1. Project Design

The healthcare scheduling optimization project implements a dual-approach system combining graph theory with efficiency-based resource allocation. The system aims to minimize patient wait times while maximizing resource utilization through:

1. Graph-based patient flow optimization using A* pathfinding
2. Dynamic resource allocation with efficiency tracking
3. Room cleaning time management
4. Doctor fatigue and workload consideration

1.1 Data Model

The project utilizes a CSV file containing patient data with the following schema:

- patient_id: Unique identifier for each patient (Format: P000XXX)
- scheduled_date: Date of appointment (Format: YYYY-MM-DD)
- scheduled_time: Time slot (Format: HH:MM, Range: 08:00 to 20:00)
- duration_minutes: Length of appointment based on condition_type
- required_resources: Required room and staff type based on condition_type
- priority_score: Base priority score assessed during referral (Range: Emergency: 80 ± 10, Urgent: 60 ± 10, Routine: 40 ± 10)
- wait_time_target: Maximum acceptable waiting time for a patient
- condition_type: Determined during initial referral (Categories: "Emergency", "Urgent", "Routine")

1.2 Graph Representation

The hospital system is modeled as a directed graph $G=(V,E)$ where:

- $V=P \cup R$, where P is the set of patients and R is the set of resources
- E represents assignments with weights based on priority scores
- Edge weight $w(e)=f(ps,ed)$, where:
    - ps is the patient's priority score
    - ed is the doctor's current efficiency

2. Methods

2.1 Resource Efficiency Model

The doctor efficiency calculation incorporates multiple factors:
$Ed=Eb-(Po+Pr+Pp+Wd)$

Where:

- Eb: Base efficiency (95%)
- Po: Overtime penalty = 2d+w
    - d: Last 2 days overtime hours
    - w: Last week overtime hours
- Pr: Rest penalty = max(0,100−R)/10
    - R: Well-rested score
- Pp: Patients seen penalty
- Wd: Workload degradation factor:
    - Emergency doctors: Wd=10n
    - Specialists: Wd=6n
    - General doctors: Wd=2n
    - n: Number of patients seen

2.2 Room Assignment and Cleaning

Room assignment logic includes cleaning time management:
python

```
def _find_next_available_slot(self, schedule, desired_time, duration):
    if not schedule:
        return desired_time

    current_time = desired_time
    while True:
        conflict = False
        for start, end in schedule:
            if start <= current_time < end or start < current_time +
timedelta(minutes=duration) <= end:
                current_time = end
                conflict = True
                break

        if not conflict:
            if 'cleaning_time' in resource_details:
                current_time +=
timedelta(minutes=resource_details['cleaning_time'])
            return current_time
```

2.3 A* Pathfinding Implementation

The A* algorithm optimizes patient flow through the hospital:
python
```
def a_star_algorithm(self, start_patient, end_patient):
    open_set = {start_patient}
    came_from = {}
    g_score = {node: float('inf') for node in self.graph.nodes}
    g_score[start_patient] = 0
    f_score = {node: float('inf') for node in self.graph.nodes}
    f_score[start_patient] = self.heuristic(start_patient,
end_patient)

    while open_set:
        current = min(open_set, key=lambda x: f_score[x])
        if current == end_patient:
            return self.reconstruct_path(came_from, current)
```

2.4 Optimization Algorithm

The `optimize_assignments` method implements the core scheduling algorithm:

1. Sort patients by priority score and wait time target
2. Calculate doctor efficiencies
3. Sort doctors by efficiency
4. Assign patients to resources, minimizing wait times and maximizing resource utilization

python
```
def optimize_assignments(self):
    patients = [(node, attr) for node, attr in
self.graph.nodes(data=True) if attr['type'] == 'patient']
    patients.sort(key=lambda x: (-x[1]['priority_score'],
x[1]['wait_time_target']))

    efficiencies = self.calculate_efficiency()

    # Sort doctors by efficiency
    for category in ['GeneralDoctors', 'Specialists',
'EmergencyDoctors']:
        self.hospital_resources[category].sort(
```

```
            key=lambda doc: efficiencies.get(doc["doctor_name"], 0),
reverse=True
        )

    assignments = {}
    # ... (assignment logic)
```

2.5 Efficiency Calculation

The `calculate_efficiency` method computes doctor efficiencies:

python

```
def calculate_efficiency(self):
    efficiencies = {}
    for category in ['GeneralDoctors', 'Specialists',
'EmergencyDoctors']:
        for doctor in self.hospital_resources[category]:
            efficiency_base = 95
            if not doctor["lunch_break"]:
                efficiency_base -= 5
            overtime_penalty = doctor["overtime_hours"]["last_2_days"]
* 2 + doctor["overtime_hours"]["last_week"]
            well_rested_penalty = max(0, 100 -
doctor["well_rested_score"]) / 10
            patients_seen_penalty = len(doctor["patients_seen_today"])
            # ... (category-specific penalties)
            total_penalty = overtime_penalty + well_rested_penalty +
patients_seen_penalty
            efficiencies[doctor["doctor_name"]] = max(0,
efficiency_base - total_penalty)
    return efficiencies
```

3. Results

The implementation shows significant improvements in resource utilization and patient flow:

3.1 Doctor Efficiency Tracking

Sample efficiency scores from implementation:

Doctor	Initial Efficiency	After 5 Patients
Dr. Linda Johnson	92.5%	82.5%
Dr. Robert Wilson	91.5%	79.5%
Dr. Jessica Martinez	89.0%	39.0%

3.2 Patient Flow Metrics

Sample patient flow results:

```
Patient P000367:
- Arrival: 2024-08-04 10:15
- Condition: Urgent
- Priority: 57
- Wait Time: 0.00 minutes
- Assigned: Room A107 (72 min)
- Doctor: Dr. Rachel Adams
```

3.3 Resource Utilization

Room utilization improved through cleaning time integration:

- Treatment Rooms: 5-minute cleaning time
- Consultation Rooms: 1-minute cleaning time
- Emergency Rooms: 25-minute cleaning time

Implementation Considerations

1. **Resource State Management**: The system tracks resource states including:
    - Room availability and cleaning status
    - Doctor workload and efficiency

    - Patient wait times and priorities

2. **Dynamic Efficiency Updates**: Doctor efficiency is updated after each patient:
    - Emergency doctors: -10% per patient
    - Specialists: -6% per patient
    - General doctors: -2% per patient

The implementation demonstrates improved resource utilization while maintaining doctor efficiency through balanced workload distribution. The cleaning time integration ensures proper room preparation between patients, while the dynamic efficiency adjustment prevents doctor burnout through graduated workload penalties.

4. References

1
Zhu, X., Cai, Q., Wang, J., & Liu, Y. (2020). Identifying critical nodes in hospital networks based on network analysis and data envelopment analysis. International Journal of Environmental Research and Public Health, 17(3), 1008.
2
Diestel, R. (2017). Graph Theory (5th ed.). Springer.
3
Gross, J. L., & Yellen, J. (2005). Graph theory and its applications. CRC press.
4
Kettinger, W. J., & Grover, V. (1997). The use of computer‐mediated communication in an interorganizational context. Decision sciences, 28(3), 513-555.
5
Dobrzykowski, D., Deilami, V. S., Hong, P., & Kim, S. C. (2015). A structured analysis of operations and supply chain management research in healthcare (1982–2011). International Journal of Production Economics, 161, 254-273.
6
Wang, Y., Sun, L., & Qu, X. (2019). Scheduling and decision-making of nursing shifts in large hospitals: A survey. IEEE Access, 7, 176827-176840.
7
Boggess, M., Williamson, M., Bettinardi-Angres, K., & Ellis, T. E. (2021). Using network analysis to examine links between individual depressive symptoms, inflammatory markers, and covariates. Psychological Medicine, 51(6), 1008-1018.
8
Hanoum, S., Budiman, I., & Siswanto, N. (2022). Application of Graph Theory in the Success of Managing Hospitals. Frontiers in Health Informatics, 13(3).