02443 Stochastic Simulation - Project 2
Project choice 2

Group 23:
Gunn Persdóttir Jacobsen - s175799
Julie Clausen - s180733
Søren Skjernaa - s223316

22. June 2023

Contents

0 Note

Group formation:
We tried to find a group on the discussion forum and by writing our contact information on all blackboards in the exercise rooms but did not get any response (likely due to the fact that we work in R, as most other groups seem to use Python). To ensure we finished the project on time we started solving it and wrote the report anyways.

Concerning the report:
All R-code used to produce this report's results is viewable in the appendix.

1 Introduction

In this project, our goal is to develop a simulation model that assesses the impact of implementing a specific distribution of bed resources across multiple wards within a hospital. The aim is to only address the problem from a technical standpoint, knowing that the decision-making process also involves organizational and political complexities.
We are considering a period where the hospital has been forced to create a new temporary ward (this ward is called F^*) due to a countrywide epidemic. Given that the hospital has limited resources, they are forced to reorganize using only staff and beds from the current inpatient wards. Currently the hospital has 5 different wards, $\mathcal{W} \in \{A, B, C, D, E\}$ each corresponding to 5 different patient types, represented by $\mathcal{P} \in \{A, B, C, D, E\}$. Every ward has a fixed bed capacity M_i. Patients of type $i \in \mathcal{P}$ arrive to ward $i \in \mathcal{W}$ with exponentially distributed inter-arrival time with rate λ_i. Furthermore, the patient stays at the hospital for an exponentially distributed amount of time with rate μ_i. The table in Figure 1 provides a summary of the parameters associated with each ward and patient.

Ward and patient type	Bed capacity	Arrivals per day (λ_i)	Mean length-of-stay ($1/\mu_i$)	Urgency points
A	55	14.5	2.9	7
B	40	11.0	4.0	5
C	30	8.0	4.5	2
D	20	6.5	1.4	10
E	20	5.0	3.9	5
F^*	*To be decided*	13.0	2.2	*Not relevant*

Figure 1: Parameters associated with each ward and patient type. Ward F^* denotes the new ward and the urgency points (column 5) reflect the "penalty" if a patient of type i is not admitted in Ward i.

In case a patient of type i is arriving at the hospital and the corresponding ward i is full, the patient is relocated to a different ward j. If the relocated ward also is full, then the patient is lost. Figure 2 presents a table of the relocation probabilities that patients of type i are being relocated to wards of type j (note that $i \neq j$).

$\mathcal{P}$ \ $\mathcal{W}$	A	B	C	D	E	F^*
A	-	0.05	0.10	0.05	0.80	0.00
B	0.20	-	0.50	0.15	0.15	0.00
C	0.30	0.20	-	0.20	0.30	0.00
D	0.35	0.30	0.05	-	0.30	0.00
E	0.20	0.10	0.60	0.10	-	0.00
F^*	0.20	0.20	0.20	0.20	0.20	-

Figure 2: Probability, p_{ij}, of relocating a patient of type $i \in \mathcal{P}$ to an alternative Ward $j \in \mathcal{W}$. Includes the new Ward F^*

In the following we will also use $i = 1$ to denote patient type A and ward A, $i = 2$ to denote patient type B and ward B, and so on.

2 Method

We use a similar procedure for assessing the distribution of the beds effect on the number of relocated and lost patients in the hospital queuing system, for the scenario with and without F^* patients. In this section, we describe the main parts of the methods used.

As described in the introduction we use the convention that $i = 1$ corresponds to patient type A and ward type A, $i = 2$ corresponds to patient type B and ward type B, and so on.

2.1 Simulation of the hospital queuing system

We estimate the fraction of relocated and lost patients in the hospital queuing system, using the event-by-event principle. We simulate the arrival of $n = 10000$ patients, with an additional $k = 1000$ patients used for a burn-in period for the simulation.

For each patient, we use the following procedure (the procedure is illustrated without F^*-ward, but the necessary changes are easily assessed):

1. First, we sample an arrival time (time since the arrival of the previous patient), by sampling a single time from the $\exp\left(\sum_{i=1}^{5} \lambda_i\right)$ distribution.

2. Next, we sample a patient type, by drawing from the 5-point distribution with probability $P(\text{Patient type} = \text{i}) = \frac{\lambda_i}{\sum_{i=1}^{5} \lambda_i}$.

3. Using the sampled patient type i, we sample a time for the length of stay, by sampling from the $\exp(\mu_i)$ distribution.

4. Using the sampled patient type, we sample a potential relocation ward, by sampling from the 5-point distribution with probability $P(\text{Patient of type } i \text{ relocates to ward } j) = p_{ij}$.

We keep a record of each bed in the hospital, and the time left until the bed is free. When having sampled the four values for each patient, we attempt to assign the patient to a ward of the same type as the patient. If there is a free bed we adjust the time until the bed is free to match the patients length of stay. If the ward is full, we attempt to relocate the patient using the same bed procedure. If that ward is also full, we finally reject the patient from the system.

2.2 Optimizing bed distribution through Simulated Annealing

Using the urgency points given in table 1 we attempt to find an optimal distribution of the beds, which minimizes the penalty score for relocating patients.

Patient type	A	B	C	D	E	F^*
Urgency point	7	5	2	10	5	u_6

Table 1: Penalty for relocating (and or losing) patient of type i.

We implement a few different strategies for running simulated annealing and compare the obtained results. For the simulated annealing we again use a simulation with $n = 11000$ patients where the first 1000 are used for burn-in. Our simulated annealing process is described below:

0. We try two different methods for constructing the optimization problem:

    - **Method 1:** We simulate 100 patient queues, which are reused for every step of the simulated annealing optimization process. When assessing the cost of a proposed bed distribution, we run the hospital simulation on each of the 100 patients and use the mean of the 100 costs as the cost for the proposed bed distribution.

      The idea of reusing the same 100 patient queues is to avoid confounding potential improvements in the proposed bed distribution, with the randomness introduced by sampling new patient queues during optimization. We use the mean of 100 patient queues, to make the end result more robust towards different patient queues instead of only optimizing over one specific patient queue.

    - **Method 2:** We run simulated annealing where we in each step, i.e. for each new proposed bed distribution, sample a new patient queue and find the cost for the proposed bed distribution on that patient queue.

      The idea is that running simulated annealing with a new patient queue in each step should force the solution to become robust to the stochastic randomness in the distribution of the patient queue.

1. Initialize the bed distribution so that all wards have an equal amount of beds (with spare beds being given to ward A, then B, and so on).

2. Using the currently suggested bed distribution, we propose a random change to the bed distribution using the following procedure:

    (a) Sample two wards $i, j = 1, 2, 3, 4, 5$ with equal probability.

    (b) Sample a number of beds to swap $l = 1, 2, 3, 4, 5$ with probability $P(l = 1) = 0.6$, $P(l = 2) = 0.2$, $P(l = 3) = 0.1$, $P(l = 4) = 0.06$ and $P(l = 5) = 0.04$. We use decreasing probabilities, so that we in most cases only swap one bed, but in a few cases allow for the swapping of multiple beds, in order to sometimes get a large change in the bed distribution and possibly avoid being stuck in local minimas for the score.

    (c) Remove l beds from ward i and add l beds to ward j.

    (d) If all wards have at least 1 bed accept the proposed change, else reject it and return the unchanged bed distribution.

3. Calculate the cost of the proposed bed distribution, by rerunning the hospital simulation using the patient queue, and the proposed bed distribution. The cost is defined as $\text{Cost} = c + \sum_{i=1}^{5} n_i u_i$, where u_i is the urgency points for patient type i, n_i is the number of

relocated and/or lost patients of type i, and c is the penalty for relocating and/or losing patients of type F^*. We have tried two possible methods for penalizing relocations of patients of type F^*.

- **Penalty type 1:** We use an urgency score of $u_6 = 10$ as the penalty for relocating (and or losing) patients of type F^*, so that $c = n_6 u_6$.
- **Penalty type 2:** For each 0.1% above 5.0% of the patients of type F^* which are relocated, we penalize the score by 100. This means that if 5.7% of type F^* patients are relocated we penalize the score by 700. Thus $c = 100000(f_6 - 0.05)$, where f_6 is the fraction of relocated and/or lost patients of type F^*.

4. Given difference δ = Cost(proposed bed distribution) - Cost(current bed distribution), we always accept the proposed bed distribution if $\delta \geq 0$. If $\delta < 0$ we accept with probability $e^{-\delta/T_i}$, where T_i is the "temperature" at step i of the simulated annealing process.

For the cooling of the temperature, we have used a temperature function defined by $T_i = \frac{T_0}{i}$ where the initial temperature $T_0 = 10000$ was found through experimentation. The result of our temperature function is seen in the following figures.

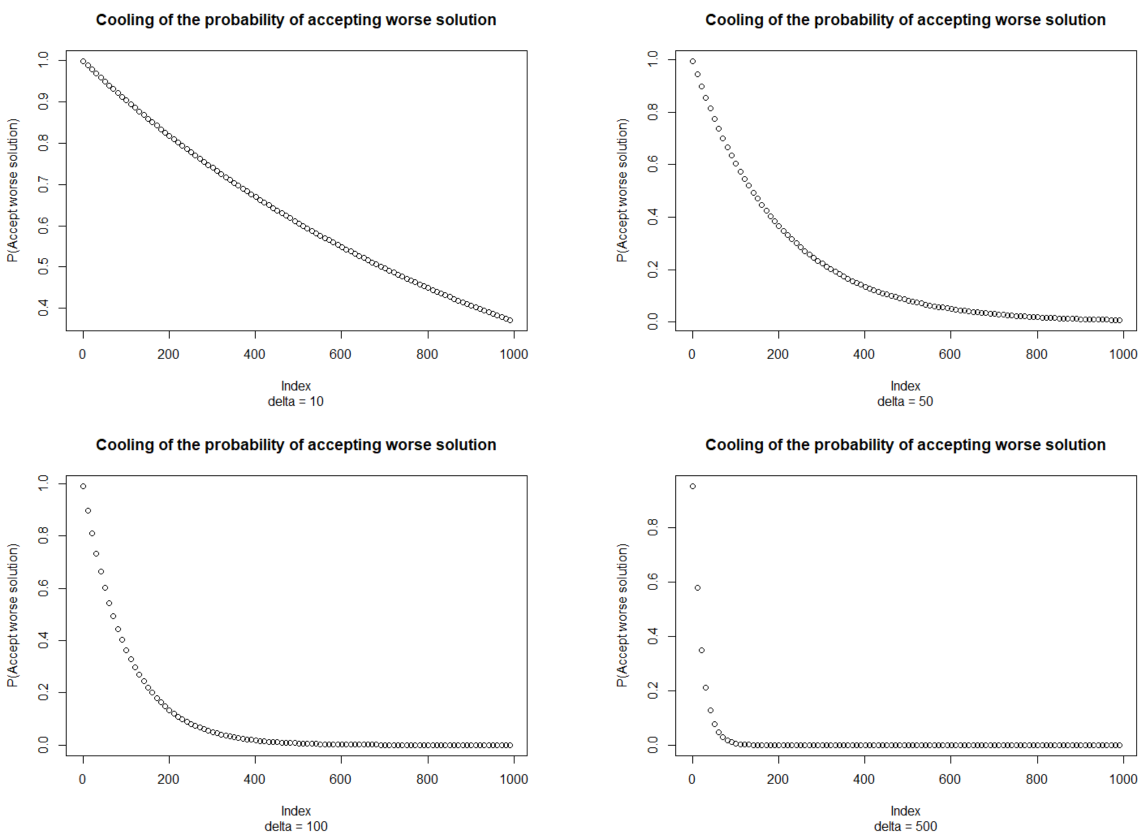

Figure 3: Probability of accepting worse solutions for $\delta = 10, 50, 100, 500$

We see that even towards the end of the simulated annealing process, we accept solutions that worsen the score by $\delta = 10$ with approximately 40% probability. For solutions, which are worse by $\delta = 50$ the probability of accepting them drops to 40% after 200 steps, while it drops to 40% after 100 steps for $\delta = 100$. Thus, in the beginning, we are more open to large changes, while towards the end we do not accept solutions that change worsen the score by much.

3 Hospital simulation without F^*-patients and ward

We initiate the project by first, considering the starting case where we have 5 wards and 5 patient types and thereby exclude F^* at first. We then simulate the patient flow based on the given parameters in the hospital setting. Based on the simulation we conduct further calculations to study the probabilities for bed occupancy, admissions, relocations, and losses, and finally, we also consider the penalty scores. By doing so we are able to assess the impact of various factors on the efficiency of the hospital's ward management system and the allocation of beds in the different wards.

The parameters for this part are given in figure 1 and figure 2. When simulating $n = 10^4$ patients one time (with a burn-in period of 10^3) we get the following distribution of admitted, relocated, and lost patients.

Attribute	Ward A	Ward B	Ward C	Ward D	Ward E
N: patients	3254.000	2377.000	1795.000	1482.000	1092.000
N: admitted	3100.000	1909.000	1156.000	1423.000	795.000
N: relocated	97.000	321.000	532.000	48.000	215.000
N: lost	57.000	147.000	107.000	11.000	82.000

Table 2: Simulation Results

One notable finding in table 2 is that ward C has a relatively higher number of patient relocations compared to the other wards. This suggests that bed availability in ward C might have been insufficient, leading to the relocation of patients to different wards. But this also might make sende concidereing figure 2, since ward C has the highest mean length-of-stay per patient and only 30 beds. We also compute the penalty score, based on the urgency of patient types and the number of relocations and losses. The score for this particular simulation is equal to 6771. We use the score to assess the impact of different bed allocation scenarios or strategies within the simulated ward system later on.

Next, we run the simulation for $k = 10^2$ number of simulations and $n = 10^4$ number of simulated patients with a burn-in period of 10^3. We run the simulation and calculate the probability of all beds being occupied upon arrival for each of the 5 wards.

Patient type	Mean	Variance	Lower 95% CI	Upper 95% CI
A	0.0459	0.0001	0.0437	0.0481
B	0.2112	0.0005	0.2070	0.2154
C	0.3473	0.0006	0.3426	0.3520
D	0.0289	0.0001	0.0267	0.0311
E	0.2887	0.0007	0.2833	0.2942

Table 3: Estimate and uncertainty for the probability of all beds being occupied upon arrival for each patient type.

In Table 3 we see the probabilities represented by the mean of each ward having no available beds for incoming patients. Higher probabilities suggest a higher likelihood of encountering a fully occupied ward upon arrival. Additionally, we have the variance which represents the spread or variability of the data around the mean. This means that a lower variance suggests that the values for a particular patient type are relatively close to the mean, while a higher variance indicates greater dispersion. In table 3, we see that the variance values are quite small, indicating that the data points for each patient type closely align with their respective means.

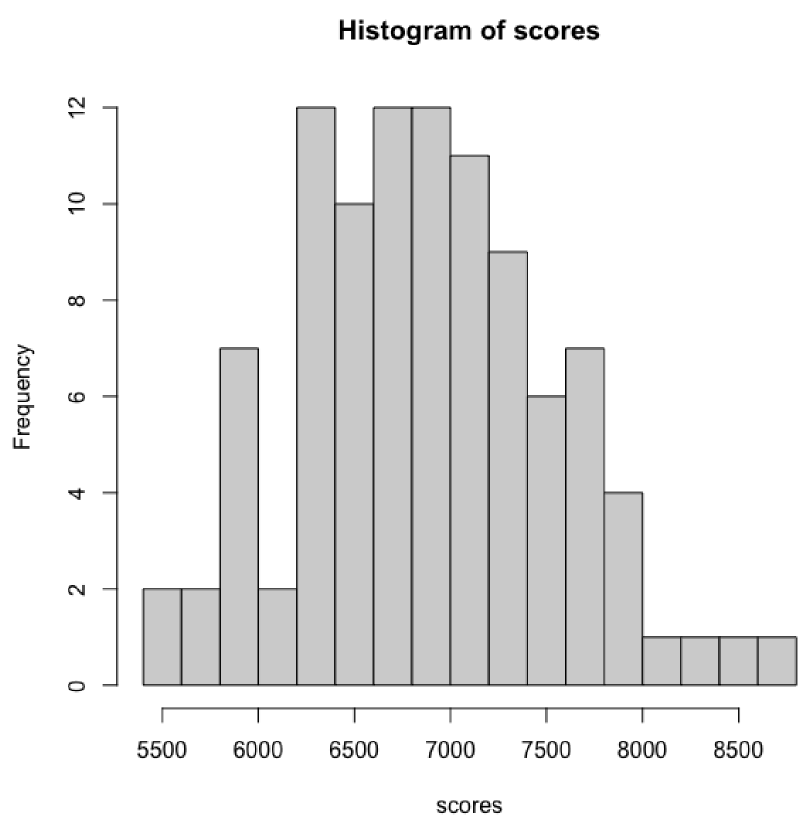

Figure 4: Histogram of the scores from the 100 simulations.

In Figure 4 the score distribution represents the distribution of penalty scores obtained from the simulations of patient allocation. The histogram divides the range of scores into 17 bins and displays the frequency or count of scores falling within each bin. When we calculate the mean, variance, and confidence interval we get the following values.

	Mean	Variance	Lower 95% CI	Upper 95% CI
Score	6883.33	433565.9	6752.68	7013.98

Table 4: Estimate and uncertainty for the score using the initial bed distribution.

The value of the mean represents the estimated score of the initial bed distribution. While the variance explains the variability of the scores around the mean. The confidence interval shows that this means that with a certain level of confidence of 95%, the true mean score is estimated to be within this range. Notice that the previous score we got with only one simulation is contained in this confidence interval.

4 Hospital simulation with F^*-patients and ward

In this section, we simulate the hospital queuing system, in the presence of patients of type F^*. The goal is to find an optimal distribution of the beds, where a maximum of 5% of patients of type F^* are being relocated (and/or lost).

In this section, we assume a total capacity of $k = 165$ beds across all wards of the hospital and we simulated the arrival of $n = 10000$ patients (with an additional 1000 patients used for burn-in of the simulation).

4.1 Initial analysis of the impact of creating ward F^*

As an initial test of the impact of including patients of type F^* in the simulation and creating ward F^*, we run 100 simulations of the hospital queuing system, where we use the bed distribution described in the introduction and move 20% of the beds from each of the wards A, B, C, D, E to ward F^*. Thus the bed distribution is given by:

Ward	A	B	C	D	E	F^*
Beds	44	32	24	16	16	33

Table 5: Bed distribution across the wards for the initial analysis.

Running the 100 simulations, we get the following probabilities for relocating (and/or losing) patients of each type:

Patient type	Mean	Variance	Lower 95% CI	Upper 95% CI
A	0.2392	0.0003	0.2355	0.2428
B	0.3885	0.0004	0.3843	0.3927
C	0.5415	0.0004	0.5375	0.5455
D	0.2169	0.0008	0.2114	0.2224
E	0.5473	0.0006	0.5426	0.5520
F^*	0.0608	0.0002	0.0582	0.0634

Table 6: Estimate and uncertainty for the probability of all beds being occupied upon arrival for each patient type.

We note that 6.08% of patients of type F^* are relocated (and/or lost) which does not meet the goal of only 5% of these patients being relocated. From the simulations we get the following estimate of the score when we use penalty type 1:

	Mean	Variance	Lower 95% CI	Upper 95% CI
Score	15532.44	591708.57	15379.81	15685.07

Table 7: Estimate and uncertainty for the score using the initial bed distribution.

The distribution of the scores (using penalty type 1) is furthermore seen in Figure 5

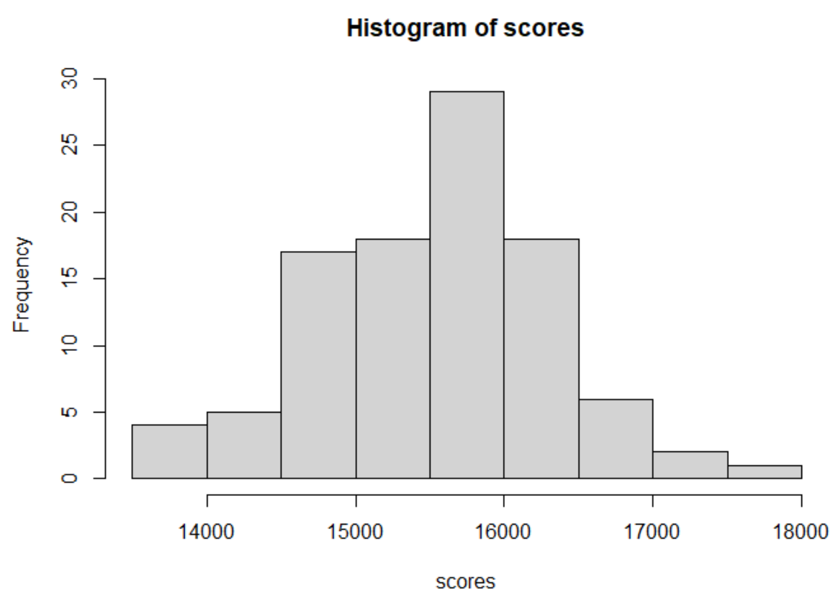

Figure 5: Histogram of the scores from the 100 simulations

4.2 Optimizing the bed distribution for a hospital with ward F^*

We use the different simulated annealing processes described in section 2.2 in order to find an optimal bed distribution w.r.t. the urgency points and the criteria that a maximum of 5% of patients of type F^* are not admitted to ward F^*.

Method 1, Penalty type 1: Figure 6, shows the development in the solution cost, when we ran simulated annealing for $k = 800$ steps reusing the same $l = 100$ patient queues in each step and used $u_6 = 10$ as the penalty for relocating patients of type F^*. We note, that improvements to the solution seem to have died out after 200 steps.

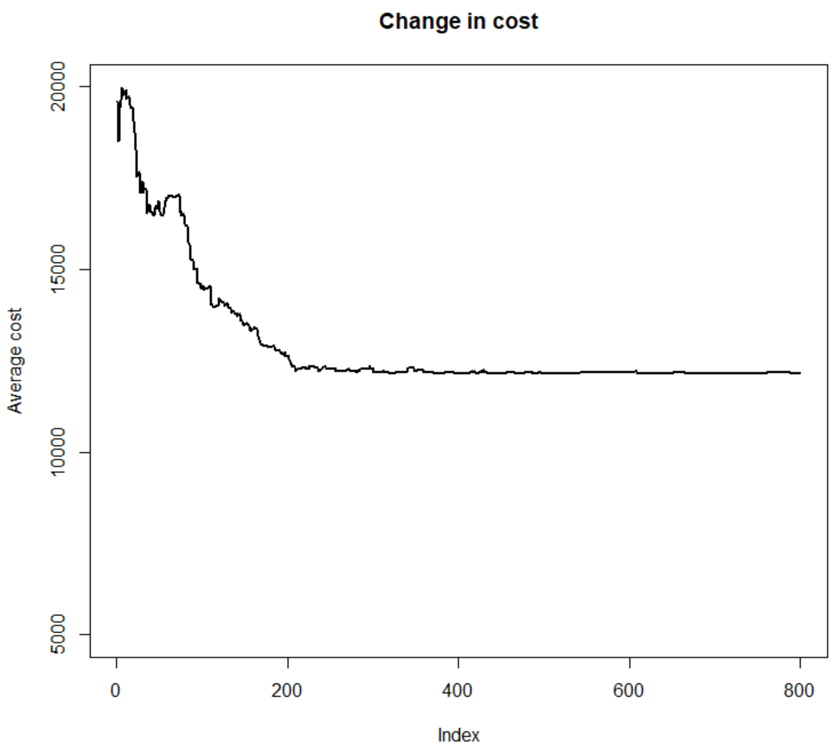

Figure 6: Development in the solution cost, when using method 1 and penalty type 1.

We found the minimal cost to be 12149.37 which was obtained when we used the following optimal bed distribution:

Ward	A	B	C	D	E	F^*
Beds	59	42	1	25	1	37

Table 8: Optimal bed distribution, when using method 1 and penalty type 1.

Method 1, Penalty type 2: Figure 7, shows the development in the solution cost, when we ran simulated annealing for $k = 400$ steps reusing the same $l = 100$ patient queues in each step and used a penalty of 100 for each 0.1% of patients of type F^* that was relocated above 5%. Again, we see that improvements to the solution largely seem to have died out after 200 steps and fully after 300 steps.

We found the minimal cost to be 11375.96 which was obtained when we used the following optimal bed distribution:

Ward	A	B	C	D	E	F^*
Beds	59	44	1	25	1	35

Table 9: Optimal bed distribution, when using method 1 and penalty type 2.

We see that there is no large difference between the solutions of penalty type 1 and 2.

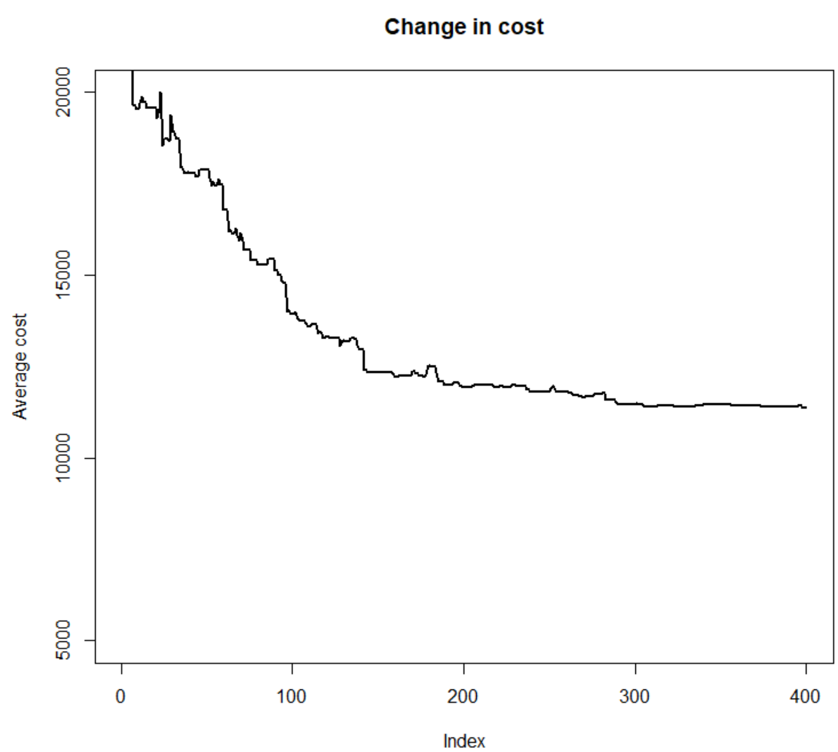

Figure 7: Development in the solution cost, when using method 1 and penalty type 2.

Method 2, Penalty type 1: Figure 8, shows the development in the solution cost, when we ran simulated annealing for $k = 400$ steps where we sampled a new patient queue in each step and used $u_6 = 10$ as the penalty for relocating patients of type F^*. We see that improvements to the solution largely seem to have died out after 150 steps.

We found the minimal cost to be 14201 which was obtained when we used the following optimal bed distribution:

Ward	A	B	C	D	E	F^*
Beds	37	25	22	27	19	35

Table 10: Optimal bed distribution, when using method 2 and penalty type 1.

Method 2, Penalty type 2: Figure 9, shows the development in the solution cost, when we ran simulated annealing for $k = 400$ steps where we sampled a new patient queue in each step and used a penalty of 100 for each 0.1% of patients of type F^* that was relocated above 5%. We see that improvements to the solution largely seem to have died out after 80 steps. When examining the proposed solution at each step we found that the small dip in the cost around step 390 is not due to a changed bed distribution, but a more optimal patient queue.

We found the minimal cost to be 13873 which was obtained when we used the following optimal bed distribution:

Ward	A	B	C	D	E	F^*
Beds	31	28	26	23	24	33

Table 11: Optimal bed distribution, when using method 2 and penalty type 2.

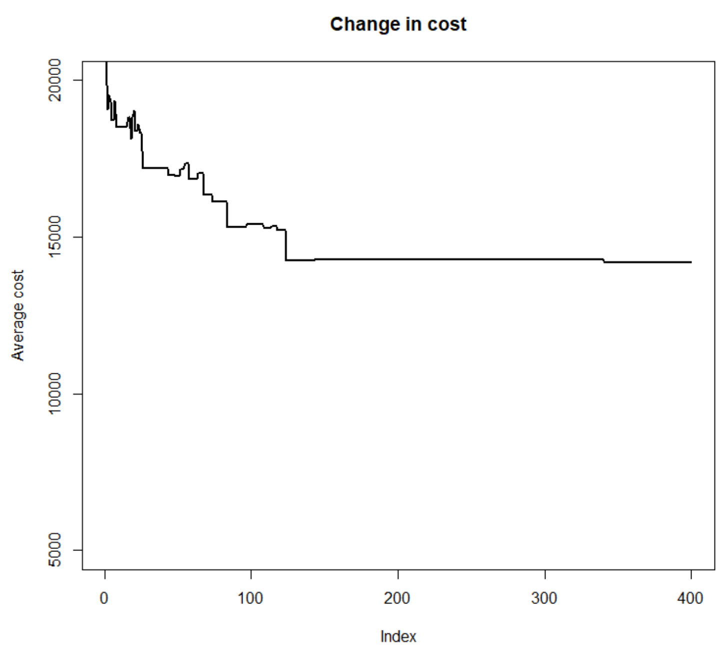

Figure 8: Development in the solution cost, when using method 2 and penalty type 1.

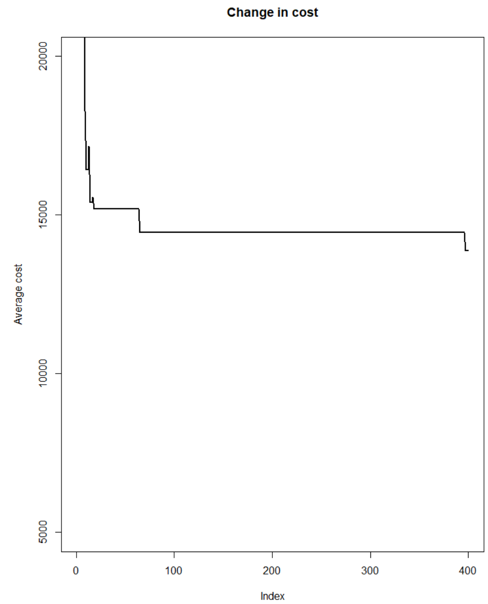

Figure 9: Development in the solution cost, when using method 2 and penalty type 2.

4.3 Comparing the found optimal solutions

While we do not see a large difference in the optimal bed distributions between penalty types 1 and 2, we do notice a large difference between methods 1 and 2. To compare the methods, we run 100 simulations for each of the four optimal solutions from tables 8, 9, 10 and 11 and calculate the obtained scores for each of the methods. The score is calculated using penalty type 1 for all four optimal solutions.

The distribution of the scores for each method is seen in figure 11, while table 12 shows the estimated scores with uncertainty

Score	Mean	Variance	Lower 95% CI	Upper 95% CI
Method 1, Penalty 1	12193.33	197319.23	12105.19	12281.47
Method 1, Penalty 2	12211.3	267870.7	12108.6	12314.0
Method 2, Penalty 1	16063.73	399628.32	15938.30	16189.16
Method 2, Penalty 2	17169.68	409433.01	17042.72	17296.64

Table 12: Estimate and uncertainty for the score using the initial bed distribution.

We see that the penalty type does not have a large influence on method 1. Furthermore, we see that method 1 outperforms method 2. When using method 2, we get worse score for both penalty type 1 and 2, than the scores we got from the initial test in table 7.

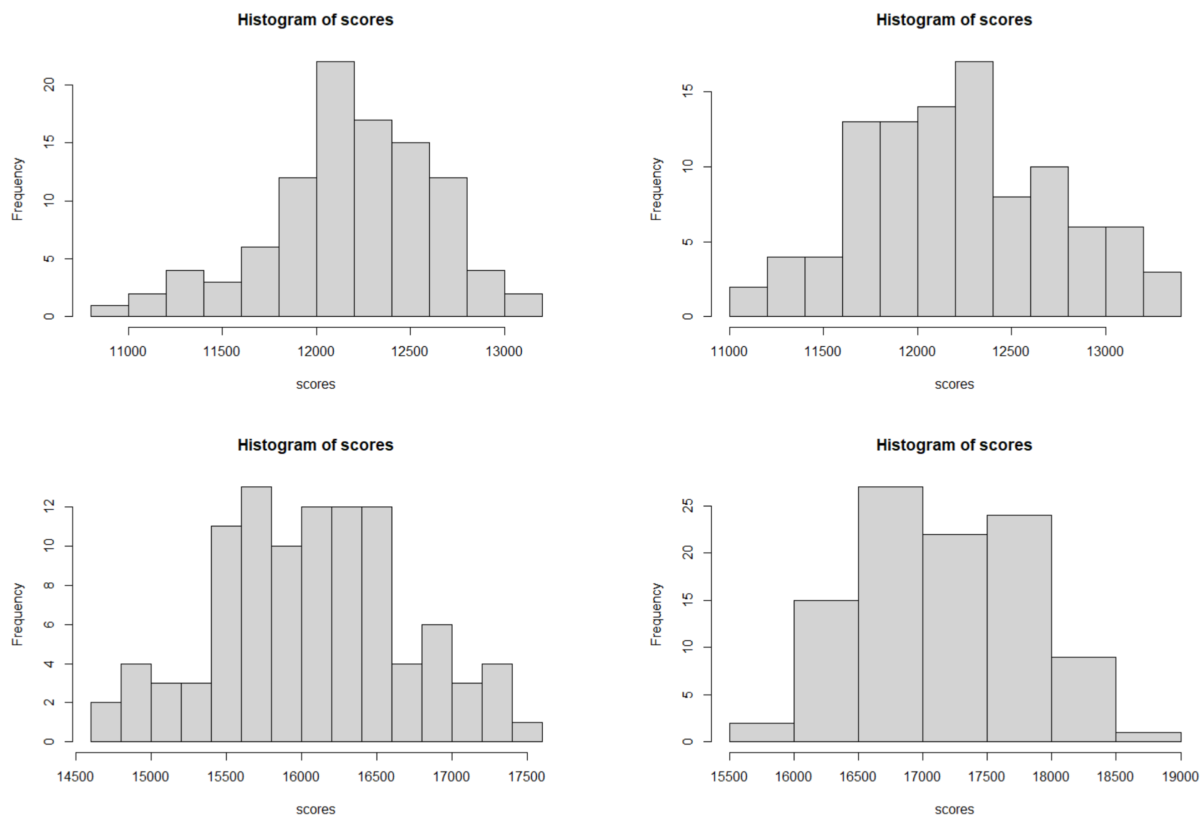

Figure 10: Histogram of the scores for each of the four methods. From left to right: Method 1 Penalty 1, Method 1 Penalty 2, Method 2 Penalty 1, Method 2 Penalty 2.

Table 13 below shows the probability of being relocated and/or lost for each patient type when using the optimal solution found from method 1 and penalty type 1. We note that patients of type F^* are relocated in less than 5% of the cases as we required from a solution.

Patient type	Mean	Variance	Lower 95% CI	Upper 95% CI
A	0.0976	0.0002	0.0946	0.1006
B	0.2304	0.0004	0.2264	0.2345
C	0.9815	0.0000	0.9809	0.9822
D	0.0666	0.0003	0.0632	0.0700
E	0.9708	0.0000	0.9696	0.9720
F^*	0.0296	0.0001	0.0280	0.0311

Table 13: Estimate and uncertainty for the probability of all beds being occupied upon arrival for each patient type for the optimal bed distribution found for method 1 and penalty type 1.

4.4 Discussion

We notice that for method 1 the optimal bed distribution was almost identical for penalty type 1 and 2, where as there was a small change for method 2. For both penalty type 1 and 2, method 2 failed to outperform our initial analysis where 20% of the beds from each ward were relocated to ward F^*. We suspect that this is more reflective of the way method 2 works, than the two penalty types.

When implementing method 2, we saw that the potential gains from proposing a new bed distribution was confounded with the increasing or decreasing cost of the new simulated patient queue in each step of the simulated annealing process. As such, improvements to the bed distribution quickly died out when using method 2. A reason for this may be that the algorithm is more likely to be stuck in a local minima which arose from a favorable patient queue in one step, and not because of a preferable bed distribution. This is avoided when using method 1, where the robustness towards changes in the patient queue comes from simulating and reusing multiple queues and finding a bed distribution that performs well on average over the queues.

We see from the optimal bed distributions that method 1 reduces ward C and E to only one bed each, and relocates the beds to wards A, B, D and F^*. This is in contrast to method 2, which also increases the number of beds in ward A and B but only slightly reduces ward C and E from the initial 27 beds in each ward. The result is that around 98% of all patients for ward C and E are relocated for method 1.

We see that method 1 outperforms method 2 in terms of the score. As ward C have the lowest urgency point and ward E is tied for second lowest together with ward B, method 1 optimizes the score by "ignoring" patients of type C and E in favour of reducing the number of relocations for the other patient types. As the mean length of stay is almost equal for patients B and E (4.0 and 3.9 respectively), and the arrival rate of type B is higher (11.0 compared to 5.0), the optimal bed distribution lowers the score by prioritizing beds for ward B before ward E.

We note that patients of type E have a 60% chance of being relocated to ward C. With both ward C and E having only one bed, patients of type E are likely to be completely lost, and not fill up beds in any of the other wards with higher priority patients. As there is no extra penalty for completely losing patients compared to only relocating them, method 1 obtains a better score by simply "turning away" patients of type E.

The bed distribution obtained from method 1, may reflect a real life scenario where in allocating resources for treating a new disease, we down prioritize certain patient groups and postpone their treatment. A more realistic model, which may avoid the strategy of "closing" certain wards, may be to add an extra penalty for completely losing patients, or to increase the length of stay for patients which are treated in a ward that does not match their patient type. We did not investigate such a simulation before having to hand in.

5 Sensitivity Analysis

5.1 Change in arrival distribution

In the sensitivity analysis of the system, we start out by testing how sensitive the system is to the length-of-stay distribution by replacing the exponential distribution with lognormal(μ, σ^2). For simplistic reasons, we have chosen to compare the system with the initial situation where the bed distribution was a given fixed distribution and where there was no ward or patients of type F^*.
To draw a lenght-of-stay from lognormal(μ, σ^2), we first find $\tilde{\boldsymbol{\mu}}$ (the log mean) such the the mean length-of-stay is the same as in the initial set up. We do this by solving for $\tilde{\mu}_i$ in the following equation for $i = 1, ..., 5$

$$\exp\left(\tilde{\mu}_i + \frac{\sigma^2}{2}\right) = \frac{1}{\mu_i}.$$

We'll be considering 3 different consecutively increasing variances, $\sigma_1^2 = 2/\tilde{\mu}_i{}^2$, $\sigma_2^2 = 3/\tilde{\mu}_i{}^2$ and $\sigma_3^2 = 4/\tilde{\mu}_i{}^2$. The following 3 tables show estimated mean and uncertainty for the probability of all beds being occupied upon arrival for each patient type, when the length-of-stay is drawn from lognormal$(\tilde{\mu}, \sigma_1^2)$, lognormal$(\tilde{\mu}, \sigma_2^2)$ and lognormal$(\tilde{\mu}, \sigma_3^2)$ respectively.

Patient type	Mean	Variance	Lower 95% CI	Upper 95% CI
A	0.0099	0.0001	0.0077	0.0121
B	0.1027	0.0017	0.0946	0.1108
C	0.1832	0.0026	0.1731	0.1932
D	0.0061	0.0001	0.0047	0.0075
E	0.1420	0.0035	0.1303	0.1536

Table 14: Estimate and uncertainty for the probability of all beds being occupied upon arrival for each patient type. The log-normal distribution is given $\sigma^2 = 2/\mu_i^2$

Patient type	Mean	Variance	Lower 95% CI	Upper 95% CI
A	0.4347	0.0019	0.4260	0.4435
B	0.5287	0.0023	0.5193	0.5382
C	0.6585	0.0023	0.6490	0.6681
D	0.3750	0.0032	0.3638	0.3862
E	0.6977	0.0017	0.6895	0.7059

Table 15: Estimate and uncertainty for the probability of all beds being occupied upon arrival for each patient type. The log-normal distribution is given $\sigma^2 = 3/\mu_i^2$

Patient type	Mean	Variance	Lower 95% CI	Upper 95% CI
A	0.6749	0.0010	0.6685	0.6813
B	0.7213	0.0010	0.7151	0.7276
C	0.8189	0.0008	0.8134	0.8244
D	0.6433	0.0026	0.6331	0.6534
E	0.8582	0.0006	0.8532	0.8632

Table 16: Estimate and uncertainty for the probability of all beds being occupied upon arrival for each patient type. The log-normal distribution is given $\sigma^2 = 4/\mu_i^2$

Starting by comparing table 14, 15 and 16 with table 3 we see that the estimated mean is somewhat lower in the first case when using σ_1^2, but noticeable higher when using σ_2^2 and σ_3^2. This is indicating that the system is somewhat sensitive of the variance of the time patients are staying at the hospital.
A substantial increase in the estimated mean for the probability of all beds being occupied results in a higher number of relocated and lost patients compared to the initial setup. Consequently, this leads to a larger penalty score, as depicted in the following tables.

	Mean	Variance	Lower 95% CI	Upper 95% CI
Score	3015.19	784608.3	2839.43	3190.95

Table 17: Estimate and uncertainty for the score using length-of-stay distribution as log-normal with variance $2/\mu_i^2$

	Mean	Variance	Lower 95% CI	Upper 95% CI
Score	27909.34	3312184	27548.22	28270.46

Table 18: Estimate and uncertainty for the score using length-of-stay distribution as log-normal with variance $3/\mu_i^2$

	Mean	Variance	Lower 95% CI	Upper 95% CI
Score	41003.78	1561715	40755.82	41251.74

Table 19: Estimate and uncertainty for the score using length-of-stay distribution as log-normal with variance $4/\mu_i^2$

The distribution of the scores from the 3 different log-normal distributions is furthermore seen in figure 11

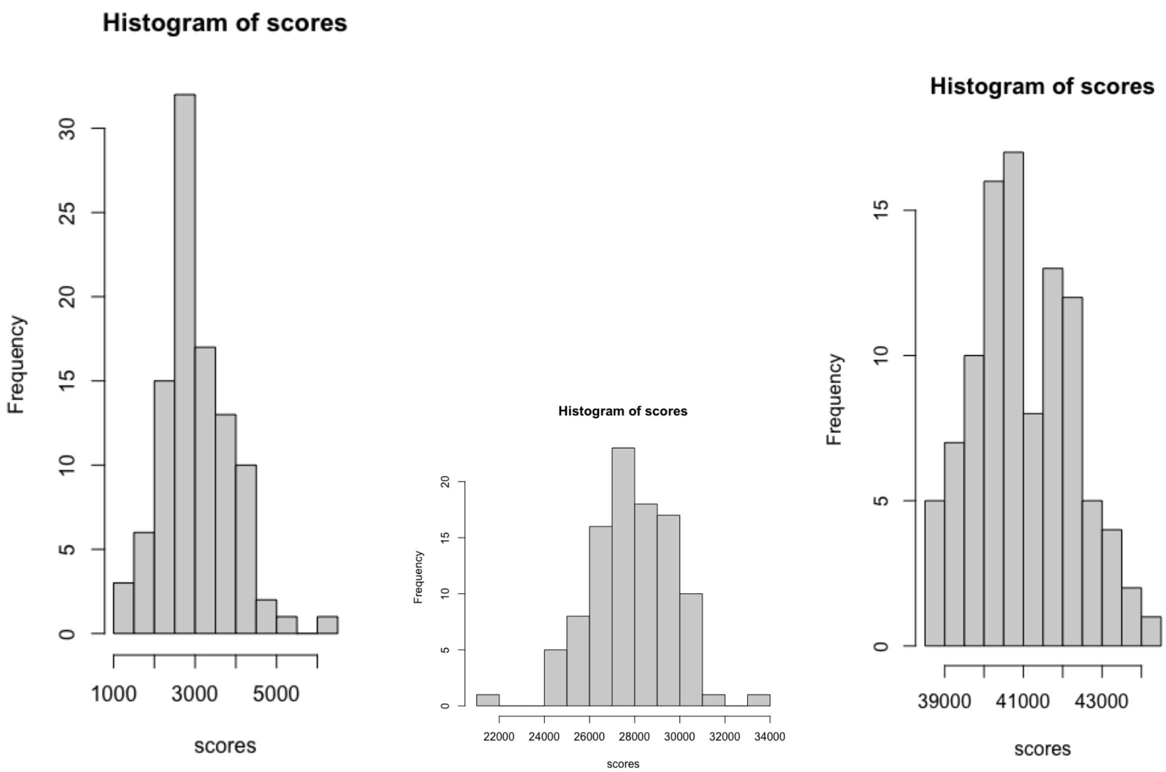

Figure 11: Histogram of the scores for each of the 3 log-normal distributions. From left to right: log-normal with variance $2/\mu_i^2$, log-normal with variance $3/\mu_i^2$, log-normal with variance $4/\mu_i^2$

5.2 Change in maximum capacity

Next we evaluate the hospital queuing systems sensitivity to a change in the total amount of beds. Again, we choose to focus on the original scenario without patients of type F^*. Furthermore, we distribute the beds among wards A, B, C, D and E following the same distribution as original, i.e we use the distribution given in table 20. This is done to show how stable the original distribution is to changes in the total bed capacity.

Ward	A	B	C	D	E
Fraction of total beds	0.34	0.25	0.18	0.12	0.12

Table 20: Bed distribution across the wards.

We run 100 simulations for each of the total bed capacities $150, 155, 160, \ldots, 185$. Averaging the probabilities of being relocated and/or lost for each patient type we get the results seen in figure 12.

Estimating the scores from the 100 simulations we get the estimates and 95% confidence intervals seen in table 21.

Total capacity	Mean	Variance	Lower 95% CI	Upper 95% CI
150	11216	600033	11063	11370
155	9702	462138	9567	9837
160	8130	400034	8005	8256
165	6755	295188	6647	6863
170	5625	283504	5519	5731
175	4802	250729	4702	4901
180	3961	174101	3878	4043
185	3433	177886	3350	3517

Table 21: Estimate and uncertainty for the scores for the different total capacities.

As we would expect the score and the probability of being relocated and/or lost drops as the total number of beds increase. The probability of being relocated drops to close to zero for patients A and D as the total bed capacity increases to 185. These two wards have the lowest mean length of stay (2.0 and 1.4 respectively). Thus, when wards A and D have 62 and 22 beds respectively almost no patients are relocated, and there is minimal gain from increasing the amount of beds on these two wards.

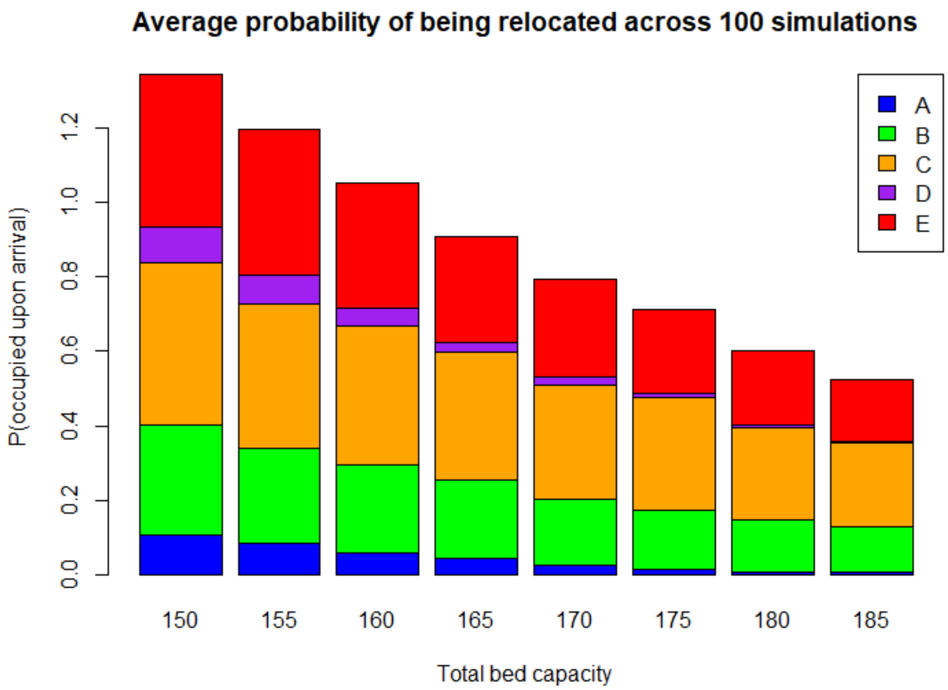

Figure 12: Probability of being relocated for each patient type.

6 Discussion/Conclusion

Based on the simulation of the hospital queuing system without the F^* ward, we obtained several important results. Firstly, the simulation provided insights into the distribution of admitted, relocated, and lost patients across the different wards and patient types. We observed variations in these numbers, indicating differences in the efficiency of bed allocation among the wards. For example, ward C had a higher number of patient relocations compared to the other wards, suggesting a potential issue with bed availability in that ward. On the other hand, ward B had a higher number of lost patients, indicating inefficiencies in accommodating patients in that particular ward.

Moving on to adding a new ward F^* and testing different strategies in this setting. We simulated the hospital queuing system with the inclusion of patients of type F^* and the goal of finding an optimal distribution of beds while ensuring a maximum of 5% of F^* patients are relocated or lost.

The initial analysis evaluates the impact of creating a new ward, F^*, by running 100 simulations with a bed distribution that reallocates 20% of beds from each existing ward to F^*. The simulations reveal that 6.08% of F^* patients are relocated or lost, exceeding the desired limit of 5%. To optimize the bed distribution, two methods are employed: Method 1, which reuses the same patient queues in each step, and Method 2, which samples a new patient queue in each step. Each method is tested with two penalty types: Penalty type 1, where a fixed penalty is applied to relocating F^* patients, and Penalty type 2, where the penalty increases for each percentage point above the 5% relocation limit. Method 1 with Penalty type 1 yields a minimal cost of 12149.37 and an optimal bed distribution where beds are reallocated from wards C and E to other wards, reducing ward C and E to only one bed. Method 1 with Penalty type 2 results in a minimal cost of 11375.96, with a similar bed distribution. On the other hand, Method 2 with Penalty type 1 achieves a minimal cost of 14201, while Method 2 with Penalty type 2 attains a minimal cost of 13873.

Comparing the optimal solutions, it is observed that penalty type does not significantly impact Method 1, and Method 1 outperforms Method 2 regardless of the penalty type. However, all optimal solutions obtained from the methods have higher costs than the initial analysis without the inclusion of F^* ward. The probability of relocation or loss for each patient type in the optimal solution of Method 1 with Penalty type 1 is provided. Notably, F^* patients are relocated in less than 5% of cases, meeting the goal.

In the end Method 1 is more effective due to its ability to account for changes in patient queues, while Method 2 may get stuck in local minima. Furthermore, Method 1 redistributes beds from wards C and E to other wards, while Method 2 exhibits only minor changes. Leading to the analysis suggesting that Method 1 with Penalty type 1 provides the best results for optimizing the bed distribution while limiting the relocation of F^* patients.

Additionally, we examined the system's sensitivity to changes in the length-of-stay distribution in the sensitivity analysis. By replacing the exponential distribution with a lognormal distribution, we observed that the estimated mean probability of all beds being occupied upon arrival varied with different variances. The estimated mean generally increased as the variance of the lognormal distribution increased, indicating the sensitivity of the system to the variance of patients' hospital stays. This resulted in a higher number of relocated and lost patients, leading to larger penalty scores.
Furthermore, we evaluated the system's sensitivity to changes in the total bed capacity. We kept the bed distribution among wards consistent with the original scenario and conducted simulations for different total bed capacities. As the total number of beds increased, the probability of being relocated and/or lost decreased, and the penalty scores decreased accordingly. Wards with lower mean length of stay showed a smaller impact from increasing the bed capacity, as the probability of relocation decreased significantly for those wards.

7 Appendix

7.1 R - code

Simulation of hospital wards

```
# Preamble #####################################################################

## File Description ############################################################
#
#    Sren  Skjernaa - s223316
#   22/06-2023
#
#   Stochastic Simulation
#   Project 2.2
#
#   Notes:
#       ...
#
###############################################################################

## Clean up ####################################################################
rm(list = ls())
if(!is.null(dev.list())) dev.off()



# _ ############################################################################
# Ward simulation ##############################################################

### Arguments:
#       - n        = Number of patients to simulate
#       - burn_in = Burn_in period for the simulation
#       - capacity = Capacities of the different wards
#       - lambda  = Arrival rate of patients of different types
#       - mu      = Length of stay rate of patients of different types
#       - P       = Probabilities of patient i transitioning to ward j
###


### Return:
#       - patients = Number of arrived patients of type i
#       - admitted = Number of admitted patients of type i to ward i
#       - relocated = Number of relocated patients of type i
#       - lost     = Number of lost patients of type i
###


```

```r
ward_sim <- function(n, burn_in, capacity, lambda, mu, P){

    # number of different wards and types of patients
    m <- length(capacity)

    # initialize vectors for storage of results
    patients <- numeric(m)
    admitted <- numeric(m)
    relocated <- numeric(m)
    lost <- numeric(m)

    # Initialize beds (time until bed is free)
    max_capacity <- max(capacity)
    temp <- c()
    for (i in 1:m){
        temp <- c(temp,
                  rep(0, capacity[i]),
                  rep(NA, max_capacity - capacity[i]))
    }
    beds <- matrix(temp, nrow = m, ncol = max_capacity, byrow = TRUE)


    # Main loop of simulation - simulate journey of patient i
    for (i in 1:(burn_in + n)){

        # Sample time until one of the type of arrivals occur
        arrival_time <- rexp(1, rate = sum(lambda))

        # Reduce the time until bed is free
        beds <- beds - arrival_time

        # Sample type of patient that arrived
        p_patient_type <- lambda / sum(lambda)
        patient_type <- sample(1:m, 1, prob = p_patient_type)

        # record patient
        if (i > burn_in){

            patients[patient_type] <- patients[patient_type] + 1
        }

        # Assign patient to ward of same type as patient
        assigned_ward <- patient_type

        # Sample length of stay for patient that arrived
        length_of_stay <- rexp(1, rate = mu[patient_type])

```

```
        # Check originally intended ward for free beds
        bed_index <- which.min(beds[assigned_ward,])
        bed_time <- beds[assigned_ward, bed_index]

        if (bed_time <= 0){      # Assign patient to intended ward

            beds[assigned_ward, bed_index] <- length_of_stay

            # record admission
            if (i > burn_in){

                admitted[patient_type] <- admitted[patient_type] + 1
            }

        } else{                  # Try to relocate patient

            # Sample a relocation
            assigned_ward <- sample(1:m, 1, prob = P[patient_type,])

            # Check relocated ward for free beds
            bed_index <- which.min(beds[assigned_ward,])
            bed_time <- beds[assigned_ward, bed_index]

            if (bed_time <= 0){  # Assign patient to relocation ward

                beds[assigned_ward, bed_index] <- length_of_stay

                # record relocation
                if (i > burn_in){

                    relocated[patient_type] <- relocated[patient_type] + 1
                }

            } else{              # Turn patient away

                # record lost
                if (i > burn_in){

                    lost[patient_type] <- lost[patient_type] + 1
                }
            }
        }
    }



    # Return results
```

```r
    result <- list("patients" = patients,
                   "admitted" = admitted,
                   "relocated" = relocated,
                   "lost" = lost)

    return(result)
}

# _ ############################################################################
# Simulation - without F #######################################################

## Test simulation - 1 run #####################################################

### Parameters -----------------------------------------------------------------
k <- 10^2                                       # Number of simulations to run
n <- 10^4                                       # Number of simulated patients
burn_in <- 10^3                                 # Burn in period
wards <- c("A", "B", "C", "D", "E")             # Name of wards
capacity <- c(55, 40, 30, 20, 20)               # Ward bed capacity
lambda <- c(14.5, 11.0, 8.0, 6.5, 5.0)          # Ward arrivals per day
mu <- c(1/2.9, 1/4.0, 1/4.5, 1/1.4, 1/3.9)  # Ward mean length of stay

P <- matrix(c(0, 0.05, 0.10, 0.05, 0.80,        # Probability patient i
                  0.20, 0, 0.50, 0.15, 0.15, # relocates to ward j
                  0.30, 0.20, 0, 0.20, 0.30,
                  0.35, 0.30, 0.05, 0, 0.30,
                  0.20, 0.10, 0.60, 0.10, 0),
                nrow = 5, byrow = TRUE)

urgency <- c(7, 5, 2, 10, 5)                    # Penalty for relocating
                                                # or losing patient of type i

### Run simulation -------------------------------------------------------------

# Run simulation
sim <- ward_sim(n, burn_in, capacity, lambda, mu, P)
patients <- sim$patients
admitted <- sim$admitted
relocated <- sim$relocated
lost <- sim$lost

# Find probability of all beds occupied upon arrival
occupied <- 1 - (admitted / patients)

# Calculate penalty score
score <- sum(urgency * (relocated + lost))

```

```r

### Results -------------------------------------------------------------------

# Format results in a table
temp <- matrix(c(occupied, patients, admitted, relocated, lost),
               ncol = 5, byrow = TRUE,
               dimnames = list(Attribute = c("P(occupied bed upon arrival)",
                                             "N: patients",
                                             "N: admitted",
                                             "N: relocated",
                                             "N: lost"),
                               Ward = wards))
result_table <- as.table(temp)

# Print results
options(scipen = 999)
round(result_table, 3)
options(scipen = 0)

#Print score
score



## Find distribution of score ##################################################

### Run simulations ------------------------------------------------------------

# Initialize vector to store the scores
scores <- numeric(k)

# Run the simulations
for (i in 1:k){

    sim <- ward_sim(n, burn_in, capacity, lambda, mu, P)

    # Calculate score
    scores[i] <- sum(urgency * (sim$relocated + sim$lost))
}

### Show results ---------------------------------------------------------------

# Hisogram of scores
hist(scores, breaks = 15)

# Confidence intervals
mu <- mean(scores)
```

```r
S2 <- var(scores)
t <- qt(0.975, df = k - 1)
mu
S2
c(mu - sqrt(S2 / k) * t, mu + sqrt(S2 / k) * t)



# _ ############################################################################
# Simulation - with F ##########################################################

## Test simulation - 1 run #####################################################

### Parameters -----------------------------------------------------------------
k <- 10^2                                    # Number of simulations to run
n <- 10^4                                    # Number of simulated patients
burn_in <- 10^3                              # Burn in period
wards <- c("A", "B", "C", "D", "E", "F")     # Name of wards
# capacity <- c(28, 28, 28, 27, 27, 27)        # Ward bed capacity
lambda <- c(14.5, 11.0, 8.0, 6.5, 5.0, 13.0) # Ward arrivals per day
mu <- c(1/2.9, 1/4.0, 1/4.5, 1/1.4, 1/3.9, 1/2.2) # Ward mean length of stay

P <- matrix(c(0, 0.05, 0.10, 0.05, 0.80, 0,  # Probability patient i
             0.20, 0, 0.50, 0.15, 0.15, 0,   # relocates to ward j
             0.30, 0.20, 0, 0.20, 0.30, 0,
             0.35, 0.30, 0.05, 0, 0.30, 0,
             0.20, 0.10, 0.60, 0.10, 0, 0,
             0.20, 0.20, 0.20, 0.20, 0.20, 0),
           nrow = 6, byrow = TRUE)

urgency <- c(7, 5, 2, 10, 5, 10)             # Penalty for relocating
                                             # or losing patient of type i

### Run simulation -------------------------------------------------------------

# Run simulation
sim <- ward_sim(n, burn_in, capacity, lambda, mu, P)
patients <- sim$patients
admitted <- sim$admitted
relocated <- sim$relocated
lost <- sim$lost

# Find probability of all beds occupied upon arrival
occupied <- 1 - (admitted / patients)

# Calculate penalty score
score <- sum(urgency * (relocated + lost))
```

```r

### Results ---------------------------------------------------------------------

# Format results in a table
temp <- matrix(c(occupied, patients, admitted, relocated, lost),
               ncol = 6, byrow = TRUE,
               dimnames = list(Attribute = c("P(occupied bed upon arrival)",
                                             "N: patients",
                                             "N: admitted",
                                             "N: relocated",
                                             "N: lost"),
                              Ward = wards))
result_table <- as.table(temp)

# Print results
options(scipen = 999)
round(result_table, 3)
options(scipen = 0)

#Print score
score

# _ ###############################################################################
# Test of four optimal solutions ################################################

## Parameters -------------------------------------------------------------------
k <- 10^2
n <- 10^4
burn_in <- 10^3
wards <- c("A", "B", "C", "D", "E", "F")
lambda <- c(14.5, 11.0, 8.0, 6.5, 5.0, 13.0)
mu <- c(1/2.9, 1/4.0, 1/4.5, 1/1.4, 1/3.9, 1/2.2)

P <- matrix(c(0, 0.05, 0.10, 0.05, 0.80, 0,
              0.20, 0, 0.50, 0.15, 0.15, 0,
              0.30, 0.20, 0, 0.20, 0.30, 0,
              0.35, 0.30, 0.05, 0, 0.30, 0,
              0.20, 0.10, 0.60, 0.10, 0, 0,
              0.20, 0.20, 0.20, 0.20, 0.20, 0),
           nrow = 6, byrow = TRUE)

urgency <- c(7, 5, 2, 10, 5, 10)

# capacity <- c(59, 44, 1, 24, 1, 36)
# capacity <- c(59, 44, 1, 25, 1, 35)
# capacity <- c(37, 25, 22, 27, 19, 35)
capacity <- c(31, 28, 26, 23, 24, 33)
```

```r


## Run simulations ---------------------------------------------------------------
p_occupied <- matrix(0, k, 6)
scores <- numeric(k)

for (i in 1:k){

    # Simulation
    sim <- ward_sim(n, burn_in, capacity, lambda, mu, P)

    # P(relocated)
    temp <- (sim$relocated + sim$lost) / sim$patients
    p_occupied[i,] <- temp

    # Score
    scores[i] <- sum(urgency * (sim$relocated + sim$lost))
}

## Show results ------------------------------------------------------------------
hist(scores, breaks = 10)

# Scores
m <- mean(scores)
S2 <- var(scores)
t <- qt(0.975, df = k-1)
CI <- c(m - sqrt(S2 / k) * t, m + sqrt(S2 / k) * t)
capacity
print(c(m, S2, CI))


# P(relocated)
for (i in 1:6){
    mu <- mean(p_occupied[,i])
    S2 <- var(p_occupied[,i])
    t <- qt(0.975, df = k-1)
    CI <- c(mu - sqrt(S2 / k) * t, mu + sqrt(S2 / k) * t)
    print(round(c(i, mu, S2, CI),4))
}


# _ ##############################################################################

# Sensitivity ####################################################################

## Change in max capacity ########################################################

```

```r
### Parameters ----------------------------------------------------------------
k <- 10^2
n <- 10^4
burn_in <- 10^3
wards <- c("A", "B", "C", "D", "E")
lambda <- c(14.5, 11.0, 8.0, 6.5, 5.0)
mu <- c(1/2.9, 1/4.0, 1/4.5, 1/1.4, 1/3.9)
P <- matrix(c(0, 0.05, 0.10, 0.05, 0.80,
              0.20, 0, 0.50, 0.15, 0.15,
              0.30, 0.20, 0, 0.20, 0.30,
              0.35, 0.30, 0.05, 0, 0.30,
              0.20, 0.10, 0.60, 0.10, 0),
            nrow = 5, byrow = TRUE)
urgency <- c(7, 5, 2, 10, 5)

max_capacity <- seq(150, 185, 5)
original_capacity <- c(55, 40, 30, 20, 20)
bed_distribution <- original_capacity / sum(original_capacity)


### Run simulations ------------------------------------------------------------

m <- length(max_capacity)
p_occupied <- array(0, dim = c(k, 5, m))
scores <- matrix(0, k, m)

for (j in 1:m){

    # Print iteration number
    print(j)

    # Divide beds among wards
    capacity <- floor(bed_distribution * max_capacity[j])
    temp <- c(rep(1, max_capacity[j] - sum(capacity)),
              rep(0, 5 - (max_capacity[j] - sum(capacity))))
    capacity <- capacity + temp

    for (i in 1:k){

        # Run simulation
        sim <- ward_sim(n, burn_in, capacity, lambda, mu, P)

        # P(relocated)
        temp <- (sim$relocated + sim$lost) / sim$patients
        p_occupied[i,,j] <- temp

        # Score
```

```
        scores[i,j] <- sum(urgency * (sim$relocated + sim$lost))
    }
}


### Results --------------------------------------------------------------------

# Find means
mean_scores <- colMeans(scores)
mean_occupied <- colMeans(p_occupied)


# bar chart
barplot(mean_occupied,
        main = "Average probability of being relocated across 100 simulations",
        xlab = "Total bed capacity", ylab = "P(occupied upon arrival)",
        names.arg = max_capacity,
        col = c("blue", "green", "orange", "purple", "red"))
legend("topright", legend = c("A", "B", "C", "D", "E"),
       fill = c("blue", "green", "orange", "purple", "red"))

# Scores
plot(max_capacity, mean_scores, type = "p", pch = 16, col = "blue",
     main = "Average score across 100 simulations",
     xlab = "Total bed capacity", ylab = "Score")

# CI for scores
for (i in 1:m){
    m <- mean(scores[,i])
    S2 <- var(scores[,i])
    t <- qt(0.975, df = k-1)
    CI <- c(m - sqrt(S2 / k) * t, m + sqrt(S2 / k) * t)
    print(round(c(max_capacity[i], m, S2, CI),0))
}


### Show results ---------------------------------------------------------------
hist(scores, breaks = 10)

# Scores
m <- mean(scores)
S2 <- var(scores)
t <- qt(0.975, df = k-1)
CI <- c(m - sqrt(S2 / k) * t, m + sqrt(S2 / k) * t)
capacity
print(c(m, S2, CI))

```

```r

# P(relocated)
for (i in 1:6){
    mu <- mean(p_occupied[,i])
    S2 <- var(p_occupied[,i])
    t <- qt(0.975, df = k-1)
    CI <- c(mu - sqrt(S2 / k) * t, mu + sqrt(S2 / k) * t)
    print(round(c(i, mu, S2, CI),4))
}
```

Simulated Annealing

```r
# Preamble #####################################################################
## File Description ############################################################
#
#    Sren  Skjernaa - s223316
#   22/06-2023
#
#   Stochastic Simulation
#   Project 2.2
#
#   Notes:
#       ...
#
###############################################################################

## Clean up ####################################################################
rm(list = ls())
if(!is.null(dev.list())) dev.off()



# _ ############################################################################
# Set up #######################################################################

## Create test problems ########################################################

### Arguments:
#       - patients = Number of patients to simulate in each queue
#       - queues   = Number of patient queues to simulate
#       - lambda   = Arrival rate of patients of different types
#       - mu       = Length of stay rate of patients of different types
#       - P        = Probabilities of patient i transitioning to ward j
###


### Return:
```

```r
#       - queue   = An array of dim(patients, 4, queue) storing the patient
#                   queues
###

patient_queue <- function(patients, queues, lambda, mu, P){

    # Initialize arrays to store patient queues
    queue <- array(0, dim = c(patients, 4, queues))

    # Create patient queues
    for (i in 1:queues){
        for (j in 1:patients){

            # Number of different type of patients
            m <- length(lambda)

            # Draw arrival time
            arrival_time <- rexp(1, rate = sum(lambda))

            # Sample type of patient that arrived
            p_patient_type <- lambda / sum(lambda)
            patient_type <- sample(1:m, 1, prob = p_patient_type)

            # Sample length of stay for patient that arrived
            length_of_stay <- rexp(1, rate = mu[patient_type])

            # Sample a relocation
            relocation_ward <- sample(1:m, 1, prob = P[patient_type,])

            # Store patient in queue
            queue[j, ,i] <- c(arrival_time, patient_type,
                              length_of_stay, relocation_ward)
        }
    }

    # Return
    return(queue)
}


## Initial solution ############################################################

### Arguments:
#       - wards            = Number of different type of wards
#       - capacity         = Total number of beds across all wards
#       - initial_strategy = Either "uniform" or "random" initial distribution
###
```

```r


### Return:
#       - bed_distribution = Initial distribution of beds across wards
###

initial_distribution <- function(wards, capacity, initial_strategy){

    if (initial_strategy == "random"){

        # Generate random bed distribution across the wards
        bed_distribution <- runif(wards)
        bed_distribution <- bed_distribution / sum(bed_distribution)

        # Scale the distribution so that it matches the max capacity
        bed_distribution <- floor(bed_distribution * capacity)

        # Add spare beds to ward with fewest beds
        temp1 <- which.min(bed_distribution)
        temp2 <- setdiff(1:wards, temp1)
        bed_distribution[temp1] <- capacity - sum(bed_distribution[temp2])

    } else if (initial_strategy == "uniform"){

        # Generate uniform bed distribution accros the wards
        bed_distribution <- rep(floor(capacity / wards), wards)
        temp <- rep(1, capacity - sum(bed_distribution))
        temp <- c(temp, rep(0, wards - length(temp)))
        bed_distribution <- bed_distribution + temp
    }

    # Return
    return(bed_distribution)
}


# _ ##########################################################################
# Hospital simulation ########################################################

### Arguments:
#       - wards            = Number of wards and different types of patients
#       - queue            = Simulated patient queues
#       - bed_distribution = Current distribution of beds
#       - burn_in          = Burn in period of simulation
###

### Return:
```

```r
#       - patients = Number of arrived patients of type i
#       - admitted = Number of admitted patients of type i to ward i
#       - relocated = Number of relocated patients of type i
#       - lost     = Number of lost patients of type i
###

hospital_sim <- function(wards, queue, bed_distribution, burn_in){

    # number of simulated queues and patients
    m <- dim(queue)[3]
    n <- dim(queue)[1]

    # initialize vectors for storage of results
    patients <- matrix(0, m, wards)
    admitted <- matrix(0, m, wards)
    relocated <- matrix(0, m, wards)
    lost <- matrix(0, m, wards)

    # Loop over all simulated queues
    for (j in 1:m){

        # Initialize time until bed is free counter
        max_capacity <- max(bed_distribution)
        temp <- c()
        for (i in 1:wards){
            temp <- c(temp,
                      rep(0, bed_distribution[i]),
                      rep(NA, max_capacity - bed_distribution[i]))
        }
        beds <- matrix(temp, nrow = wards, ncol = max_capacity, byrow = TRUE)

        # Main loop of simulation - simulate journey of patient i
        for (i in 1:n){

            # Retrieve patients queue data
            arrival_time <- queue[i, 1, j]
            patient_type <- queue[i, 2, j]
            length_of_stay <- queue[i, 3, j]
            relocation_ward <- queue[i, 4, j]


            # Reduce the time until bed is free
            beds <- beds - arrival_time

            # record patient
            if (i > burn_in){

```

```
                patients[j, patient_type] <- patients[j, patient_type] + 1
            }

            # Assign patient to ward of same type as patient
            assigned_ward <- patient_type

            # Check originally intended ward for free beds
            bed_index <- which.min(beds[assigned_ward,])
            bed_time <- beds[assigned_ward, bed_index]


            if (bed_time <= 0){      # Assign patient to intended ward

                beds[assigned_ward, bed_index] <- length_of_stay

                # record admission
                if (i > burn_in){

                    admitted[j, patient_type] <- admitted[j, patient_type] + 1
                }

            } else{                  # Try to relocate patient

                # Assign patient to relocated ward
                assigned_ward <- relocation_ward

                # Check relocated ward for free beds
                bed_index <- which.min(beds[assigned_ward,])
                bed_time <- beds[assigned_ward, bed_index]


                if (bed_time <= 0){  # Assign patient to relocation ward

                    beds[assigned_ward, bed_index] <- length_of_stay

                    # record relocation
                    if (i > burn_in){

                        relocated[j, patient_type] <-
                            relocated[j, patient_type] + 1
                    }

                } else{              # Turn patient away

                    # record lost
                    if (i > burn_in){

```

```r
                        lost[j, patient_type] <- lost[j, patient_type] + 1
                    }
                }
            }
        }
    }

    # Return results
    result <- list("patients" = patients,
                   "admitted" = admitted,
                   "relocated" = relocated,
                   "lost" = lost)
    return(result)
}


# _ ###########################################################################
# Evaluate solution ###########################################################

## Cost of solution ###########################################################

### Arguments:
#       - urgency  = Penalty for relocating patients
#       - relocated = Number of patients that are relocated (and/or lost)
#       - patients = Number of patients of each type that arrives
#       - penalty  = Penalty for relocating patients of type F (more than 5%)
###


### Return:
#       - score = Average score across all simulated queues
###


cost <- function(urgency, relocated, patients, penalty){

    # Number of simulated queues
    m <- dim(relocated)[1]

    # Initialize vector to store the scores
    scores <- numeric(m)

    # Loop over all simulated queues
    for (i in 1:m){

        # Calculate penalty for relocating patients of type F
        if (penalty == 0){   # Penalty type 1
```

```
            temp <- 0

        } else{              # Penalty type 2

            temp <- (relocated[i, 6] / patients[i, 6]) * 1000
            temp <- floor(max(0, temp - 50)) * penalty
        }

        # Calculate total score and save it
        score <- sum(relocated[i,] * urgency) + temp
        scores[i] <- score
    }

    # Return average score
    result <- mean(scores)
    return(result)
}


## Plot solution ################################################################

### Arguments:
#       - current_distribution = Current bed distribution
#       - current_cost       = Current solution cost
#       - index              = Index of current solution
###

plot_solution <- function(current_distribution, current_cost, index){

    # Number of wards and total capacity
    m <- length(current_distribution)
    capacity <- sum(current_distribution)

    # Labels for plotting
    main_label <- paste("Capacity: ", capacity, " - Cost: ", current_cost)
    sub_label <- paste("Index:", index)

    # Plot solution
    plot(1:m, current_distribution,
         main = main_label, sub = sub_label, xlab = "Ward",
         ylab = "Number of beds in wards",
         type = "l", lwd = 2, col = "blue")
}


# _ ############################################################################
```

```
# Annealing strategy ############################################################

## Temperature cooling ##########################################################

temperature <- function(delta, index, T0){

    # Calculate temperature (decreasing function of index)
    # T = -log(index + 1)
    # T <- 1 / sqrt(1 + index)
    # T <- T0 * 0.90^index
    T <- T0 / index

    # Probability to accept a worse solution
    p <- exp(- delta / T)
    # p <- 0.25

    return(p)
}

### Plots of cooling ############################################################
T0 <- 10^4
x <- seq(1, 10^3, 10)


plot(x, temperature(10, x, T0),
     main = "Cooling of the probability of accepting worse solution",
     sub = "delta = 10", xlab = "Index", ylab = "P(Accept worse solution)")

plot(x, temperature(50, x, T0),
     main = "Cooling of the probability of accepting worse solution",
     sub = "delta = 50", xlab = "Index", ylab = "P(Accept worse solution)")

plot(x, temperature(100, x, T0),
     main = "Cooling of the probability of accepting worse solution",
     sub = "delta = 100", xlab = "Index", ylab = "P(Accept worse solution)")

plot(x, temperature(500, x, T0),
     main = "Cooling of the probability of accepting worse solution",
     sub = "delta = 500", xlab = "Index", ylab = "P(Accept worse solution)")

plot(x, temperature(1000, x, T0),
     main = "Cooling of the probability of accepting worse solution",
     sub = "delta = 1000", xlab = "Index", ylab = "P(Accept worse solution)")



## Change solution ##############################################################
```

```r

### Arguments:
#       - bed_distribution = Current distribution of beds
###


### Return:
#       - proposal_distribution = New proposed distribution of beds
###

change_solution <- function(bed_distribution){

    # Save old bed distribution
    old_distrbution <- bed_distribution

    # Number of wards
    m <- length(bed_distribution)

    # Sample two random wards to take n beds from ward i and add to ward j
    n <- sample(1:5, size = 1, prob = c(0.6, 0.2, 0.1, 0.06, 0.04))
    swap <- sample(1:m, size = 2, replace = FALSE)
    bed_distribution[swap[1]] <- bed_distribution[swap[1]] - n
    bed_distribution[swap[2]] <- bed_distribution[swap[2]] + n

    # Return new bed distribution
    if (any(bed_distribution <= 0)){

        return(old_distrbution)

    } else{

        return(bed_distribution)
    }
}


# _ ############################################################################
# Simulated Annealing ##########################################################

## Main simulated annealing loop ###############################################

### Arguments:
#       - steps          = Steps to run the simulated annealing for
#       - patients       = Number of patients to simulate in each queue
#       - burn_in        = Burn in period to use
#       - queues         = Number of hospital queues to simulate
#       - capacity       = Maximum capacity across all wards
```

```
#       - lambda           = Arrival rate of patients of different types
#       - mu               = Length of stay rate of patients of different types
#       - P                = Probabilities of patient i transitioning to ward j
#       - urgency          = Penalty for relocating (or losing) patient i
#       - initial_strategy = Uniform or random initial distribution of beds
#       - T0               = Initial temperature
#       - penalty          = Penalty for relocating patients of type F
#       - plot_costs       = Logical variable controlling if development in
#                            plots should be shown during optimization.
###


### Return:
#       - solutions = Distribution of beds at each index
#       - costs     = Costs of solution at each index
#       - delta     = Improvement in solution for each proposal
#       - patients = Distribution of patient types in simulation
#       - admitted = Number of admitted patients
#       - relocated = Number of relocated patients
#       - lost      = Number of lost patients
###

sim_annealing <- function(steps,
                          patients, burn_in, queues,
                          capacity, lambda, mu, P, urgency,
                          initial_strategy, T0, penalty,
                          plot_costs){

    # Parameters
    total_patients <- patients + burn_in
    wards <- length(lambda)

    # Store costs and solution at each index
    solutions <- matrix(0, steps, wards)
    costs <- numeric(steps)
    deltas <- numeric(steps)

    # Create test problem (patient queues)
    queue <- patient_queue(total_patients, queues, lambda, mu, P)

    # Create initial solution (initial bed distribution)
    bed_distribution <- initial_distribution(wards, capacity, initial_strategy)

    # Find initial cost
    sim <- hospital_sim(wards, queue, bed_distribution, burn_in)
    total_relocations <- sim$relocated + sim$lost
    current_cost <- cost(urgency, total_relocations, sim$patients, penalty)
```

```
    # Simulated annealing loop
    for (i in 1:steps){

        # Loop counter (print to console)
        print(i)

        # Propose new solution by swapping adding bed from one ward to another
        proposal_distribution <- change_solution(bed_distribution)

        # Calculate cost of proposed bed_distribution
            # queue <- patient_queue(total_patients, queues, lambda, mu, P)
        sim <- hospital_sim(wards, queue, proposal_distribution, burn_in)
        total_relocations <- sim$relocated + sim$lost
        proposal_cost <- cost(urgency, total_relocations, sim$patients, penalty)


        # Calculate change to cost
        delta <- proposal_cost - current_cost


        # Accept proposal if drop in cost: delta <= 0
        if (delta <= 0){

            bed_distribution <- proposal_distribution
            current_cost <- proposal_cost

            # Accept proposal with probability according to cooling scheme
        } else if (runif(1) <= temperature(delta, i, T0)) {

            bed_distribution <- proposal_distribution
            current_cost <- proposal_cost
        }

        # Store current_cost
        costs[i] <- current_cost
        solutions[i,] <- bed_distribution
        deltas[i] <- delta

        # Plot development in cost during optimization
        if (plot_costs){
            plot(1:i, costs[1:i],
                 xlim = c(1, steps), ylim = c(5000, 20000),
                 type = "l", col = "black", lwd = 2,
                 main = "Change in cost", xlab = "Index", ylab = "Average cost")
        }
    }
```

```r

    # Run simulation a final time with the found optimum
    sim <- hospital_sim(wards, queue, bed_distribution, burn_in)

    # Return costs and solutions
    result <- list("solutions" = solutions, "costs" = costs, "delta" = deltas,
                   "patients" = sim$patients, "admitted" = sim$admitted,
                   "relocated" = sim$relocated, "lost" = sim$lost)
    return(result)
}


# _ ###############################################################################
# Test Simulated Annealing #####################################################

## Parameters ------------------------------------------------------------------
steps <- 800
patients <- 10^4
burn_in <- 10^3
queues <- 100

capacity <- 165
lambda <- c(14.5, 11.0, 8.0, 6.5, 5.0, 13.0)
mu <- c(1/2.9, 1/4.0, 1/4.5, 1/1.4, 1/3.9, 1/2.2)
P <- matrix(c(0, 0.05, 0.10, 0.05, 0.80, 0,
             0.20, 0, 0.50, 0.15, 0.15, 0,
             0.30, 0.20, 0, 0.20, 0.30, 0,
             0.35, 0.30, 0.05, 0, 0.30, 0,
             0.20, 0.10, 0.60, 0.10, 0, 0,
             0.20, 0.20, 0.20, 0.20, 0.20, 0),
           nrow = 6, byrow = TRUE)
urgency <- c(7, 5, 2, 10, 5, 10)
penalty <- 0

initial_strategy <- "uniform"
T0 <- 10000
plot_costs <- TRUE

## Run simulated annealing ----------------------------------------------------
sim <- sim_annealing(steps,
                     patients, burn_in, queues,
                     capacity, lambda, mu, P, urgency,
                     initial_strategy, T0, penalty,
                     plot_costs)

# Summary of results
sim
```

```
round((sim$relocated + sim$lost) / sim$patients, 3)
sim$solutions[steps,]
sim$costs[steps]
(sim$relocated + sim$lost) * urgency
sum((sim$relocated + sim$lost) * urgency)

test <- (sim$relocated + sim$lost) / sim$patients
mean(test[,1])
mean(test[,2])
mean(test[,3])
mean(test[,4])
mean(test[,5])
mean(test[,6])
```