Machine minimization problem for vaccine scheduling

Cristian Grosu 0721808
Utrecht University Faculty of Natural Sciences, Netherlands

Tim Brouwer 6153798
Utrecht University Faculty of Natural Sciences, Netherlands

Zhadyra Khattar 0033324
Utrecht University Faculty of Natural Sciences, Netherlands

Marwan Ait Addi 8260125
Utrecht University Faculty of Natural Sciences, Netherlands

Allison Lo 1465015
Utrecht University Faculty of Natural Sciences, Netherlands

Abstract

In this paper, the problem of machine minimization was analyzed through the perspective of vaccine scheduling. To be precise, two types of problems were tackled. The first one is the offline problem where the input is completely known by the algorithm before scheduling. The second one is the online problem where the program will consequently process its input piece-by-piece. For the offline problem, two linear programming models were designed and their time complexity with their representation were compared. Linear programming techniques were used to solve the offline problem. For the online problem, a general lower bound for any deterministic online algorithm was proposed. Moreover, two online algorithms were discussed and analyzed in terms of lower and upper bound for competitive ratio.

2012 ACM Subject Classification Theory of computation → Scheduling algorithms

Keywords and phrases Machine minimization, optimization problem, vaccine scheduling, online algorithms, linear programming, competitive ratio

Digital Object Identifier 10.4230/LIPIcs.VS.2021.1

1 Introduction

In computer science, mathematics and economics, an optimization problem is the problem of finding the best solution from all feasible solutions. Optimization problems can be divided into two categories, depending on whether the variables are continuous or discrete:

- An optimization problem with discrete variables is known as a discrete optimization, in which an object such as an integer, permutation or graph must be found from a countable set.
- A problem with continuous variables is known as a continuous optimization, in which an optimal value from a continuous function must be found. They can include constrained problems and multi-modal problems.

Discrete optimization problem for vaccine scheduling is tackled in this paper.

1.1 Machine minimization problem definition

In this subsection, the machine minimization problem for vaccine scheduling is described. We consider the problem where the vaccination is with two-phase vaccine jabs. Each vaccine jab has two doses, and a certain time gap is required between the two doses. Each dose must be given by a hospital/machine. After the time slot a patient having a dose, the patient

Vaccine Scheduling Assignment 2021.
Editors: Cristian Grosu, Tim Brouwer, Zhadyra, Marwan, Allison; Article No. 1; pp. 1:1–1:18
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

may be required to stay further for examinations for a number of additional time slots, directly after receiving the dose. The time slot of having the dose together with the ones of staying for examinations form a contiguous time interval. The length of the time interval is the processing time of this dose. Each hospital/machine can only take care (either give a dose or examine) of at most one patient per time slot. A patient can submit an interval of time slots that he/she is willing to have the first dose. The goal is to assign the time slots of the two doses for each patient (where both the preferences of the patients and the time gap between the two doses for every jabs must be satisfied) with the minimum number of hospitals/machines to operate the vaccine tasks. An instance to our problem has the following values:

- Global parameters, which are the same for all patients
  - the processing time of the first dose: $p_1 \geq 1$
  - the processing time of the second dose: $p_2 \geq 1$
  - and the time gap between the first and the second doses: g.
- A set of jobs $J = \{J_1, J_2, \ldots, J_n\}$. Each job represents one patient. Each job has the following information:
  - For job $J_i \in J$:
    - The first feasible interval $I_{i,1} = [r_{i,1}, d_{i,1}]$ for the first dose (given by the patient)
    - The first dose is scheduled at start time $t_i \in I_{i,1}$ such that $t_{i,1} + p_1 - 1 \leq d_{i,1}$.
    - The patient-dependent delay x_i, where $x_i \geq 0$
    - The patient-dependent (second) feasible interval length l_i where $l_i \geq p_2$
    - The second feasible interval $I_{i,2} = [t_{i,1} + p_1 + g + x_i, t_i, 1 + p_1 + g + x_i + l_i - 1]$ for the second dose. Note that this interval depends on the start time for the first dose as was determined by the program, and the given values g, x_i, and l_i.)
    - The second dose is scheduled at start time $[t_{i,2}, t_{i,2} + p_2 - 1] \in I_{i,2}$. (This value also has to be determined by the program.)
- Machine (hospital): at any time step, there can be at most one job executing. i.e., at each time step, each hospital can have at most 1 patient who received a dose or is in observation.
- Feasible schedule: For any job J_i, the first dose is scheduled at time interval $[t_{i,1}, t_{i,1} + p_1 - 1] \subseteq I_{i,1}$, and the second dose is scheduled at time $[t_{i,2}, t_{i,2} + p_2 - 1] \subseteq I_{i,2}$.

The objective of this problem is to minimize the number of machines (i.e., hospitals). Two versions of this particular problem are discussed in this paper:

- Offline problem, where the whole input is known at the beginning;
- Online problem, where the input to the problem is revealed partially during the running time of the algorithm that solves it, namely the algorithm must decide on each patient separately. Moreover, algorithm decisions are irrevocable.

In the offline version of the problem, the global parameters p_1, p_2, g , and the set of jobs with for each job the first feasible interval, second feasible interval, patient-depending delay, and patient-depending second interval length are given. The task is to find for each patient the time slot for the first and for the second dose such that all constraints are fulfilled, and the number of hospitals is as small as possible.

The online problem is described as follows. At the start, the global parameters p_1, p_2, g are given. Then, we have for each patient one round. At round i, we obtain all information for the i^{th} patient: these are four integers, where the first two give the interval for the first dose, the third the delay for this patient, and the fourth the length of the second feasible interval. The online program then has to schedule this patient: give the time and hospital when and where the first dose is given, the time and hospital when and where the second dose is given. These should fulfill the conditions as explained earlier.After this, we start the next round with the next patient.

1.2 Related works

1.2.0.1 Related works of offline problem

Our problem is a special case of a more general problem, namely machine minimization problem with jobs interval constraints. This kind of problem is widely used in real life application such as personnel scheduling for work, and telecommunication. These problem can be classified in two categories, continuous and discrete. In 2002 J. Chuzhoy et al. [1] proved an $O(\sqrt{\frac{\log n}{\log\log n}})$ approximation via a relaxation of the problem in an ILP. However, a lot of people are still focused on finding the best online algorithm as it is still an open question and an efficient online algorithm would have more practical application than the offline version, since it is more closely to real life situations when inputs are revealed partially.

1.2.0.2 Relative works of online problem

In the online version of vaccine scheduling problem each job has a release time, a processing time, and must be completed by its deadline. Since the jobs are revealed online over time, the algorithm must decide whether to activate/open a new machine depending only on the information of the jobs that have been released so far, without any knowledge of the future jobs. The aim is to design online algorithm that minimizes the number of activated machines to ensure that all the jobs are completed by their deadlines.
The machine minimization problem studies can be classified to two categories if we discuss whether the job preemption is allowed. When preemption is allowed, any job in process may be paused and resumed later, possibly on a different machine.
The preemptive version of online machine minimization problem has been investigated extensively. Phillips et al.(2002) [2] showed that there is an $O(\log \frac{p_{max}}{p_{min}})$ -competitive online algorithm, where p_{max} and p_{min} are the maximum and minimum processing times of jobs. However, the competitive ratio is far from optimal compared with a lower bound of 5/4 they provided. Nearly two decades later, Chen et al.(2018) [3] improved the competitive ratio to $O(\log m)$, where m is the optimal number of machines in the offline setting (all the jobs are known in advance). Based on this novel work, the competitive ratio was further improved to $O(\frac{\log m}{\log\log m})$ Azar and Cohen (2018) [4] and to $O(\log\log m)$ Im et al.(2017) [5].
The non-preemptive version of online machine minimization problem is more challenging. Saha (2013) [6] showed that no algorithm can achieve a competitive ratio better than Ω(log pmax /pmin), which is unbounded if $\frac{p_{max}}{p_{min}}$ is unbounded(p_{max} and p_{min} are the maximum and minimum processing times of jobs). Thus, more research focused on special cases of the non-preemptive problem, for example, uniform processing times and with a common deadline. Kao et al.(2012) [7] provided a 5.2-competitive algorithm and a lower bound of 2.09 for the online machine minimization problem with uniform job processing times. Devanur et al.(2014) [8] improved the competitive ratio to e (natural number, mathematical constant approximately equal to 2.71828) and showed that no deterministic algorithm for this problem has a competitive ratio less than e. Devanur et al.(2014) [8] also gave a 16-competitive algorithm for the special case of jobs with equal deadlines. We studied these related works discussed by Chen et al.(2020) [9] in their paper, although this paper is not very related to our problem, the related paper above did provide us some idea of solving online problem.

1.3 Implementation details

In this subsection we give some implementation details, as the solver we used for the offline problem, programming language we used, and characteristics of the machines we used for the experiments. This section also presents the main concept of greedy algorithm that we will use later on.

1.3.0.1 Optimization Modeling tool - Gurobi/gurobipy

The offline problem can be formulated as an integer linear programming (ILP) problem as we will see in the next sections.We solve the ILP model using a well known Optimization Modeling tool - Gurobi/gurobipy.

Gurobi comes with a Python extension module called “gurobipy” that offers convenient object-oriented modeling constructs and an API to all Gurobi features. The Gurobi distribution also includes a Python interpreter and a basic set of Python modules that are sufficient to build and run simple optimization models.

Gurobipy is written in C++ and is an advanced solver for LP and ILP models. Several advanced techniques as branch and bound along with cutting off are used for the optimization of ILP models. Other solvers like CPLEX or GLPK was also considered.

The Gurobi Optimizer is capable of solving all major problem types (convex and non-convex):

- Linear programming (LP)
- Mixed-integer linear programming (MILP)
- Quadratic programming (QP)
- Mixed-integer quadratic programming (MIQP)
- Quadratically-constrained programming (QCP)
- Mixed-integer quadratically

The main steps of using this tool are listed below:

- Creating the model
- Adding variables to the model
- Setting the objective
- Adding constraints to the model
- Optimizing the model
- Reporting results - attributes

1.3.0.2 Greedy algorithm

Greedy algorithm is a simple, intuitive algorithm that is used as a heuristic in optimization problems. The algorithm makes the optimal choice at each step as it attempts to find the overall optimal way to solve the entire problem. However, the algorithm does not always achieve the optimal value of the problem. In this paper, we will apply this concept to our algorithm for the online problem, and will later discuss more about the performance through the lower and upper bounds of competitive ratio.

1.3.0.3 Programming Environments

Python was used as the programming language for our algorithms. We run the online problem on google colab, with python version 3.6.9. and with max 12 GB RAM. For the offline problem , we run it on local laptop, the version of gurobipy is gurobipy-9.1.2 with academic licence. The RAM size of the computer is 8GB with OS Windows10.

2 Offline problem

In this section, we firstly prove that the offline version of our problem can be formulated as an integer linear programming problem and give two mathematical models for it.

Since the input is known, a first approach is to use a heuristic such as Greedy algorithm, however, this approach will not bring the optimal result in all the cases. Another approach is to tackle the problem as a linear programming one. In this approach, the problem is designed in terms of minimizing or maximizing an objective function subject to several constraints of the problem. In our case we deal with a minimization problem, namely minimizing the number of hospitals/machines. Due to the aforementioned facts and because patients have preferences for vaccine jabs treating the problem as an ILP is naturally.

2.1 ILP models

In this subsection we will introduce two ILP models. In both models, we considered n as the number of jobs that need to be scheduled. In order to avoid the confusion the name of the variable x_j, that represents the patient j delay, was changed to α_j.

2.1.0.1 First model

In the first model we will use a set of variables, namely:

- $x_j \in \mathbb{N}$ the starting time of the first jab of job j, $\forall j \in 1, ..., n$
- $x_j \in \mathbb{N}$ the starting time of the second jab of job j, $\forall j \in n, ..., 2n$
- $z_{j,i} = \begin{cases} 1, if\ a\ jab\ j\ is\ assigned\ on\ machine\ i \\ 0, otherwise \end{cases} \quad \forall i \in \{1, \dots, n\}, \forall j \in \{1, \dots, 2n\}$
- $t_i = \begin{cases} 1, if\ machine\ i\ is\ used \\ 0, otherwise \end{cases} \quad \forall i \in \{1, \dots, n\}$
- $u_{j,j'} = \begin{cases} 1, if\ jab\ j\ is\ overlapping\ with\ jab\ j' \\ 0, otherwise \end{cases} \quad \forall j, j' \in \{1, \dots, 2n\}$

Note that we are dealing with $2n + n^2 + n + n^2 = 2n^2 + 3n$ variables. The model itself is described below by equations (1) - (17).

Machine minimization linear programming model

$$\text{Objective function: } \min \sum_{i=1}^{n} t_i \tag{1}$$

Subject to: (2)

$$x_j \geq r_j \quad \forall j \in \{1, ..., n\}$$

lower bound for starting time of the first jab of job j (3)

$$x_j \leq d_j - p_1 + 1 \quad \forall j \in \{1, ..., n\}$$

upper bound for starting time of the first jab of job j (4)

$$x_{j+n} \geq x_j + p_1 + \alpha_j + g \quad \forall j \in \{1, ..., n\}$$

lower bound for starting time of the second jab of job j (5)

$$x_{j+n} \leq x_j + p_1 + \alpha_j + g + l_j \quad \forall j \in \{1, ..., n\}$$

upper bound for starting time of the second jab of job j (6)

$$t_i \geq z_{j,i} \quad \forall j \in \{1, ..., 2n\} \quad \forall i \in \{1, ..., n\} \quad \text{machine i is used} \tag{7}$$

$$\sum_{i=1}^{n} z_{j,i} = 1 \quad \forall j \in \{1, ..., 2n\}$$

a jab is assigned exactly to one machine (8)

$$z_{j,i} + z_{j',i} + u_{j,j'} + u_{j',j} \leq 3 \quad \forall j, j' \in \{1, ..., 2n\} \quad j \neq j' \quad \forall i \in \{1, ..., n\}$$

if jabs overlap they can not be assigned on the same machine (9)

$$x_j - x_{j'} - L\ u_{j,j'} \leq -1 \quad \forall j, j' \in \{1, ..., 2n\} \quad j \neq j'$$

if jabs j and j' overlap (10)

$$x_j - x_{j'} - L\ u_{j,j'} \leq -(\ c_j\ u_{j',j} + c_{j'}\ u_{j,j'}) \quad \forall j, j' \in \{1, ..., 2n\} \quad j \neq j'$$

if jabs j and j' overlap (11)

$$x_j \geq 0 \quad \forall j \in \{1, ..., 2n\} \tag{12}$$

$$x_j \in \mathbb{N} \quad \forall j \in \{1, ..., 2n\} \tag{13}$$

$$t_i \in \{0, 1\} \quad \forall i \in \{1, ..., n\} \tag{14}$$

$$z_{j,i} \in \{0, 1\} \ \forall j \in \{1, ..., 2n\} \ \forall i \in \{1, ..., n\} \tag{15}$$

$$u_{j,j'} \in \{0, 1\} \quad \forall j, j' \in \{1, ..., 2n\} \tag{16}$$

(17)

Where $c_j = \begin{cases} p_1, if\ j \leq n \\ p_2, if\ j > n\ and\ j \leq 2n \end{cases}$

The number of constraints for this model are $5n + 2n^2 + 2n + 4 \cdot 4 \cdot n^2 + 4n + n + 2n^2 + 4n^2 = 24n^2 + 12n$

2.1.0.2 Second model

For this model we first define a constant T which represent the last time slot possible in that instance.

$$T = p_1 + g + \max_{j \in \{1, ..., n\}} (d_j + \alpha_j + l_j) \tag{18}$$

Variables used in this model are defined as follows:

- $x_{j,i,t} = \begin{cases} 1, & if\ job\ j\ is\ assigned\ on\ machine\ i\ on\ day\ t \\ 0, & otherwise \end{cases}$
  $\forall j, i \in \{1, \dots, n\}, \forall t \in \{1, \dots, T\}$

- $y_{j,i,t} = \begin{cases} 1, & if\ first\ jab\ of\ job\ j\ is\ starting\ on\ machine\ i\ on\ day\ t \\ 0, & otherwise \end{cases}$
  $\forall j, i \in \{1, \dots, n\}, \forall t \in \{1, \dots, T\}$

- $u_{j,i,t} = \begin{cases} 1, & if\ second\ jab\ of\ job\ j\ is\ starting\ on\ machine\ i\ on\ day\ t \\ 0, & otherwise \end{cases}$
  $\forall j, i \in \{1, \dots, n\}, \forall t \in \{1, \dots, T\}$

- $z_i = \begin{cases} 1, if\ machine\ i\ is\ used \\ 0, otherwise \end{cases} \quad \forall i \in \{1, \dots, n\},$

Second model is described by the equations (19) - (34) and uses $3 \cdot n^2 \cdot T + n$ variables. Note that T could be bigger than n. This model use more variables than the previous one. Moreover, in the second model there are $n \cdot T + n + 2n^2 + 5n + 4n^2 \cdot T + n = 4n^2 \cdot T + 2n^2 + n \cdot T + 7n$ constraints. Due to the fact that T is a big constant we can affirm that for most of the instances the second model will have more constraints than the first. Therefore, the second model is more complex in terms of running times, we will see this as well in the experiments section of this paper.

Machine minimization ILP model

$$\text{Objective function:} \quad \min \sum_{i=1}^{n} z_i \tag{19}$$

$$\text{Subject to:} \tag{20}$$

$$\sum_{j=1}^{n} x_{j,i,t} \leq 1 \quad \forall i \in \{1,...,n\}, \forall t \in \{1,...,T\} \tag{21}$$

$$\sum_{t=1}^{T} \sum_{i=1}^{n} x_{j,i,t} \leq 1 \quad \forall j \in \{1,...,n\} \tag{22}$$

$$\sum_{t=k}^{k+p_1} x_{j,i,t} \geq p_1 \; y_{j,i,k}$$

$$\forall j, i \in \{1,...,n\} \quad \forall k \in \{r_j,...,d_j - p_1 + 1\} \tag{23}$$

$$\sum_{t=k}^{k+p_2} x_{j,i,t} \geq p_2 \; u_{j,i,k} \quad \forall j, i \in \{1,...,n\}$$

$$\forall k \in \{r_j + p_1 + \alpha_j + g, ..., d_j + \alpha_j + g + l_j - p_2 + 1\} \tag{24}$$

$$\sum_{i=1}^{n} \sum_{t=r_j}^{d_j - p_1 + 1} y_{j,i,t} = 1 \quad \forall j \in \{1,...,n\} \tag{25}$$

$$\sum_{i=1}^{n} \sum_{t=r_j+p_1+\alpha_j+g}^{r_j+p_1+\alpha_j+g+l_j-p_2+1} u_{j,i,t} = 1 \quad \forall j \in \{1,...,n\} \tag{26}$$

$$\sum_{i=1}^{n} \sum_{t=1}^{T} y_{j,i,t} + u_{j,i,t} = 2 \quad \forall j \in \{1,...,n\} \tag{27}$$

$$\sum_{i=1}^{n} \Big(\sum_{t=r_j+p_1+\alpha_j+g}^{r_j+p_1+\alpha_j+g+l_j-p_2+1} t \; u_{j,i,t} - \sum_{t=r_j}^{d_j-p_1+1} t \; y_{j,i,t} \Big) \geq p_1 + \alpha_j + g$$

$$\forall j \in \{1,...,n\} \tag{28}$$

$$\sum_{i=1}^{n} \Big(\sum_{t=r_j+p_1+\alpha_j+g}^{r_j+p_1+\alpha_j+g+l_j-p_2+1} t \; u_{j,i,t} - \sum_{t=r_j}^{d_j-p_1+1} t \; y_{j,i,t} \Big) \leq p_1 + \alpha_j + g + l_j - p_2$$

$$\forall j \in \{1,...,n\} \tag{29}$$

$$z_i \geq y_{j,i,t} \quad \forall j, i \in \{1,...,n\} \quad \forall t \in \{1,...,T\} \tag{30}$$

$$x_{j,i,t} \in \{0,1\} \quad \forall j, i \in \{1,...,n\} \quad \forall t \in \{1,...,T\} \tag{31}$$

$$y_{j,i,t} \in \{0,1\} \quad \forall j, i \in \{1,...,n\} \quad \forall t \in \{1,...,T\} \tag{32}$$

$$u_{j,i,t} \in \{0,1\} \quad \forall j, i \in \{1,...,n\} \quad \forall t \in \{1,...,T\} \tag{33}$$

$$z_i \in \{0,1\} \quad \forall i \in \{1,...,n\} \tag{34}$$

$$\tag{35}$$

3 Online problem

In this section we introduce a couple of theorems along with their proofs in order to find bounds for the competitive ratio to the online version of our problem.
Firstly, a general lower bound for the problem is introduced. An online algorithm then is described and an analysis for competitive ratio bounds is provided. Our analysis is provided in terms of number of jobs.

3.1 Competitive ratio general lower bound for the online problem

▶ **Theorem 1.** *There is no $O(1)$-competitive deterministic algorithm for the vaccine scheduling problem.*

Proof. We will first define a few useful series. Then we will create a series of adversaries Adv_c that will have an optimal solution using at most a single hospital, while the algorithm will use at least c hospitals. Finally, we will prove that the existence of this series implies that there is no $O(1)$-competitive deterministic algorithm for the vaccine scheduling problem.
For this we will first define two numbers progressions as follows:

$$x_1 = 1 \qquad x_k = x_{k-1}^2 + x_{k-1} \quad \forall k > 1 \tag{36}$$

and

$$y_1 = 1 \qquad y_k = x_{k-1}^2 = x_k - x_{k+1} \quad \forall k > 1 \tag{37}$$

We design the adversary Adv_c to generate an instance I as follows:

1. $p_1 = p_2 = 1, g = x_c + 1$
2. $k = c, n = x_c$
3. Let z_k be the starting time of any interval with length x_k where ALG has up to this point placed at least $(c-k)x_k$ appointments.
4. Generate x_{k-1}^2 jobs in with $r_{j,1} = z_k$, $d_{j,1} = z_k + x_k$, $x_i = 0$, and $l_i = 1$ $\forall j \in \{1, .., n - y_k\}$
5. $k = k - 1$
6. if $k = 0$ stop else go to 3

We will prove that for any $n \in \mathbb{N}$ the following claims are true:
Given any deterministic algorithm ALG for the vaccination problem,the instance generated by the adversary Adv_c has the following properties:
- Adv_c can find an interval with the constraints defined in the third line of the adversary.
- Adv_c will use x_c patients in total.
- There exists an offline solution to the instance I using only a single hospital.
- ALG will give a solution to this instance with at least c hospitals.

First, we will prove by downwards induction to k that Adv_c can find such an interval for every $0 < k \leq c$.
Induction basis: If $k = c$, then any interval will do.
Induction step: Assume that the theorem holds for $k = i$. We will prove that the theorem holds for $k = i - 1$. We will take a look at the interval $[z, z + x_i)$. We know that there are at least $(c - i)x_i = (x_{i-1} + 1)(c - i)x_{i-1}$ appointments in this interval as the theorem holds for $k = i$, and the adversary created $x_{i-1}^2 = (x_{i-1} - 1)(x_{i-1} + 1) + 1$ extra for a total of at least $(x_{i-1} + 1)((c - i)x_{i-1} + x_{i-1} - 1) + 1$.
Now we will take a look at the $x_{i-1} + 1$ intervals $[z, z + x_{i-1}), [z + x_{i-1}, z + 2 \cdot x_{i-1}), \cdots, [z + x_{i-1}^2, z + x_{i-1}^2 + x_{i-1})$. Note that these intervals are all disjunct and their union forms the interval $[z, z + x_i)$ by the definition of x_i. As we have just seen, there are at least $(x_{i-1} + 1)((c - i)x_{i-1} + x_{i-1} - 1) + 1 > (x_{i-1} + 1)((c - i)x_{i-1} + x_{i-1} - 1)$ appointments in this interval. Every appointment is in exactly one of the $x_{i-1} + 1$ intervals mentioned before. According to the pigeonhole principle that means that there is at least one interval with at least $(c - i)x_{i-1} + x_{i-1} = (c - (i - 1))x_{i-1}$ appointments.

Second, we will prove by induction to $q \in \mathbb{N}_{\leq c}$ that Adv_c will use x_k patients from the step where $k = q$ is defined onwards.
Induction basis: If $q = 1$ then from step $k = 1$ onwards Adv_c will only use 1 patient. Induction step: Assume the theorem holds for $q = i$. Then Adv_c uses only k_q patients from the step where $k = i$ is defined onwards. Therefore, the amount of patients Adv_c uses from the step where $k = i + 1$ is defined onwards is x_i plus the amount of patients Adv_c uses when $k = i + 1$, which is equal to $x_i + x_i^2 = x_{i+1}$.

Now we use $q = c$ in this result to get our second claim that Adv_c uses x_c patients in total.

Third, we will prove by downwards induction to k that every time Adv_c starts step 3, there exists a solution using at most one hospital that does not place any appointments in the intervals $[z_k, z_k + x_k)$ and $[z_k + x_c + 1, z_k + x_k + x_c + 1)$.
Induction basis: If $k = c$ this is trivial, as no patients are provided so the empty schedule suffices.
Induction step: Assume this holds for $k = i$. Then we will prove that if $i > 0$, this will hold for $k = i - 1$. There are exactly x_{i-1}^2 patients we have to schedule. First we will look at the first appointments. We will use the schedule for $k = i$ as a basis, and only add appointments in $[z_i, z_i + x_i)/[z_{i-1}, z_{i-1} + x_{i-1})$. This interval has size $x_i - x_{i-1} = x_{i-1}^2$, so we can schedule all first appointments there. All second appointments are scheduled exactly $x_c + 1$ timeslots later, so the same argument holds there.
To prove our claim, we use this result with $k = 1$ to get that there exists a solution with at most one hospital that does not use the intervals $[z_1, z_1 + 1)$ and $[z_1 + x_c + 1, z_1 + 1 + x_c + 1)$. The last patient has to be placed exactly there, so there indeed exists a schedule using one hospital.

The fourth claim is that ALG will give a solution using at least c hospitals. We have already proved that the interval in step 3 can be found, so when $k = 1$ we know that there exists an interval with length 1 where ALG has up to this point placed at least $c - 1$ appointments. After that, the adversary forces ALG to place an appointment exactly there, so there is a timestep with at least c appointments. As all these appointments require different hospitals, this proves that ALG uses at least c hospitals.

Assume that there exists an $O(1)$-competitive deterministic algorithm. Let d be the competetive ratio. Then there exists a α such that $ALG \leq d \cdot OPT + \alpha$ for every input. However, if we take $Adv_{\lceil d+\alpha+1 \rceil}$, then $\lceil d + \alpha + 1 \rceil \leq d + \alpha$, which is clearly untrue. Therefore, there does not exist a $O(1)$-competitive deterministic algorithm. ◀

▶ **Theorem 2.** *There is an $O(\log\log n)$-competitive ratio lower bound for any deterministic algorithm for n-vaccine scheduling problem with $n > 1$.*

Proof. Let $s_1, s_2, \cdots$ be Sylvester's sequence, defined as

$$s_1 = 2 \qquad s_k = s_{k-1}(s_{k-1} - 1) + 1 \quad \forall k > 1 \tag{38}$$

. We will prove by induction to k that $s_k = x_k + 1$.
Induction basis: For $k = 1$ we have that $x_1 = 1$, and $s_1 = 2$.
Induction step: Assume that $s_i = x_i + 1$. Then $s_{i+1} = s_{i-1}(s_{i-1} - 1) + 1 = (x_{i-1} + 1)x_{i-1} + 1 = x_{i-1}^2 + x_{i-1} + 1 = x_i + 1$.
It is a well-known fact that $s_n = \lfloor E^{2^{n+1}} + \frac{1}{2} \rfloor$ where E is approximately 1.26408, so

$$x_n = \left\lfloor E^{2^{n+1}} - \frac{1}{2} \right\rfloor$$

Now, assume that there exists a deterministic algorithm ALG with a competitive ratio not conforming to the lower bound of $\Omega(\log\log n)$. Then,

$$\forall \alpha > 0, \forall n_0 > 0, \exists n > n_0, \forall \text{input with } n \text{ patients} : \frac{ALG}{OPT} < \alpha \cdot \log\log n.$$

Therefore, it also holds for $\alpha = \frac{1}{2}$, and $n_0 = 2$, so it implies that there is an $n > 2$ for which for every input we have $\frac{ALG}{OPT} < \frac{1}{2}\log_2\log_2 n$. Let q be the highest integer for which x_q is smaller than n Therefore this also holds for the adversary which uses Adv_q as a basis and adds patients with start time $2 \cdot n + 2$, end time $3 \cdot n + 2$, gap 0, and interval for the second jab 1, There is a schedule which uses only 1 hospital for this, as these appointments can be made at any time. However, ALG uses at least q hospitals. Now, as $x_{q+1} \geq n$, we can say that

$$\left\lfloor E^{2^{q+2}} - \frac{1}{2} \right\rfloor \geq n$$

Therefore we have that

$$E^{2^{q+2}} - \frac{1}{2} \geq n$$

$$(E^4)^{2^q} \geq n$$

$$(E^4)^{2^q} \geq n$$

$$q \geq \log_{E^4}\log_2 n.$$

And, as we know that $E^4 < 4$, that means that $q \geq \log_4\log_2 n = \frac{1}{2}\log_2\log_2 n$. Therefore we have that

$$\frac{1}{2}\log_2\log_2 n > \frac{ALG}{OPT} \geq q \geq \frac{1}{2}\log_2\log_2 n.$$

Which is a contradiction. This proves that no such algorithm can exist, and therefore there is a lower bound of $O(\log\log n)$. ◀

3.2 Algorithms for online problem and their competitive ratio

3.2.0.1 Online algorithms

In this paragraph we present two algorithms that can be used for solving the online machine minimization problem for vaccine scheduling. First we introduce an algorithm we called **FirstFit**. The strategy of our algorithm is as follows:

- For the first patient open a machine and place it on this machine;
- For the next patients.
  - If a patient can be placed on the opened machines then the algorithm will place it on the first available time slots on the first machine it founds
  - If there is no machine on which the patient can be placed open a new machine and place patient on that machine.

A more detailed pseudo-code for this algorithm is presented in the appendix of this paper. A similar algorithm to **FirstFit** was also considered. We called this algorithm **LastFit** and the difference is that a patient is placed on the last available time slots on an available machine if possible.

3.2.0.2 Competitive ratio bounds for the online algorithms

In this paragraph we show the lower bound for the introduced algorithms, namely **FirstFit** and **LastFit**.

▶ **Theorem 3.** *For the **FirstFit** algorithm there is a $O(\log n)$-competitive ratio lower bound.*

Proof. An adversary *Adv* can be designed to give instances with a sufficient large interval for the second jab, in order for it to always fit on one machine, in this case we can ignore the second jab entirely from our proof. *Adv* design an instance I for n-vaccination problem as follows:

For $n \in [2^k, 2^{k+1}]$

1. Place $n - 2^k$ jobs in interval $[2^k \cdot p_1 + 1, 2^k \cdot p_1 + 1 + (n - 2^k) \cdot p_1]$
2. Place 2^{k-1} jobs in interval $r_j = 0$ and $d_j = 2^k \cdot p_1 \;\; \forall j \in 1, ..., 2^{k-1}$
3. $k = k - 1$
4. if $k = 0$ then place 1 job in interval $[0, p1]$ else go to 2

We will prove that for any $n \in \mathbb{N}$ with $n \in [2^k, 2^{k+1})$ the instance generated by the adversary *Adv* has the following properties:

- There exists an offline solution to the instance I using only a single machine,
- *ALG* will give a solution to this instance with at least $k + 1$ machines.

Let $n \in [2^k, 2^{k+1}]$, then:
$n - 2^k$ jobs are generated in interval $[2^k \cdot p_1 + 1, 2^k \cdot p_1 + 1 + (n - 2^k) \cdot p_1]$, which means all this jobs will be assigned on one machine by FirstFit algorithm. Notice that for the offline solution this jobs also will be assigned on a single machine. Next 2^{k-1} jobs can be assigned on the same machine by both FirstFit algorithm and offline solution since their intervals don't overlap with the intervals of previous $n - 2^k$ jobs. Note that for the 2^{k-1} jobs the interval is $[0, 2^k \cdot p_1]$ and the first 2^{k-1} time slots on the one machine we are using so far are assigned to our jobs by the FirstFit algorithm. Now for the next 2^{k-2} jobs the interval is $[0, 2^{k-1} \cdot p_1]$, but there are no free time slots in this interval on the machine we use so the FirstFit algorithm can only open/use another machine to place the jobs. Meanwhile, the offline solution can "move" the jobs on the first machine and free up the interval $[0, 2^{k-1} \cdot p_1]$ for the 2^{k-2} jobs that just came, namely the first 2^{k-1} jobs will be assigned one after another

in the interval $[2^{k-1} \cdot p_1 + 1, 2^k \cdot p_1]$. Therefore the offline solution can still use only one machine.

We can use the same argumentation $k-1$ times and at each step decreasing k by one. When $k = 0$ the adversary generates one more job in interval $[0, p1]$ but this interval is already busy on all the machines was used so far so we use another machine. In total our algorithm will use $1 + k - 1 + 1 = k + 1$ machines. On the other hand the offline solution will assign all the jobs to one machine. Therefore for any $n \in N$ the claims holds. The competitive ratio is $c = \frac{ALG(I)}{OPT(I)} = \frac{k+1}{1} = k + 1$.

$2^{k+1} \geq n \Rightarrow c = k + 1 \geq \log_2 n$

Therefore, the FirstFit algorithm is $O(logn)$-competitive.

◀

▶ **Theorem 4.** *The **LastFit** algorithm has $O(\log n)$-competitive ratio lower bound.*

Proof. Using a similar proof to the proof from theorem (3). But with $r_j = 2^k \cdot p_1$ and $d_j = 2^{k+1} \cdot p_1$. ◀

In order to prove an upper bound, we will first prove a related lemma.

▶ **Lemma 5.** *Let $I = [a, b)$ be a time interval where in the solution of FirstFit every single day has at least q patients. Then there exists an interval $I' = [a', b')$ with $I \subseteq I'$, I' has length at least $2 * |I|$, and in the solution of FirstFit every single day in I' has at least $q - (2 * OPT)$ patients.*

Proof. Let I be defined as before, and let $I' = [a', b')$ be the largest interval for which $I \subseteq I'$, and in the solution of FirstFit every single day in I' has at least $q - (OPT + 1)$ patients. We will prove that I' has length at least $2 * |I|$.

We will look at the execution of FirstFit. For every single patient p scheduled in I', we will look the number of patients already scheduled on that day. If that is $q - 2 * OPT$ or more, then we take a closer look at where the patient is placed in the optimal schedule. If there is a possibility to schedule the patient outside of I' in the optimal schedule, then FirstFit would have scheduled the patient outside of I' as well, as that would mean either $a' - 1$ or b' is in the interval. However, at those times FirstFit does not schedule $q - 2 * OPT$ patients, otherwise I' can be larger. Therefore, in the optimal schedule p has to be scheduled in I'.

In total, there are $(2 * OPT)|I| + k$ such patients. Therefore, there are at least that many patients in I' in the optimal solution. However, there are no more than $|I'| * OPT$ patients in the optimal solution. Therefore,

$$(2 * OPT)|I| \leq |I'| * OPT$$

$$2 * |I| \leq |I'|$$

This proves the lemma. ◀

▶ **Theorem 6.** *For the **FirstFit** algorithm there is a $O(\log n)$-competitive ratio upper bound, assuming only one jab and $p = 1$.*

Proof. Now to use the lemma. As we know, the amount of patients at the day with the most patients according to FirstFit is ALG. Let that day be day d, and $I_0 = [d, d+1)$. Let $I_i = I'_{i-1}$ as defined in the lemma above. Repeated use of that lemma says that there are at least $ALG - 2 * i * OPT$ patients in interval I_i, and the length of I_i is at least 2^i. Let $b = \frac{ALG}{OPT}$. Let I_w be an interval with more than 0 patients according to this. Then the number of patients in I_w is at least 2^w, and at most n. Therefore $2^w \leq num \leq n$, so $w \leq \log_2(n)$. Now we take a look at $I_{\frac{1}{2}\frac{ALG}{OPT}}$. If there are no patients, then $ALG - 2 * (\frac{1}{2}\frac{ALG}{OPT}) * OPT < 0$. Therefore, $ALG - ALG < 0$, which is a contradiction. Therefore, there are patients. Thus, we have $\frac{1}{2}\frac{ALG}{OPT} \leq \log_2(n)$, and $\frac{ALG}{OPT} \leq \log_2(n)$.

◀

4 Experiments and results

In this section, we will provide a series of computational experiments for our approaches on both offline and online problems. Firstly, the experiments for offline models are presented. In this series of experiments, we measured the running times of our ILP models on different instances with different number of patients. Test instances were generated by several teams of people that worked on that problem. The average time of 5 independent runs for each instance is presented in the table 1 along with the optimal number of machines needed for that instance. For some instances, because of the fact that the constant T is too big, the second models has too much variables and constraints and it is impossible to be solved in reasonable time, therefore, we indicated running time as $-$. The running times of the second model are smaller than for the first one, but this happens just for some special instances. In average the running times of the first model are much smaller than for the second one. These results prove our claim that the model that use starting times of the jabs for the patients can be solved more faster than the model that use a variable for each time slot.

The experiments for online algorithm are presented in table 2. The instances we used for those experiments are different for those used in the experiments for offline problem. A number of 5 independent runs where held for each instance and the average time along with the average solution were considered. Moreover, for some small instances we run the first ILP model in order to get the optimal solution, and computed the rapport of number of machines used by our online algorithm to the optimal solution.

5 Conclusions and Future Work

In our paper, we proposed two mathematical models for the offline problem and two algorithms for the online problem. In addition, we provided an analysis for competitive ratio bounds and presented a comparative analysis of different ILP models. Our computational experiments and results, along with theorems and proofs confirm the aforementioned. Further improvements include the reduction of time complexity for offline problem by designing more smarter ILP models. For the online problem an important direction for future works is to extend our proof and find an upper bound that considers the vaccination problem with two jabs. Online deterministic and non-deterministic algorithms that have a competitive ratio closer to general lower bound also present a high interest for future works.

References

1 J. Chuzhoy, S. Guha, S. Khanna and J. S. Naor, "Machine minimization for scheduling jobs with interval constraints," 45th Annual IEEE Symposium on Foundations of Computer Science, 2004, pp. 81-90.

2 Phillips CA, Stein C, Torng E, Wein J (2002) Optimal time-critical scheduling via resource augmentation. Algorithmica 32(2):163–200

3 Chen L,Megow N, Schewior K (2018) An O(log m)-competitive algorithm for online machine minimization.SIAM J Comput 47(6):2057–2077

4 Azar Y, Cohen S (2018) An improved algorithm for online machine minimization. Oper Res Lett 46(1):128–133

5 Im S, Moseley B, Pruhs K, Stein C (2017) An O(log log m)-competitive algorithm for online machine minimization. In: 2017 IEEE real-time systems symposium, RTSS 2017, Paris, France, December 5–8, 2017, IEEE Computer Society, pp 343–350

6 Saha B (2013) Renting a cloud. In: Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS) 2013, Schloss Dagstuhl–Leibniz-Zentrum fur Infor-matik, Leibniz International Proceedings in Informatics (LIPIcs),vol 24, pp 437–448

7 Kao M, Chen J, Rutter I, Wagner D (2012) Competitive design and analysis for machine-minimizing job scheduling problem. In: Chao K, Hsu T, Lee D (eds) Algorithms and computation—23rd international symposium, ISAAC 2012, Taipei, Taiwan, December 19–21, 2012. Proceedings, lecture notes in computer science, vol 7676, pp 75–84. Springer, Berlin

8 N. R. Devanur, K. Makarychev, D. Panigrahi, and G. Yaroslavtsev,(2014) Online algorithms for machine minimization, CoRR, abs/1403.0486

9 Chen, C., Zhang, H. and Xu, Y., (2020) Online machine minimization with lookahead. J Comb Optim., https://doi-org.proxy.library.uu.nl/10.1007/s10878-020-00633-w

Algorithm 1 FirstFit

```
procedure FILL HOSPITALS
    start ← the first day of a jab
    end ← the last day of a jab
    machine-idx ← the index of a hospital
    patient-idx ← the index of a patient
    for i ←start to end do
        machines[machine-idx][i] ← patient-idx
procedure FIND INTERVAL
    i-start ← the first available day for a patient
    i-end ← the last available day for a patient
    processing-time ← the processing time of a jab
    for machine-idx ← 0 to length of used machines − 1 do
        machine ← machines[machine-idx]
        if i-start >= len(machine) then
            while len(machine) <= i-start + processing-time do
                append 0 to machine
            return machine-idx, i-start, i-start + processing-time - 1

        while len(machine) <= i-end do
            append 0 to machine
        for day ←i-start to i-end − processing-time + 1 do
            is-full← False
            for day-of-processing ← 0 toprocessing-time do
                if machine[day + day-of-processing] ≠ 0 then
                    is-full← True
                    break
            if !is-full then
                return machine-idx, day, day + processing-time - 1
    append empty list to machines
    length ← 0
    while length <= i-start + processing-time do
        length ← length + 1
        append 0 to machines[-1]
    return length of machines - 1, i-start, i-start + processing-time - 1
procedure FIRSTFIT
    machines ← empty list
    p1 ← input
    p2 ← input
    gap ← input
    patient ← 0
    date ← input
    while data does not contain 'x' do
        patient ← patient + 1
        map data to r, d, x, l
        machine-idx-1, first-dose-start, first-dose-end ← Find Interval(r, d, p1)
                ▷ machine-idx-1 is the hospital that provides first jab for incoming patient
        Fill hospitals (first-dose-start, first-dose-end, machine-idx-1, patient)
        machine-idx-2, second-dose-start, second-dose-end ← Find Interval((first-dose-
    start + p1 + gap + x, first-dose-start + p1 + gap + x + l, p2))
                ▷ machine-idx-1 is the hospital that provides first jab for incoming patient
        Fill hospitals (second-dose-start, second-dose-end, machine-idx-2, patient)
        print first-dose-start, machine-idx-1, sec-dose-start, machine-idx-2
        data ← input
    print length of machines
```

Table 1 Experiments for the offline problem

Results for offline models.			
Test instance	Offline running time model 1	Offline running time model 2	Offline solution
0	0.00099 sec	0.0 sec	0
1	0.00879 sec	– sec	1
2-1	0.00677 sec	0.016853 sec	1
2-2	0.00965 sec	– sec	1
3-1	0.01884 sec	0.033725 sec	2
3-2	0.02810 sec	0.017344 sec	1
3-3	0.01820 sec	0.042550 sec	1
4-1	0.06633 sec	0.033726 sec	2
4-2	0.07157 sec	0.032771 sec	3
4-3	0.04113 sec	– sec	2
4-4	0.00501 sec	0.101718 sec	2
5-1	0.10608 sec	0.383249 sec	3
5-2	0.03889 sec	0.639411 sec	4
5-3	0.06038 sec	0.119503 sec	3
5-4	0.02518 sec	0.207083 sec	1
6-1	0.0624 sec	0.860603 sec	2
6-2	0.074734 sec	0.916763 sec	2
6-3	0.055260 sec	0.135616 sec	1
7-1	0.142165 sec	1.832611 sec	2
7-2	0.084716 sec	0.163283 sec	1
9	0.165624 sec	1.144028 sec	1
10-1	0.7462 sec	10.1332 sec	3
10-2	0.4541 sec	6 min 2.9382 sec	1
10-3	0.1314 sec	18.8081 sec	1
12	1.37468 sec	37.8269 sec	3
15	0.31250 sec	11.8696 sec	3
20	1.64017 sec	1 min 24.5422 sec	2
23	1.98096 sec	15 min 26.2508 sec	1
45	8.35741 sec	– sec	5

Results for online algorithm.				
Test instance	FirstFit running time	Online solution	Offline solution	ALG/OPT
0	0.000031 sec	0	0	-
1	0.000036 sec	1	1	1
3-1	0.000106 sec	3	1	3
3-2	0.000096 sec	2	1	2
3-3	0.000061 sec	2	1	2
3-4	0.000084 sec	2	1	2
3-5	0.000076 sec	2	1	2
3-6	0.000064 sec	3	-	-
3-7	0.000078 sec	3	2	1.5
4-1	0.000204 sec	2	2	1
4-2	0.000073 sec	3	2	1.5
4-3	0.000086 sec	2	1	2
4-4	0.000141 sec	2	2	1
4-5	0.000068 sec	3	2	1.5
5-1	0.000073 sec	1	1	1
5-2	2.094662 sec	3	2	1.5
6	0.000145 sec	4	2	2
7-1	0.000123 sec	2	1	2
7-2	0.000119 sec	3	2	1.5
7-3	0.000179 sec	3	2	1.5
8	0.000096 sec	2	1	2
10	0.000197 sec	5	2	2.5
14	0.000208 sec	4	-	-
20-1	0.000268 sec	2	1	2
20-2	0.000276 sec	4	2	2
45	0.004281 sec	13	5	2.6
46	0.050989 sec	33	29	1.138
50-1	0.003321 sec	12	-	-
50-2	0.000803 sec	7	-	-
60	0.001890 sec	10	4	2.5
69	0.004452 sec	34	-	-
96-1	0.003485 sec	18	-	-
96-2	0.003961 sec	16	-	-
96-3	0.004670 sec	29	-	-
96-4	0.002880 sec	19	-	-
116	0.078250 sec	27	-	-
1000	0.215953 sec	49	-	-
5000	1.338015 sec	110	-	-
10000	0.381817 sec	23	-	-

Table 2 Results for the online algorithm