3, 0.1])
        server_prefs.append(pref)
    preferences[server['Server_ID']] = server_prefs

# Create preference matrix
pref_matrix = pd.DataFrame(preferences, index=shifts['Shift_ID']).T
print("\n\nSERVER PREFERENCES (0=CANNOT WORK, 1=hates it, 10=loves it):")
print("=" * 60)
print(pref_matrix.to_string())
print("\nAvailability Constraints:")
for server_id, shifts_unavailable in unavailable.items():
    server_name = servers[servers['Server_ID'] == server_id].iloc[0]
['Name']
    print(f"  {server_name} ({server_id}): Cannot work {',
'.join(shifts_unavailable)}")

print("SERVERS AVAILABLE:")
print("=" * 60)
print(servers.to_string(index=False))

print("\n\nSHIFT REQUIREMENTS:")
```

```python
print("=" * 60)
print(shifts.to_string(index=False))

print("\n\nTHE PROBLEM:")
print("=" * 60)
print(f"Total experienced servers needed:
{shifts['Min_Experienced'].sum()}")
print(f"Total experienced servers available: {(servers['Type'] ==
'Experienced').sum()}")
print(f"SHORTAGE: {shifts['Min_Experienced'].sum() - (servers['Type'] ==
'Experienced').sum()} experienced servers!")
print("\nComplex factors:")
print(" • HARD CONSTRAINTS: Some servers cannot work certain shifts
(pref=0)")
print(" • Varying shift lengths (4-6 hours) affect total labor cost")
print(" • Large penalty amounts (€500-€1200) make choices strategic")
print(" • Server preferences (1-10) affect morale and quality")
print(" • Lower preferences = unhappy staff = significant quality penalty")
print(" • Availability constraints create interdependencies between
assignments")
```

```
SERVER PREFERENCES (0=CANNOT WORK, 1=hates it, 10=loves it):
============================================================
Shift_ID  FD  FL  SL  SD  UL  UD
E01        0   0   0   7   0   9
E02       10   0   0   8   0   0
E03        0   0   0   0   5   7
E04        9   0   0   0   0   8
E05       10   0   0   0   0   9
E06        0   0   0   8   0   9
J01        2   8   3   4   7   7
J02        0   7   5   0   7   0
J03        7   7   5   7   8   4
J04        5   6   3   5   8   5
J05        0   9   3   0   5   0
J06        5   8   8   5   8   5
J07        6   9   6   6   5   5
J08        0   6   6   0   7   0
J09        6   6   6   2   8   7
J10        4   9   6   4   7   5
J11        7   8   8   7   5   5
J12        6   8   6   6   7   6

Availability Constraints:
  François (E01): Cannot work FD, UL, SL, FL
  Amélie (E02): Cannot work UD, UL, SL, FL
  Baptiste (E03): Cannot work SD, FD, SL, FL
  Céline (E04): Cannot work SD, SL, UL, FL
  Damien (E05): Cannot work SD, SL, UL, FL
  Élise (E06): Cannot work FD, UL, SL, FL
  Margot (J02): Cannot work SD, UD, FD
```

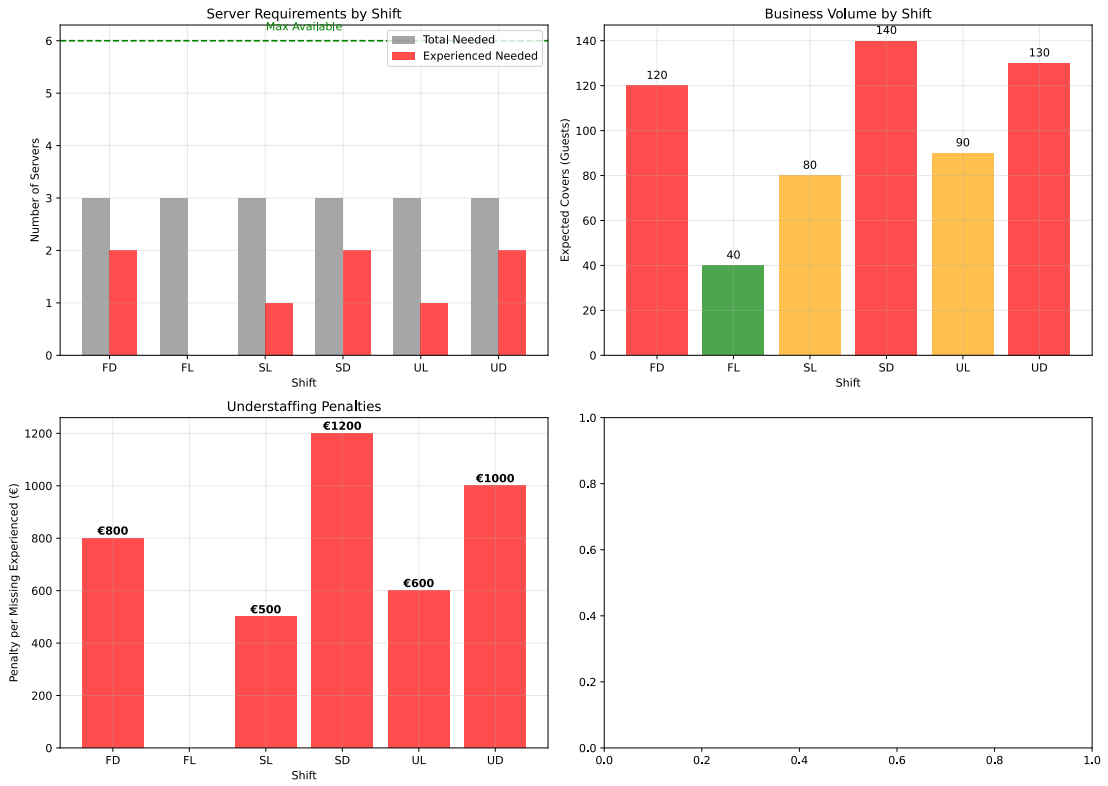
```
  Léa (J05): Cannot work SD, UD, FD
  Louis (J08): Cannot work SD, UD, FD
SERVERS AVAILABLE:
============================================================
Server_ID        Type  Hourly_Rate     Name
      E01 Experienced           75 François
      E02 Experienced           75   Amélie
      E03 Experienced           75 Baptiste
      E04 Experienced           75   Céline
      E05 Experienced           75   Damien
      E06 Experienced           75    Élise
      J01      Junior           25    Jules
      J02      Junior           25   Margot
      J03      Junior           25     Théo
      J04      Junior           25  Camille
      J05      Junior           25      Léa
      J06      Junior           25     Hugo
      J07      Junior           25     Emma
      J08      Junior           25    Louis
      J09      Junior           25    Chloé
      J10      Junior           25    Lucas
      J11      Junior           25    Manon
      J12      Junior           25    Ethan


SHIFT REQUIREMENTS:
============================================================
Shift_ID      Shift_Name      Day        Time  Hours  Required_Servers
Min_Experienced  Expected_Covers  Penalty_Per_Missing
      FD   Friday Dinner   Friday 18:00-00:00      6                 3
2              120                   800
      FL     Friday Late   Friday 23:00-03:00      4                 3
0               40                     0
      SL  Saturday Lunch Saturday 11:00-15:00      4                 3
1               80                   500
      SD Saturday Dinner Saturday 17:00-21:00      4                 3
2              140                  1200
      UL    Sunday Lunch   Sunday 11:00-16:00      5                 3
1               90                   600
      UD   Sunday Dinner   Sunday 17:00-22:00      5                 3
2              130                  1000


THE PROBLEM:
============================================================
Total experienced servers needed: 8
Total experienced servers available: 6
SHORTAGE: 2 experienced servers!

Complex factors:
 • HARD CONSTRAINTS: Some servers cannot work certain shifts (pref=0)
 • Varying shift lengths (4-6 hours) affect total labor cost
 • Large penalty amounts (€500-€1200) make choices strategic
 • Server preferences (1-10) affect morale and quality
```

- Lower preferences = unhappy staff = significant quality penalty
- Availability constraints create interdependencies between assignments

Understanding the Costs

```
# Visualize the requirements and cost impacts
fig, axes = plt.subplots(2, 2, figsize=(14, 10))

# Top Left: Shift requirements
ax = axes[0, 0]
x = range(len(shifts))
width = 0.35
ax.bar([i - width/2 for i in x], shifts['Required_Servers'], width,
       label='Total Needed', color='gray', alpha=0.7)
ax.bar([i + width/2 for i in x], shifts['Min_Experienced'], width,
       label='Experienced Needed', color='red', alpha=0.7)
ax.set_xlabel('Shift')
ax.set_ylabel('Number of Servers')
ax.set_title('Server Requirements by Shift')
ax.set_xticks(x)
ax.set_xticklabels(shifts['Shift_ID'])
ax.legend()
ax.axhline(y=6, color='green', linestyle='--', label='Available
Experienced')
ax.text(2.5, 6.2, 'Max Available', fontsize=10, color='green', ha='center')
ax.grid(True, alpha=0.3)

# Expected covers (business volume)
ax = axes[0, 1]
colors = ['red' if covers > 100 else 'orange' if covers > 60 else 'green'
          for covers in shifts['Expected_Covers']]
bars = ax.bar(shifts['Shift_ID'], shifts['Expected_Covers'], color=colors,
alpha=0.7)
ax.set_xlabel('Shift')
ax.set_ylabel('Expected Covers (Guests)')
ax.set_title('Business Volume by Shift')
ax.grid(True, alpha=0.3)

# Add value labels
for bar, val in zip(bars, shifts['Expected_Covers']):
    ax.text(bar.get_x() + bar.get_width()/2, bar.get_height() + 2,
            str(val), ha='center', va='bottom')

# Penalty structure
ax = axes[1, 0]
penalties = shifts['Penalty_Per_Missing'].values
bars = ax.bar(shifts['Shift_ID'], penalties, color=['red' if p > 0 else
'green' for p in penalties], alpha=0.7)
ax.set_xlabel('Shift')
ax.set_ylabel('Penalty per Missing Experienced (€)')
ax.set_title('Understaffing Penalties')
ax.grid(True, alpha=0.3)
```

```
# Add annotations
for bar, val in zip(bars, penalties):
    if val > 0:
        ax.text(bar.get_x() + bar.get_width()/2, bar.get_height() + 5,
                f'€{val}', ha='center', va='bottom', fontweight='bold')

plt.tight_layout()
plt.show()
```



Helper Functions

```
def calculate_schedule_cost(schedule: Dict[str, List[str]],
                            servers_df: pd.DataFrame,
                            shifts_df: pd.DataFrame) -> Tuple[float, Dict]:
    """
    Calculate total cost of a schedule including labor, penalties, and
preference scores.

    Returns:
        total_cost: Total cost in euros
        cost_breakdown: Dictionary with detailed cost breakdown
    """
    labor_cost = 0
    penalty_cost = 0
    preference_penalty = 0
    availability_violation_penalty = 0
    violations = []
```

```
    preference_score = 0
    availability_violations = 0

    for shift_idx, shift_row in shifts_df.iterrows():
        shift_id = shift_row['Shift_ID']

        if shift_id not in schedule:
            continue

        assigned_servers = schedule[shift_id]

        # Calculate labor cost for this shift
        for server_id in assigned_servers:
            server = servers_df[servers_df['Server_ID'] ==
server_id].iloc[0]
            labor_cost += server['Hourly_Rate'] * shift_row['Hours']

            # Check availability constraints
            server_pref = pref_matrix.loc[server_id, shift_id]

            if server_pref == 0:
                # CRITICAL: Cannot assign unavailable servers!
                # Massive penalty to make this worse
                availability_violation_penalty += 10000
                availability_violations += 1
            else:
                # Penalty for low preferences (encourage happy staff)
                # Scale: 1 (worst) = €180 penalty, 10 (best) = €0 penalty
                preference_penalty += (10 - server_pref) * 20
                preference_score += server_pref

        # Check experienced server requirement
        experienced_count = sum(1 for sid in assigned_servers
                                if servers_df[servers_df['Server_ID'] ==
sid].iloc[0]['Type'] == 'Experienced')

        shortage = max(0, shift_row['Min_Experienced'] - experienced_count)
        if shortage > 0:
            penalty = shortage * shift_row['Penalty_Per_Missing']
            penalty_cost += penalty
            violations.append({
                'shift': shift_row['Shift_Name'],
                'needed': shift_row['Min_Experienced'],
                'assigned': experienced_count,
                'shortage': shortage,
                'penalty': penalty
            })

    total_cost = labor_cost + penalty_cost + preference_penalty +
availability_violation_penalty

    cost_breakdown = {
        'labor_cost': labor_cost,
        'penalty_cost': penalty_cost,
```

```
        'preference_penalty': preference_penalty,
        'availability_violation_penalty': availability_violation_penalty,
        'availability_violations': availability_violations,
        'total_cost': total_cost,
        'violations': violations,
        'average_preference': preference_score / (18 -
availability_violations) if (18 - availability_violations) > 0 else 0
    }

    return total_cost, cost_breakdown

def validate_schedule(schedule: Dict[str, List[str]],
                      servers_df: pd.DataFrame,
                      shifts_df: pd.DataFrame) -> Tuple[bool, List[str]]:
    """
    Validate that a schedule meets all hard constraints.

    Returns:
        is_valid: Boolean indicating if schedule is valid
        errors: List of constraint violations
    """
    errors = []

    # Check each shift has exactly 3 servers
    for shift_row in shifts_df.itertuples():
        if shift_row.Shift_ID not in schedule:
            errors.append(f"Shift {shift_row.Shift_Name} has no
assignment")
        elif len(schedule[shift_row.Shift_ID]) !=
shift_row.Required_Servers:
            errors.append(f"Shift {shift_row.Shift_Name} has
{len(schedule[shift_row.Shift_ID])} servers, needs
{shift_row.Required_Servers}")

    # Check each server works exactly once
    all_assignments = []
    for shift_servers in schedule.values():
        all_assignments.extend(shift_servers)

    server_counts = pd.Series(all_assignments).value_counts()

    # Check no duplicates
    for server_id, count in server_counts.items():
        if count > 1:
            errors.append(f"Server {server_id} assigned {count} times
(should be 1)")

    # Check all servers are assigned
    all_server_ids = set(servers_df['Server_ID'])
    assigned_ids = set(all_assignments)

    if len(assigned_ids) != len(all_server_ids):
        missing = all_server_ids - assigned_ids
        if missing:
```

```
            errors.append(f"Servers not assigned: {', '.join(missing)}")

    # Check availability constraints (preference = 0 means cannot work)
    for shift_id, assigned_servers in schedule.items():
        for server_id in assigned_servers:
            pref = pref_matrix.loc[server_id, shift_id]
            if pref == 0:
                shift_name = shifts_df[shifts_df['Shift_ID'] ==
shift_id].iloc[0]['Shift_Name']
                errors.append(f"AVAILABILITY VIOLATION: {server_id}
assigned to {shift_name} but is unavailable (preference=0)")

    is_valid = len(errors) == 0
    return is_valid, errors

def visualize_schedule(schedule: Dict[str, List[str]],
                       servers_df: pd.DataFrame,
                       shifts_df: pd.DataFrame,
                       cost_breakdown: Dict):
    """Create a visual representation of the schedule."""

    fig, axes = plt.subplots(2, 2, figsize=(14, 10))

    # 1. Schedule Grid
    ax = axes[0, 0]

    # Create schedule matrix
    schedule_matrix = np.zeros((len(servers_df), len(shifts_df)))

    for shift_idx, shift_row in shifts_df.iterrows():
        shift_id = shift_row['Shift_ID']
        if shift_id in schedule:
            for server_id in schedule[shift_id]:
                server_idx = servers_df[servers_df['Server_ID'] ==
server_id].index[0]
                schedule_matrix[server_idx, shift_idx] = 1

    # Color based on server type
    colors = []
    for server in servers_df.itertuples():
        if server.Type == 'Experienced':
            colors.append('red')
        else:
            colors.append('blue')

    # Plot
    im = ax.imshow(schedule_matrix, cmap='YlOrRd', aspect='auto')
    ax.set_xticks(range(len(shifts_df)))
    ax.set_xticklabels(shifts_df['Shift_ID'])
    ax.set_yticks(range(len(servers_df)))
    ax.set_yticklabels([f"{row.Server_ID} ({row.Type[0]})" for row in
servers_df.itertuples()],
                       fontsize=8)
    ax.set_xlabel('Shift')
```

```
    ax.set_ylabel('Server')
    ax.set_title('Schedule Assignment Matrix')

    # Add grid
    for i in range(len(servers_df) + 1):
        ax.axhline(i - 0.5, color='gray', linewidth=0.5)
    for i in range(len(shifts_df) + 1):
        ax.axvline(i - 0.5, color='gray', linewidth=0.5)

    # 2. Cost Breakdown
    ax = axes[0, 1]
    costs = {
        'Labor': cost_breakdown['labor_cost'],
        'Penalties': cost_breakdown['penalty_cost']
    }
    bars = ax.bar(costs.keys(), costs.values(), color=['green', 'red'],
alpha=0.7)
    ax.set_ylabel('Cost (€)')
    ax.set_title(f"Total Cost: €{cost_breakdown['total_cost']:.2f}")

    for bar, val in zip(bars, costs.values()):
        ax.text(bar.get_x() + bar.get_width()/2, bar.get_height() + 10,
                f'€{val:.0f}', ha='center', va='bottom', fontweight='bold')

    # 3. Violations
    ax = axes[1, 0]
    if cost_breakdown['violations']:
        violations_df = pd.DataFrame(cost_breakdown['violations'])
        x = range(len(violations_df))
        ax.bar(x, violations_df['shortage'], color='red', alpha=0.7)
        ax.set_xticks(x)
        ax.set_xticklabels(violations_df['shift'], rotation=45, ha='right')
        ax.set_ylabel('Missing Experienced Servers')
        ax.set_title('Understaffing by Shift')
        ax.grid(True, alpha=0.3)
    else:
        ax.text(0.5, 0.5, 'No Violations!\n(All requirements met)',
                ha='center', va='center', fontsize=14, color='green')
        ax.set_title('Violations')
    ax.axis('on')

    # 4. Server Type Distribution
    ax = axes[1, 1]
    shift_composition = []
    for shift_id in shifts_df['Shift_ID']:
        if shift_id in schedule:
            exp_count = sum(1 for sid in schedule[shift_id]
                            if servers_df[servers_df['Server_ID'] ==
sid].iloc[0]['Type'] == 'Experienced')
            jun_count = len(schedule[shift_id]) - exp_count
            shift_composition.append({'Shift': shift_id, 'Experienced':
exp_count, 'Junior': jun_count})

    if shift_composition:
```

```
        comp_df = pd.DataFrame(shift_composition)
        x = range(len(comp_df))
        width = 0.35
        ax.bar([i - width/2 for i in x], comp_df['Experienced'], width,
               label='Experienced', color='red', alpha=0.7)
        ax.bar([i + width/2 for i in x], comp_df['Junior'], width,
               label='Junior', color='blue', alpha=0.7)
        ax.set_xticks(x)
        ax.set_xticklabels(comp_df['Shift'])
        ax.set_ylabel('Number of Servers')
        ax.set_title('Staff Composition by Shift')
        ax.legend()
        ax.grid(True, alpha=0.3)

    plt.tight_layout()
    plt.show()

# Test with a random valid schedule
def create_random_schedule(servers_df, shifts_df):
    """Create a random valid schedule for testing."""
    schedule = {}
    available_servers = list(servers_df['Server_ID'])
    random.shuffle(available_servers)

    server_idx = 0
    for shift_row in shifts_df.itertuples():
        schedule[shift_row.Shift_ID] =
available_servers[server_idx:server_idx + shift_row.Required_Servers]
        server_idx += shift_row.Required_Servers

    return schedule

# Test the helper functions
test_schedule = create_random_schedule(servers, shifts)
test_cost, test_breakdown = calculate_schedule_cost(test_schedule, servers,
shifts)
is_valid, errors = validate_schedule(test_schedule, servers, shifts)

print(f"\nRANDOM SCHEDULE TEST:")
print(f"Valid: {is_valid}")
print(f"Total Cost: €{test_cost:.2f}")
print(f"  - Labor: €{test_breakdown['labor_cost']:.2f}")
print(f"  - Penalties: €{test_breakdown['penalty_cost']:.2f}")
if test_breakdown['violations']:
    print(f"Violations: {len(test_breakdown['violations'])} shifts
understaffed")
```

```
RANDOM SCHEDULE TEST:
Valid: False
Total Cost: €58140.00
  - Labor: €3400.00
```

```
  - Penalties: €3800.00
Violations: 3 shifts understaffed
```

Your Task

Design the optimal server schedule that minimizes total cost while managing multiple complex constraints.

The Challenge

This is NOT a simple assignment problem! You must balance:

- Varying shift lengths (4-6 hours) that affect labor costs differently
- Server preferences (1-10) where lower values mean unhappy staff and quality issues
- Strategic penalty decisions - penalties range from €0 to €1200
- Three cost components: labor + experience penalties + preference penalties

Approach Options

You can use ANY approach you prefer:

1. Smart Greedy - Consider preferences AND penalties in your rules
2. Local Search - Start with valid schedule (respecting 0 preferences), improve with constraint-aware swaps
3. Metaheuristics - Use AI to implement SA, GA, etc. that handle hard constraints
4. Hybrid - Combine approaches (e.g., greedy for feasibility, then SA for optimization)
5. Creative - Surprise me with novel constraint-handling techniques!

Remember: Unhappy staff (low preferences) lead to quality issues and small penalties that add up!

Your Solution

Implement your solution below. You can use greedy, metaheuristic, or hybrid approaches.

AI-Assisted Metaheuristic Template

```
# Template for using AI to implement metaheuristics
"""
You are an expert in constraint-handling techniques for metaheuristic
problems.

I need to solve a complex [PROBLEM NAME] using [ALGORITHM NAME].

Problem details: [PROVIDE ALL PROBLEM DETAILS HERE]

Constraints: [PROVIDE ALL CONSTRAINTS YOU NEED TO MEET HERE]

Objective: [PROVIDE THE OBJECTIVE AND ALL POSSIBLE PENALTIES]
```

```
Data: [PROVIDE DATA OR SAVE DATA AS CSV AND LINK FILES]

Please assist me and provide:
1. Solution representation
2. Neighborhood/move function that considers [E.G. INFEASIBILITY, ETC.]
3. Cost calculation including all components
4. Main algorithm loop with optimization
5. Parameter recommendations for this specific problem

Plan first and discuss all open questions with me before the
implementation.
"""
```

```
'\nYou are an expert in constraint-handling techniques for metaheuristic
problems.\n\nI need to solve a complex [PROBLEM NAME] using [ALGORITHM
NAME].\n\nProblem details: [PROVIDE ALL PROBLEM DETAILS
HERE]\n\nConstraints: [PROVIDE ALL CONSTRAINTS YOU NEED TO MEET
HERE]\n\nObjective: [PROVIDE THE OBJECTIVE AND ALL POSSIBLE
PENALTIES]\n\nData: [PROVIDE DATA OR SAVE DATA AS CSV AND LINK
FILES]\n\nPlease assist me and provide:\n1. Solution representation\n2.
Neighborhood/move function that considers [E.G. INFEASIBILITY, ETC.]\n3.
Cost calculation including all components\n4. Main algorithm loop with
optimization\n5. Parameter recommendations for this specific
problem\n\nPlan first and discuss all open questions with me before the
implementation.\n'
```

Step 1: Implementation

Implement your solution below:

```
# YOUR SOLUTION CODE HERE

def my_scheduling_algorithm(servers_df, shifts_df):
    """
    Your scheduling algorithm implementation.
    Can be greedy, metaheuristic, or hybrid.
    """
    # YOUR CODE HERE

    # ... your implementation ...

    # For now, returning a random solution as placeholder
    return create_random_schedule(servers_df, shifts_df)

# Run your solution
my_schedule = my_scheduling_algorithm(servers, shifts)
my_cost, my_breakdown = calculate_schedule_cost(my_schedule, servers,
shifts)

# Validate
is_valid, validation_errors = validate_schedule(my_schedule, servers,
shifts)
```

```python
print("MY SOLUTION:")
print("=" * 60)
if not is_valid:
    print("INVALID SCHEDULE:")
    for error in validation_errors:
        print(f"  - {error}")
else:
    print("Valid Schedule")

print(f"\nTotal Cost: €{my_cost:.2f}")
print(f"  Labor: €{my_breakdown['labor_cost']:.2f}")
print(f"  Penalties: €{my_breakdown['penalty_cost']:.2f}")

print(f"\nImprovement over random: {(test_cost - my_cost)/
test_cost*100:.1f}%")

# Visualize your solution
if is_valid:
    visualize_schedule(my_schedule, servers, shifts, my_breakdown)
```

```
MY SOLUTION:
============================================================
INVALID SCHEDULE:
  - AVAILABILITY VIOLATION: E03 assigned to Friday Dinner but is
unavailable (preference=0)
  - AVAILABILITY VIOLATION: E06 assigned to Friday Late but is unavailable
(preference=0)
  - AVAILABILITY VIOLATION: E01 assigned to Sunday Lunch but is unavailable
(preference=0)
  - AVAILABILITY VIOLATION: J08 assigned to Sunday Dinner but is
unavailable (preference=0)

Total Cost: €47220.00
  Labor: €3600.00
  Penalties: €2700.00

Improvement over random: 18.8%
```

Step 2: Create Your Submission

Prepare a one-slide presentation (PDF) containing:

- Your Best Schedule: Total cost prominently displayed (labor + penalties + preferences)
- Approach: Which solution generation method(s) did you use?
- Cost Breakdown: Visualization showing the three cost components (labor, experience penalties, preference penalties)
- Shift Assignments: Table showing each shift with experienced vs junior server counts
- Strategy Justification: 2-3 sentences explaining why your schedule is optimal

Tips for Success

1. Start simple: Try random generation or greedy assignment first to understand the solution space
2. Check feasibility: Always verify availability constraints (preference $\neq 0$) before assigning
3. Preferences matter: Low preference scores create unhappy staff and add €180+ in penalties
4. Strategic penalties: Some shifts have €0 penalties (Friday Late), use these wisely
5. Balance all three costs: Labor, experience penalties, AND preference penalties all matter

Common Pitfalls to Avoid

- Violating availability constraints: Assigning servers to shifts where preference = 0 triggers massive penalties
- Ignoring preferences: Low-preference assignments (1-3) add up quickly in costs
- Missing the shortage: You need 8 experienced servers but only have 6, plan which penalties to accept
- Incorrect cost calculation: Must include all three components: labor + experience penalties + preference penalties

Final Checklist

Before submitting your solution, verify:

- ☐ Schedule assigns each server exactly once
- ☐ Each shift has exactly 3 servers
- ☐ No availability violations (preference = 0 assignments)
- ☐ Cost calculation includes all three components
- ☐ Justification explains your strategic penalty decisions (2-3 sentences)
- ☐ One-slide presentation ready with cost breakdown and shift assignments

Good Luck!

May the best schedule win!

Bibliography