IND ENG 174 Final Report
Simulation Model for ICU

Kenny Chan
Student ID: 3040782826
kennychan@berkeley.edu

Sizhe Li
Student ID: 3040838258
sizheli@berkeley.edu

Siqi Yao
Student ID: 304798244
siqiyao2024@berkeley.edu

Yiyao Li
Student ID: 3040803340
yiyaoli@berkeley.edu

Project Overview and Objectives

Our project addresses the challenge of resource optimization within an Intensive Care Unit (ICU), where high patient demand and constrained capacity—particularly in terms of bed availability and staffing—create persistent operational difficulties. To tackle this, our aim is to develop a queuing simulation model based on MIMIC-IV demo dataset [1].

Within this framework, we categorizes patients by severity and urgency of treatment to reflect real-world hospital needs. Time-based penalties reflective of patient severity and urgency of treatment are incorporated to evaluate the performance of different queuing strategies. Based on this, we developed three distinct simulation optimization solvers to optimize the allocation of beds and servers. Additionally, we conducted a sensitivity analysis to ensure the robustness of our models and indicate the potential directions for future improvement.

Code and Materials Access

All of our code, materials, and videos related to this project can be accessed via the following links:

- Google Drive Folder
- GitHub Repository

Problem Setting

The ICU system we model involves complex patient flow and resource allocation processes. These processes are influenced by dynamic factors such as fluctuating patient arrival rates and varying levels of care request based on patient severity. To illustrate the overall structure of our problem, we provide a flowchart that outlines the key elements of our simulation framework (see Figure 1).

- **Patient Arrival Process**: Patients arrive at the ICU based on fluctuating demand throughout the day, denoted by varying arrival rates $\lambda(t)$. For example, during mid-morning hours, the arrival rate tends to be higher as many patients are transferred from operating rooms or other facilities following morning surgeries. Conversely, during late evening and early morning hours, the arrival rate is typically lower due to reduced hospital activity and fewer scheduled procedures. These patients have different severity levels, denoted by k (with $k = 1$ representing "Mild", $k = 2$ representing "Moderate" and $k = 3$ representing "Severe"), influencing their care requests and lengths of stay in the ICU.
- **ICU Capacity Management**: The ICU can only accommodate a limited number of patients at a time. When the ICU reaches capacity C, new patients must wait at least until a bed becomes available.

- **Caregiver Service Process**: Each patient who are already in the ICU requires a certain level of service, which is categorized into small-scale, medium-scale, and large-scale tasks. The likelihood of requesting larger-scale tasks is higher for more severe patients. For patients with the same severity, the likelihood of requesting different tasks varies across different stages of stay. Their care requests are not deterministic but rather probabilistic, reflecting a tendency rather than a certainty. For instance, a critically ill patient may have a higher chance of requesting large-scale services like ventilator management, while less severe cases might primarily require small-scale tasks such as routine monitoring. Similarly, during the early stages of hospitalization, patients are often in more critical condition and may be more likely to frequently request large-scale tasks. However, as they approach the recovery phase closer to discharge, their condition tends to improve significantly, making them more inclined to request medium- or small-scale tasks. Throughout their stay, patients can repeatedly request services as their conditions evolve, leading to a dynamic and fluctuating demand on the limited number of Caregivers, N, responsible for these tasks. When all caregivers are occupied, patients requesting services may experience delays, which can impact their outcomes depending on the level of their care requests.

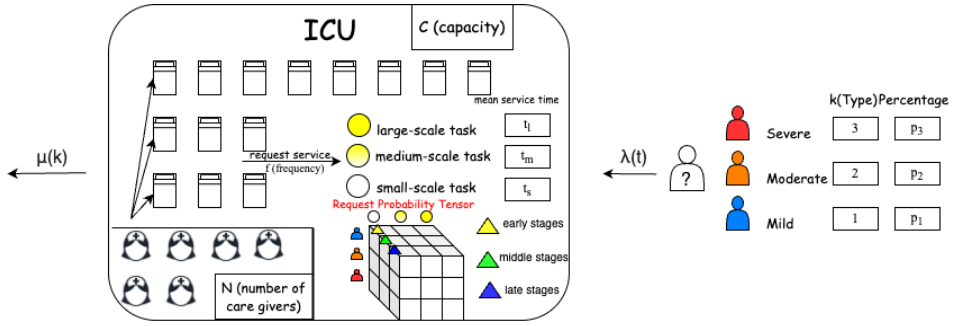

Figure 1: Flowchart of Problem Setting

Challenges

The complexity of ICU operations stems from its inherently dynamic and unpredictable nature. Unlike simpler systems that can be effectively modeled using static assumptions, ICU environments demand a more nuanced approach to capture their real-world behavior. This is due to the continuous evolution of patient states, the variability in service requests, and the intricate interplay between resource availability and patient outcomes. Our work focuses on addressing these challenges by developing a robust simulation framework that accounts for the following key difficulties:

- **Dynamic Nature of ICU Service Processes**
  The real-time fluctuation in the number of patients within the ICU complicates the service processes. The state and needs of different patients evolve over time, introducing dynamic elements that are challenging to capture using static models.

- **Frequent and Recurrent Service Requests**
  Patients who have completed their service can request additional services without restrictions and potentially at a high frequency. At certain time points, patients currently being served or waiting for service are unable to submit new service requests. This dynamic behavior deviates from traditional M/M/c queuing models, making it difficult to fully model the ICU process. Consequently, this adds complexity to the simulation, requiring consideration of dynamic queuing and unpredictable service requests.

Implementation

To address the challenges inherent in ICU operations, we have developed a simulation model that allows users to customize various parameters to capture the specifics of their own settings. This flexibility ensures that our model can adapt to different ICU environments, reflecting the unique dynamics of patient flow and resource allocation (see Table 1).

Symbol	Definition
T	Time horizon
$\lambda(t)$	Arrival rates, varies by time of day
k	Severity level of a patient (1-Mild, 2-Moderate, 3-Severe)
$p[=(p_1, p_2, p_3)]$	Probability distribution of severity
C	ICU bed capacity
N	Total number of caregivers
$t[=(t_s, t_m, t_l)]$	Mean service time
A	Request probability tensor
f	Request frequency
b_1	The cutoff point between the early and middle stages
b_2	The cutoff point between the middle stages and late stages
$\mu(k)$	departure rates(1/Length of Stay), varies by severity
m, α	Parameters in the penalty function

Table 1: Parameters

Sequential Simulation Approach

We observed that the Caregiver Service Process depends on the outcomes of the ICU Queue, which allows for a sequential simulation approach. Specifically, by first simulating the ICU Queue (part 1), we can establish the patient flow and queue dynamics, which serve as the input for simulating the Caregiver Process (part 2). This approach not only simplifies the complexity of the overall system but also enables us to analyze each component in detail while preserving their interconnected behavior.

Simulation details

Part1: ICU Queue

The ICU queue simulation model, which follows a classic M/M/c queuing framework, is designed to handle patient arrivals, assign severity levels, simulate patient departure times. This model captures the stochastic nature of patient arrivals and service times, leveraging exponential distributions to reflect real-world dynamics in a multi-server setting.

- **Arrival Process Simulation**: The model uses an **acceptance-rejection method** to simulate this arrival process. We set variable **arrival rates** at different times of the day, capturing peak hours between 9 AM and 12 PM and lower demand between 8 PM and 4 AM to reflect real-world fluctuations in patient arrivals. Upon arrival, each patient is assigned a **severity level** (Mild, Moderate, or Severe) based on predefined probabilities p, representing a realistic distribution of case types where moderate cases are the most common, followed by severe and mild cases.

- **Departure Process Simulation**: Patients' **lengths of stay** are drawn from an exponential distribution based on their severity level, with average stays of 3, 7, and 15 days for mild, moderate, and severe cases, respectively. To manage the ICU's **limited capacity**, a **min-heap structure** maintains the departure times of currently admitted patients. When the ICU reaches capacity, incoming patients must wait at least until a bed becomes available. In these cases, a patient's **start time** is determined either by their arrival time or by the earliest available ICU bed. Based on our model setting, we simulated different queuing strategies:

  (0) FIFO (baseline): Patients are admitted and discharged strictly based on their order of arrival, regardless of severity level.

  (1) Priority Queue: Patients are prioritized based on their severity level, more severe cases receiving earlier access to ICU resources.

  (2) Reserved Beds: A small number of ICU beds are reserved exclusively for the most severe cases. These reserved beds remain unused unless all regular beds are occupied.

  (3) Priority Queue + Reserved Beds: This strategy combines the benefits of reserved beds and severity-based prioritization.

(4) Dynamic Priority Queue: Patients' priorities are dynamically adjusted by a heuristic function, which is based on their waiting time and severity level. This prevents milder cases from being indefinitely delayed while ensuring that critically ill patients remain the top priority.

(5) Dynamic Priority Queue + Reserved Beds: This strategy combines the benefits of reserved beds and severity-based prioritization.

The following figure is a diagram illustrating these strategies.

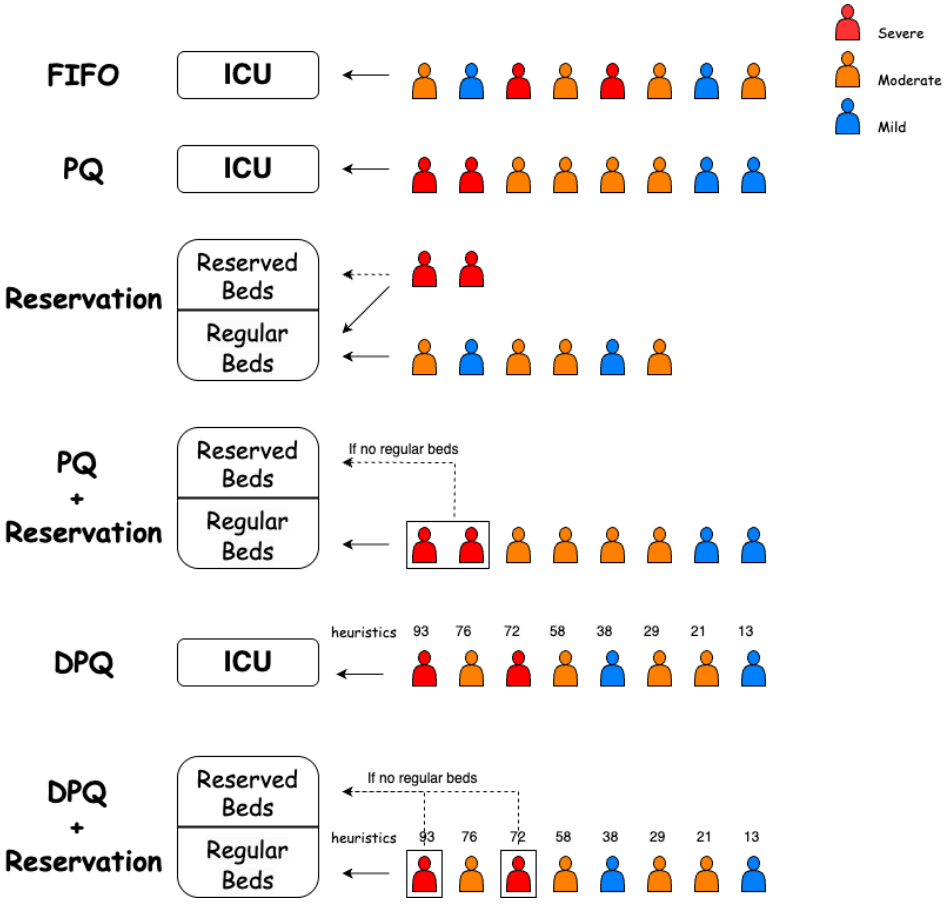

Figure 2: Graphical Illustration

Part2: Caregiver Process

The caregiver simulation model is designed to manage the workflow and service delivery of caregivers in an ICU, simulating the process of service requests and caregiver availability. Unlike a traditional M/M/c framework, this process incorporates dynamic and recurrent service requests, where patients can repeatedly request multiple types of services during their stay. Each patient's stay is divided into three distinct stages, and the distribution of service requests varies across these stages.

- **Data Structures**: The model utilizes an array **patient states** to track the status of each patient, where **state 0** indicates not currently in the ICU, **state 1** means the patient is in the ICU and is neither in service nor waiting to be served, and **state 2** indicates being in service or waiting to be served.(see Figure 3) **service start times** and **service end times** are recorded in lists to monitor when caregivers start and finish service for each hospital bed. Additionally, an **index list** maps patients to their assigned beds, while **service waiting times** record the waiting times for each service request along with **severity cor waiting times** recording severity levels corresponding to the waiting time. A **min-heap structure** is still used to get the **start time** for each patient's service request.

- **Simulation Loop**: The main simulation loop iterates over time, generating patient service requests based on an exponential distribution to mimic real-world scenarios. At each time increment, the model first updates the states of all patients based on the following transitions:
  - Patients entering the ICU transition from **state 0** to **state 1**.
  - Patients completing their service transition from **state 2** to **state 1**.
  - Patients being discharged transition from **state 1** or **state 2** to **state 0**.

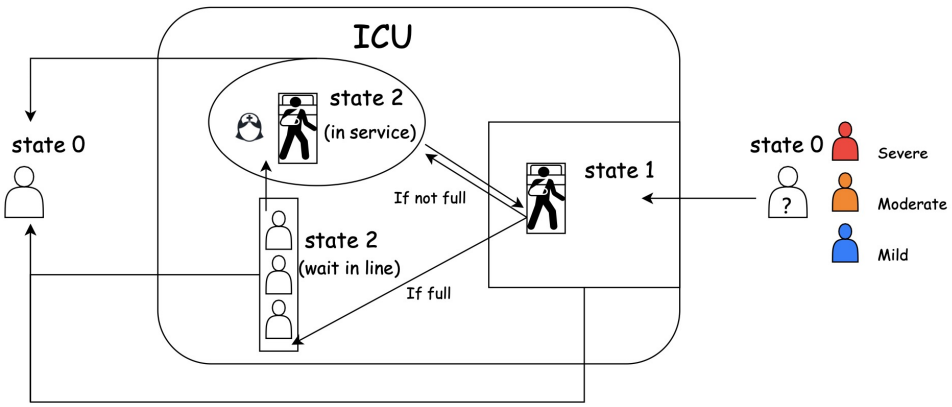

Figure 3: Patient States

Performance evaluation

To capture the loss of waiting times of patients before entering the ICU and the waiting times of ICU patients for services, associated with the severity of their conditions, we decided to introduce a penalty function, which provides a reasonable metric that reflects both the operational efficiency of the ICU and the impact on patient outcomes.

We have designed the penalty function in the following form:

$$\mathcal{P}_1 = m_1 \cdot \sum_{i}^{N_p} (e^{\alpha_1 \cdot k_i^{(I)} \cdot t_i^{(I)}} - 1)$$

$$\mathcal{P}_2 = m_2 \cdot \sum_{i}^{N_s} (e^{\alpha_2 \cdot k_i^{(II)} \cdot t_i^{(II)}} - 1)$$

$$\mathcal{P} = \mathcal{P}_1 + \mathcal{P}_2$$

where m_1, m_2 are the trade-off and scaling parameters, α_1, α_2 can be explained as the time-sensitivities, N_p is the total number of patients, N_s is the total number of services, $t_i^{(I)}$, $t_i^{(II)}$ are the waiting time of the i-th patient for getting admitted into ICU (in hours) and the patient that received the i-th service (in minutes), $k_i^{(I)}, k_i^{(II)}$ are the severity levels of the i-th patient and the patient that received the i-th service.

The rationale behind this penalty function lies in our observation that prolonged waiting times are particularly critical for patients awaiting ICU admission, as these delays can have life-threatening consequences. Therefore, we employ an exponential form to capture the escalating impact of longer waiting times, especially for patients with severe conditions. m_1, m_2 are chosen to balance the penalty 1 and penalty 2 in our baseline to avoid the focus on the only one process. Moreover, we aim to avoid disproportionately penalizing cases with minimal or zero waiting times, as doing so would shift the focus away from the scenarios we are most concerned with: those involving extended delays and high-severity patients. m_1, m_2 are chosen to balance the penalty 1 and penalty 2 in our baseline to avoid the focus on the only one process. This design ensures that our evaluation prioritizes the most critical cases, aligning the penalty with both the operational and clinical priorities of ICU management.

Results and Implications

We implement naive FIFO and 5 other queuing strategies mentioned above using the parameters listed below (see Table 2 and Table 13). Each scenario is simulated 100 times to ensure statistical robustness. The results and implications are presented as follows.

Basecase

Symbol	Values
T	10
$\lambda(t)$	$\begin{cases} 3.0, & \text{if } 9 \le t < 12 \\ 1.5, & \text{if } 12 \le t < 17 \\ 2.5, & \text{if } 17 \le t < 20 \\ 0.7, & \text{if } 20 \le t < 24 \text{ or } 0 \le t < 4 \\ 1.0, & \text{if } 4 \le t < 9 \end{cases}$
$p[= (p_1, p_2, p_3)]$	$(0.2, 0.5, 0.3)$
$t[= (t_s, t_m, t_l)]$	$(0.2, 0.5, 1)$
f	2
$\mu(k)$	$(\frac{1}{3}, \frac{1}{7}, \frac{1}{15})$
A	$\left[\begin{bmatrix} 0.1 & 0.3 & 0.6 \\ 0.3 & 0.5 & 0.2 \\ 0.6 & 0.3 & 0.1 \end{bmatrix}, \begin{bmatrix} 0.1 & 0.2 & 0.7 \\ 0.2 & 0.5 & 0.3 \\ 0.5 & 0.3 & 0.2 \end{bmatrix}, \begin{bmatrix} 0.1 & 0.1 & 0.8 \\ 0.2 & 0.3 & 0.5 \\ 0.5 & 0.3 & 0.2 \end{bmatrix}\right]$
b_1	0.2
b_2	0.9
m_1, α_1	1, 0.005
m_2, α_2	0.01, 0.1

Table 2: Parameters

We investigate FIFO under different settings of C (number of beds) and N (number of care givers).
(a)

Symbol	Values
C	100
N	50

Table 3: Parameters

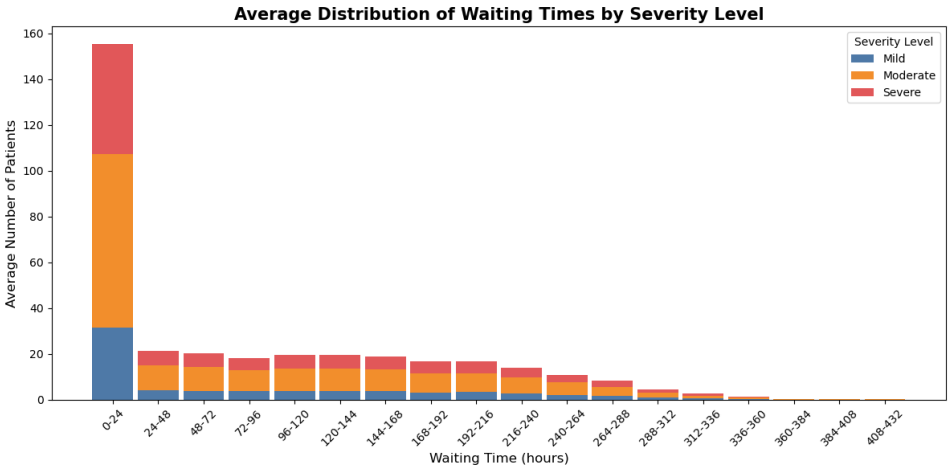

Figure 4: $C = 100$, $N = 50$ (Waiting times for ICU admission)

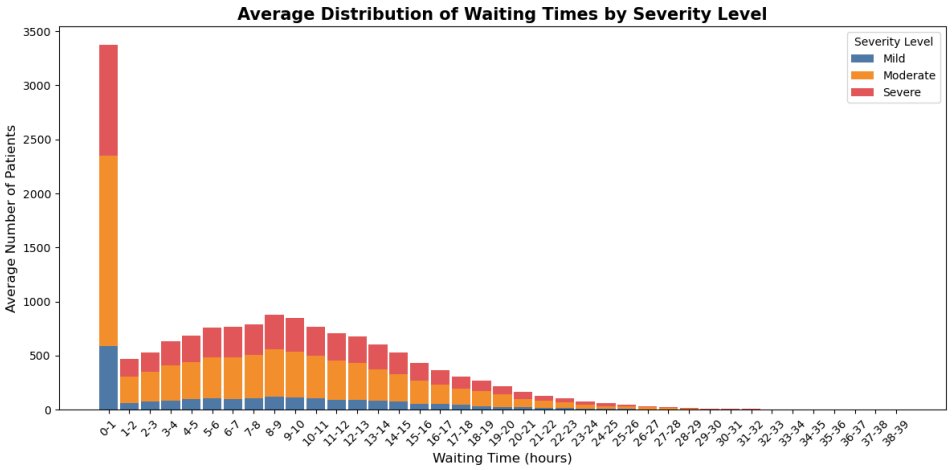

Figure 5: $C = 100$, $N = 50$ (Waiting times for services)

Symbol	Values
$\mathcal{P}_1$	1331.9382
$\mathcal{P}_2$	7955.7605
$\mathcal{P}$	9287.6987

Table 4: Penalties

Symbol	95% Confidence Interval
$\mathcal{P}_1$	$(1175.0679, 1488.8084)$
$\mathcal{P}_2$	$(1738.7312, 14172.7898)$

Table 5: Confidence Intervals

(b)

Symbol	Values
C	120
N	50

Table 6: Parameters

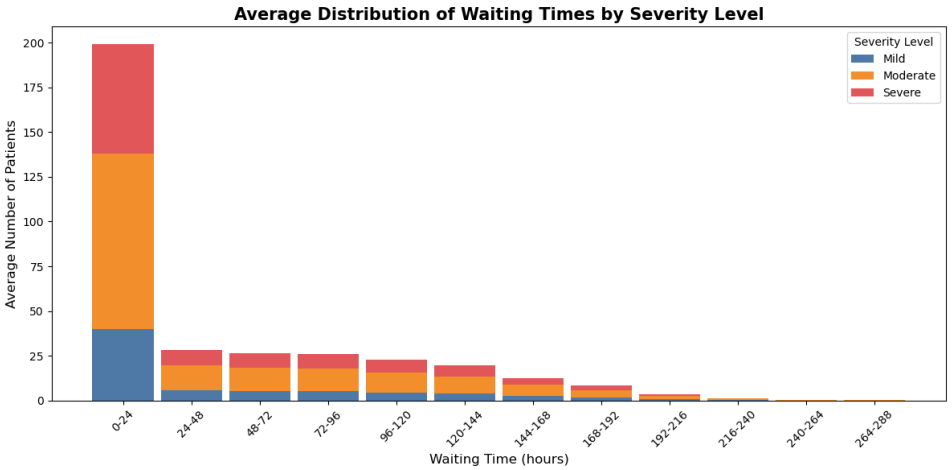

Figure 6: $C = 120$, $N = 50$ (Waiting times for ICU admission)

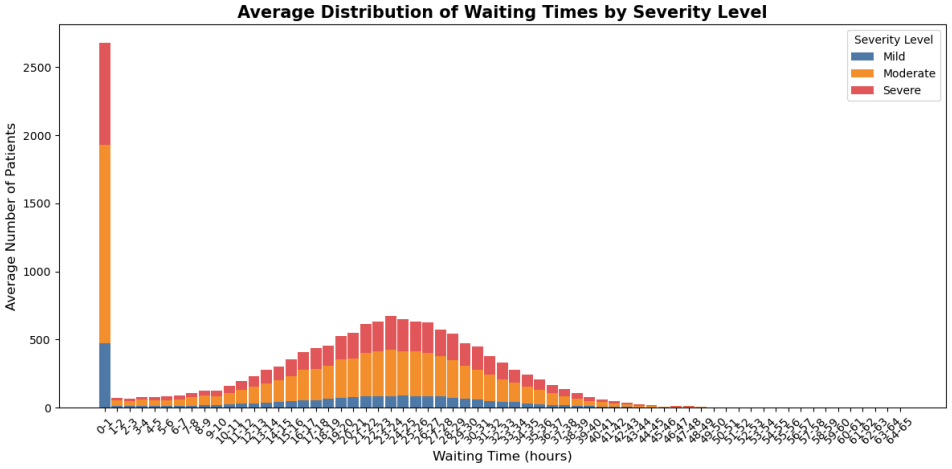

Figure 7: $C = 120$, $N = 50$ (Waiting times for services)

Symbol	Values
$\mathcal{P}_1$	328.0033
$\mathcal{P}_2$	1551855.5927
$\mathcal{P}$	1552183.5960

Table 7: Penalties

Symbol	95% Confidence Interval
$\mathcal{P}_1$	$(294.2007, 361.8059)$
$\mathcal{P}_2$	$(1251411.9626, 1852299.2228)$

Table 8: Confidence Intervals

(c)

Symbol	Values
C	100
N	60

Table 9: Parameters

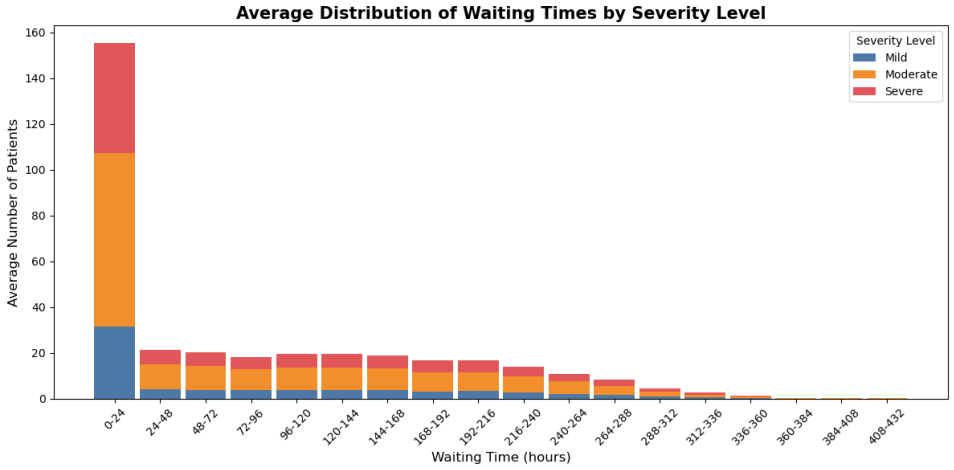

Figure 8: $C = 100$, $N = 60$ (Waiting times for ICU admission)

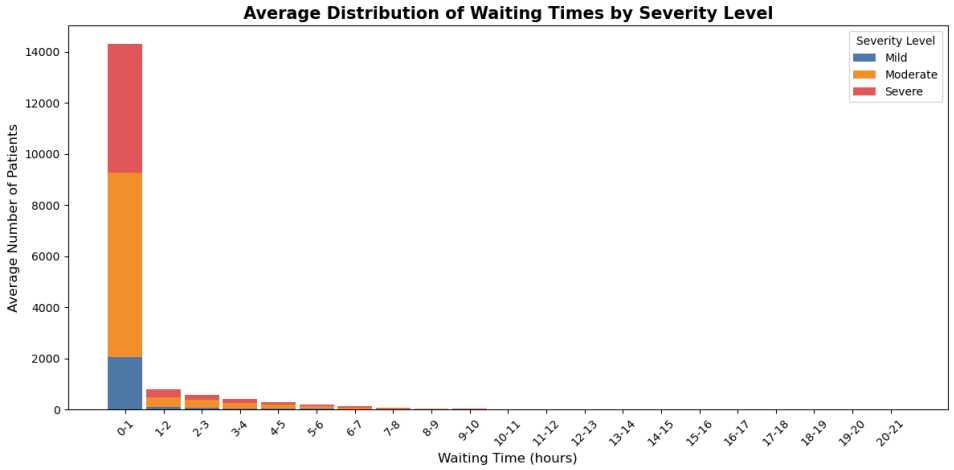

Figure 9: $C = 100$, $N = 60$ (Waiting times for services)

Symbol	Values
$\mathcal{P}_1$	1331.9382
$\mathcal{P}_2$	55.6669
$\mathcal{P}$	1387.6051

Table 10: Penalties

Symbol	95% Confidence Interval
$\mathcal{P}_1$	$(1175.0679, 1488.8084)$
$\mathcal{P}_2$	$(45.0793, 66.2545)$

Table 11: Confidence Intervals

Settings	$\mathcal{P}_1$	$\mathcal{P}_2$	$\mathcal{P}$
C=100, N=50	1331.9382	7955.7605	9287.6987
C=120, N=50	328.0033	1551855.5927	1552183.5960
C=100, N=60	1331.9382	55.6669	1387.6051

Table 12: Summary

These results align well with our intuition. While increasing the number of ICU beds effectively reduces the waiting times for ICU admission, it may lead to an imbalance between the number of beds and caregivers. This imbalance can significantly increase the waiting times for caregiver services within the ICU, potentially worsening the overall situation despite the initial improvement in admission waiting times.

On the other hand, increasing the number of caregivers appears to be a more stable strategy, as the caregiver process operates independently of the ICU queue process. If we aim to increase both bed and caregiver capacities simultaneously, it is crucial to maintain a balanced ratio between caregivers and beds to avoid compromising the quality of care and system efficiency.

It can be observed that in the waiting time distribution for ICU admission, the number of people waiting between 0–24 hours significantly exceeds the waiting numbers in other time intervals. This is primarily because patients who arrive early do not need to wait and can be immediately admitted to available beds. Similarly, in the waiting time distribution for services, the majority of waiting times fall within the 0–1 minute range due to similar reason: caregivers are often available to provide services immediately upon request especially at the very beginning.

Furthermore, in waiting time distribution for services, the percentage of services for *Mild Patients* is significantly lower than that of other patient categories. This is primarily due to two factors: the lower proportion of mild patients (only 20% of the total patient population) and their shorter ICU stay duration (an average of 3 days, compared to 7 days for *Moderate Patients* and 15 days for *Severe Patients*). These two factors collectively contribute to the reduced service demand for mild patients.

However, the current model does not differentiate treatment based on patient severity levels, which motivates us to explore different queuing strategy for ICU admission process.

Different Queuing Strategies and Implications

Symbol	Values
T	10
$\lambda(t)$	$\begin{cases} 3.0, & \text{if } 9 \le t < 12 \\ 1.5, & \text{if } 12 \le t < 17 \\ 2.5, & \text{if } 17 \le t < 20 \\ 0.7, & \text{if } 20 \le t < 24 \text{ or } 0 \le t < 4 \\ 1.0, & \text{if } 4 \le t < 9 \end{cases}$
$p[= (p_1, p_2, p_3)]$	$(0.2, 0.5, 0.3)$
$t[= (t_s, t_m, t_l)]$	$(0.2, 0.5, 1)$
f	2
$\mu(k)$	$(\frac{1}{3}, \frac{1}{7}, \frac{1}{15})$
A	$\left[\begin{bmatrix} 0.1 & 0.3 & 0.6 \\ 0.3 & 0.5 & 0.2 \\ 0.6 & 0.3 & 0.1 \end{bmatrix}, \begin{bmatrix} 0.1 & 0.2 & 0.7 \\ 0.2 & 0.5 & 0.3 \\ 0.5 & 0.3 & 0.2 \end{bmatrix}, \begin{bmatrix} 0.1 & 0.1 & 0.8 \\ 0.2 & 0.3 & 0.5 \\ 0.5 & 0.3 & 0.2 \end{bmatrix}\right]$
b_1	0.2
b_2	0.9
m_1, α_1	1, 0.005
m_2, α_2	0.01, 0.1
C	100
N	50

Table 13: Parameters

(0) FIFO (Baseline)

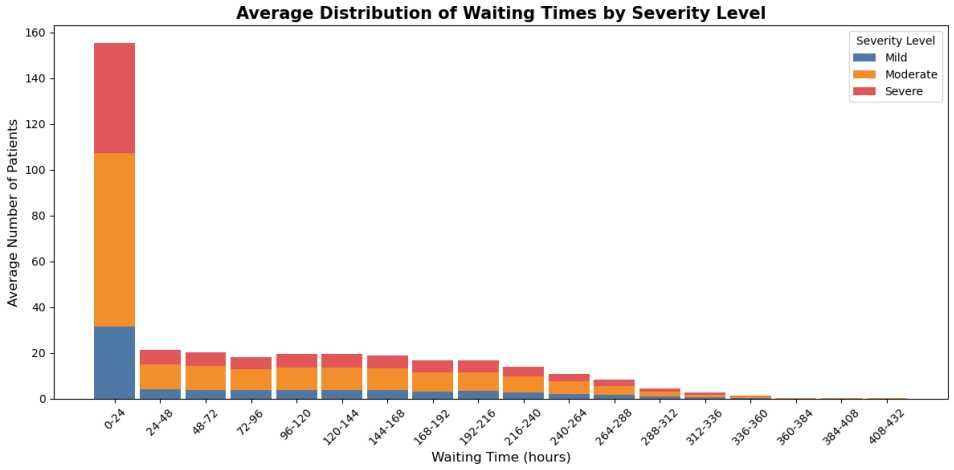

Figure 10: $C = 100$, $N = 50$ (Waiting times for ICU admission)

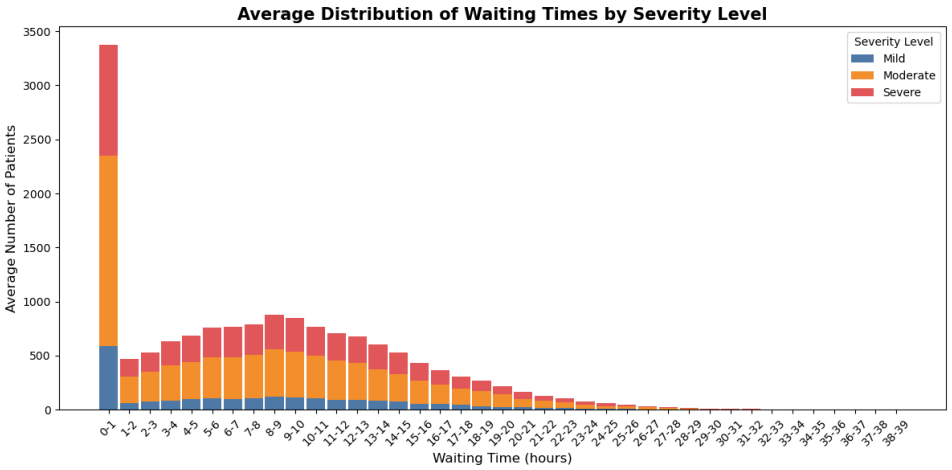

Figure 11: $C = 100$, $N = 50$ (Waiting times for services)

Symbol	Values
$\mathcal{P}_1$	1331.9382
$\mathcal{P}_2$	7955.7605
$\mathcal{P}$	9287.6987

Table 14: Penalties

(1) Priority Queue

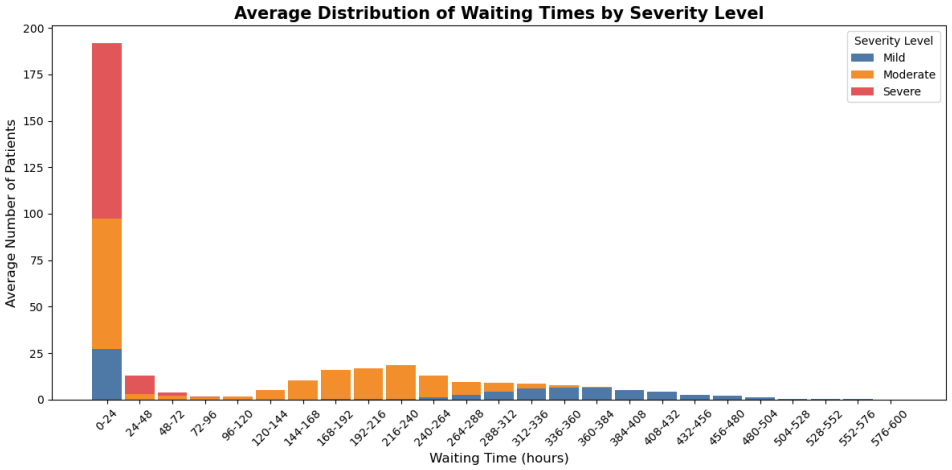

Figure 12: Applying Priority Queue (Waiting times for ICU admission)

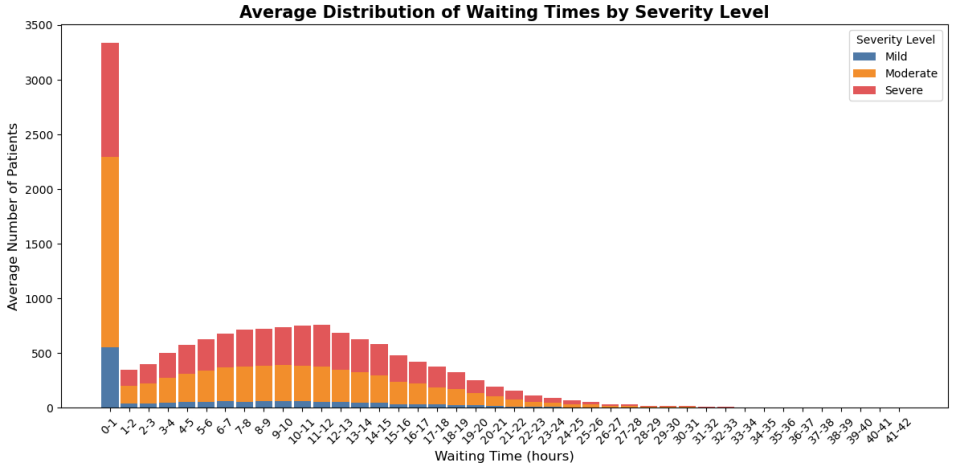

Figure 13: Applying Priority Queue (Waiting times for services)

Symbol	Values
$\mathcal{P}_1$	975.7926
$\mathcal{P}_2$	9640.1645
$\mathcal{P}$	10615.9571

Table 15: Penalties

Symbol	95% Confidence Interval
$\mathcal{P}_1$	$(873.3748, 1078.2104)$
$\mathcal{P}_2$	$(8243.4431, 11036.8858)$

Table 16: Confidence Intervals

We can see that for ICU admission, the number of patients waiting between 0-24 hours remains the highest, for reasons previously mentioned.

Almost all severe cases are waiting within the 0-24 hour window, while in other time intervals, the waiting times for mild cases are often longer than those for moderate cases. This is because Priority Queue ensures that more severe patients are always prioritized when queuing. Compared to the baseline, $\mathcal{P}_1$ decreases since we prioritize more severe patients in ICU admission. However, this could also lead to a situation where the ICU simultaneously accommodates a high number of severe patients. This, in turn, may result in more frequent large-scale service requests, causing $\mathcal{P}_2$ to increase.

It is evident that this naive PQ approach may not be the most perfect solution all the time, as it crudely disregards patients with milder conditions, which is also reflected on the higher $\mathcal{P}_2$ compared with Baseline Model. In reality, as milder patients wait longer, their medical condition may deteriorate, so it would be inappropriate to simply make milder cases wait indefinitely.

(2) Reserved Beds (30 beds is the default setting)

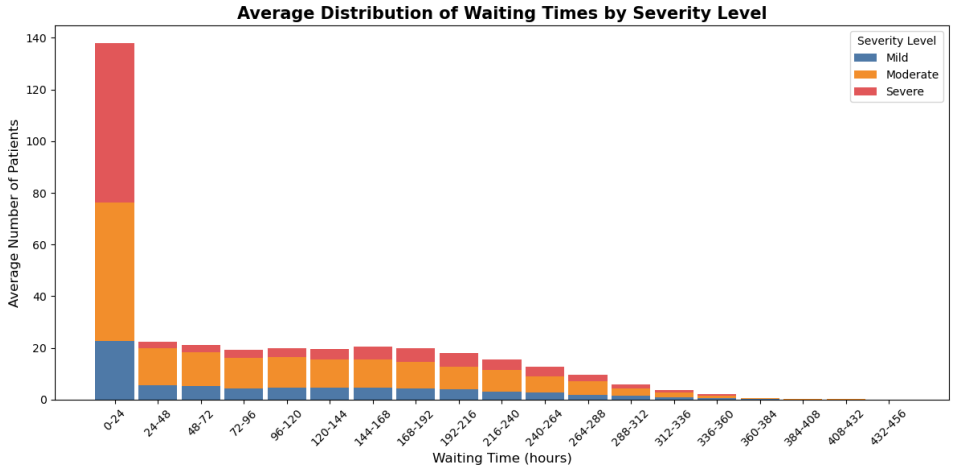

Figure 14: With 30 reserved beds (Waiting times for ICU admission)

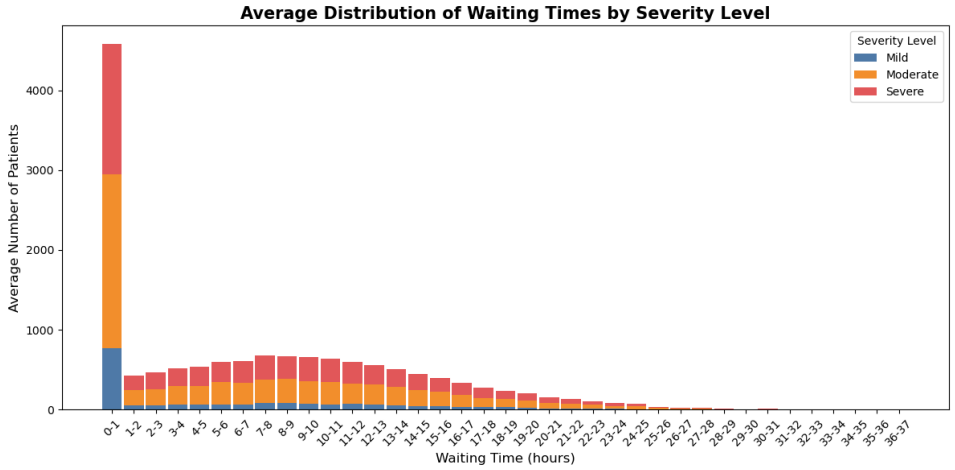

Figure 15: With 30 reserved beds (Waiting times for services)

Symbol	Values
$\mathcal{P}_1$	1512.0821
$\mathcal{P}_2$	13294.0456
$\mathcal{P}$	14806.1277

Table 17: Penalties

Symbol	95% Confidence Interval
$\mathcal{P}_1$	$(1340.2031, 1683.9611)$
$\mathcal{P}_2$	$(5697.4013, 20890.6898)$

Table 18: Confidence Intervals

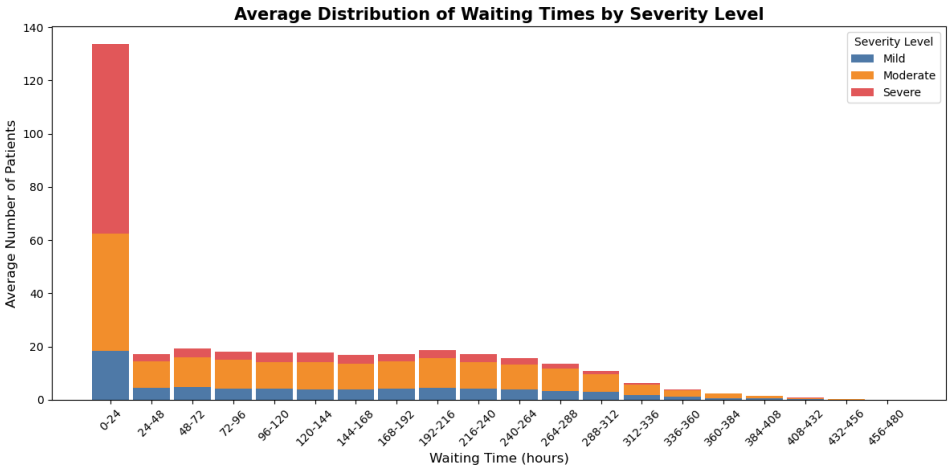

Figure 16: With 40 reserved beds (Waiting times for ICU admission)

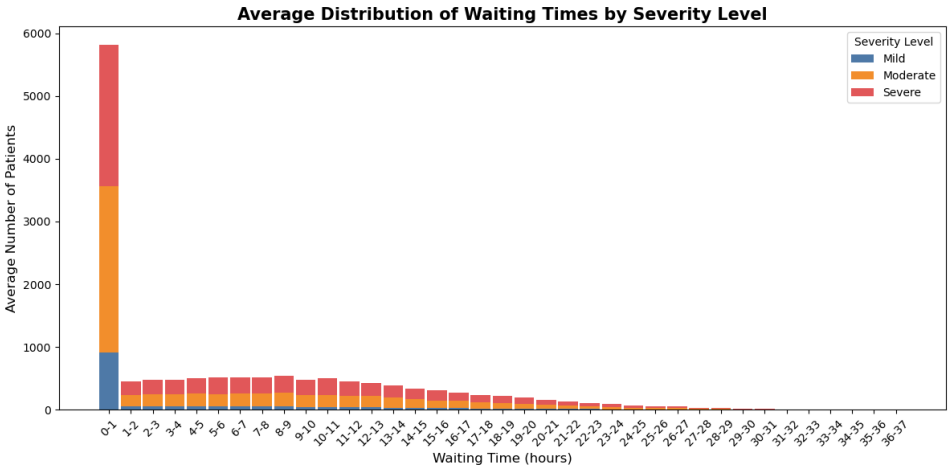

Figure 17: With 40 reserved beds (Waiting times for services)

Symbol	Values
$\mathcal{P}_1$	1629.8764
$\mathcal{P}_2$	7335.8895
$\mathcal{P}$	8965.7659

Table 19: Penalties

Symbol	95% Confidence Interval
$\mathcal{P}_1$	$(1456.5944, 1803.1583)$
$\mathcal{P}_2$	$(3583.3135, 11088.4655)$

Table 20: Confidence Intervals

We can observe that when reserving 40 beds, the number of severe patients waiting beyond the 0-24 hour window notably decreases. However, due to the reduction in regular bed availability, the waiting times for mild and moderate patients increase substantially: the maximum waiting interval increases

from 432-456 hours with 30 reserved beds to 456-480 hours with 40 reserved beds. The phenomenon becomes more evident as the number of reserved beds increases. This reveals a problem similar to the naive Priority Queue approach—neglecting patients with less severe symptoms. Moreover, reserving too many beds is obviously unrealistic.

(3) Priority Queue + Reserved Beds

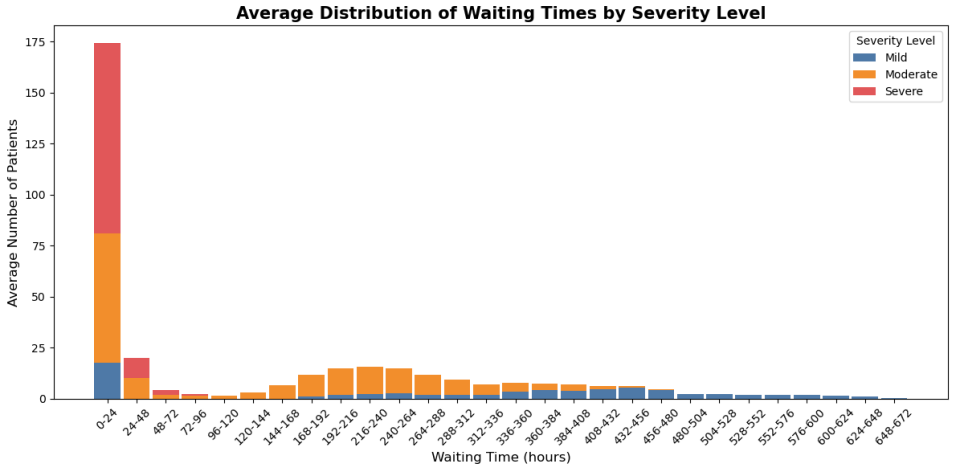

Figure 18: Apply Priority Queue and Reserved Beds simultaneously (Waiting times for ICU admission)

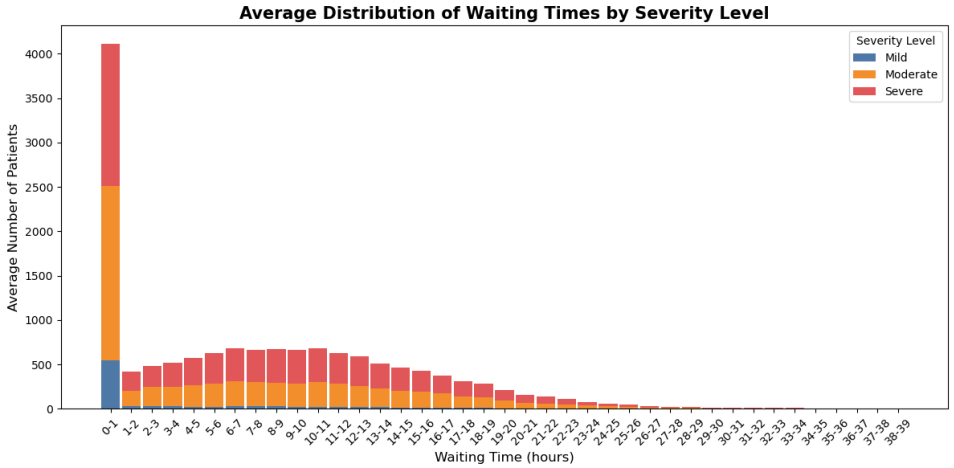

Figure 19: Apply Priority Queue and Reserved Beds simultaneously (Waiting times for services)

Symbol	Values
$\mathcal{P}_1$	1589.0669
$\mathcal{P}_2$	9958.7207
$\mathcal{P}$	11547.7876

Table 21: Penalties

Symbol	95% Confidence Interval
$\mathcal{P}_1$	$(1394.8939, 1783.2400)$
$\mathcal{P}_2$	$(6125.2325, 13792.2088)$

Table 22: Confidence Intervals

We observe that the results are quite similar to results from using Priority Queue only, but the maximum waiting time interval has increased dramatically to 648-672 hours. This aligns with our previous analysis of the Priority Queue and Reserved Beds strategies, which tend to ignore milder patients. Under the simultaneous influence of these two strategies, this tendency to ignore milder patients becomes particularly pronounced.

(4) Dynamic Priority Queue

Heuristic Function

To prioritize patients dynamically based on both severity and waiting time, we designed a heuristic function that calculates a priority score for each patient. This score is updated continuously during the simulation and is defined as:

$$\text{Priority Score} = m_1 \cdot (\text{severity}^2) + \alpha_1 \cdot (\text{waiting\_time}^{1.5})$$

The heuristic function is designed to balance the trade-off between patient severity and waiting time. Severity is given a quadratic weight to ensure that more critical cases are prioritized exponentially higher than less severe cases. The waiting time is weighted with an exponent of 1.5 to account for the increasing urgency of patients who have been waiting longer. This prevents cases of mild or moderate severity from being deprioritized indefinitely, thereby maintaining fairness in resource allocation. The parameters m_1 and α_1 were calibrated to reflect the realistic urgency of ICU operations, where severity matters more.

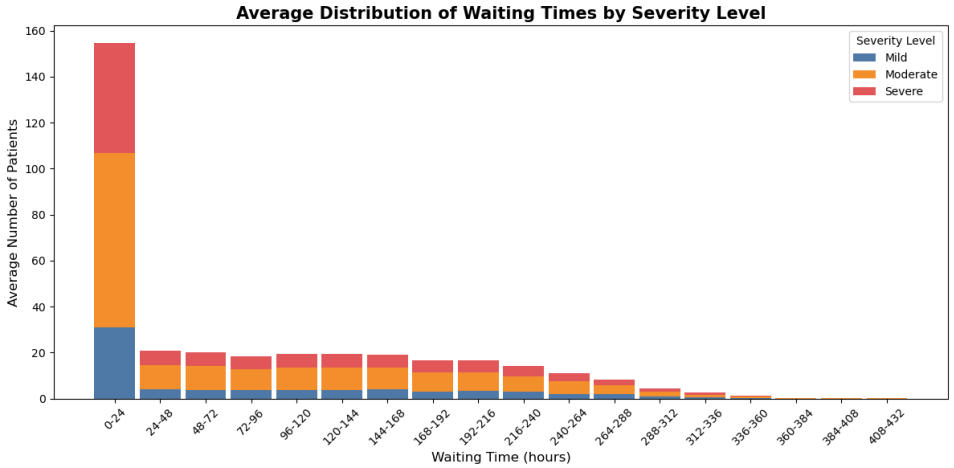

Figure 20: Applying Dynamic Priority Queue (Waiting times for ICU admission)

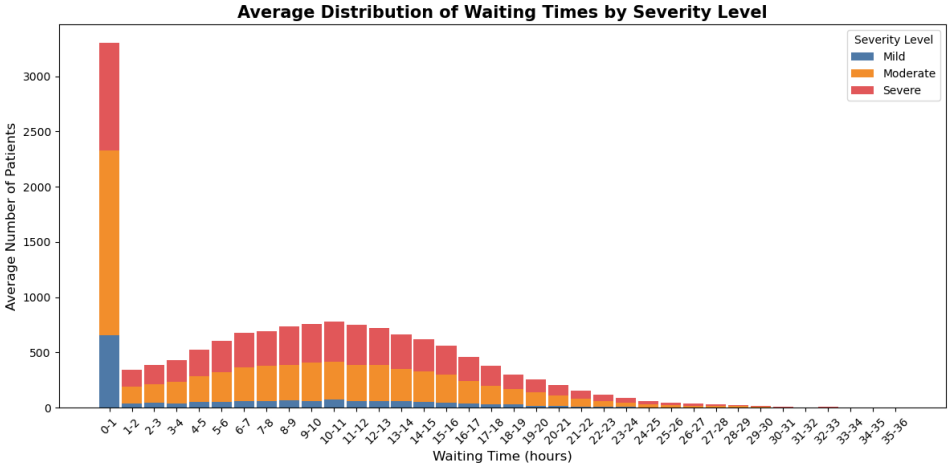

Figure 21: Applying Dynamic Priority Queue (Waiting times for services)

Symbol	Values
$\mathcal{P}_1$	1346.9915
$\mathcal{P}_2$	8266.7400
$\mathcal{P}$	9613.7315

Table 23: Penalties

Symbol	95% Confidence Interval
$\mathcal{P}_1$	$(1188.6344, 1505.3486)$
$\mathcal{P}_2$	$(3335.3962, 13198.0838)$

Table 24: Confidence Intervals

In the Dynamic Priority Queue results, Severe patients are still mostly distributed in the 0-24 hour window. Notably, the maximum waiting time interval for Mild patients remains 408-432 hours, identical to the FIFO Model, and the waiting times of Mild patients are not strictly larger than that of Moderate patients. This observation aligns with our earlier discussion regarding the reduced effect of deprioritizing less severe patients.

Comparison of figures further illuminates this effectiveness: the Dynamic Priority Queue not only reduces waiting times for Severe patients but also maintains waiting times for Mild and Moderate patients.

(5) Dynamic Priority Queue + Reserved Beds

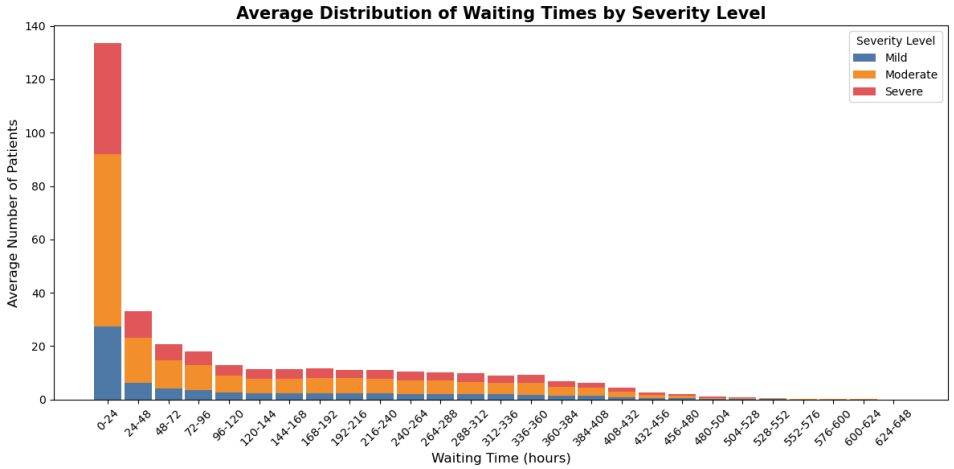

Figure 22: Apply Dynamic Priority Queue and Reserved Beds simultaneously (Waiting times for ICU admission)

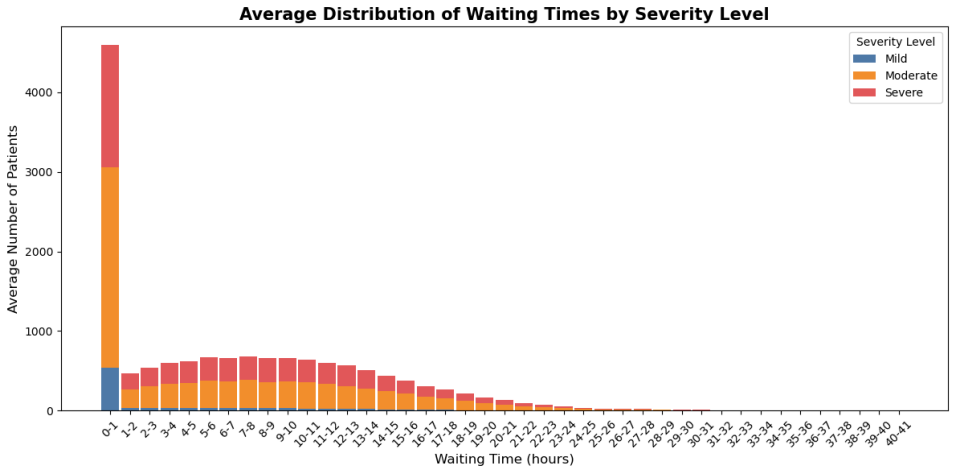

Figure 23: Apply Dynamic Priority Queue and Reserved Beds simultaneously (Waiting times for services)

Symbol	Values
$\mathcal{P}_1$	6179.9721
$\mathcal{P}_2$	4718.9788
$\mathcal{P}$	10898.9509

Table 25: Penalties

Symbol	95% Confidence Interval
$\mathcal{P}_1$	$(5108.2707, 7251.6736)$
$\mathcal{P}_2$	$(1380.8532, 8057.1045)$

Table 26: Confidence Intervals

The combination of the Dynamic Priority Queue (DPQ) and Reserved Beds exhibits characteristics influenced by the interaction between the two strategies. DPQ dynamically adjusts patient priorities based on severity and waiting time, while Reserved Beds allocate a fixed number of beds specifically for severe patients. The simultaneous application of these strategies can introduce redundancy and alter the balance of resource allocation, affecting the overall behavior of the queuing system.

Summary

We summarize our results in the following table.(Table 27).

Strategies	$\mathcal{P}_1$	$\mathcal{P}_2$	$\mathcal{P}$
FIFO (Baseline)	1331.9382	7955.7605	9287.6987
Priority Queue	975.7926	9640.1645	10615.9571
30 Reserved Beds	1512.0821	13294.0456	14806.1277
40 Reserved Beds	1629.8764	7335.8895	8965.7659
Priority Queue + Reserved Beds	1589.0669	9958.7207	11547.7876
Dynamic Priority Queue	1346.9915	8266.7400	9613.7315
Dynamic Priority Queue + Reserved Beds	6179.9721	4718.9788	10898.9509

Table 27: Summary

We want to emphasize that, in such a complex system, there is no inherently dominating or dominated strategy. While a given strategy may perform better in certain aspects, it is highly likely to impact other parts of the system in ways that are difficult to predict theoretically. This underscores the importance of developing a simulation model, as only a model capable of reflecting real-world dynamics can capture these subtle interactions that are often beyond the reach of theoretical analysis. Such simulation models provide valuable insights to inform decision-making in real-world scenarios.

Sensitivity Analysis

To understand the influence of different parameters on the ICU penalty function, we conducted a sensitivity analysis. This analysis uses the standardized sensitivity index defined as:

$$\text{Sensitivity Index} = \frac{\Delta\text{Penalty}}{\text{Baseline Penalty} \times \Delta\text{Parameter}}$$

Due to the exponential nature of the penalty function, small change in parameters can lead to significant and rapid changes in the penalty value, which makes a direct comparison using just the difference in penalties ($\Delta\mathcal{P}$) impractical. Therefore, we chose index normalization, which provides a consistent way to compare the relative impact of each parameter.

The baseline total penalty value, calculated under the default settings, is:

$$\mathcal{P}_{\text{baseline}} = 7423.76$$

Sensitivity Analysis Results

The sensitivity indices for the four parameters analyzed are shown in Table 28. These indices reveal the relative impact of each parameter on the total penalty.

Table 28: Standardized Sensitivity Analysis Results

Parameter	**Sensitivity Index**
Mean Service Time	61.12
Length of Stays	7.45
Arrival Rates	2.44
Request Frequency	2.13

Figure 24 provides a Tornado Chart visualization of the sensitivity analysis. The parameters are sorted by their sensitivity index, with the most impactful parameter (*Mean Service Time*) at the bottom.

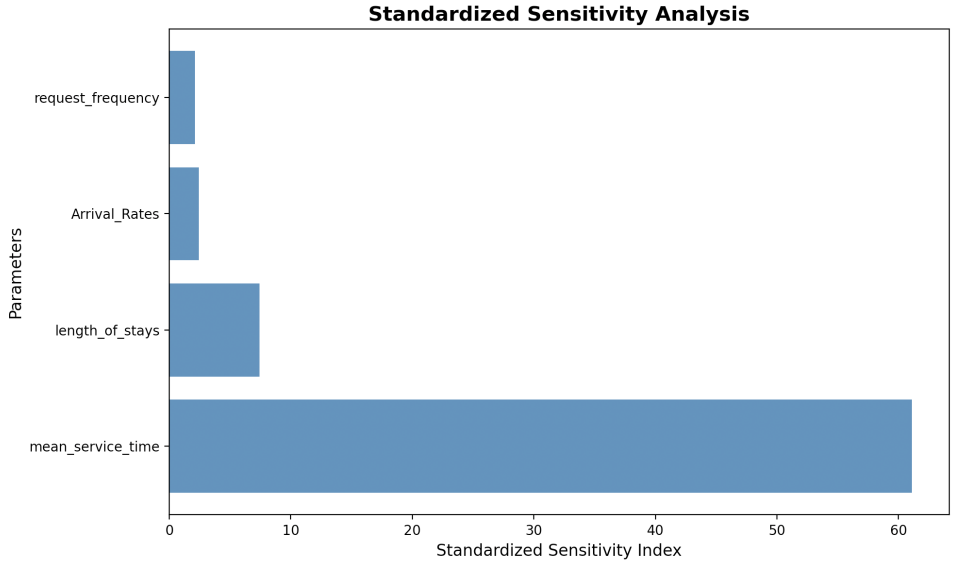

Figure 24: Tornado Chart of Sensitivity Analysis Results

Interpretation of Results

- **Mean Service Time (Sensitivity Index: 61.12)**: This parameter has the largest impact on the penalty. A small increase in the mean service time causes significant growth in the penalty due to the exponential penalty function, which emphasizes the importance of efficient service delivery. Optimizing service time is critical for reducing delays and improving ICU performance.

- **Length of Stays (Sensitivity Index: 7.45)**: As the length of stays directly affects resource availability and patient flow, the change of it also matters a lot on penalty, which encourage us to get more accurate parameters.
- **Arrival Rates (Sensitivity Index: 2.44)**: While the arrival rates impact the penalty, their sensitivity is relatively low. This suggests that the ICU system can handle fluctuations in patient arrival rates within reasonable limits without severe disruptions.
- **Request Frequency (Sensitivity Index: 2.13)**: The frequency of service requests has the smallest impact among the parameters analyzed. This indicates that the system is less sensitive to variations in how often patients request caregiver services, likely due to the capacity of the caregivers being sufficient to handle these demands.

Simulation Optimization

Optimization Strategies for ICU Resource Allocation

In modern healthcare systems, optimizing resource allocation is critical for improving efficiency and enhancing patient outcomes. **Simulation optimization** provides a systematic framework to achieve this by combining simulation models with optimization techniques. This approach is particularly valuable when additional financial resources are available, allowing decision-makers to explore how different investments can enhance system performance under dynamic and uncertain conditions. By simulating real-world operations and applying optimization strategies, healthcare administrators can identify bottlenecks and determine the most effective allocation of beds and caregivers to achieve optimal outcomes.

In our specific setting of Intensive Care Unit (ICU) resource management, we consider scenarios where a supplementary budget, such as \$100,000, can be allocated to improve efficiency. This budget can be used for purchasing additional ICU beds (e.g., \$10,000 per bed, including space and equipment) or hiring additional caregivers (e.g., \$1,000 per caregiver for a given time horizon). Additionally, administrators can adjust queueing strategies and determine the number of reserved beds for critical patients. Our goal is to leverage our simulation model to guide these decisions, ensuring that investments lead to tangible improvements in patient care by reducing patient waiting times for ICU admission and caregiver services, with a focus on severity-sensitive priorities to address the needs of critical patients effectively.

To achieve this, we propose three distinct optimization models, each offering different trade-offs between computational cost and solution quality:

1. **Exhaustive Search**: This method evaluates all possible combinations of bed and caregiver allocations to identify the optimal solution. While it provides a comprehensive baseline, it is computationally expensive and impractical for large-scale problems due to its inefficiency.

2. **Pareto Front Search**: This approach reduces computation time compared to exhaustive search by focusing on a set of optimal trade-offs between the two conflicting objectives: the number of beds and the number of caregivers. While it is more efficient, it still requires a substantial amount of computation time, especially for complex scenarios with many variables.

3. **Tabu Search**: This heuristic method accelerates the optimization process by navigating the search space intelligently and avoiding previously visited solutions. It relies on an **initial guess (prediction)**, and both the computation time and the quality of the resulting local minimizer depend heavily on the accuracy of this prediction. While faster, it often sacrifices the guarantee of finding a global optimum and instead converges to a local optimal solution.

By comparing these models, we aim to illustrate the trade-offs between solution quality and computational cost. The exhaustive search, while thorough, is often impractical for real-world use due to its inefficiency. The Pareto front search offers a balanced approach, significantly improving efficiency while maintaining solution quality. Tabu search provides the fastest results but relies on the quality of the initial prediction to achieve effective outcomes. This comparative analysis helps decision-makers choose the most suitable strategy based on their specific operational constraints and optimization goals.

ICU Resource Optimization Problem Formulation

In this section, we present a formal formulation of the ICU resource optimization problem. The objective is to determine the optimal allocation of beds and caregivers, select appropriate queueing strategies, and decide on the number of reserved beds for critical patients. The goal is to minimize patient waiting times for both ICU admission and caregiver services, particularly prioritizing those with severe conditions. This formulation serves as the foundation for evaluating and comparing different optimization strategies, including exhaustive search, Pareto front search, and Tabu search.

We simplify our simulation model into a black-box model denoted by:

$$\mathcal{M}(C, N, Q, r)$$

where $C \in \mathbb{Z}_+$ is the ICU capacity (number of beds), $N \in \mathbb{Z}_+$ is the number of caregivers, $Q \in \mathbb{Q}$ is the queueing strategy with $\mathbb{Q} = \{\texttt{FIFO}, \texttt{PQ}, \texttt{DPQ}, \texttt{Reservation}, \texttt{PQ+Reservation}, \texttt{DPQ+Reservation}\}$, and $r \in \mathbb{Z}_+$ is the number of reserved beds for critical patients.

Suppose the ICU initially has $C_0 \in \mathbb{Z}_+$ beds and $N_0 \in \mathbb{Z}_+$ caregivers. Given an additional budget of $K \in \mathbb{R}_+$ dollars, resources can be expanded by adding $C' \in \mathbb{N}$ beds, each costing $p \in \mathbb{R}_+$ dollars, and hiring $N' \in \mathbb{N}$ caregivers, each costing $q \in \mathbb{R}_+$ dollars. The budget constraint is expressed as

$$pC' + qN' \leq K$$

To ensure consistency with the selected queueing strategy, we define the set of strategies that support reservations as

$$\mathbb{Q}_R = \{\texttt{Reservation}, \texttt{PQ+Reservation}, \texttt{DPQ+Reservation}\}$$

The number of reserved beds r must be zero if the selected strategy does not support reservations. This constraint can be written as

$$r \cdot \mathbb{1}[Q \notin \mathbb{Q}_R] = 0$$

In practice, reserving too many beds for critical patients is unrealistic. Therefore, we introduce a parameter δ to limit the number of reserved beds to a fraction of the total capacity. This constraint is given by

$$r \leq \delta \cdot C$$

ensuring that r does not exceed δ times the expanded capacity C.

To measure the internal performance of the ICU system, we use the previously defined penalty function $\mathcal{P} : \mathbb{M} \to \mathbb{R}$ as our evaluation metric. The penalty function $\mathcal{P}$ captures the delays experienced by patients for ICU admission and caregiver services, with sensitivity to the severity of their conditions. Therefore, our goal is to minimize $\mathcal{P}$ as the primary objective in our optimization problem.

Given the constraints and parameters defined above, the ICU resource allocation task can be formulated as the following optimization problem:

$$\begin{aligned} \min_{C', N', Q, r} \quad & \mathcal{P}(\mathcal{M}(C, N, Q, r)) \\ \text{s.t.} \quad & p \cdot C' + q \cdot N' \leq K \\ & r \leq \delta \cdot C \\ & r \cdot \mathbb{1}[Q \notin \mathbb{Q}_R] = 0 \\ & C = C_0 + C' \\ & N = N_0 + N' \end{aligned}$$

Due to the black-box nature of the model $\mathcal{M}(C, N, Q, r)$ and the integer programming characteristics of the problem, employing non-zero-order (e.g., first-order methods like gradient descent or second-order methods like Newton's method) algorithms is inefficient for solving this problem. The lack of differentiability and the discrete nature of the decision variables make these methods unsuitable for our setting.

Therefore, we primarily adopt **zero-order algorithms** for optimization. These methods do not rely on gradient information and instead evaluate the objective function P directly. The zero-order approach is well-suited for navigating the discrete and black-box nature of our ICU resource optimization problem, enabling us to efficiently explore and identify near-optimal solutions.

Algorithm 1: Exhaustive Search

Exhaustive search is a brute-force optimization technique that systematically explores all possible combinations of the decision variables (C', N', Q, r) in feasible space to identify the optimal solution. Given the constraints and the objective function above, the algorithm evaluates each feasible configuration and selects the one that minimizes the penalty function $\mathcal{P}$.

Pseudocode of Exhaustive Search

The step-by-step procedure of the algorithm is outlined below.

Algorithm 1: Exhaustive Search

Input: Simulation model $\mathcal{M}$, penalty function $\mathcal{P}$, initial capacity C_0, initial number of caregivers N_0, total budget K, cost of adding a bed p, cost of hiring a caregiver q, maximal percentage of reserved beds δ, set of queueing strategies $\mathbb{Q}$, set of reservation-supporting strategies $\mathbb{Q}_R$

Output: Optimal number of added beds C'_{opt}, caregivers N'_{opt}, queueing strategy Q_{opt}, and number of reserved beds r_{opt}

```
Initialize P_min ← ∞;
Initialize (C'_opt, N'_opt, Q_opt, r_opt) ← (0, 0, FIFO, 0);
for C' ← 0 to ⌊K/p⌋ do
    for N' ← 0 to ⌊K/q⌋ do
        if pC' + qN' ≤ K then
            for Q ∈ ℚ \ ℚ_R do
                Evaluate P ← P(M(C_0 + C', N_0 + N', Q, 0));
                if P < P_min then
                    P_min ← P;
                    (C'_opt, N'_opt, Q_opt, r_opt) ← (C', N', Q, 0);
                end
            end
            for Q ∈ ℚ_R do
                for r ← 0 to ⌊δ · (C_0 + C')⌋ do
                    Evaluate P ← P(M(C_0 + C', N_0 + N', Q, r));
                    if P < P_min then
                        P_min ← P;
                        (C'_opt, N'_opt, Q_opt, r_opt) ← (C', N', Q, r);
                    end
                end
            end
        end
    end
end
return (C'_opt, N'_opt, Q_opt, r_opt);
```

Run-Time Analysis of Exhaustive Search

Theorem 1. Assuming that each simulation and penalty function evaluation $\mathcal{P}(\mathcal{M}(C, N, Q, r))$ takes constant time and there is sufficient flexibility in the number of reserved beds, such that $\delta C_0 \gg 1$, the run-time complexity of the Exhaustive Search Algorithm is

$$\mathcal{O}\left(\delta|\mathbb{Q}_R|\left(C_0 + \frac{K}{p}\right) \cdot \frac{K^2}{pq}\right)$$

Proof. The run-time complexity is determined by the number of evaluations of the penalty function $\mathcal{P}$ in the nested loops of the algorithm. The number of iterations for the outer loop over C' is bounded by $\frac{K}{p}$, and the number of iterations for the middle loop over N' is bounded by $\frac{K}{q}$.

The algorithm evaluates the penalty function for all queueing strategies Q, which can be divided into two categories: non-reservation strategies ($Q \in \mathbb{Q} \setminus \mathbb{Q}_R$) and reservation-supporting strategies ($Q \in \mathbb{Q}_R$).

For non-reservation strategies, the number of evaluations is given by:

$$\frac{K}{p} \cdot \frac{K}{q} \cdot |\mathbb{Q} \setminus \mathbb{Q}_R| = \mathcal{O}\left(\frac{K^2}{pq} \cdot |\mathbb{Q} \setminus \mathbb{Q}_R|\right)$$

For reservation-supporting strategies, the number of possible reserved beds r is bounded by $\delta(C_0+C')$. Therefore, the number of evaluations for reservation-supporting strategies is:

$$\frac{K}{p} \cdot \frac{K}{q} \cdot |\mathbb{Q}_R| \cdot \delta(C_0 + C') \approx \frac{K}{p} \cdot \frac{K}{q} \cdot |\mathbb{Q}_R| \cdot \delta C_0 + \frac{K}{p} \cdot \frac{K}{q} \cdot |\mathbb{Q}_R| \cdot \delta C'$$

Since C' can be at most $\frac{K}{p}$, this becomes:

$$\mathcal{O}\left(\frac{K^2}{pq} \cdot |\mathbb{Q}_R| \cdot \delta C_0 + \frac{K^3}{p^2 q} \cdot |\mathbb{Q}_R| \cdot \delta\right)$$

Combining the two parts, the overall run-time complexity is:

$$\mathcal{O}\left(\frac{K^2}{pq} \cdot |\mathbb{Q} \setminus \mathbb{Q}_R| + \frac{K^2}{pq} \cdot |\mathbb{Q}_R| \cdot \delta C_0 + \frac{K^3}{p^2 q} \cdot |\mathbb{Q}_R| \cdot \delta\right)$$

Since $\delta C_0 >> 1$ and $|\mathbb{Q} \setminus \mathbb{Q}_R|$,$|\mathbb{Q}_R|$ are almost balanced, this eventually evaluates to

$$\mathcal{O}\left(\delta |\mathbb{Q}_R| \left(C_0 + \frac{K}{p}\right) \cdot \frac{K^2}{pq}\right)$$

□

Algorithm 2: Pareto Front Search

In this section, we introduce the Pareto Front Search algorithm, which considers the trade-off between the two primary objectives: the **number of added beds** C' and the **number of added caregivers** N'. Instead of optimizing a single penalty function, we aim to identify a set of *non-dominated solutions* that balance these two objectives under the given constraints.

Definition 1 (Pareto Optimality). Given a total budget K, let $C', N' \in \mathbb{N}$ denote the number of added beds and added caregivers, respectively, such that $pC' + qN' \leq K$. A solution (C'^*, N'^*) is said to be **Pareto optimal** if there does not exist another solution $(C', N') \in \mathbb{N} \times \mathbb{N}$ satisfying the budget constraint such that:

$$C' \geq C'^* \quad \text{and} \quad N' \geq N'^* \quad \text{with at least one strict inequality.}$$

In other words, a solution is Pareto optimal if no other solution exists that improves one objective (e.g., increases the number of added beds C') without worsening the other objective (e.g., reducing the number of added caregivers N') while remaining within the budget K.

Motivation for Pareto Front Search

Exploring the Pareto front is essential in this problem because it allows for a systematic evaluation of trade-offs between the two primary objectives: increasing the number of added beds C' and the number of added caregivers N'. The rationale for this approach is outlined below:

- *Effect of Increasing C' (Beds):* Increasing C' while keeping N' constant does not always reduce the penalty. In some cases, it can significantly increase the penalty due to proportional imbalances in resource allocation. This is tied to the sequential nature of the problem: when C' is increased without a corresponding increase in N', the caregiver-to-patient ratio worsens, leading to inefficiencies in patient care.
- *Effect of Increasing N' (Caregivers):* In contrast, increasing N' while keeping C' constant reliably decreases the penalty, or at worst, leaves it unchanged. Adding caregivers directly addresses bottlenecks in patient care by reducing service waiting times. In a multi-server cyclic queueing model, increasing the number of servers (caregivers) reduces delays and enhances overall system performance without disrupting resource proportionality.
- *Simulation Model Limitations:* The inherent randomness of the simulation model, combined with the dynamic characteristics of the ICU system, introduces additional complexity. Unlike a classic multi-server cyclic queueing system, the ICU model is influenced by patient inflows and outflows, as well as variations in resource demand and allocation. These factors make the system behavior highly context-dependent, and conclusions drawn from the simulation rely on empirical observations rather than deterministic guarantees.

Pseudocode of Pareto Front Search

The step-by-step procedure of the algorithm is outlined below.

Algorithm 2: Pareto Front Search

Input: Simulation model $\mathcal{M}$, penalty function $\mathcal{P}$, initial capacity C_0, initial number of caregivers N_0, total budget K, cost of adding a bed p, cost of hiring a caregiver q, maximal percentage of reserved beds δ, set of queueing strategies $\mathbb{Q}$, set of reservation-supporting strategies $\mathbb{Q}_R$

Output: Optimal number of added beds C'_{opt}, caregivers N'_{opt}, queueing strategy Q_{opt}, and number of reserved beds r_{opt}

Initialize $\mathcal{F} \leftarrow \emptyset$;
Initialize $\mathcal{P}_{\min} \leftarrow \infty$;
Initialize $(C'_{\text{opt}}, N'_{\text{opt}}, Q_{\text{opt}}, r_{\text{opt}}) \leftarrow (0, 0, \texttt{FIFO}, 0)$;
for $C' \leftarrow 0$ **to** $\left\lfloor \frac{K}{p} \right\rfloor$ **do**
  for $N' \leftarrow 0$ **to** $\left\lfloor \frac{K}{q} \right\rfloor$ **do**
    if $pC' + qN' \leq K$ **then**
      if (C', N') *is not dominated by any solution in* $\mathcal{F}$ **then**
        Remove all solutions in $\mathcal{F}$ that are dominated by (C', N');
        Add (C', N') to $\mathcal{F}$;
      end
    end
  end
end
for $(C', N') \in \mathcal{F}$ **do**
  for $Q \in \mathbb{Q} \setminus \mathbb{Q}_R$ **do**
    Evaluate $\mathcal{P} \leftarrow \mathcal{P}(\mathcal{M}(C_0 + C', N_0 + N', Q, 0))$;
    if $\mathcal{P} < \mathcal{P}_{\min}$ **then**
      $\mathcal{P}_{\min} \leftarrow \mathcal{P}$;
      $(C'_{\text{opt}}, N'_{\text{opt}}, Q_{\text{opt}}, r_{\text{opt}}) \leftarrow (C', N', Q, 0)$;
    end
  end
  for $Q \in \mathbb{Q}_R$ **do**
    for $r \leftarrow 0$ **to** $\lfloor \delta \cdot (C_0 + C') \rfloor$ **do**
      Evaluate $\mathcal{P} \leftarrow \mathcal{P}(\mathcal{M}(C_0 + C', N_0 + N', Q, r))$;
      if $\mathcal{P} < \mathcal{P}_{\min}$ **then**
        $\mathcal{P}_{\min} \leftarrow \mathcal{P}$;
        $(C'_{\text{opt}}, N'_{\text{opt}}, Q_{\text{opt}}, r_{\text{opt}}) \leftarrow (C', N', Q, r)$;
      end
    end
  end
end
return $(C'_{\text{opt}}, N'_{\text{opt}}, Q_{\text{opt}}, r_{\text{opt}})$;

Run-Time Analysis of Pareto Front Search

Theorem 2. Assuming that each simulation and penalty function evaluation $\mathcal{P}(\mathcal{M}(C, N, Q, r))$ takes constant time and there is sufficient flexibility in the number of reserved beds, such that $\delta C_0 \gg 1$, the run-time complexity of the Pareto Front Search Algorithm is

$$\mathcal{O}\left(\delta|\mathbb{Q}_R|\left(C_0 + \frac{K}{p}\right) \cdot \frac{K}{\max\{p, q\}}\right)$$

Proof. The run-time complexity is determined by the number of evaluations of the penalty function $\mathcal{P}$ for all solutions in the Pareto front, considering all queueing strategies and reserved bed configurations.

The size of the Pareto front $|\mathcal{F}|$ is bounded by:

$$|\mathcal{F}| = \min\left\{\frac{K}{p}, \frac{K}{q}\right\}.$$

The algorithm evaluates the penalty function for all reservation-supporting queueing strategies $Q \in \mathbb{Q}_R$ and iterates over all possible values of reserved beds r, which are bounded by $\delta(C_0+C')$. Substituting $C' = \frac{K}{p}$, the number of evaluations becomes:

$$|\mathcal{F}| \cdot |\mathbb{Q}_R| \cdot \delta(C_0 + C') \approx \min\left\{\frac{K}{p}, \frac{K}{q}\right\} \cdot |\mathbb{Q}_R| \cdot \delta\left(C_0 + \frac{K}{p}\right).$$

Breaking this into two terms, we get:

$$\mathcal{O}\left(\min\left\{\frac{K}{p}, \frac{K}{q}\right\} \cdot |\mathbb{Q}_R| \cdot \delta C_0 + \min\left\{\frac{K}{p}, \frac{K}{q}\right\} \cdot |\mathbb{Q}_R| \cdot \delta\frac{K}{p}\right).$$

Simplifying further and using the fact that $\min\{p, q\}$ affects the Pareto front size, the complexity becomes:

$$\mathcal{O}\left(\delta|\mathbb{Q}_R|\left(C_0 + \frac{K}{p}\right) \cdot \frac{K}{\max\{p, q\}}\right)$$

□

Algorithm 3: Tabu Search

Tabu Search is a heuristic optimization algorithm designed to navigate the search space efficiently by iteratively improving a candidate solution while avoiding revisiting previously evaluated solutions. Unlike Exhaustive Search, which systematically explores the entire feasible space, and Pareto Front Search, which identifies non-dominated solutions and optimizes trade-offs among them, Tabu Search focuses on refining solutions within a local region of the Pareto front. It achieves this by performing a targeted search in the neighborhood of a candidate solution while maintaining a tabu list to prevent cycling.

This localized search approach is guided by two key hyperparameters:

- ω_{PF}: The search width along the Pareto front, which controls the breadth of exploration across non-dominated solutions.
- ω_r: The search width for the number of reserved beds, which limits the range of adjustments to the reserved beds variable r.

By adjusting these hyperparameters, Tabu Search balances computational cost and solution quality, allowing the algorithm to efficiently identify high-quality solutions within a practical time frame.

Key Features of Tabu Search

- *Initial Solution:* The algorithm starts with an initial guess for the decision variables, $(\hat{C}', \hat{N}', \hat{Q}, \hat{r})$. This guess provides the starting point for iterative improvements.
- *Neighborhood Exploration:* At each iteration, the algorithm explores the neighborhood of the current solution, $\mathcal{N}(C', N', Q, r)$, as defined above. The exploration involves adjusting the decision variables C' (number of added beds), N' (number of added caregivers), Q (queueing strategy), and r (number of reserved beds).
- *Tabu List:* The algorithm maintains a tabu list to track visited solutions or moves. This mechanism prevents the algorithm from revisiting these solutions, thereby avoiding cycling and encouraging exploration of new areas in the search space.

This algorithm is particularly useful when computational resources are limited or when a global optimal solution is not strictly required. By focusing on local improvements, Tabu Search sacrifices the guarantee of global optimality for a significant reduction in computation time. The quality of the solution and the required computational time are strongly influenced by the initial guess and the design of the tabu list.

Definition 2 (Neighborhood). Given the Pareto front $\mathcal{F}$ and search width ω_{PF} and ω_r, the *neighborhood* of a solution (C', N', Q, r), denoted by $\mathcal{N}(C', N', Q, r, \mathcal{F}, \omega_{PF}, \omega_r)$, is the set of solutions obtained

by making controlled adjustments to the decision variables (C', N', Q, r) within specified bounds defined by the hyperparameters ω_{PF} and ω_r. Formally:

$$\mathcal{N}(C', N', Q, r, \mathcal{F}, \omega_{PF}, \omega_r) = \left\{ (C'_n, N'_n, Q_n, r_n) \;\middle|\; \begin{array}{l} (C'_n, N'_n) \in \{(C'', N'') \in \mathcal{F} \mid \max\left\{|C'' - C'|, |N'' - N'|\right\} \leq \left\lceil \omega_{PF} \cdot \frac{\max\{p,q\}}{\min\{p,q\}} \right\rceil \right\} \\ r_n \cdot \mathbb{1}\{Q_n \notin \mathbb{Q}_R\} = 0 \\ |r_n - r| < \omega_r, \quad r_n \in \big(0, \delta(C_0 + C'_n)\big), \quad \forall Q_n \in \mathbb{Q}_R \end{array} \right\}$$

Pseudocode of Tabu Search

The step-by-step procedure of the algorithm is outlined below.

Algorithm 3: Tabu Search

Input: Simulation model $\mathcal{M}$, penalty function $\mathcal{P}$, initial capacity C_0, initial number of caregivers N_0, total budget K, cost of adding a bed p, cost of hiring a caregiver q, maximal percentage of reserved beds δ, set of queueing strategies $\mathbb{Q}$, set of reservation-supporting strategies $\mathbb{Q}_R$, initial guess $(\hat{C}', \hat{N}', \hat{Q}, \hat{r})$, search width along the Pareto front ω_{PF}, search width for number of reserved beds ω_r

Output: High-quality solution $(C'_{\text{opt}}, N'_{\text{opt}}, Q_{\text{opt}}, r_{\text{opt}})$

Initialize Tabu List $\mathcal{T} \leftarrow \emptyset$;
Initialize $\mathcal{F} \leftarrow \emptyset$;
Initialize $(C'_{\text{opt}}, N'_{\text{opt}}, Q_{\text{opt}}, r_{\text{opt}}) \leftarrow (\hat{C}', \hat{N}', \hat{Q}, \hat{r})$;
Initialize $\mathcal{P}_{\min} \leftarrow \mathcal{P}(\mathcal{M}(C_0 + \hat{C}', N_0 + \hat{N}', \hat{Q}, \hat{r}))$;
for $C' \leftarrow 0$ **to** $\left\lfloor \frac{K}{p} \right\rfloor$ **do**
  for $N' \leftarrow 0$ **to** $\left\lfloor \frac{K}{q} \right\rfloor$ **do**
    if $pC' + qN' \leq K$ **then**
      if (C', N') *is not dominated by any solution in* $\mathcal{F}$ **then**
        Remove all solutions in $\mathcal{F}$ that are dominated by (C', N');
        Add (C', N') to $\mathcal{F}$;
      end
    end
  end
end
while *improved* **do**
  improved $\leftarrow$ False;
  for $(C'_n, N'_n, Q_n, r_n) \in \mathcal{N}(C'_{opt}, N'_{opt}, Q_{opt}, r_{opt}, \mathcal{F}, \omega_{PF}, \omega_r)$ **do**
    if $(C'_n, N'_n, Q_n, r_n) \notin \mathcal{T}$ **then**
      Evaluate $\mathcal{P}_{\text{current}} \leftarrow \mathcal{P}(\mathcal{M}(C_0 + C'_n, N_0 + N'_n, Q_n, r_n))$;
      if $\mathcal{P}_{current} < \mathcal{P}_{min}$ **then**
        $(C'_{\text{opt}}, N'_{\text{opt}}, Q_{\text{opt}}, r_{\text{opt}}) \leftarrow (C'_n, N'_n, Q_n, r_n)$;
        $\mathcal{P}_{\min} \leftarrow \mathcal{P}_{\text{current}}$;
        improved $\leftarrow$ True;
      end
      Add (C'_n, N'_n, Q_n, r_n) to Tabu List $\mathcal{T}$;
    end
  end
end
return $(C'_{\text{opt}}, N'_{\text{opt}}, Q_{\text{opt}}, r_{\text{opt}})$;

Run-Time Analysis of Tabu Search

To analyze the efficiency of Tabu Search, we introduce the concept of *prediction loss*, which quantifies the discrepancy between the predicted high-quality solutions and the actual optimal solutions identified during the search process.

Definition 3 (Neighborhood Optimal). A solution $X = (C', N', Q, r)$ is said to be *neighborhood optimal* with respect to the search widths ω_{PF}, ω_r and Pareto front $\mathcal{F}$ if it achieves the minimum penalty $\mathcal{P}(\mathcal{M}(C', N', Q, r))$ within its neighborhood $\mathcal{N}(C', N', Q, r, \mathcal{F}, \omega_{PF}, \omega_r)$. Formally, X is neighborhood optimal if:

$$\mathcal{P}(\mathcal{M}(C', N', Q, r)) \leq \mathcal{P}(\mathcal{M}(C, N, Q', r')) \quad \forall (C, N, Q', r') \in \mathcal{N}(C', N', Q, r, \mathcal{F}, \omega_{PF}, \omega_r)$$

where the neighborhood $\mathcal{N}(C', N', Q, r, \mathcal{F}, \omega_{PF}, \omega_r)$ is defined as:

$$\mathcal{N}(C', N', Q, r, \mathcal{F}, \omega_{PF}, \omega_r) = \left\{ (C'_n, N'_n, Q_n, r_n) \;\middle|\; \begin{array}{l} (C'_n, N'_n) \in \left\{ (C'', N'') \in \mathcal{F} \mid \max\{|C'' - C'|, |N'' - N'|\} \leq \left\lceil \omega_{PF} \cdot \frac{\max\{p,q\}}{\min\{p,q\}} \right\rceil \right\} \\ r_n \cdot \mathbb{1}\{Q_n \notin \mathbb{Q}_R\} = 0 \\ |r_n - r| < \omega_r, \quad r_n \in \left(0, \delta(C_0 + C'_n)\right), \quad \forall Q_n \in \mathbb{Q}_R \end{array} \right\}.$$

Definition 4 (Prediction Loss). Given a predicted solution $\hat{X} = (\hat{C}', \hat{N}', \hat{Q}, \hat{r})$ on Pareto front, the global optimal solution $X_{\text{opt}} = (C'_{\text{opt}}, N'_{\text{opt}}, Q_{\text{opt}}, r_{\text{opt}})$, the nearest neighborhood optimal solution subject to $C' \leq \hat{C}'$ is $X_{\text{L}} = (C'_{\text{L}}, N'_{\text{L}}, Q_{\text{L}}, r_{\text{L}})$ and the nearest neighborhood optimal solution subject to $C' > \hat{C}'$ is $X_{\text{R}} = (C'_{\text{R}}, N'_{\text{R}}, Q_{\text{R}}, r_{\text{R}})$, we define the following loss functions:

$$\mathcal{L}_1(\hat{X}, X_{\text{opt}}) = \left\lfloor \max\{|\hat{C}' - C'_{\text{opt}}|, |\hat{N}' - N'_{\text{opt}}|\} \cdot \frac{\min\{p,q\}}{\max\{p,q\}} \right\rfloor$$

$$\mathcal{L}_2(\hat{X}, X_{\text{opt}}) = |\hat{r} - r_{opt}|$$

$$\mathcal{L}_3(\hat{X}, X_{\text{L}}, X_{\text{R}}) = \left\lfloor \max\{\max\{|\hat{C}' - C'_{\text{L}}|, |\hat{N}' - N'_{\text{L}}|\}, \max\{|\hat{C}' - C'_{\text{R}}|, |\hat{N}' - N'_{\text{R}}|\}\} \cdot \frac{\min\{p,q\}}{\max\{p,q\}} \right\rfloor$$

Theorem 3. Assuming that each simulation and penalty function evaluation $\mathcal{P}(\mathcal{M}(C, N, Q, r))$ takes constant time. When $\mathcal{L}_1 \leq \omega_{PF}$ and $\mathcal{L}_2 \leq \omega_r$, the Tabu Search algorithm are guaranteed to achieve the global optimality with run-time complexity

$$\mathcal{O}\left(\omega_{PF} \cdot \omega_r \cdot |\mathbb{Q}_R|\right)$$

Proof. Tabu Search begins by exploring the neighborhood of the initial guess $(\hat{C}', \hat{N}', \hat{Q}, \hat{r})$, defined by the hyperparameters ω_{PF} (the search width along the Pareto front) and ω_r (the search width for the number of reserved beds). The algorithm evaluates all feasible solutions within this neighborhood, including any potential global optimal solution.

Given the conditions $\mathcal{L}_1 \leq \omega_{\text{PF}}$ and $\mathcal{L}_2 \leq \omega_r$, the global optimal solution $(C'_{\text{opt}}, N'_{\text{opt}}, Q_{\text{opt}}, r_{\text{opt}})$ is guaranteed to lie within the initial neighborhood. Therefore, in the first iteration, the algorithm will discover the global optimal solution by evaluating all solutions in this neighborhood.

In the second iteration, the algorithm re-evaluates the neighborhood around the previously found optimal solution. Since the global optimal solution has already been identified and the search space has been sufficiently explored, no better solution exists within the neighborhood. Consequently, the algorithm recognizes the current solution as locally optimal and terminates.

Thus, the run-time complexity is determined by the number of evaluations in the first iteration, which involves searching through ω_{PF} possible adjustments for (C', N'), ω_r possible adjustments for r, and $|\mathbb{Q}_R|$ queueing strategies. Therefore, the overall complexity is:

$$\mathcal{O}\left(\omega_{\text{PF}} \cdot \omega_r \cdot |\mathbb{Q}_R|\right).$$

□

Theorem 4. Assuming that each simulation and penalty function evaluation $\mathcal{P}(\mathcal{M}(C, N, Q, r))$ takes constant time, when $\omega_r \geq \delta \cdot \left(C_0 + \left\lfloor \frac{K}{p} \right\rfloor\right)$, the Tabu Search algorithm is guaranteed to achieve local optimality with worst-case run-time complexity:

$$\mathcal{O}\left(\delta |\mathbb{Q}_R| \left(C_0 + \frac{K}{p}\right) \cdot \mathcal{L}_3\right).$$

Proof. Tabu Search explores the neighborhood defined as:

$$\mathcal{N}(C', N', Q, r, \mathcal{F}, \omega_{PF}, \omega_r) = \left\{ (C'_n, N'_n, Q_n, r_n) \;\middle|\; \begin{array}{l} (C'_n, N'_n) \in \left\{ (C'', N'') \in \mathcal{F} \mid \max\{|C'' - C'|, |N'' - N'|\} \leq \left\lceil \omega_{PF} \cdot \frac{\max\{p,q\}}{\min\{p,q\}} \right\rceil \right\}, \\ r_n \cdot \mathbb{1}\{Q_n \notin \mathbb{Q}_R\} = 0, \\ |r_n - r| < \omega_r, \quad r_n \in \left(0, \delta(C_0 + C'_n)\right), \quad \forall Q_n \in \mathbb{Q}_R. \end{array} \right\}.$$

Since the search is restricted to the Pareto front $\mathcal{F}$, the number of adjustments to (C', N') is bounded by $\mathcal{L}_3$, which represents the number of feasible points on the Pareto front that the algorithm needs to explore.

For each solution (C'_n, N'_n) on the Pareto front, the algorithm evaluates the reserved beds r_n within a range defined by ω_r. The condition $\omega_r \geq \delta \cdot \left(C_0 + \left\lfloor \frac{K}{p} \right\rfloor\right)$ ensures that the entire feasible range for the reserved beds is covered within this neighborhood.

At each neighborhood point, the algorithm considers all reservation-supporting queueing strategies $Q_n \in \mathbb{Q}_R$. Therefore, the total number of evaluations can be expressed as:

$$\mathcal{L}_3 \cdot \delta \left(C_0 + \frac{K}{p}\right) \cdot |\mathbb{Q}_R|.$$

Thus, the worst-case run-time complexity of the Tabu Search algorithm is:

$$\mathcal{O}\left(\delta |\mathbb{Q}_R| \left(C_0 + \frac{K}{p}\right) \cdot \mathcal{L}_3\right).$$

□

Theorem 3 shows that by increasing the search widths ω_{PF} and ω_r, the Tabu Search algorithm has a greater likelihood of achieving global optimality. However, this increase comes at the cost of higher computational time, as the run-time complexity scales with $\omega_{\text{PF}} \cdot \omega_r \cdot |\mathbb{Q}_R|$. When both search widths are sufficiently large, the algorithm effectively explores all feasible neighborhoods along the Pareto front. In this case, Tabu Search degenerates to Pareto Front Search, systematically evaluating all non-dominated solutions while securing global optimality. Therefore, the search widths ω_{PF} and ω_r offer a way to balance computational cost and solution quality.

Theorem 4, on the other hand, implies that while it is difficult for the initial guess to directly hit the global optimum, or even be very close to it, Tabu Search can still efficiently identify high-quality solutions. If the initial guess is sufficiently close to some local optimal solutions (which is assumed to be neighborhood optimal under the given search widths) and ω_r is large enough, the algorithm can quickly converge to these solutions. This significantly reduces search time compared to exploring the entire Pareto front. Consequently, even without global optimality, Tabu Search can achieve a good solution within a practical time frame. These results highlight the flexibility of Tabu Search: by adjusting the search widths, the algorithm can either aim for global optimality with higher computational cost or focus on obtaining a high-quality solution quickly when an initial guess is reasonably close to a local optimum.

Despite the advantages of Tabu Search, one notable drawback is its relatively constrained search direction. The algorithm is prone to being misled by seemingly minor local optima within the current neighborhood, which can steer the search towards suboptimal regions. This behavior resembles a short-term focus that sacrifices potential long-term benefits. In the context of the current problem setting, we consider this a fundamentally challenging issue to resolve. As a result, under adversarial setting, when both ω_{PF} and ω_r are small, the time complexity of Tabu Search can degenerate to that of Pareto Front Search, while still lacking a guarantee of global optimality.

Nevertheless, despite the lack of theoretical lower bounds for Tabu Search with small search widths, we argue that in practice, the algorithm performs exceptionally well. The ability to balance computational cost and solution quality, combined with the flexibility to adjust search parameters, makes Tabu Search a highly effective heuristic approach for the problem at hand.

Theorem 5. Assuming that each simulation and penalty function evaluation $\mathcal{P}(\mathcal{M}(C, N, Q, r))$ takes constant time, the Tabu Search algorithm's worst-case run-time complexity:

$$\mathcal{O}\left(\delta |\mathbb{Q}_R| \left(C_0 + \frac{K}{p}\right) \cdot \frac{K}{\max\{p, q\}}\right)$$

Proof. To construct an adversarial scenario that forces Tabu Search to explore as many points as possible, we design the penalty function $\mathcal{P}$ such that the algorithm is guided through a series of misdirected searches before converging to a local optimum. This construction exploits the limited search direction of Tabu Search and the inherent randomness of the simulation process, which prevents the algorithm from consistently identifying the global trend of the penalty values.

Step 0: Initial Guess

Let the initial guess (where $(\hat{C}, \hat{N}) \in \mathcal{F}$) be $(\hat{C}, \hat{N}, \hat{Q}, \hat{r})$. W.L.O.G., suppose $\hat{Q} \in \mathbb{Q}_R$ and $\hat{C}, \hat{r}$ satisfies the condition:

$$\hat{C} > \left\lfloor \frac{K}{2p} \right\rfloor$$

$$\hat{r} > \frac{\delta \cdot (C_0 + \hat{C})}{2}$$

(We can also construct similar adversarial scenarios for the cases where $\hat{C} > \left\lfloor \frac{K}{2p} \right\rfloor$ and $\hat{r} \leq \frac{\delta \cdot (C_0 + \hat{C})}{2}$; $\hat{C} \leq \left\lfloor \frac{K}{2p} \right\rfloor$ and $\hat{r} > \frac{\delta \cdot (C_0 + \hat{C})}{2}$; or $\hat{C} \leq \left\lfloor \frac{K}{2p} \right\rfloor$ and $\hat{r} \leq \frac{\delta \cdot (C_0 + \hat{C})}{2}$.)

Step 1: Vertical Misdirection

We design the penalty function $\mathcal{P}$ such that $C_{opt} > \hat{C}$ and the optimal solution in the current neighborhood has a penalty value that encourages the search to move vertically downward in the reserved beds dimension r. Specifically, set the penalty values such that, arbitrarily choose a $Q_1 \in \mathbb{Q}_R$:

$$\mathcal{P}_1 = \mathcal{P}(\mathcal{M}(C_0 + \hat{C}, N_0 + \hat{N}, Q_1, \hat{r} - \omega_r))$$

is the minimum penalty within the neighborhood centered at $(\hat{C}, \hat{N}, \hat{Q}, \hat{r})$. Then further design the penalty function so that, arbitrarily choose $Q_2 \in \mathbb{Q}_R$:

$$\mathcal{P}_2 = \mathcal{P}(\mathcal{M}(C_0 + \hat{C}, N_0 + \hat{N}, Q_2, \hat{r} - 2\omega_r)) < \mathcal{P}_1.$$

And $\mathcal{P}_2$ is the minimum penalty within the neighborhood centered at $(\hat{C}, \hat{N}, Q_1, \hat{r} - \omega_r)$. This construction forces the search to continue moving downward along the r dimension while keeping (C, N) constant.

We can repeatedly apply the above method, guiding the search downward in the r dimension while keeping C and N constant.

After m iterations, this process yields a point $(\hat{C}, \hat{N}, \hat{Q}, r - m \cdot \omega_r)$, where $r - m \cdot \omega_r < \omega_r$. This systematic descent ensures that the algorithm explores the entire feasible range for r within the current neighborhood before reaching the lower boundary.

Step 2: Horizontal Misdirection

Once the vertical exploration reaches the lower boundary for r, we redirect the search horizontally by designing the penalty function to guide the algorithm to an adjacent region in the Pareto front $\mathcal{F}$. Specifically, set the penalty function such that the optimal point in the neighborhood is obtained by decreasing C and increasing N while keeping r constant. Formally, we set:

$$\mathcal{P}_{m+1} = \mathcal{P}\left(\mathcal{M}(C_0 + \hat{C}_{-1}, N_0 + \hat{N}_{+1}, Q_m, r - m \cdot \omega_r)\right),$$

where $Q_m \in \mathbb{Q}_R$; $(\hat{C}_{-1}, \hat{N}_{+1})$ is the adjacent point to $(\hat{C}, \hat{N})$ on Pareto front $\mathcal{F}$ such that $\hat{C}_{-1} < \hat{C}$ and $\hat{N}_{+1} > \hat{N}$; $\mathcal{P}_{m+1} < \mathcal{P}_m$ and $\mathcal{P}_{m+1}$ is the minimum penalty within the neighborhood centered at $(\hat{C}, \hat{N}, Q_m, r - m \cdot \omega_r)$.

After this first horizontal redirection, we apply another horizontal redirection by setting:

$$\mathcal{P}_{m+2} = \mathcal{P}\left(\mathcal{M}(C_0 + \hat{C}_{-2}, N_0 + \hat{N}_{+2}, Q_{m+1}, r - m \cdot \omega_r)\right),$$

where $Q_{m+1} \in \mathbb{Q}_R$; $(\hat{C}_{-2}, \hat{N}_{+2})$ is the adjacent point to $(\hat{C}_{-1}, \hat{N}_{+1})$ on Pareto front $\mathcal{F}$ such that $\hat{C}_{-2} < \hat{C}_{-1}$ and $\hat{N}_{+2} > \hat{N}_{+1}$; $\mathcal{P}_{m+2} < \mathcal{P}_{m+1}$ and $\mathcal{P}_{m+2}$ is the minimum penalty within the neighborhood centered at $(\hat{C}_{-1}, \hat{N}_{+1}, Q_{m+1}, r - m \cdot \omega_r)$.

After these two horizontal redirections, we use the same method as in the initial step to guide the search vertically upward along the r dimension. This ensures that the search explores a new vertical sequence of points in the updated neighborhood, effectively covering a different region of the search space.

Step 3: Repetition

By repeating the pattern of vertical **(step 1)** and horizontal **(step 2)** movements, we force the algorithm to explore multiple disjoint vertical regions. This adversarial design ensures that the search visits a large number of points before finally converging to a local optimum near the boundary of the search space.

Complexity Analysis

In this adversarial scenario, the number of explored points is proportional to the total number of feasible (C', N') pairs, which is bounded by:

$$\frac{K}{\max\{p, q\}}.$$

For each (C', N') pair, the number of feasible values for r is proportional to $\delta(C_0 + C')$. Considering the reservation-supporting queueing strategies $|\mathbb{Q}_R|$, the total number of evaluations becomes:

$$\left(C_0 + \frac{K}{p}\right) \cdot \frac{K}{\max\{p, q\}} \cdot \delta \cdot |\mathbb{Q}_R|.$$

Therefore, the worst-case run-time complexity of the Tabu Search algorithm is:

$$\mathcal{O}\left(\delta|\mathbb{Q}_R|\left(C_0 + \frac{K}{p}\right) \cdot \frac{K}{\max\{p, q\}}\right).$$

□

Remarks

We always construct our adversarial attack on Reservation Strategy Set $\mathbb{Q}_R$, if the initial guess $\hat{Q} \notin \mathbb{Q}_R$, we misdirect it into $\mathbb{Q}_R$ at the very first step.

Summary

We summarize our theoretical analysis of the algorithms in the table below (see Table 29). This table provides a concise comparison of the assumptions, solution quality, and time complexity for Exhaustive Search, Pareto Front Search, and Tabu Search under various conditions.

Table 29: Comparison of Algorithms

Algorithm	Assumptions	Solution Quality	Time Complexity
Exhaustive Search	$\delta C_0 \gg 1$	Global Optimality	$\mathcal{O}\left(\delta\lvert\mathbb{Q}_R\rvert\left(C_0 + \frac{K}{p}\right) \cdot \frac{K^2}{pq}\right)$
Pareto Front Search	$\delta C_0 \gg 1$	(Almost) Global Optimality	$\mathcal{O}\left(\delta\lvert\mathbb{Q}_R\rvert\left(C_0 + \frac{K}{p}\right) \cdot \frac{K}{\max\{p,q\}}\right)$
Tabu Search	$\omega_{\text{PF}} \geq \mathcal{L}_1, \omega_r \geq \mathcal{L}_2$	(Almost) Global Optimality	$\mathcal{O}\left(\omega_{\text{PF}} \cdot \omega_r \cdot \lvert\mathbb{Q}_R\rvert\right)$
	$\omega_r \geq \delta\left(C_0 + \left\lfloor\frac{K}{q}\right\rfloor\right)$	Local Optimality	$\mathcal{O}\left(\delta\lvert\mathbb{Q}_R\rvert\left(C_0 + \frac{K}{q}\right) \cdot \mathcal{L}_3\right)$
	N/A	Local Optimality	$\mathcal{O}\left(\delta\lvert\mathbb{Q}_R\rvert\left(C_0 + \frac{K}{p}\right) \cdot \frac{K}{\max\{p,q\}}\right)$

Results

We conducted a series of experiments to validate our algorithms and their theoretical analyses. These experiments were designed to observe the performance of Exhaustive Search, Pareto Front Search, and Tabu Search under various configurations and parameter settings. By analyzing the experimental results, we aimed to establish connections with our theoretical findings, particularly in terms of solution quality and computational efficiency. The empirical observations provide insights into how the assumptions and hyperparameters influence the practical performance of each algorithm, confirming the trade-offs identified in our theoretical analysis.

(i) Exhaustive Search v.s. Pareto Front Search v.s. Tabu Search

Due to the expensive time cost associated with Exhaustive Search, we decided to use the following parameters (see Table 30) to restrict the feasible sets to a relatively small range.

Table 30: Parameters

Symbol	Values
K (budget)	50
p (price of the bed)	10
q (price of the caregiver)	3
C_0 (original number of beds)	100
N_0 (original number of caregivers)	50
δ (maximal percentage of reservation)	0.1
$(\hat{C}, \hat{N}, \hat{Q}, \hat{r})$ (initial guess)[in Tabu Search]	(2, 10, `FIFO`, 0)
ω_{PF}(search width along Pareto front)[in Tabu Search]	1
ω_r(search width along reserved beds)[in Tabu Search]	5

(i)(a) Exhaustive Search

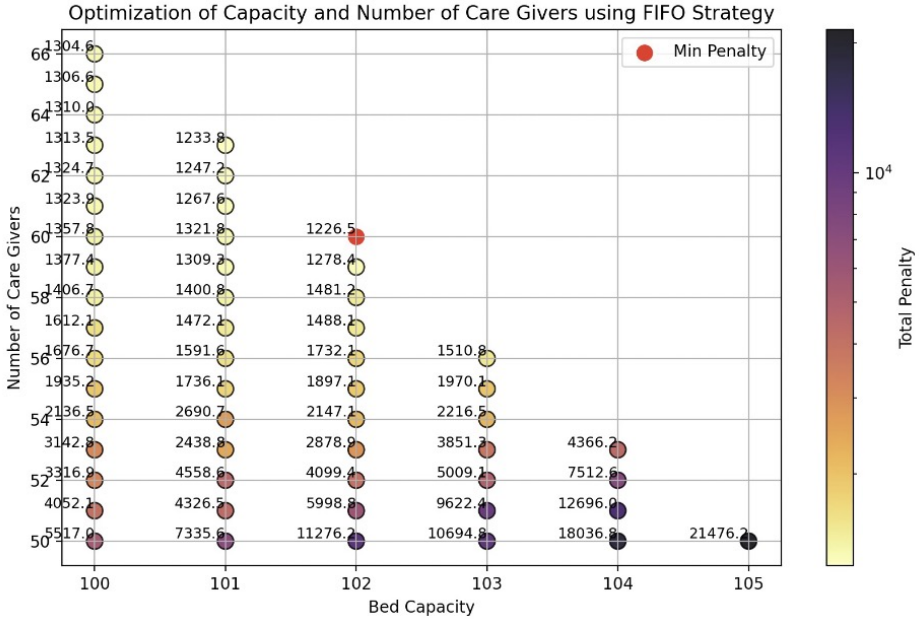

Figure 25: Result of Running Exhaustive Search Under FIFO

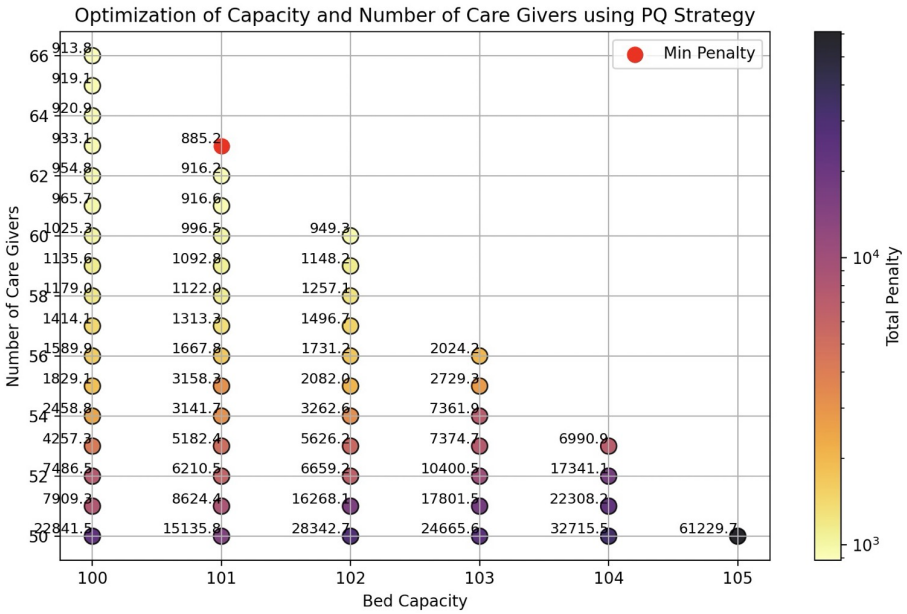

Figure 26: Result of Running Exhaustive Search Under PQ

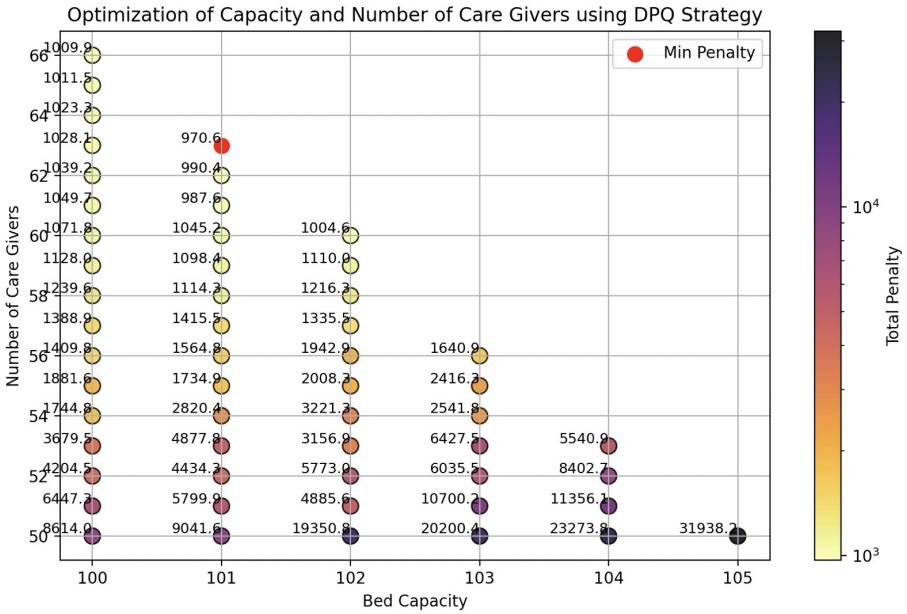

Figure 27: Result of Running Exhaustive Search Under DPQ

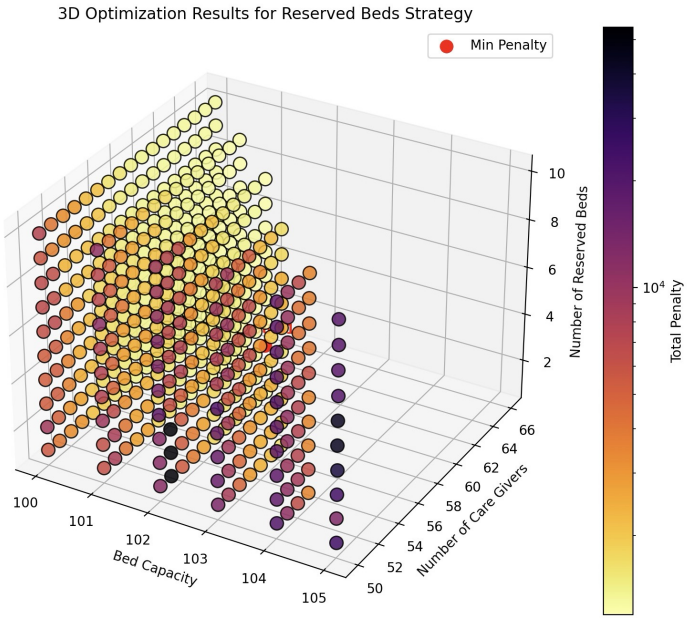

Figure 28: Result of Running Exhaustive Search Under Reservation

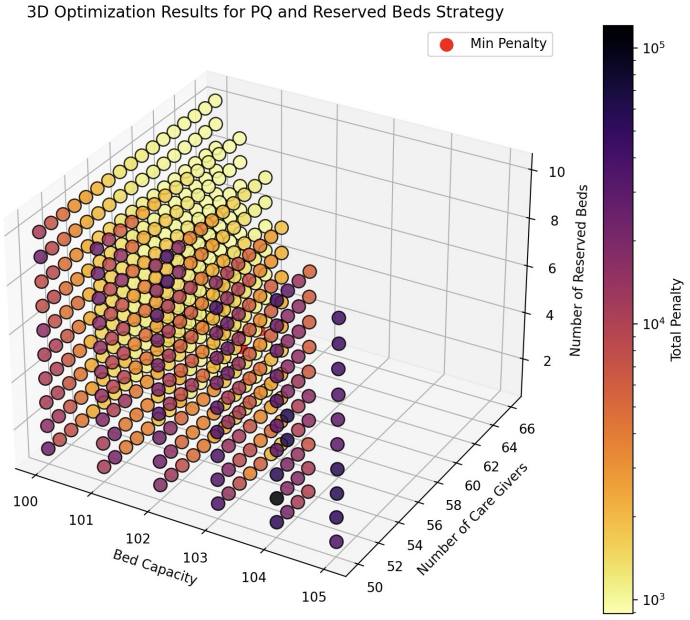

Figure 29: Result of Running Exhaustive Search Under PQ + Reservation

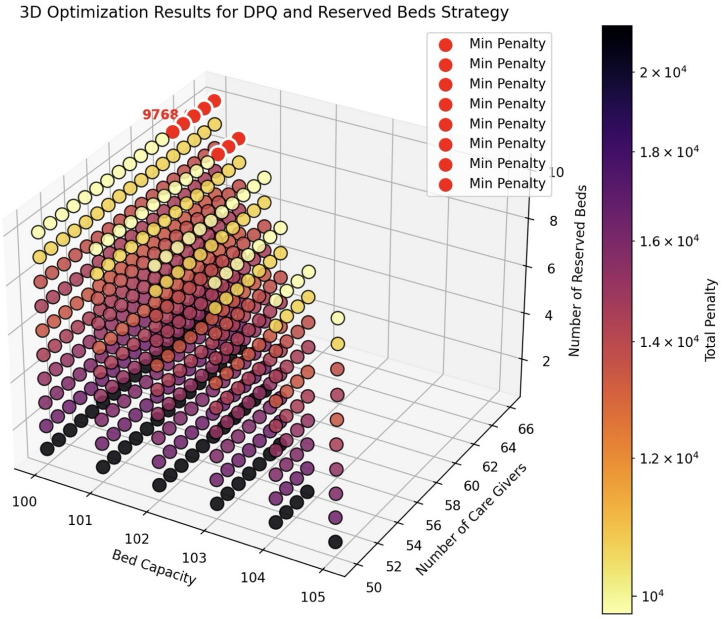

Figure 30: Result of Running Exhaustive Search Under DPQ + Reservation

Table 31: Running Results for Exhaustive Search

Configurations	**Values**
C'(Added Beds)	1
N'(Added Caregivers)	13
Q(Strategy)	PQ
r(Reserved Beds)	0
Function Execution Time	4371.18 seconds

(i)(b) Pareto Front Search

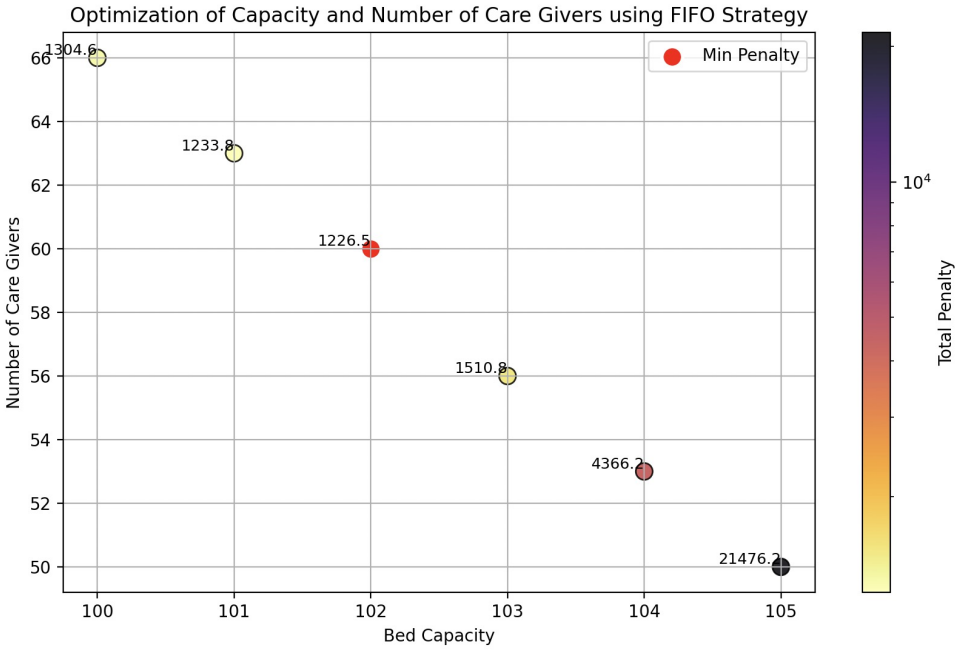

Figure 31: Result of Running Pareto Front Search Under FIFO

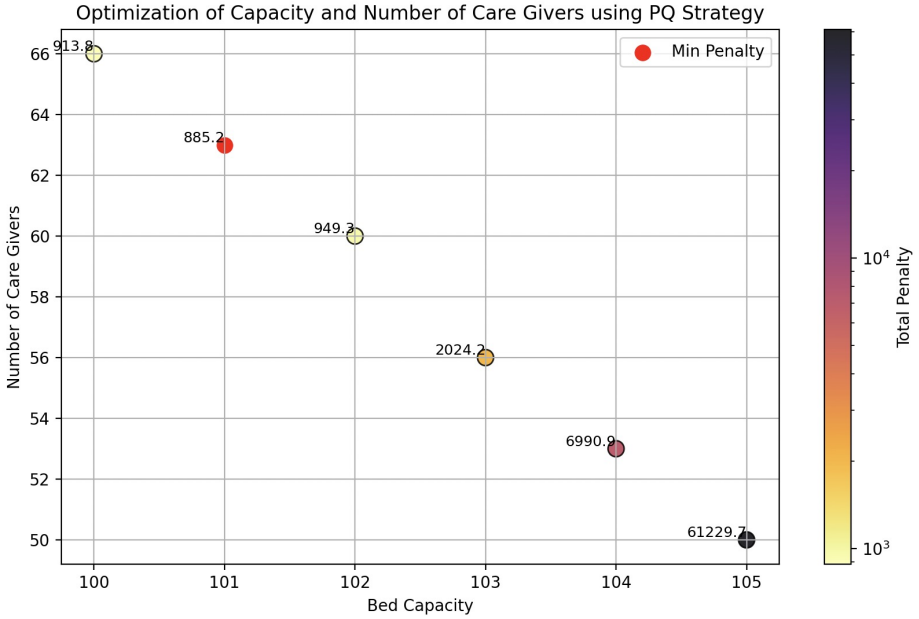

Figure 32: Result of Running Pareto Front Search Under PQ

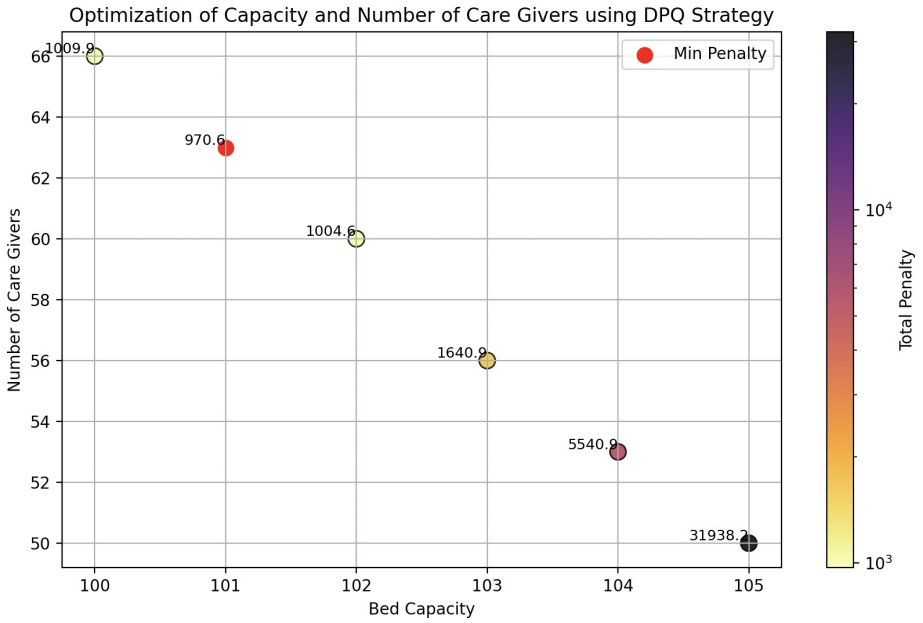

Figure 33: Result of Running Pareto Front Search Under DPQ

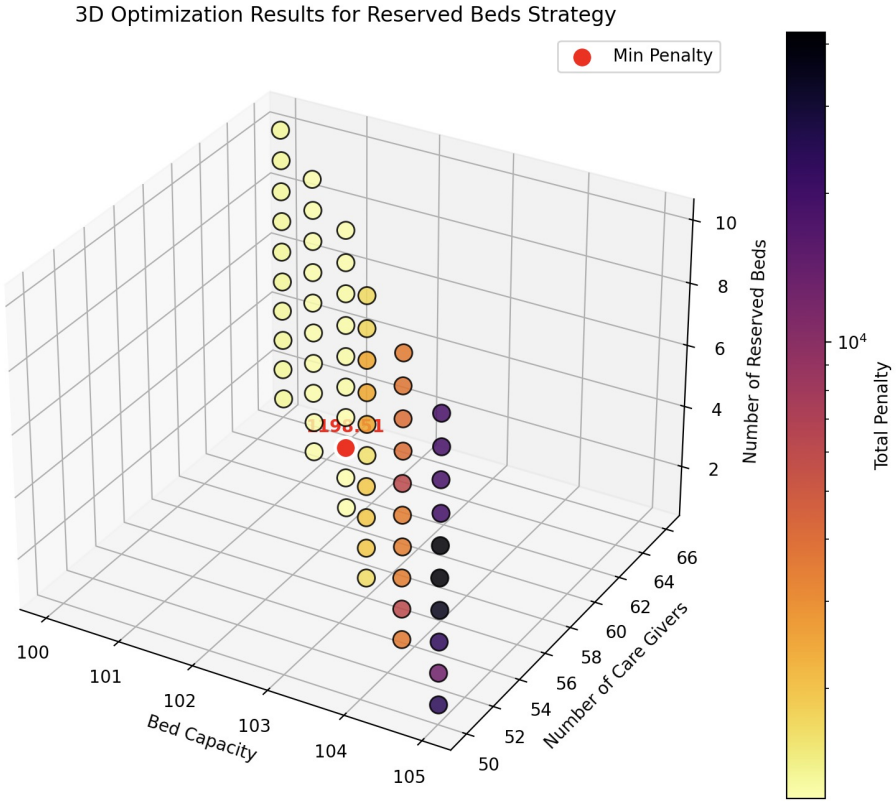

Figure 34: Result of Running Pareto Front Search Under Reserved_Beds

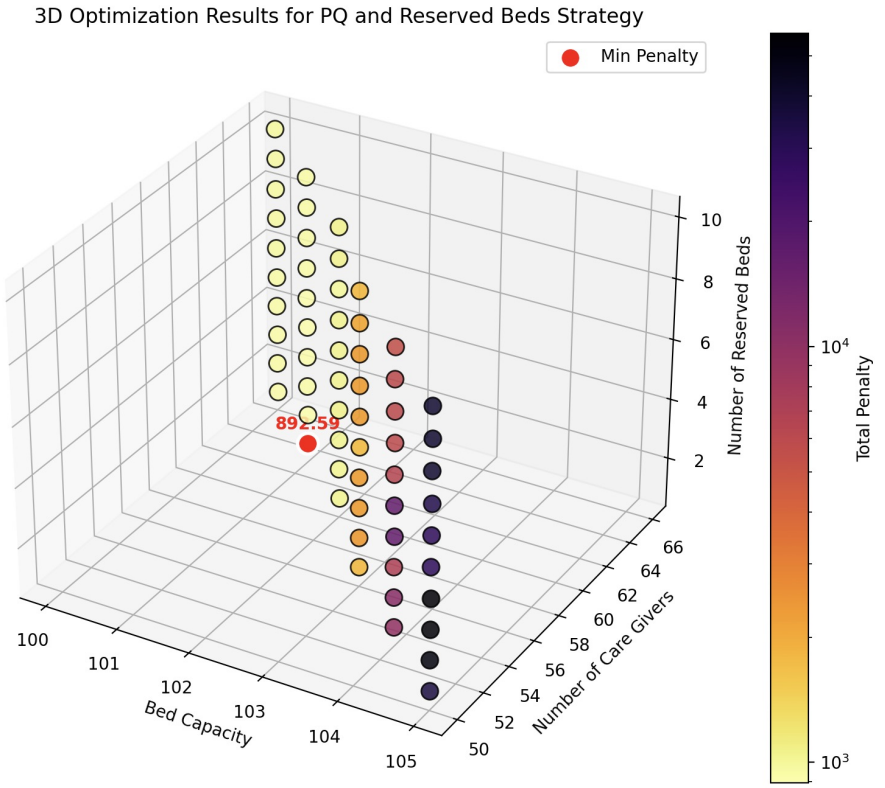

Figure 35: Result of Running Pareto Front Search Under PQ+Reserved_Beds

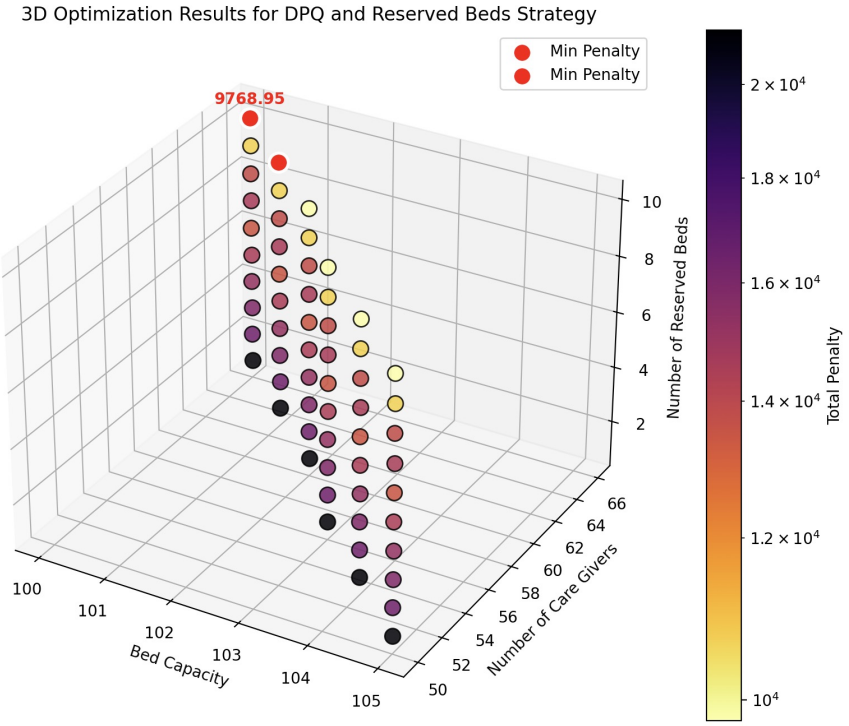

Figure 36: Result of Running Pareto Front Search Under DPQ+Reserved_Beds

Table 32: Running Results for Pareto Front Search

Configurations	**Values**
C'(Added Beds)	1
N'(Added Caregivers)	13
Q(Strategy)	PQ
r(Reserved Beds)	0
Function Execution Time	481.23 seconds

(i)(c) Tabu Search

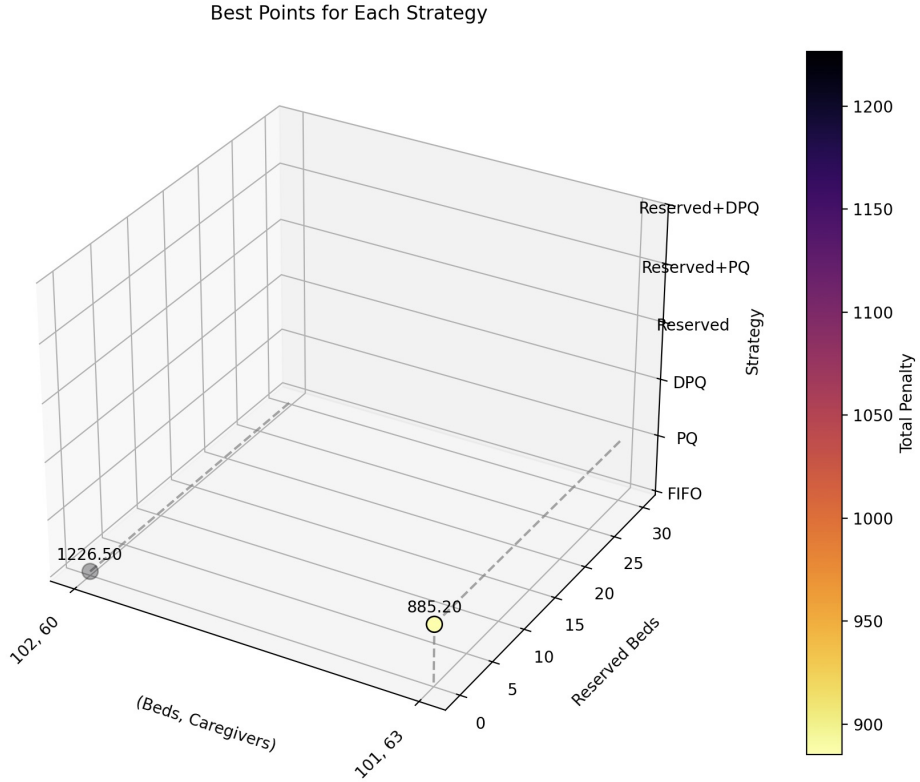

Figure 37: Result of Running Tabu Search

Table 33: Running Results for Tabu Search

Configurations	**Values**
C'(Added Beds)	1
N'(Added Caregivers)	13
Q(Strategy)	PQ
r(Reserved Beds)	0
Function Execution Time	240.41 seconds

These results provide evidence supporting our previous theoretical analysis. Due to the relatively small feasible space and the fact that the initial guess in Tabu Search is close to the optimal solution, all three algorithms converged to the same global optimal solution. As expected, in terms of execution time, Exhaustive Search $\gg$ Pareto Front Search $>$ Tabu Search.

Furthermore, we observed that the ratio of execution times for Exhaustive Search and Pareto Front Search is approximately

$$\frac{1}{2} \cdot \frac{K}{\min\{p, q\}}$$

while the ratio of execution times for Pareto Front Search and Tabu Search is approximately:

$$\frac{\delta(C_0 + \frac{K}{p}) \cdot \frac{K}{\max\{p,q\}}}{(2\omega_{\text{PF}}) \cdot (2\omega_r)}$$

These observations align closely with our theoretical analysis conducted earlier.

(ii) Theoretical and Practical Divergence: Efficiency of Tabu Search with Small Search Widths (ω_{PF}, ω_r)

As we analyzed earlier, from a theoretical perspective, Tabu Search is not a robust algorithm when the search widths are small. However, in our experiments, we observed that the practical results deviate from this theoretical limitation. This discrepancy could be attributed to the lack of extremely low-probability adversarial attacks, which only arise in theoretical settings, within the real simulation model. Using the following parameters(see Table 34), we conducted several experiments with different initial guesses (with different seeds) and found that Tabu Search typically converges rapidly to a local optimal solution.

Table 34: Parameters

Symbol	Values
K (budget)	100
p (price of the bed)	10
q (price of the caregiver)	1
C_0 (original number of beds)	100
N_0 (original number of caregivers)	50
δ (maximal percentage of reservation)	0.2
$(\hat{C}_1, \hat{N}_1, \hat{Q}_1, \hat{r}_1)$ (initial guess 1)	(2, 80, `FIFO`, 0)
$(\hat{C}_2, \hat{N}_2, \hat{Q}_2, \hat{r}_2)$ (initial guess 2)	(5, 50, `Reservation`, 8)
$(\hat{C}_3, \hat{N}_3, \hat{Q}_3, \hat{r}_3)$ (initial guess 3)	(10, 0, `PQ`, 0)
ω_{PF}(search width along Pareto front)	1
ω_r(search width along reserved beds)	2

(ii)(a) $(\hat{C}_1, \hat{N}_1, \hat{Q}_1, \hat{r}_1) = (2, 80, \texttt{FIFO}, 0)$

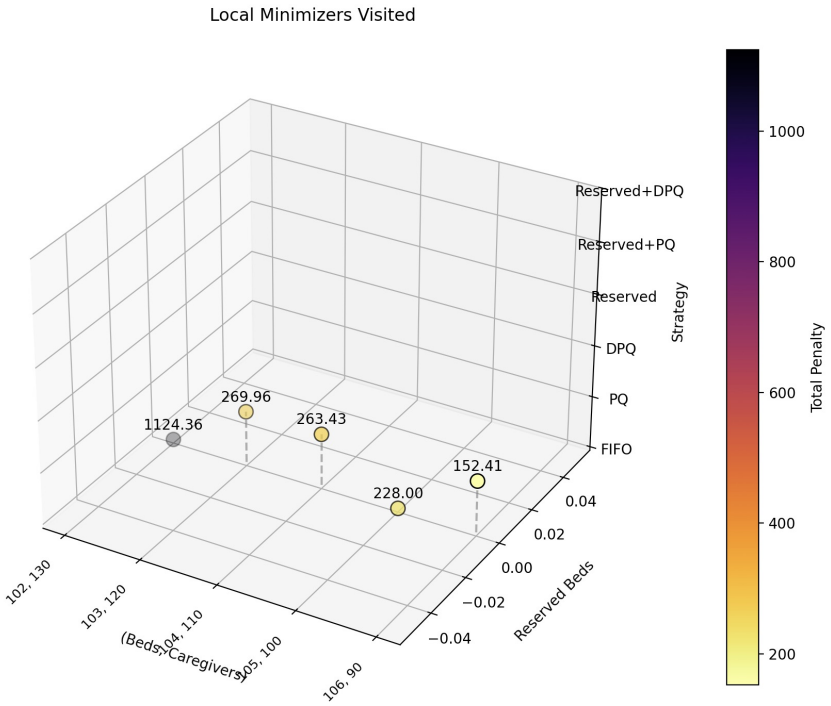

Figure 38: Result of Running Tabu Search for (a)

(ii)(b) $(\hat{C}_2, \hat{N}_2, \hat{Q}_2, \hat{r}_2) = (5, 50, \texttt{Reservation}, 8)$

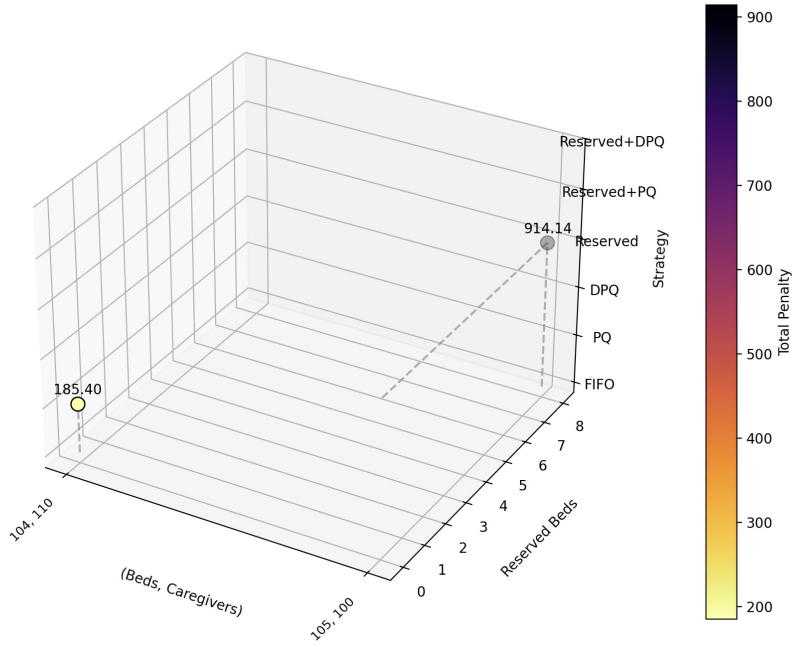

Figure 39: Result of Running Tabu Search for (b)

(ii)(c) $(\hat{C}_2, \hat{N}_2, \hat{Q}_2, \hat{r}_2) = (10, 0, \texttt{PQ}, 0)$

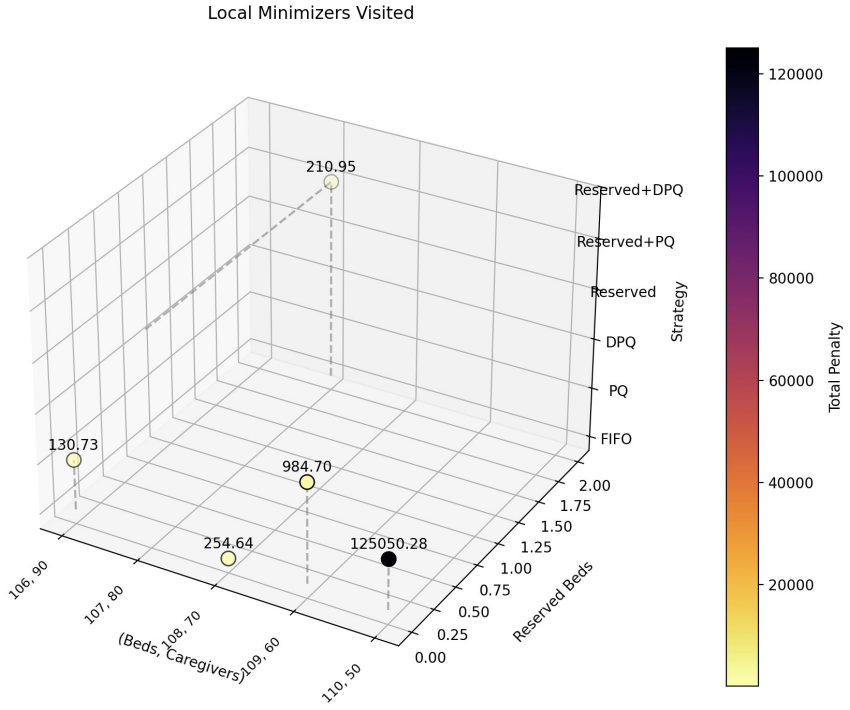

Figure 40: Result of Running Tabu Search for (c)

The above experiments demonstrate that, despite using entirely different initial guesses and setting different seeds, Tabu Search with very small search widths can effectively navigate to a high-quality

solution. Furthermore, the three converged points were identical. Using Pareto Front Search, we subsequently verified that this point represents the global optimality. Therefore, although Tabu Search with small search widths generally lacks strong theoretical guarantees for both time complexity and solution quality, it has proven to be highly efficient in practice.

Future Directions

While our current work focuses on ICU operations, the underlying sequential queuing structure and recurrent service requests are common in many domains. For instance, a restaurant with fluctuating diners, limited tables, and recurring service requests (e.g., ordering courses or drinks) faces similar challenges in managing capacity and prioritizing tasks. Similarly, call centers with varying call volumes and multiple service levels could also benefit from this framework.

Moving forward, making the model more general and customizable would broaden its applicability, allowing users to tailor input parameters to their own contexts, such as arrival patterns, task complexities, resource constraints. This would transform the model into a flexible decision-support tool for capacity optimization. Additionally, there are compelling theoretical fields to explore in **IEOR**, such as developing performance bounds, stability criteria, or integrating advanced optimization techniques like reinforcement learning to dynamically adapt policies as conditions evolve.

Acknowledgments

We would like to express our sincere gratitude to Professor Zeyu Zheng for his invaluable guidance throughout this project. We are also deeply thankful to GSIs Dilara Aykanat and Yuhang Wu for their insightful feedback and constructive suggestions, which greatly enhanced our work.

References

[1] Alistair Johnson, Lucas Bulgarelli, Tom Pollard, Steven Horng, Leo Anthony Celi, and Roger Mark. Mimic-iv clinical database demo (version 2.2). PhysioNet, 2023. https://doi.org/10.13026/dp1f-ex47.