UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

APPROXIMATED ALGORITHMIC MECHANISM FOR SCHEDULING CHILEAN OPERATING ROOMS WITH MULTI-AGENT BASED SIMULATION

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN CIENCIAS

ANDRÉS OLIVARES

COMISIÓN:

PROFESOR GUÍA:
JÉRÉMY BARBAY

PROFESOR CO-GUÍA:
JOSÉ CONTRERAS

SANTIAGO DE CHILE
2021/12/08AT 19:03:42

Abstract

Operating rooms are one of the most important parts of hospitals and the main cost and revenue source. Scheduling patients, efficiently allocating resources in operating rooms, would greatly improve service quality. However, automatic scheduling of non-emergency patients is complicated due to the lack of an objective indicator that points out who to attend. Implemented optimization solutions must be approved by a board of surgeons given low trust on the system, due to lack of decision making explainability. The changes they perform affect the optimality of results to an unknown extent and give space to misaligned incentives. Mechanism design applies game theory to deal with situations where agent valuations are taken into account. We formalize the problem and design a $2t$-approximated mechanism which is truthful in expectation. The mechanism produces a solution that takes into account the opinions (as values) of surgeons on patients risk, while also improving the scheduling. To address the practical impact of the solution, we implemented the mechanism solution in a simulation environment that uses the data of Chilean hospitals given by various public healthcare institutions. Given practical issues we opt for an implementation that minimizes the quadratic error to the truthful solution. Results are compared with simpler scheduling methods as well as a classifier trained in the original data, approximating current behavior. This comparison showed similar performance to metric specific optimizing methods, while also maximizing the social benefit which includes surgeon valuations.

Agradecimientos

Table of Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

Cooperation is a fundamental assumption that engineering systems make when social agents are involved. A common example can be seen in lines for a document or procedure, where the system focuses on the procedure and assumes a behavior from their users. However, there are cases where this assumption is harder to make such as public transport, where cooperation is lower among drivers, and biddings, where each agent wants to obtain the highest possible benefit.

The topic of public transport has a common example of the effect of greedy behavior known as Braess Paradox, where adding a road to a network can slow down the overall traffic over it [69]. On the other hand, a central planner would take advantage of the new road. However, there are some instances where there is asymmetry of information between agents and the central planner such as airport arrival times, where a company has more information about the uncertainty of arrival times of their own flights [63].

Healthcare is an area where cooperation is high. The constant effort of medical staff on improving the well-being of the population and recently on the Covid-19 pandemic are the opposite of greedy behaviour. However, regarding information asymmetry and uncertainty, the subjective opinion on urgency of attendance may give space to non optimal planning since "a measure of clinical need ... is very difficult to obtain" [87].

In Chilean healthcare, there are numerous cases where the waiting time for surgery is over 3 years [56]. Data from studies on surgery rooms show that the country is under the OECD countries on metrics such as canceled surgeries, room usage hours, first surgery start delay, surgeries per day and others [11]. Also, there is a high variation among these metrics on Chilean public healthcare institutions, while some hospitals are up to OECD standards others have serious problems on efficiency [11]. This last statement can be made due to the fact that similar institutions with different waiting list sizes have similar results [11]. Around 40% of the surgeries performed in Chilean public surgery rooms were done to patients that entered as emergency [11]. It is also important to notice that 85% of these patients were already on a waiting list [11]. There is a correlation between waiting time and the emergency entrance, it could be hypothesized that the comorbidities generated by the time extension eventually produce the collapse of the patient, however, this is not an hypothesis that is

tested in this thesis [11].

Being self-managed, hospitals have a high variability in how they work [11]. Institutions of similar resources with different list sizes can have similar average waiting times [11]. Although there is space for improvement on management and logistics, in the current state some patients may have to wait until their condition becomes life threatening, were others with the same condition get attended in an acceptable amount of time.

High variability in treatment was partially solved with the implementation of the Diagnosis Related Groups (DRG) payment system, which incentivizes hospitals to lower their costs on well defined treatment packages [83, 70]. The DRG payment system has some issues such as weakness to collusion and missreport of diagnosis [64]. Moreover, while it effectively improves the hospitals efficiency through incentives, it can also incentivize them to pick the less complex cases that fall under the same DRG. Most countries that have implemented DRG also have strict deadlines for all patients, i.e., the UK has a deadline of one year for attendance after which the patient is sent to another hospital with a penalty to the first one [11]. Given that Chile lacks such a policy, and it is also hard to implement quickly, we aim to create a system that allows effective use of surgery rooms and a prioritization.

We use medical valuation in the objective function of the scheduling program, which in the operating room context is called Operating Room Scheduling Problem (ORSP). The value given to such a subjective opinion must be the real valuation of the surgeon. To achieve this, we develop a mechanism with useful properties that allow a central system to incentivize the prioritization, make the surgeons specify their true valuations and most importantly give an schedule the personnel would not modify.

Lavi and Swany developed a general technique that uses an approximation algorithm to obtain an approximation for a mechanism which is truthful in expectation through the use of randomized metarounding [55, 19]. An example case for their technique are combinatorial auctions. We model the problem as a combinatorial auction with an underlying scheduling problem and develop a $2t$-approximation in expectation algorithm which gives a $2t$-approximation mechanism for our problem. To do this, we improve on the previous $6t$-approximation by giving a $2t$-approximation algorithm [9].

To evaluate the mechanism, we use the data from various sources that process data of Chilean public healthcare [65]. The data is sensitive and required a high amount of processing due to the inner working of the Chilean healthcare technology, which allows use of free text on most attributes. We used the data of the institutions with better data quality for the tests [60].

The prioritization rule to use in the simulation is of high importance. Given the possibility of an increase in risk over time, we studied the usage of a risk over waiting time function. However, we found no correlation on severity and waiting time for the same CIE-10 diagnosis. This implies that patients with high severity are not necessarily being attended before others with less severity and the same diagnosis. We use a *soft deadline* based on experts opinions as prioritization.

In combinatorial auctions, even approximations take a long time to compute on practice,

to measure scheduling metric results we implement an algorithm with the same bound that minimizes the square error to Vickrey-Clarke-Groves (VCG) payments with a similar approach to CORE-VCG auctions [72]. Results on the simulation using the algorithm bound show that the average, maximum and median of the waiting time are similar to the *first deadlines first* algorithm, however, the likehood of the surgeons agreeing with the schedule is higher. The likehood of agreement is higher than in the simulated current scheduling while lowering the waiting time and the maximum wait time to a range suggested by literature.

This thesis assume basic knowledge of discrete mathematics, like Chapter 1 from Diestel's "Graph Theory" book, and probabilities, such as the first two chapters of Mizenmacher and Upfal's book "Probability and Computing" [29, 66].

This thesis is structured as follows. Chapter 2 provides an overview on computational complexity, approximation algorithms and algorithmic mechanism design. Readers with a computer science background can skip to the mechanism design part in Section 2.2.

Chapter 3 reviews the Chilean healthcare, focused on operating rooms, and theoretical approaches to the problem from an algorithmic perspective. Sections 3.1 and 3.2 show the difficulties on combinatorial mechanisms and the underlying scheduling problem, focused on split interval scheduling, which we use to model the problem. Section 3.3 shows results from a previous data oriented study and serves as a motivation to the operating room scheduling problem. It also contains an overview on the problems of data given current technologies on healthcare in Chile, moreover, it reviews approaches to prioritization from data.

Chapter 4 explains theoretical results. Here, the arguments for the selected model are given, the problem is modeled as a combinatorial auction with an underlying t-interval scheduling problem on multiple identical machines. We explain the main algorithm we modify to obtain the theoretical results as well as variations of the problem, including alternative solutions with similar approximation bounds.

Chapter 5 shows the implementation decisions of the simulation. The importance of this chapter relies on two main issues: the difficulties of implementation of the approximation mechanism algorithm and the decisions made for agent based simulation. On the first topic, we remove truthfulness from the algorithm to obtain results with the theoretical bound and compare them with other algorithms. Regarding the simulation, we detail the assumptions made over data and explain the decisions on what facilities to evaluate. Finally, we show the results of the experiments.

Chapter 6 explains the results taking into account the assumptions made and decisions on implementation. Moreover, we discuss the future work on theoretical results and practical solutions for the problem.

Chapter 2

Background

This chapter is aimed at readers without experience in computer science and game theory, with the objective of giving an overview on the concepts used. On public healthcare problems in Chile, a quick revision can be found in Chapter 3.

This chapter is structured as follows. Section 2.1 begins with quick overview on algorithms execution time in order to explain complexity classes and measuring problem difficulty. We also show examples on ways to deal with problem hardness. Section 2.2 explains what game theory is and what it is used for without mathematical formality. The rest of the section focuses on the concepts of algorithmic game theory and what we look for in these types of problems.

2.1 Computational complexity

This section is mainly geared towards readers that do not have computer science knowledge with the exception of the subsection on parameterized algorithms. Section 2.1.1 shows time complexity only to the extent of big $\mathcal{O}$ notation. Section 2.1.2 talks about the main hardness classes on a superficial way, without explaining Turing machines and languages. A better explanation of these concepts is given by M. Sipser, whose book was the main source for this section as well as the following one [85]. Section 2.1.3 gives a quick explanation on approximation algorithms with a classic example, a deeper treatment on this topic can be found in Vazirani et al. [89]. Finally, Section 2.1.4 shows the difference between approximation and parameterized algorithms. Its content is based on Cygan et al. [26].

2.1.1 Time complexity

Let $L = \{a_0, a_1, \ldots, a_n\}$ a list of ordered values where $a_{\mathrm{i}} \in \mathbb{R}, \forall \mathrm{i} \in \{0, 1, \ldots, n\}$ and $a_{\mathrm{i}} \leq a_j, \mathrm{i} \leq j, \forall \mathrm{i}, j \in \{0, 1, \ldots, n\}$. For educational purposes, we would like to know the fastest way to give the position of value $b \in \mathbb{R}$ in the list. A simple solution would be to iterate over each element in the list while updating the position value until an element of equal value is found, then, output the current position value. How much time would this algorithm take to find the correct element? Given that every computer is different it is easier to compare

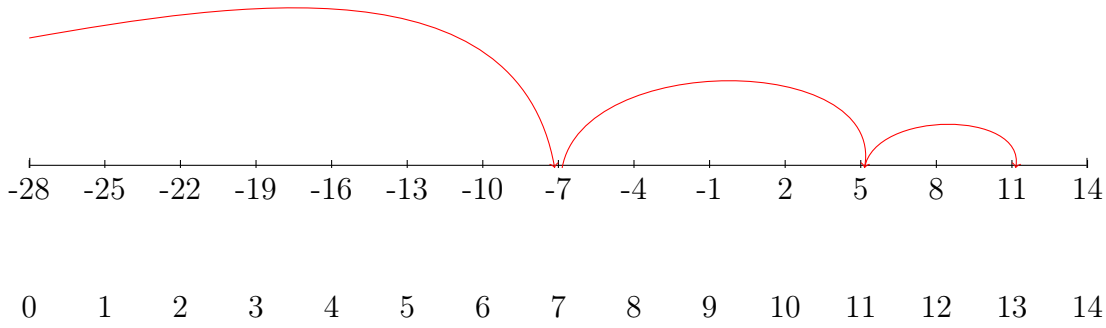

Figure 2.1: Binary search of the value 11 in an ordered array of 15 values. First the value in the leftmost middle position is inspected, since $11 \geq -7$, the right side is inspected. The middle of the positions [8,14] is the 11th position with value 5, since it is lower than 11 we inspect the right side [12,14] and check the middle value. This value, position 13, corresponds to the value in the search.

the number of operations made. In this case, the operations correspond to access elements, comparison between values, and a sum to the position value. In summary, we take three steps for every iteration. It is easy to see that in the worst case, when the value we are looking for is at the end of the list, the number of total steps is $3n$. Although is is not the only theoretical way to analyze an algorithm, using the worst case is the most common one. A good resource on alternative ways of analyzing algorithms can be found in Tim Rougharden's book [79].

Another way to solve this problem would be to skip some values in the list until we are close to the one we are looking for. Suppose we check the value in the middle of the list, the floor of the middle position if the length is even. We known that if the value we are looking for is bigger than the one inspected, we have to look for it in the right side of the list, and the opposite if it is smaller. We inspect the middle value of the corresponding side recursively. By doing this, we reduce the search space until we reach the value when there is only one value left.

An example of this algorithm can be found on Figure 2.1. Looking for the value 11, we inspect the middle value and note that this is lower than 11. We inspect the middle value of the right side, which is 5, also less than 11. Finally, we inspect the middle value on the right side of the corresponding part and arrive at the value without inspecting all values.

This way is faster than the previous one. We can check this by counting the steps like we did in the previous example. First we need to calculate the middle index to be able to jump there. This point is $\frac{hi-li}{2}$ where hi is the index of the highest value (n at the beginning) and li is the index of the lowest value (0 at the beginning). After that we check the value in the given position and compare it to the one we are looking for. We change the higher and lower indexes given the result or stop if the checked value is the correct one. The sum gives 4 steps per jump, however, given that we are reducing the search space by half on every jump the worst case happens when we do all the jumps until there is only one value in the search area. The number of jumps j is equivalent to the amount of times we have to halve n to make it

one, this means $1 = \frac{n}{2^j} \rightarrow j = \log(n)$. The total amount of steps in the worst case is $4\log(n)$, which is strictly better than the last result.

Definition 2.1 ($\mathcal{O}$ notation) *Let $f, g : \mathcal{N} \rightarrow \mathcal{R}^+$ functions. We write $f(n) = \mathcal{O}(g(n))$ if there exists positive integers c and n_0 such that $\forall n \geq n_0, f(n) \leq cg(n)$.*

When we are theoretically analyzing algorithms we make use of functions and big $\mathcal{O}$ notation to compare them, this is called asymptotic analysis. The big $\mathcal{O}$ notation is a way to simplify the comparison of functions, when we use it we remove constants and only consider the higher order term. In big $\mathcal{O}$ notation, $4\log(n)$ is shown as $\mathcal{O}(\log(n))$ which means that the algorithm runtime has an upper bound of $c\log(n)$, for $n > n_0$. This is a measurement of the time it takes for the algorithm to compute the output. There are other measures for algorithms, however they are not necessary to understand this thesis.

2.1.2 Complexity classes

We focus on another task, which is answering the question of which problem is harder to solve? In this question, we are comparing problems instead of solutions. When we think about problem A finding an element in an ordered array and problem B finding an element in an unordered array, deciding which is harder to solve seems easy, given that at least in the ordered array we have a hint about where to look at, while in the unordered the only alternative is to look at every element. However, both problems belong to the same class which is P, the class of problems that can be solved in polynomial time. Small differences between polynomials such as $\mathcal{O}(n^2)$ and $\mathcal{O}(n^3)$ are not considered here, theoretically, they are way more similar than $\mathcal{O}(n^2)$ and $\mathcal{O}(2^n)$, which imply way stronger differences between problems.

There is a vast amount of problems whose polynomial time algorithm has not been found. A second question can be asked about such problems: if we are given an answer, can we verify that it is a solution? A good example of this class of problems is the K-CLIQUE problem, which is to determine if a graph has a clique of size k. A clique is a subset of a graph where every node is connected to all nodes in the subset. While finding a clique of size k over a graph is not simple, verifying that a given answer is a clique of a given size is straightforward. We only have to check if every node is connected to the other nodes in the clique and that the total amount of nodes is k. This can be done in polynomial time. Problems where we can verify a solution in polynomial time are in the class of *nondeterministic polynomial time* (NP) problems. While clearly that $P \subseteq NP$, it is the main question of theoretical computer science to find out if $NP \subseteq NP$ and $P = NP$.

Demonstrating that a problem belongs to a class may be a hard task and intuition may tell us that two problems share similarities and thus, they should belong to the same class. We can see this case with k-Independent Set and k-Vertex Cover.

Definition 2.2 (Vertex Cover) *Let $G = (V, E)$ a graph. A vertex cover is a set of nodes S such that $\forall v \in V$ there is a vertex $u \in S$ such that $(v, u) \in E$. A k-Vertex Cover is a vertex cover where $|S| = k$.*

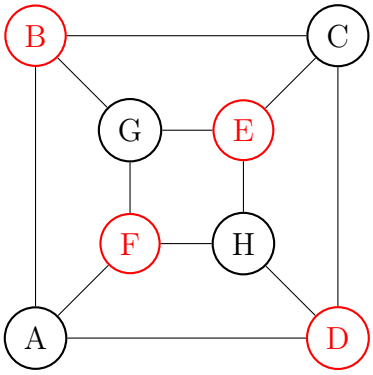

Figure 2.2: Unweighted graph with $V = \{A, B, C, D, E, F, G, H\}$. A vertex cover is represented by the vertices in red (B,D,E,F), this set is also an independent set.

Definition 2.3 (Independent Set) *Let $G = (V, E)$ a graph. An independent set I is a set of vertices such that $\forall u, v \in I$, $(u, v) \notin E$.A k-independent set is an independent set of size k.*

An example of a graph that is both, a vertex cover and an independent set can be seen in Figure 2.2.

Lemma 2.1.1 I is an independent set of $G = (V, E)$ if and only if $G - I$ is a vertex cover.

Proof.
$\rightarrow$) $\forall u, v \in I$, $(u, v) \notin E$ but $(u, w) \in E$, $w \in V(G - I)$ such that every edge is covered by $G - I$.
$\leftarrow$) $\forall (u, v) \in E$, $u \in G - I$ or $v \in G - I$ which implies that $(w, y) \notin E$, $w, y \in I$, thus I is an independent set.

□

We will formalize this notion of difficulty similarity between two problems. Notice that given a k-independent over a graph of size n, we have a vertex cover of size $n - k$. Given this result, we know that solving k-Independent Set has the same difficulty as solving $n - k$-Vertex Cover since to obtain a $n - k$-vertex cover we just have to pick every node that is not in the independent set, which takes time $\mathcal{O}(n)$. A reduction from problem A to problem B is an algorithm that transforms problem B to problem A, such that if a solution to problem B is found, then it can be used to solve problem A. It is important to know that most of the time this reduction must be done in polynomial time to be able to show that the problems belong to the same class, this is written as $A \leq_P B$. In the mentioned example Vertex Cover$\leq_P$Independent Set and Independent Set $\leq_P$Vertex Cover by using the respective side of the demonstration.

It is intriguing which problems can be reduced to others, and which ones can solve many other problems through a reduction. This raises the question on which classes beside the ones show, exist. While there are not only subclasses among the ones mentioned and even classes for space, this work only requires understanding of the mainly used classes.

It is highly important to know which problem can produce a solution to other problems. In the context of NP problems, we would like to have a problem which can solve most other problems in the class so we can focus our efforts in that problem. There is a subclass of problems that can solve every other problem in NP by using reduction. Meaning that for every problem A in NP there is a problem B in the class such that we can reduce in polynomial time A to B. The class that has this property is named NP-hard. If a problem is in NP-hard and at the same time belongs to NP, it is in the NP-complete class.

2.1.3 Approximation algorithms

There are many optimization problems such as Minimum Vertex Cover, Knapsack Problem, 3-SAT and others that are in NP and are key to many applications. Considering that solving an NP problem in polynomial time would solve a long time unsolved problem and the most important problem of theoretical computer science, some areas assume that NP$\not\subseteq$P and look for other ways to solve these problems.

Approximation algorithms deal most of the time with NP-hard problems and sacrifice optimality to obtain a polynomial time algorithm. To exemplify their usage we will use the Minimum Vertex Cover problem, which is finding the minimum size vertex cover in a graph. Given that this problem is NP-hard, we will look for a polynomial time algorithm that reaches a near-optimal solution.

A common way to build an approximation algorithm is to relax the problem to a fractional domain, such that the fractional problem can be solved in polynomial time. Our optimization problem for Minimum Vertex Cover as an integer optimization problem can be seen to the left in 2.1, where $x_\mathrm{i} = 1$ means that the vertex i is picked and $x_\mathrm{i} = 0$ if not picked. The relaxation can be seen to the right in 2.1.

$$\min \sum_{\mathrm{i} \in V} x_\mathrm{i} \quad \text{(Min VC)}$$
Subject to
$$x_\mathrm{i} + x_j \geq 1, \forall (\mathrm{i}, j) \in E$$
$$x_\mathrm{i} \in \{0, 1\}, \forall \mathrm{i} \in V$$

$$\min \sum_{\mathrm{i} \in V} x_\mathrm{i} \quad \text{(Relaxed Min VC)}$$
Subject to
$$x_\mathrm{i} + x_j \geq 1, \forall (\mathrm{i}, j) \in E$$
$$0 \leq x_\mathrm{i} \leq 1, \forall \mathrm{i} \in V$$

We will use the solution to the relaxed problem to obtain the approximation. Linear programming problems such as this one can be solved in polynomial time. However, the result may give rational values, and to obtain a vertex cover it is necessary to round the solution to integer values. Since $x_\mathrm{i} + x_j \geq 1$, $x_\mathrm{i} \geq \frac{1}{2} \forall \mathrm{i} \in V$, we pick values where $x_\mathrm{i} \geq \frac{1}{2}$. The result is a vertex cover since we pick at least one node of every $(\mathrm{i}, j) \in E$. Given that the fractional problem solutions space includes the integer solutions, its optimal value (OPT_f) is at most the optimal value of the integer problem (OPT), we picked every $x_\mathrm{i} \geq \frac{1}{2}$, which means that $x_\mathrm{i} = 1$ in our approximation, at most we double the original value. We conclude that our algorithm solution is at most $2OPT_f$ and at most $2OPT$.

Definition 2.4 (Approximation) *We say that an approximation algorithm for a minimization (maximization) problem is k-optimal if the value of the approximation is at most (least)*

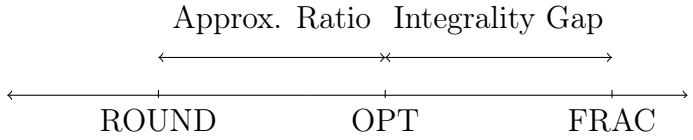

Figure 2.3: Optimal values on a line. The fractional rounding solution ROUND is at most OPT, the problem solution, the fractional solution FRAC is at least OPT. The integrality gap is the supreme of FRAC/OPT, while the approximation ratio is given by OPT/ROUND.

k ($\frac{1}{k}$) times the optimal value.

The previous algorithm is a 2-optimal approximation for Minimum Vertex Cover.

This technique of using a linear programming relaxation to obtain an approximate solution from a rounding of the fractional solution raises the question about which problems can be approximated by rounding the linear programming relaxation (LP). The answer is that every LP has an integer polytope Z (feasible solution), however, the main issue is how good the approximation actually is. We explain a maximization problem. The minimization version is analog.

Given that the solution to an integer problem IP is OPT, and the solution to the LP of such a problem is FRAC, a rounded solution ROUND can be at most OPT, an illustration of the relation can be seen in figure 2.3. The ratio FRAC/ROUND is a bound over OPT/ROUND which is the approximation ratio, however it is easier to calculate given that we do not know OPT. However, the value FRAC/OPT gives a value on how good can the approximation get, the maximum value of this ratio is the **integrality gap**. Since OPT is the maximum value of a rounded solution, an approximation whose ratio FRAC/ROUND is closer to the integrality gap is closer to be optimal.

Definition 2.5 (Integrality gap) *Given a linear programming relaxation LP for a maximization problem P, let $FRAC(I)$ be the cost of an optimal fractional solution to instance I. The **integrality gap** of the linear programming relaxation is given by:*

$$sup_I \frac{FRAC(I)}{OPT(I)}$$

2.1.4 Parameterized algorithms

Another way to deal with problems in NP-hard is to make the algorithm runtime dependant on a parameter whose value is known to be small. Again, we use Minimum Vertex Cover as an example. First, a parameter whose value does not depend on the size of the graph is necessary. A common way to make such optimization problems easier to solve is to answer if there is a k size answer, in this case a k-Vertex Cover, then we can reduce the value of k until the minimum size, since at a lower value than the minimum size the problem should output no solution.

The k-Vertex Cover does not have a polynomial algorithm, however, conditioning the

runtime on a parameter we can have an algorithm which is polynomial for a fixed parameter. We focus on the easy cases and try to reduce the search area of nodes that are in the k-Vertex Cover. Suppose we have three nodes u, v, w and $(u, v), (u, w)$ as the edges - the minimum size vertex cover is $\{u\}$. Several reasons could point to this output, however, we do not want to try every possibility to know which node to pick. A valid reason that could make us pick u without testing all possibilities is to pick it because it has the largest number of edges. Another one would be the fact that the only edge that v (or w) have is connected to u and given that u has more edges, it is better to pick it in both cases. This method of solving the easy cases to filter out possible solutions is called **kernelization**. Another simple trick to remove options on k-Vertex Cover is to pick nodes with a number of edges higher than k since those edges could not be covered otherwise.

Take a graph $G = (V, E)$ and apply all **kernelization** rules we stated previously. After this prepossessing there are k' nodes left to add to the vertex cover. If we remove the edges in the vertex cover S from E and non-edges nodes from V, our new graph G' only has nodes of edges between 2 and k', with $k' \leq k$. It could be said that this is the hard part of the problem, which we can only be solved using non-polynomial algorithms and while we could use a brute force approach on G', we can still make it a bit faster. We take an edge (u, v) and we want to know if u or v belongs to the k'-vertex cover. To do this, we assume $u \in S$ and recursively look for a $(k-1)$-vertex cover in $G' - u$, if there are no edges left when $k = 0$, the vertex cover exists and we output the result, otherwise, we try with v. Checking the existing edges takes $\mathcal{O}(|E|)$ time, the amount of total recursions we make is $2^{k'}$ since in each recursion we make a binary decision between both endpoints of an edge and the maximum depth of the recursion is k. This technique is called **bounded search trees**. Given that $|E| \leq nk$ after **kernelization** (no node has more than k edges), the total runtime of this algorithm to find the k-vertex cover is $\mathcal{O}(2^k nk)$, which is still bad in the worst case, but it is fast for small values of k compared to the brute force $\mathcal{O}(n^k)$ approach. With this algorithm, we can solve Minimum Vertex Cover by finding the lowest k that outputs a vertex cover.

Definition 2.6 (Fixed Parameter Algorithm) *Algorithms with a running time given by $f(k)n^c$ for a constant c are called fixed-parameter algorithms (FPT). Such is the case for our $\mathcal{O}(2^k nk)$ k-vertex cover, where k is a parameter independent of n.*

The difference between approximated algorithms and *fixed-parameter algorithms* is that the first ones trade-off an optimal result for a polynomial time algorithm while FPT algorithms obtain the optimal result, but their time is not polynomial on the worst case - they are polynomial over n but not over the chosen parameter. A distinction with heuristics is also necessary as these solutions do not reach the optimal in contrast to approximation algorithms, heuristics do not have a theoretical bound over the optimal value.

2.2 Algorithmic mechanism design

In this section we explain the necessary concepts on algorithmic mechanism design that we use in our work. All topics are contained in the books "Algorithmic Game Theory" by Noam Nisan et al. and "20 Lecture on Algorithmic Game Theory" by Tim Roughgarden, which are

the main source for this section [69, 78]. We begin by explaining what is game theory and the problems it deals with in part 2.2.1, we also do a quick explanation on what is algorithmic game theory. Section 2.2.2 is the main body of most of this section and explains algorithmic mechanism design, which problems it solves and the main difficulties in the area.

2.2.1 Game theory

Game theory aims to model interactions between multiple entities which decisions may affect each other. The traditional example on game theory is the Prisoner's Dilemma. On this problem, there are two prisoners uncommunicated from each other which have only two options, to confess or to remain silent. If both remain silent the authorities do not have enough evidence and they get 2 years prison time; if only one confesses, all charges go to the one who remained silent, the one who confessed gets 1 year and the other gets 10 years; if both confess they sentence is divided and both are given 5 years. The best answer for the prisoners is for both to remain silent, however, when we take into account that they are uncommunicated and that each thinks only in their own good. If *prisoner 1* remains silent, the worst that can happen is for *prisoner 2* to confess to lower his jail time and change *prisoner 1*'s jail time from 2 to 10 years. On the other hand if *prisoner 1* decides to confess, he goes from 1 to 5 years if the other confesses. Rationally, if both think in their own good the solution is for both to confess, because if they do, there is no gain in changing their decisions, confessing is the best strategy that does not depend on the other prisoner's strategy.

Definition 2.7 (Strategy) *A strategy is a decision rule that defines the action the agent will take. Let S_{i} the set of all possible strategies for an agent, $s_{\mathrm{i}} \in S_{\mathrm{i}}$ denotes the strategy taken by agent* i*. Let n the amount of agents, we write $S = (S_1, \ldots, S_n)$ the set of all possible strategy vectors and $s_{-\mathrm{i}} = (s_1, \ldots, s_{\mathrm{i}-1}, s_{\mathrm{i}+1}, \ldots, s_n)$ the vector of strategies not including agent* i*, such that $(s_{\mathrm{i}}, s_{-\mathrm{i}}) = (s_1, \ldots, s_n)$.*

Definition 2.8 (Dominant strategy solution) *The concept of a best strategy independent of the other players choices is a **dominant strategy solution**. Formally, let $s \in S$ vector of strategies, $\mathrm{i} \in I$ a player such that $u_{\mathrm{i}}(s)$ is the utility of player* i *on the strategy s. Since the utility can change given the other players strategy we write $u_{\mathrm{i}}(s, s_{-\mathrm{i}})$ to denote the utility of player* i *given the strategies of all players in $I - \mathrm{i}$. A vector of strategies $s \in S$ is a **dominant strategy solution** if:*

$$\forall \mathrm{i} \in I, \forall s' \in S : u_{\mathrm{i}}(s_{\mathrm{i}}, s'_{-\mathrm{i}}) \geq u_{\mathrm{i}}(s'_{\mathrm{i}}, s'_{-\mathrm{i}})$$

Mechanism design, which is the main subarea of game theory we use on this thesis, creates games that have **dominant strategy solutions** such that a certain objective of the mechanism creator is reached. However, we will focus on the computational side of it, on problems that require algorithms for the mechanisms to work.

A concept that is similar to **dominant strategy solution** is the **Nash equilibrium** (NE). The NE is a vector of strategies such that each player cannot improve its benefit by changing strategies assuming that they know the other players strategies. The last part is the major difference between a **dominant strategy solution** and NE, since the first one is

the strategy vector where a player cannot benefit from changing strategies without knowing the others information, a **dominant strategy solution** is also a NE.

Game theory mostly focuses on studying the NE and by extension, algorithmic game theory mostly focuses on studying how to calculate NE by the usage of algorithms. It is important to know that finding a NE is not an easy problem even for simple games, but even applying the concept of NP-completeness is not right for these problems given that a NE always exists, these problems have their own class which is the Polynomial Parity Argument Directed Case (PPAD). We do not explore these topics given that mechanism design does not require these tools for the problems we present. However, the fact that many variations of the problem of finding the NE such as answering if the game has two NE or if there is an NE which contains a strategy S are NP-complete shows that problems in the area solvable in polynomial time are not found often.

2.2.2 Algorithmic mechanism design

To illustrate the kind of problems mechanism design aims to solve we will analyze an auction. In the auction we are tasked with selling a valuable relic, naturally, there are bidders interested in such an object. We model this situation as n bidders where v_{i} is the valuation that the bidder (player) i has for the item (our relic), $\mathrm{i} \in [n]$. Given that there is only one item, each bidder reports the price they are willing to pay for the relic, if a player does not win the relic the utility is 0, otherwise it is $v_{\mathrm{i}} - p$, with p the price the item was sold for. Usually in an auction the item would be given to the highest bidder, however, a bidder that knows that he can increase his utility may bid for a lower number to try to obtain the relic for a lower price, on this case the reported value depends on the strategies of other players and may result in unpredictable behaviour. Prices do not necessarily represent valuations and the item may sell for a lower price to a bidder which valuation and bid are way lower than others. There are also situations where the seller would like to sell the item to the person that appreciates the relic the most, in the previous situation a random bidder may win the auction and in order to maximize its profits may resell the item to this person. On both cases, we do not want the bidders to lie about their values, in other words, we would like them to be **truthful**. To avoid the mentioned problems we introduce **Vickrey auctions**. In this case the concept is simple, the winning bidder is still the one who bid the highest, however the price to be paid is the second highest bid.

Proposition 2.9 (Vickrey) *For all valuations $v_0, \ldots, v_n$. Let u_{i} be the utility of player* i *if it reports its valuation v_{i} and u'_{i} if it reports a valuation v'_{i}. Then $u_{\mathrm{i}} \geq u'_{\mathrm{i}}, \forall \mathrm{i} \in [n]$.*

Proof. Suppose player i wins bidding v_{i}, let $p*$ be the second highest bid. The utility is $u_{\mathrm{i}} = v_{\mathrm{i}} - p* \geq 0$, however, if he wins by reporting a value $v'_{\mathrm{i}} \geq p*$ then the utility is still $u'_{\mathrm{i}} = v_{\mathrm{i}} - p*$. Since the utility is independent of the value reported $u_{\mathrm{i}} \geq u'_{\mathrm{i}} \forall \mathrm{i} \in [n]$. Notice that if $v'_{\mathrm{i}} \leq p*$ the utility is zero. Suppose player i loses bidding v_{i}, then $u_{\mathrm{i}} = 0$. If i reports $v'_{\mathrm{i}} \geq w_{\mathrm{i}}$, then $u'_{\mathrm{i}} = v_{\mathrm{i}} - p*$ where $p* \geq v_{\mathrm{i}}$ which implies $u'_{\mathrm{i}} \leq 0$. Concluding that $u_{\mathrm{i}} \geq u'_{\mathrm{i}} \forall \mathrm{i} \in [n]$. □

A mechanism that has this property of players maximizing their utilities by reporting

their true values is called **truthful**, **incentive compatible** or **strategy proof**. We want to formalize this notion, however, other concepts also require a definition to formalize this property.

Definition 2.10 (Social choice function) *Let A be a set of possible alternatives or actions. Let L set of linear orders on A. A social choice function is a function $f : L^n \to A$.*

Definition 2.11 (Mechanism) *Let f a social choice function $V_1, \ldots, V_n \to A$ where V_i is the valuation of player* i*. Let $p = p_1, \ldots, p_n$ a vector of payments where $p_\mathrm{i} : V_1, \ldots, V_n \to \mathbb{R}$ A mechanism is a pair (f, p) where f is a social choice function and p vector of payments.*

Definition 2.12 (Truthfulness) *A mechanism $(f, p_1, \ldots, p_n)$ is incentive compatible, also called truthful or strategy proof if for every player* i*, every $v_\mathrm{i} \in V_\mathrm{i}$, and every $v'_\mathrm{i} \in V_\mathrm{i}$ we have that $v_\mathrm{i}(f(v_\mathrm{i}, v_{-\mathrm{i}})) - p_\mathrm{i}(v_\mathrm{i}, v_{-\mathrm{i}}) \geq v_\mathrm{i}(f(v'_\mathrm{i}, v_{-\mathrm{i}})) - p_\mathrm{i}(v'_\mathrm{i}, v_{-\mathrm{i}})$.*

We will revisit truthfulness constantly during this work, moreover, our results point to another notion of truthfulness which holds a similar meaning but applies to randomized mechanisms.

While the Vickrey auction incentives the players to bid truthfully, there is another key factor related to the objective. In fact, the rule that forces the winner to pay the second highest price has a downside for the seller which could have won more if the winner just paid its bid, on the other hand, the winner wins the difference between its bid and the second highest as benefit. When we sum the whole benefit it is equivalent to the price that the highest bidder can pay, which is also the highest value that can be transferred, the highest benefit of the system. Given an alternative $a \in A$, the sum of valuations it generates is the **social welfare** $\sum_\mathrm{i} v_\mathrm{i}(a)$. We could say that while the revenue of the seller is lower, the Vickrey auction maximizes the **social welfare**.

Definition 2.13 (VCG) *A mechanism (f, p) is called a Vickrey-Clarke-Groves(VCG) mechanism if:*

- *f maximizes the social welfare.*
- *given $h_1, \ldots, h_n$ with $h_\mathrm{i} : V_\mathrm{i} \to \mathbb{R}$ we have that $v_\mathrm{i} \in V_\mathrm{i} \forall \mathrm{i} \in [n] : p_\mathrm{i}(v_1, \ldots, v_n) = h_\mathrm{i}(v_{-\mathrm{i}}) - \sum_{\mathrm{i} \neq j} v_j(f(v_1, \ldots, v_n))$*

The Vickrey auction meets both requirements. Moreover, a VCG mechanism is also incentive compatible, this is mostly given by the *Clark's pivotal rule* [69]. *The Clark's pivotal rule* sets $h_\mathrm{i} = max_{a \in A} \sum_{j \neq \mathrm{i}} v_\mathrm{i}(a)$ which gives the mechanism the properties of non-negative utilities for all players and makes the system (seller in the auction case) to not have to pay money to players.

There are different categories of auctions in the mechanism design area. The amount of bidders, items, bidding instances, objective functions and other constraints make each

problem different from the other. Multiple and single unit auctions are types where the amount of items change, on the same topic, the stock of each item may also be higher than one, another example of a different category of auctions are iterative auctions, where the auction is performed many times with a winner and payment on each iteration, while objective functions may be different, the main focus is revenue and social welfare, finally, an example of an auction with constraints are spectrum auctions. However, the knowledge necessary for this work only involves multi-unit auctions outside of mentions to the other types, moreover, we focus on multi-unit auctions where the bids are done in bundles instead of picking a single item, these type of auctions are called **combinatorial auctions**.

In the context of an auction of multiple items where multiple bidders want certain bundles, aggrupations of items which hold a higher value in a bundle than the sum of individual values of each one, who should win the items? The answer is complex independent of what we decide to maximize, social welfare or the revenue of the seller. Giving the item to the bidder with the bundle of the highest value does not necessarily get a good result, a higher value may be obtained by selling the items in that bundle to independent buyers with lower bidding prices, and other alternatives have the similar conclusions. Evidently, these type of auctions main problem is determining who won the auction, this problem is NP-complete in the general case.

Combinatorial auctions are formalized as follows: We have a set of m indivisible items concurrently auctioned among n bidders. Every bidder i has a valuation function v_{i} such that for a bundle or set of items S, $v_{\mathrm{i}}(S)$ is the valuation of the bundle and $v_{\mathrm{i}}(S) \leq v_{\mathrm{i}}(T), S \subseteq T$.

For bidders to specify their valuations, which bundles they prefer over other and what item they would exchange in each bundle for another without changing their price is a complex task. In a more practical way of bidding, we focus on *single-minded valuations.*

Definition 2.14 (Single minded valuations) *A valuation v is single-minded if there is a bundle S^* and number $a \in \mathbb{R}$ such that $v(S) = a, \forall S^* \in S$ and $v(S) = 0$ for all other cases.*

By only giving value only to bundles that contain a bundle S and making every other possibility a zero value, the bidding is not only easier in the practical side but also for processing. However, allocation of items over single-minded bidders is NP-hard while the decision problem of whether there is a social welfare of at least k is NP-complete. As stated before, when facing NP-hard problems we look for away around by giving up optimality. An approximation algorithm with an $\sqrt{m}$ approximation guarantee can be obtained using a greedy approach for combinatorial single-minded auctions, the greedy mechanism is also incentive compatible. However, another way of obtaining such an algorithm would be to use the same strategy we pointed at *approximation algorithms section*, which is relaxing the optimization problem. We will explain this approach under the chapter *literature*.

Chapter 3

Literature review

The areas of algorithmic mechanism design, healthcare data and scheduling are extensive. In this work we used results and techniques from all of them to model and solve the problem of scheduling operating rooms in a game theoretic setting. While scheduling as a problem has a long history with many variations through the years, scheduling in operating rooms predates the mathematical formalization. In Section 3.1 we detail the operating room scheduling problem (ORSP). Algorithmic mechanism design has been extensively studied in the latest years. In Section 3.2 we point at the limits of computation in algorithmic mechanism design. We do so in the context of our problem. The most important part being Section 3.2.1 which explains a general technique for approximations in mechanism design.

Finally, in Section 3.3 we review the healthcare system in Chile focused on the operating rooms and data, its challenges and ultimately, how it can help in prioritization.

3.1 Scheduling

The main problem of scheduling consists of allocating n jobs on m machines, an schedule consists of an allocation for each job to one or more intervals. Most scheduling problems require the schedule to satisfy certain conditions. Jobs consist of a number of operations, each one with a processing requirement. These requirements usually are a release date, processing time, specific machines, precedence constraints, and others [16]. The objectives also variate, with cost minimization, due date minimization, weight maximization and others.

In this work we focus on interval scheduling, the problem of allocating jobs to machines with a given starting and processing times such that jobs do not overlap. Unlike *shop scheduling*, our objective function corresponds to job weight maximization. Job scheduling main problems and solutions can be found in Brucker's book as well as Anand and Pannerselvam review [16, 4]. Interval scheduling problems are also referred as *fixed job* scheduling and *k-track assignment*. A survey on interval scheduling problems variations can be found in Kolen et al. and Kovalyov et al. [51, 53]. A recent review on overall time-dependent scheduling is given by Stanislaw Gawiejnowicz [40].

Given that each surgeon or agent has a set of blocks it wants to schedule for its patients, we

detail literature on scheduling multiple intervals for the same job. These disjoint intervals of a job are also referred as *operations* in the *shop scheduling* literature [16]. When there is only one block per agent and identical machines, WEIGHT MAXIMIZING INTERVAL SCHEDULING can be solved in polynomial time by solving a MINIMUM COST k-FLOW PROBLEM [14].

Interval scheduling on parallel machines is not a simple task. Slight variations such as adding a deadline and tardiness penalty to the objective function make the problems NP-hard, a practical approach on this variation is studied by Sterna et al. [86]. Multiple blocks scheduling can also be seen as a special case of scheduling with dependencies, this kind of problems are usually seen in the context of servers such as Pedarsani et al. [73]. There is work on a game theoretical version of the server scheduling setup by Thielen and Krumke [88]. Lately, this idea has been applied to cloud computing, but without jobs starting times [39].

Interval scheduling with blocks per agent is referred as t-interval scheduling in the literature. Interval graphs are graphs that represent the time intervals of the jobs, such that each job has a vertex and there is an edge between two vertices of the jobs they represent intersect [71]. Solving MAXIMUM WEIGHT INDEPENDENT SET(MWIS) on interval graphs gives a solution to the interval scheduling problem. MWIS in proper interval graphs, which correspond to the case where no interval completely overlaps with another, has a polynomial time solution [18, 8]. Proper graphs belong to the class of claw-free graphs. Berman gave a d/2-approximation for MWIS on d-claw-free graphs [10]. A more general case with an approximation are bounded degree graphs [6, 81]. Approximations for other variations are studied by Kako et al. [47].

An alternative to scheduling with dependencies is multistage scheduling. The main difference is that jobs are restricted to a stage, we point to work on multistage scheduling on t-interval graphs [45, 46].

For scheduling on t-interval graphs, Bar-Yehuda et al. gave a $2t$-approximation for MWIS. Bar-Yehuda and Rawitz explored other variations for the problem such as t-interval scheduling with demands, the case where each job as a demand and the machines have a demand resource [9]. Butman et al. explore similar problems on these graphs [17].

3.1.1 Operating Room Scheduling

Scheduling in healthcare has been studied for a long time. Medical research had always been looking for efficient usage of the resources. Healthcare management and policies also look for measures and protocols that improve the efficiency or avoid common time sinks. Most famous problems on healthcare scheduling are PATIENT ADMISSION SCHEDULING PROBLEM(PASP), NURSE ROSTERING PROBLEM(NRP) and OPERATING ROOM SCHEDULING PROBLEM(ORSP). An excellent review on these problems is given by Abdalkareem et al. [1]. In this thesis as well as this section, we focus on ORSP.

The ORSP consists of two parts, **operating room** and **recovery room** [1]. The operating room part involves resources such as personnel, facilities and equipment. The recovery room consists of the allocation of patients on recovery beds post surgery. We give the definitions of different classifications of ORSP problems [1]:

- Scheduling strategy: how are blocks selected.
  - Open: there is no time slot reservation, nor day reservation.
  - Block: a group of surgeons are assigned to several time blocks.
  - Modified block: time slot reserves are allowed but uses the free space with block strategy.
- Advanced scheduling: scheduling of surgery dates.
  - Dynamic: given at consultation time.
  - Static: from the waiting list.
- Allocation scheduling: allocate resources and time for the surgery.

Zhu et al. points that open scheduling is usually done with a first-come-first-served (FCFS) strategy [90]. Another important difference is the patients origin. Inpatients and outpatients as well as elective and non-elective patients. In the last few decades, ambulatory surgeries have increased. Outpatients represent a higher volume of surgeries. Since one of the objectives of the project that inspired this work was to increase the number of ambulatory surgeries, we use these cases for our work. A good review in scheduling outpatients can be found in [20, 15]. Elective patients are part of the usual scheduling, in contrast, non-elective patients are the ones that add uncertainty to the system, usually they are emergency cases [90].

On the economical side, private and public healthcare facilities look for a decrease in costs. While not lowering the quality of attention, these costs are mostly related to problems such as surgery cancellations, inventory problems and unexpected overtime on surgeries. These factors give space to different solutions combined with the scheduling strategies, a review on the convenient method for these problem variations can be found in [2].

The emergency room also plays a role in the uncertainty. Even if there are assigned emergency operating rooms, common operating rooms may be used if needed. In this work we do not address the uncertainty of these factors since we require provable guarantees fist. Most solutions on this subtopic rely on optimization solves and simulations. A review on this topic can be found in [37, 76]. On the other hand, optimality bounds can be obtained if the problem's constraints are reduced [57]. We do this in our work in order to prove guarantees on the mechanism.

In Chile, research and implementation of optimization with prioritization to allow fairness has only been done by G. Durán et al [32].

Another solution to operating room scheduling is through simulation. Present in other areas of healthcare, simulation has a long history defined by the limits of computing power [43]. Among simulation techniques, the most popular are Discrete Event Simulation (DES), System Dynamics (SD) and Agent-Based Simulation (ABS). DES deals with agents passing through events in the systems, SD mainly uses flows and storages, while the relatively new ABS simulates the interactions among agents through defined rules [62].

Hospitals are complex systems, which are impossible to completely simulate through DES and arguably any other method [43]. However, there are simulations combining DES and ABS under specific cases, like patients decisions [50]. Work in agent based simulation for elective surgeries has answered important questions on operating room management [52, 80].

While originally this thesis was oriented to use both simulation methods on a near-complete hospital simulation, data and complexity did not make it possible. We address the simulation from an ABS perspective in Chapter 5.

3.2 Algorithmic mechanism design

Usage of auctions and mechanism design in practice is extensive. Most popular cases are school choice, kidney donations, matching markets, online auctions and frequency allocation [77, 21, 69]. A key aspect of landing slots mechanism design applies to surgery rooms and is a strong argument for the usage of mechanism design for this problem, information asymmetry [82].

Mechanisms for kidney donation do not involve money since it is illegal [77]. However, these mechanisms have additional problems that are usually circumvented by payments, one of these is ordered preferences [74]. As result, combinatorial auctions, a famous algorithmic mechanism design problem is impossible to solve in the no money case [38]. Solutions have to make weaker assumptions such as value verification to make it manageable [38]. Resource assignment problems, known as Generalized Assignment Problem (GAP) have multiple related uses and have known approximations based on work by Lavi and Swamy on the no money case [31, 55].

Clearly, GAP approximations are better in the case with payments. Improving from $\mathcal{O}(\log(n))$ to $(1-1/\mathrm{e})$ [35]. Fadaei and Bichley use the method proposed by Lavi and Swamy to obtain a $(1-1/\mathrm{e})$-approximation for GAP (slightly worst than the best $(1-1/\mathrm{e}+p)$ method [35]. This technique has high importance in this thesis and will be explained in Section 3.2.1. Closer problems to scheduling patients fall in the complementary items category, since doctors would prefer to have sequential usage of the room rather than highly separated hours. Nguyen et al. propose a solution to assignment problems (bundle allocation) that satisfy efficiency, envy-freeness and asymptotic strategy-proofness [68]. The trade-off for these results is the addition of a constraint on bundle sizes, this has similarities with our solution since our approximation depends on the bundle size.

To the knowledge of the author Liu et al. is the only author that approached the combination of ORSP with mechanism design [58]. However, this work does not allow multiple bids nor presents guarantees such as truthfulness. Moreover, it follows a straightforward solution using iterative bidding, which may be troublesome for the surgical team given that they must select new schedules until every space is filled. Finally, these kind of solutions do not behave well with complementary items [78].

In contrast, mechanism design approaches to scheduling have been studied in numerous works. In particular, machine scheduling has been studied from the perspective of a decentralized system where machines decide which jobs to accept. A review on this problem is given by Heydenreich et al. [12]. Abed et al. study a variation of mechanism design and scheduling were the strategic agents want jobs processed [3]. Another variation consists of stages controlled by different players [45]. Ananth and Chandra Sekaran provide a review on applications of game theory to scheduling in cloud computing [5]. Kress et al. gives a general review over the classification of the different variations by shared characteristics [54].

bidding languages for combinatorial auctions Failures of the VCG Mechanism in Combinatorial Auctions and Exchanges Declining valuations in sequential auctions No known mechanism is equivalent to the problem we propose. We decide to use combinatorial auctions not only due to a practical issue, but an optimality one too [49]. Although combinatorial auctions and VCG have several problems [24]. Following the success of the technique, we base our mechanism on the work of Lavi and Swamy [55].

3.2.1 Randomized metarounding for AMD

We explored in section 2.2.2 that many problems in mechanism design are NP-complete. We also summarized how approximation algorithms are a workaround for NP-hard problems. While some problems in mechanism design have an approximation algorithm, they are usually tailored to the mechanism and their techniques will not necessarily work on other problems. Lavi and Swamy developed a general technique, which gives an *r-approximation mechanism* which is also truthful in expectation [55]. Their work is based on a clever usage of *randomized metarounding* to preserve truthfulness.

Randomized metarounding is a technique with applications to approximation algorithms developed by Carr and Vempala. It builds a convex decomposition which purpose is to deal with known issues of randomized rounding over optimization, specifically, using a a convex decomposition instead of randomized solution assures that conditions are always met, such that no solution is unfeasible, the main example of a problem that would require such a technique is Multicast Congestion [19].

Randomized rounding is a standard technique in approximation algorithms problems. The technique is similar to the example problem in section 2.1.3, which is relaxing an optimization problem and round the resulting variable values to an integer value. On a generalized integer optimization problem IP:

$$
\begin{aligned}
&\min cx \\
&\text{s.t} \\
&Ax \geq b && (3.1)\\
&x \in \{0,1\} && (3.2)\\
& && (3.3)
\end{aligned}
$$

We relax the x variable and allow it to have values $x \in [0,1] \in \mathbb{R}$, we name this relaxed version P. A convex optimization problem such as P can be solved in polynomial time using the ellipsoid method. Let $x*$ be the optimal value for P, for the integer solution assign $x_{\mathrm{i}} = 1$ with probability ${x^*}_{\mathrm{i}}$ and 0 with probability $1 - {x^*}_{\mathrm{i}}$. This solution has value cx^* in expectation and fulfills $Ax^* \leq b$ in expectation.

However, *randomized rounding* has a clear problem, a solution picked using such probability distributions may break some constraints, there are problems where breaking a constraint is unnaceptable. *Randomized metarounding* deals with this problem trading-off optimality to fulfill the conditions, giving an approximate solution. To do this, we require an

r-approximation heuristic.

Definition 3.1 (Approximation heuristic) *A polynomial time algorithm A is an r-approximation heuristic to a minimizing integer program (min-IP) with positive objective function and linear program relaxation P, if, A finds a solution which value is at most r times the optimal value of P.*

Observation : An *r-approximation heuristic* is also an *r-approximation algorithm.*

The usage of such an *heuristic* may seem counter-intuitive, but this technique was created for Multicast Congestion, where an *approximation heuristic* was originally used for solving multiple Steiner Trees, this cannot be done using *randomized rounding.* For mechanism design, Lavi and Swamy rely on *randomized metarounding*'s ability to keep properties and keep truthfulness after applying an *approximation heuristic*, we simplify the explanation focusing for such a case.

We want to find the convex decomposition such that $rx^* \geq \sum_j \lambda_j x^j$ where x^j is an extreme point of P for all j and x^* is the optimal value of P the linear relaxation of IP. Ultimately, we would like to have an equality such that $rx^* = \sum_{\mathrm{i}} \lambda_{\mathrm{i}} x^j$, we will show that such an equality would keep properties such as truthfulness, however, we have to show how to obtain the inequality first.

The overall requirements for randomized decomposition are a linear programming relaxation P of an integer problem with positive objective function and an *r-approximation heuristic* A as well as a feasible solution $x*$ to P. We list the extreme points of P as $(x^j | j \in J)$. We would like to solve the following linear optimization problem.

$$\begin{aligned} \max \quad & \sum_{j \in J} \lambda_j \\ \text{s.t} \quad & \\ & \sum_{j \in J} \lambda_j x^j \leq rx* \\ & \sum_{j \in J} \lambda_j \leq 1 \\ & \lambda_j \geq 0 \end{aligned} \tag{3.4}$$

The solution to this optimization problem gives a convex decomposition $\lambda*$ where $rx* \geq \lambda_{\mathrm{i}} x^j$ holds. However, (3.4) has an exponential number of variables, since there is an exponential number of λ_j. To solve this issue Carr and Vempala solve the dual problem:

$$
\begin{aligned}
\min \quad & rx * \cdot w + z \\
\text{s.t} \quad & \\
& x^j \cdot w + z \geq 1 \ , \forall j \in J \\
& w \geq 0 \\
& z \geq 0
\end{aligned}
\tag{3.5}
$$

Although there is an exponential number of constraints in (3.5), the ellipsoid method can solve the problem in polynomial time through the use of the approximation heuristic A as a separation oracle. This way, we have a convex decomposition such that $rx* \geq \sum_j \lambda_j x^j$ in polynomial time.

Lavi and Swamy elaborated a way to obtain an r-approximated truthful in expectation mechanism from an r-approximation heuristic. The method is based on the usage of *randomized metarounding*. Take a combinatorial auction problem (CAP) as shown in problem 3.6. It is a known result that the solution of this problem when the objetive is social welfare maximization gives a truthful mechanism, however, such a problem is NP-hard [55]:

$$
\begin{aligned}
\max \quad & \sum_{\mathrm{i}, S \neq \phi} v_{\mathrm{i}}(S) x_{\mathrm{i},S} \\
\text{s.t} \quad & \\
& \sum_{S \neq \phi} x_{\mathrm{i},S} \leq 1 \quad \forall \mathrm{i} \\
& \sum_{\mathrm{i}} \sum_{S : j \in S} x_{\mathrm{i},S} \leq 1 \quad \forall j \\
& x_{\mathrm{i},S} \in \{0,1\} \quad \forall \mathrm{i}, S
\end{aligned}
\tag{3.6}
$$

Here, $x_{\mathrm{i},S} = 1$ if player i receives item set S. Let M^F a relaxation of the mechanism represented in problem 3.6 where the variable $x_{\mathrm{i},S} \in \{0,1\}$ is relaxed to $x_{\mathrm{i},S} \geq 0$. Let x^* the optimal solution of M^F, where p^F are the payments using VCG, this solution holds truthfulness. Using *randomized metarounding* it is posible to obtain a convex decomposition such that $rx^* \geq \lambda_{\mathrm{i}} x^j$, the crucial improvement made by Lavi and Swamy is to modify the decomposition to get a result such that $rx^* = \lambda_{\mathrm{i}} x^j$ [55]. This decomposition preserves the structure of the solution, such that the decomposition is equal to a fractional solution, where payments are given by $\frac{p^F}{r}$, which allows the decomposition to keep truthfulness. Naturally, the value given by the solution is $\frac{x^*}{r}$.

For this exact decomposition there are two requirements, one is to have a quasi-linear objective function, the second is for the problem solutions to have the *packaging property*: let $\mathcal{Z} = \{x^l\}_{l \in \mathcal{I}}$ where $\mathcal{I}$ is the set of integer solutions for the problem relaxation P, if $x \in \mathcal{Z}$ and $y \leq x$ then $y \in \mathcal{Z}$ [55]. It is easy to see that CAP has this property.

We do not dig further into how the mechanism that uses this decomposition, which is a *random mechanism* is transformed to a *deterministic mechanism*, such explanation can

be found in the work of Lavi and Swamy [55]. The result of the exact decomposition and later transform into a deterministic mechanism gives an r-approximation mechanism that is truthful in expectation.

3.3 Healthcare data and prioritization

3.3.1 Data processing technology in chilean healthcare system

Chilean patients data is registered in a Electronic Health Record (EHR), "Ficha Clinica Electrónica" (FCE) in Chile. EHR's contain around 60% of a patient's data [48]. The health record was regulated in 2012, however, the electronic document FCE is not obligatory and is not regulated [60]. The SIDRA project, started in 2008 was given the task of implementing the FCE in 2012. However, the project failed in the task only digitalizing independent tasks [60].A unique register of the waiting list was announced in 2018. The register was meant to be accesible through a webpage, with the register of the patients without a deadline, this announce was not fulfilled [60].

The lack of an EHR national integration on top of system decentralization with poor interoperability make data analysis hard. Not only different institutions use different codes, but they have different protocols. A review on Chilean healthcare data quality (in spanish) can be found in Lobos and Olivares [60]. The main drawback of the data comes from the presence of free text input and institution specific protocols. An example of this last issue is the optional registration of ambulatory surgeries until 2017 [60].

3.3.2 Problems in chilean operating rooms

From an abstract point of view, operating rooms inner working is simple. There are different kind of resources such as sanitized instruments and items (i.e facemasks) and others more related to the treatment itself such as a hip replacement. Human resources is composed of surgeons, nurses, anesthesiologist and auxiliary personnel, these last ones usually perform a role similar to what nurses do and are referred as TENS (Superior Education Nursery Technician), they have 3 years of higher education [11]. Although surgery requires at least two surgeons, a main surgeon and a medical student is enough to suffice the quorum. There are usually many surgery rooms, and some of these are reserved or have characteristics for special cases such as labor rooms. While there are also separated operating rooms for urgency, these cases are above the scheduled less urgent ones and may take any room available. In the context of our problem, we focus on the patients scheduled from the waiting list with an exact appointment date.

Excesive waiting times are an issue on many countries [84]. Chilean high complexity hospitals are self managed, which means that the way they do the logistics and resource management vary. This variability in managing policies is reflected in the waiting times, with an average of 330 days and a maximum difference of 600 days over all public high complexity hospitals in the country [11]. These results are not tied to the waiting list size, an illustration of this is that there are two 500 days average waiting time facilities with 2000 and 15000 patients in the waiting list [11]. There is a subset of patients that have specific attention deadlines, these are usually prioritized since the state pays additionally

for them, these are the patients under a GES (Explicit Health Guarantee) condition. The National Health Service (NHS) from the United Kingdom has a set deadline of 18 weeks for all non-special cases (i.e cancer) after which the centre has to solve the situation by sending the patient to another centre with an incentive, their fulfillment of this policy is of 86%. Catalunya set three possibilities for non-special cases: 90, 180 and 365 days [11]. The consequences of the lack of such policies are shown later.

Another topic that helps to increase the size of the waiting lists in Chile is the lack of ambulatorization, surgeries without bed stay. In Chile, the percentage of ambulatory surgeries is under 30% while the average on developed countries: Canada, USA, Norway and Denmark is around 60% [11]. Low surgery room usage is an additional factor: out of 9 hours, the average usage of a room is 4.8 hours which pales in comparison to the UK, where the average use is 6.4 hours out of 9. Another common issue in surgery rooms is start and end time, the first surgery starts with an average of 40 minutes of delay and the last one ends on average 2 hours before the closing time, while discussion on the causes may vary, the fact that most surgeons prefer not to attend people near the closing time is affected by the uncertainty on end time which affects their schedules outside of surgery rooms. While some hospitals reach an average under 10 minutes in delay, which is the UK benchmark, many others present a bad performance. These problems conclude in a 3.3 average surgery per day in Chile against 5.1 in UK.

Lack of staff is usually pointed at as the cause of problems in Chilean healthcare. However, before the COVID, logistics of staff generated could have fixed this problem on surgery rooms, resources such a anesthetists are not enough on 19% of the hospitals, on the other hand, issues with absenteeism with nurses and TENS required better working conditions to be fixed. At the time, sudden personnel absence as well as logistic issues with material resources produced sudden surgery cancellations.

Yardstick competition is the base theory for the Diagnosis Related Group (DRG) payment system [23, 83]. This system procures to diminish the variability among practices by defining bundles with the same price, based on the average price among institutions. Hospitals which save costs earn more by spending less than the bundle payment, the opposite happens for the less efficient ones [23]. This system has been implemented in many zones such as Europe, Scandinavia, United States, Canada, Australia and others [61, 64]. Problems with DRG are mostly due to collusion and miss-reports [64]. In Chile, the DRG codes have been used in the FCE, however, its usage as payment system was approved in 2020 [30].

Most issues mentioned in this section are not worked in this thesis, while we work on incentives with schedules to improve room usage, we focus on other issues for which it is necessary to describe the inner working of patients selection and scheduling. We could point that most surgeries are scheduled in the following way:

1. There are various clinical services with different specialties.
2. Each team has certain time blocks assigned.
3. Each team decides which patients from the waiting or hospitalized list to schedule in their blocks and which surgeon treats the patient.

.

The extremely high number of patients in the waiting list with the addition of communication problems between system-patient and hospitals-APS suggest that surgeons cannot analyze the whole list of patients with enough scrutiny. In fact, **prioritization given risk factors cannot be done** since diagnosis are not registered on the no GES list.

At least 40% of the surgery waiting list without deadline (no GES) are cases of low complexity. . Moreover, an alarming 85% of the patients that entered by the emergency door were on one of the waiting lists [11]. Studies show that 11 million Chileans have more than one comorbidity [11]. Although correlation cannot be proven explicitly, it is feasible that waiting for attendance provoked more comorbidities until the patients life is on serious risk. Given that 55% of the surgeries entered as an emergency case, even programming surgeries gets hard and ultimately concludes on a high amount of cancellations.

In this work, we focus on the issue stated in the last paragraph. Given that there is no current possibility of prioritization in the no deadlines (no-GES) waiting list, and that even with the registered diagnosis there is no enough information for a perfect definition of risk and urgency, moreover, urgency of attendance and risk has no correlation currently as we will show in Chapter 5. We use the surgeons opinion on the case as a value of risk and urgency, however, we also keep waiting time and national quota of cases fulfillment as priorities of the central planner. Using these factors, we look for a solution that incentivizes opportune attention while considering the doctors opinion.

3.3.3 Prioritization

A key aspect of any healthcare related system is prioritization. Protocols have defined age, diseases, socio-economic factors that define which patient should be attended first. For elective surgery, key factors of a prioritization system are equity, transparency, certainty and validation [25]. While the other factors are related to the implementation of the system, equity is a value that should define a priority, a score. Scoring systems are used in Canada and New Zealand, while in Australia every institution decides its system (like Chile). Testi et al. compare score and clinical assessment prioritization methods with positive results for the scoring system [87]. The main issues with score based prioritization are scientific validity, lack of provable guarantees for fairness and transparency [25].

Mullen reviews different methods for prioritization in the literature [67]. While the easiest to implement system consists of deadlines, better (but harder) ways of prioritization exist. As we point out in Section 3.3, condition deterioration due to waiting time is one of the main causes of cost increase, waiting times and other public health issues. Gudex et al. proposed in 1990 to prioritize the waiting lists by expected health benefit [42]. Besides Edwards and Barlow attempt to measure QALY by using a computer simulation, no other attempts on this method have been seen by the authors [34]. We argue that current technology and data may improve results on this task. In Chapter 5 we show negative results on producing a QALY index based on the urgency index that Chilean healthcare assigns to patients. Experience with diagnosis based prioritization has given positive results in Chile [32].

Chapter 4

Patient oriented operating room scheduling

Although operating room scheduling is a well known problem with many variants, as we saw in Section 3.1, the problem itself is not centered on the patients selected. Given the Chilean context, the possibility of adding prioritization to patients is an issue of high importance. Moreover, in Section 3.3 we pointed that there is a clear preference on using the operating room at certain hours. In this chapter we formalize our version of operating room scheduling and proof an approximation solution to it, we show that we can use such approximation on the mechanism version of the problem.

In Section 4.1 we describe the original operating room scheduling problem and discuss how we choose the variables important to our version. In Section 4.2 we recreate the solution to the scheduling problem given by Bar-Yehuda et al. [8]. Section 4.3 gives a fine grained analysis of this algorithm. Finally, Section 4.4 shows a $2t$-approximation and express how it can be used to solve the mechanism with a truthful in expectation guarantee.

4.1 Modeling the problem

In Chapter 1 and Section 3.3 we declared reasons to solve the resource allocation using the surgeons valuations on urgency of attendance of patients. We require a mechanism that optimizes the social welfare, where the values in the objective are the valuations of surgeons give to attending patients. However, to promote equality of attention, this mechanism has to incentivize surgeons to pick certain patients that have been waiting longer, which we include as a central planner that benefits from surgeons choosing certain patients, this benefit function is explored in Chapter 5, in this chapter we assume such function is quasilinear.

Although the Operating Room Scheduling Problem (ORSP) has many variations, research is focused on costs or avoiding deadlines [90]. ORSP in practice usually includes a high number of parameters, however, while preferences are mentioned in Zhu et al. as a parameter, no model in the reviewed literature includes them [90, 2]. Moreover, as seen in Section 3.3, it is common for doctors to avoid performing surgery at certain hours, a strong

assumption in most ORSP variations is that surgeons will operate at any hour [90, 2]. We include this factor in the valuations, since it is recommended to use hour preferences as incentive in the Chilean setting [11].

We model this problem as a combinatorial auction, where surgeons choose bundles of triplets (patient, date, hour) and the algorithm solving the mechanism outputs a valid assignation of bundles to surgeons, this assignation is also called the Winner Determination Problem [69]. One of the common ways to solve combinatorial auctions is to transform them into iterative auctions, where instead of bundles there are several rounds where agents pick only one item as studied by Liu et al. [69, 58]. As most surgeons want similar times to operate, there is a potentially high number of iterations to obtain an schedule and it is possible that surgeons get a high time difference between operations (such as one in the morning and one at night).

In Section 3.2.1 we showed a technique developed by Lavi and Swamy that transforms an α-approximation heuristic into an α-aproximation mechanism [55]. Recall, as seen in Section 3.1, that an α-approximation algorithm regarding the solution to the linear programming relaxation of the problem. Thus, we only require to solve the optimization problem under quasilinear valuations and show the integrality gap of the solution.

In the following part we describe the optimization problem in detail. Let A the set of integers corresponding to agents (doctors), P set of integers corresponding to patients and $T \times D$ tuple of time and days, both sets, were time is discretized by some value (assume we only care about minutes). We denote an element of $T \times D$ as a block. Finally, we denote R as the set of integers representing operating rooms. Each agent A has some preference over patients and over blocks, formally this is represented as a function $v : A \times P \times T \times D \to \mathbb{R}$. Suppose that the benefit of the central planner when patient p is attended is c_p, we count this as an additional agent c in A where its value $v(a, p, t, \mathrm{d}) = v(a', p, t', \mathrm{d}'), \forall a, a' \in A, p \in P, t, t' \in T, \mathrm{d}, \mathrm{d}' \in D$. A strong assumption we make is that the valuation of the planner for attending a patient is the same over each day in D, since we plan to run this algorithm weekly, this means that this value does not change over the week. Let $S_a \subseteq \bigcup_{(p,t,\mathrm{d},r) \in P \times T \times D \times R} \{(a, p, t, \mathrm{d}, r)\}$, set of configurations of patients and times for an agent a, we denote $x(S_a) = 1$ if the configuration is chosen in the solution and 0 otherwise. The problem, denoted P_O, consists of maximizing the social benefit $\sum_{a \in A} \sum_{s \in \bigcup_{S_a}} x(s) v(a, s)$, where $v(a, s)$ is the valuation of the configuration s for agent a independent of the room (we use notation overloading with v). Additionally, we have the following constraints:

1. An agent cannot attend two patients at the same time.
2. An agent cannot be on two rooms at the same time.
3. A room cannot have more than one surgery being performed at the same time.

One could imagine more restrictions, however, we make some assumptions regarding the behaviour of the agents: Agents will not bid for schedules outside of the functioning hours of the surgery rooms. Most importantly, agents that bid for two self conflicting bundles cannot receive the conflicting bundles. This is a key assumption that greatly reduces the complexity of the problem, otherwise, we would be including the $\vee$ operation in the bidding language [13, 69]. At last, patients in Chile are assigned a doctor in the waiting list, which means that

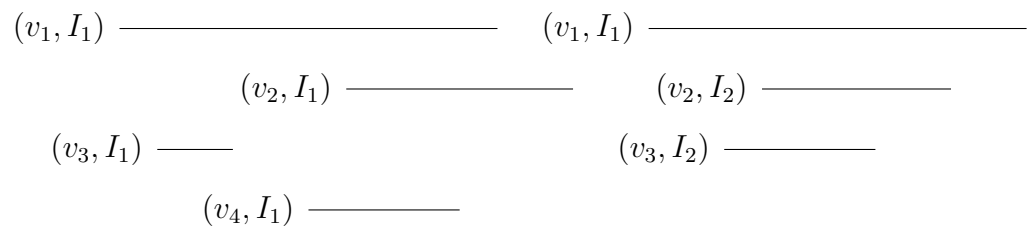

Figure 4.1: A 2-interval graph as segments where every segment represents a task from a job. Segments that have the same horizontal coordinates (time) such as (v_1, I_1) and (v_2, I_1) are segments that intersect each other.

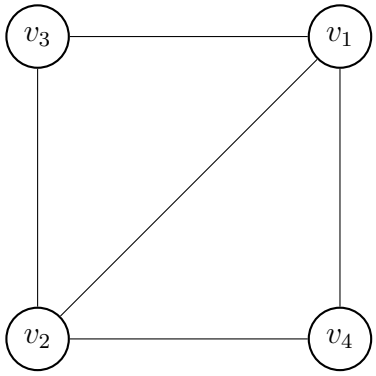

Figure 4.2: A 2-interval graph. Representing the same segments of Figure 4.1, intersecting vertices such as v_1 and v_2 share an edge.

no patient can be picked by a different doctor.

We focus on the similarity between P_O and t-Interval Scheduling. The mentioned problem consists of scheduling n non-intersecting jobs, where each job J_j is associated with an interval I_j which consists of up to t non-intersecting segments in the real line, $t \geq 1$ and weight w_j [8]. Two jobs are in conflict if their intervals intersect. The objective is to schedule a subset of non-conflicting jobs of maximum weight [8]. This problem is equivalent to finding an independent set on the corresponding t-interval graph [8].

Definition 4.1 (t-Interval Graph) *A graph where each vertex corresponds to a an interval that has been split into t parts, or segments. Vertices u and v are adjacent if and only if a segment from u intersects a segment from v.*

An example of t-interval graphs can be seen in Figure 4.2, which is the corresponding interval graph of the intervals shown in Figure 4.1.

A Maximum Weight Independent Set(MWIS) on t-interval graphs is a solution to t-Interval Scheduling [8].

Lemma 4.1.1 P_O is equivalent to MWIS on t-interval graphs, for $|R| = 1$.

Proof.
$P_O \leq_p$ MWIS:
For every bundle s bade by an agent $a \in A$, we create a vertex corresponding to a job J with weight $v(a, s)$. Each tuple $(a, p, t, \mathrm{d}, r) \in s$ corresponds to a segment where the initial and finish time is given by $[t, t + p_t]$, where $p_t \in \mathbb{N}$ corresponds to the time it takes to attend patient p. For every pair of vertices v, u, they are adjacent if and only if there is a segment in v that intersects a segment in u. The obtained graph G corresponds to a t-interval graph. The MWIS over G corresponds to maximizing the objective function of P_O $\sum_{a \in A} \sum_{s \in \bigcup_{S_a}} x(s)v(a, s)$. The independent set implies that no job intersects each other, moreover, since $|R| = 1$ there cannot be two patients in the same room.
MWIS $\leq_p$ P_O:
Given $G = (V, E)$ t-interval graph. For every vertex create a job and agent, assign the weight of the vertex to the job. For every edge $(u, v) \in E$ add a segment to the jobs representing u and v, both segments start and end between $[k, k+1]$, where $k \in N$, we can assign and update k iteratively such that no three segments intersect each other, but there is an intersection for every edge. Solving P_O on this graph gives a set of non intersecting jobs, which are the vertices in the original graph, such that the weight is maximized. □

Corollary 4.1.1.1 P_O is NP-complete and APX-hard.

MWIS on t-interval graphs is APX-hard and NP-complete for $t \geq 2$ [8]. Moreover, since t-union graphs are a subclass of t-interval graphs and MWIS on t-union graphs is equivalent to t-DIMENSIONAL MATCHING, the problem cannot be approximated within $\mathcal{O}(t/\log t)$ (unless P=NP) [9].

Fortunately, a $2t$-approximation algorithm is given by Bar-Yehuda et al. [8]. Although the main result is the usage of the Local Ratio technique, we review its alternative demonstration based on multicoloring [7, 8, 44].

4.2 MWIS in t-interval graphs

The following section contains a redo of the proof given by Bar-Yehuda et al. on a $2t$-approximation for the MWIS in t-interval graphs, we focus on the multicoloring version [8]. Later, in Section 4.4 we build on this result for scheduling t-intervals on multiple identical machines with flexible jobs, where jobs can be separated on different machines as long as all segments are scheduled without conflicts.

We first give an overview of the proof. Given P a relaxed version of the problem and x a feasible solution to P, $x(v) \in [0, 1]$ the value of vertex v in x. We show that the sum of the neighbors values $\sum_{u \in N(v)} x(u)$ is bounded by $2t$. This works for any feasible solution, thus, it is possible to use it repetitively on every vertex, such that the sum of all neighbors is always bounded by $2t$ for any vertex. This result allows to pick any vertex and their neighbors and map their $x(v)$ values to a real line with values in $[0, 2t]$. We do this such that no vertex uses the same mapped values as their neighbors, even if the mapping of a vertex ends up in a non-contiguous segment. We partition the line on $2t$ parts such that no part contains the mapping of two adjacent vertices, this means that all vertices mapped to a partition form an

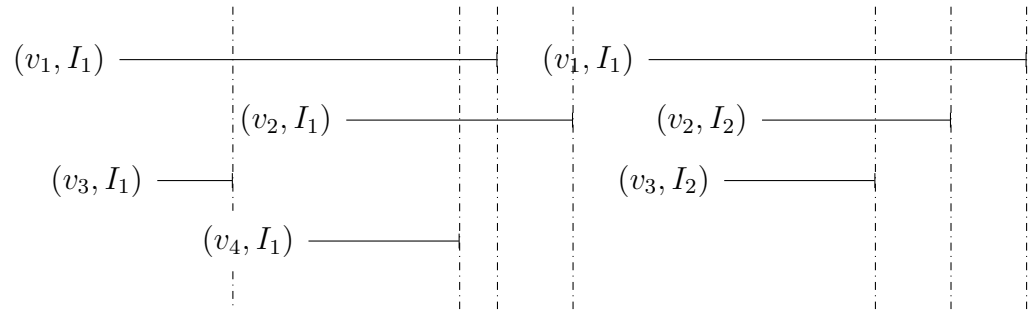

Figure 4.3: Jobs represented by segments where the dotted line corresponds to the right endpoint. By only scanning the intersections with the right endpoint it is possible to know all conflicts between jobs.

independent set. Since every vertex with $x(v) \geq 0$ is mapped to a partition, the sum of all partitions gives the same value as $\sum_{v \in V} x(v)w(v)$, such that there is at least one partition with value $\frac{x}{2t}$ or $\frac{x^*}{2t}$ when x^* is the optimal solution.

Let $G = (V; E)$ a t-interval graph. Were each $v \in V$ of weight $w(v) : V \to \mathbb{R}$ is mapped to a set of at most t non-intersecting segments over the real line. We denote $N(v)$ as the set of neighbors of v. The corresponding integer programming problem is shown in t-MWIS.

$$\begin{aligned} \max \quad & \sum_{v \in V} x(v) \cdot w(v) \quad (t\text{-MWIS}) \\ \text{s.t} \quad & \\ & \text{for each clique } \mathcal{C} \in G: \quad \sum_{v \in \mathcal{C}} x(v) \leq 1 \\ & \forall v \in V: \quad x(v) \in \{0, 1\} \end{aligned}$$

We relax the variable in t-MWIS such that $\forall v \in V, x(v) \in [0, 1]$, we will refer to this relaxation when using $x(v)$ in what remains of the proof. We denote the relaxed problem as P

We use the notation (v, I) as equivalent to $I \in v$, where v represents the job to which segment I belongs. We denote the segments intersected by the right endpoint of I (finish time) as $R(I)$. Notice that for each segment I, the segments touched by its right side form a clique as seen in Figure 4.3. We define a clique $\mathcal{C}$ is an *interval clique* if for every vertex $v \in \mathcal{C}$, there exists a segment $I \in v$ such that $(v, I) \cap \{(u, J) | u \in \mathcal{C}\} \neq \emptyset$. All interval cliques can be defined by the right endpoints of segments. Therefore, given that there is a polynomial number of constraints, we can solve the relaxed problem P in polynomial time.

The original proof of Bar-Yehuda et al. uses $x(v, I)$ as the notation for the relaxation of the indicator of I, we use $y(v, I)$ for this task. In other words $y(v, I) \in [0, 1]$ corresponds to the relaxation of an integer value that is 1 when segment $I \in v$ is picked and 0 otherwise. Since we only need to use the interval cliques to define all cliques, we can rewrite t-MWIS as shown in 4.1, for the rest of the proof we will refer to this problem as P.

$$
\begin{aligned}
\max \quad & \sum_{v \in V} x(v) \cdot w(v) \\
\text{s.t} \quad & \\
& \text{for each clique } \mathcal{C} \in G: \quad \sum_{(v,I) \in \mathcal{C}(I)} y(v,I) \leq 1 \\
& \forall v \in V, \forall I \in v : y(v,I) - x(v) \geq 0 \\
& \forall v \in V, \forall I \in v : \quad x(v), y(v,I) \geq 0
\end{aligned} \tag{4.1}
$$

Since $x(v)$ is only in the objective function, $y(v,I) = x(v), \forall v \in V$.

Lemma 4.2.1 Let x be a feasible solution to P. Then, there exists a vertex $v \in V$ satisfying:

$$\sum_{u \in N[v]} x(u) \leq 2t$$

Proof. We use the *weighted* averaging argument. Given the sum $\sum_{v \in V} \sum_{u \in N[v]} x(v)x(u)$, an upper bound can be obtained considering that, if (u, J) is a neighbor of (v, I) the the product of their values are added twice, thus, we only require to sum the product $y(u,j)y(v,I)$ given $(u,j) \in R(I)$(the right endpoint of I) and multiply by 2. Since $y(u,J) = x(u)$, we have that:

$$\sum_{v \in V} \sum_{u \in N[v]} x(v)x(u) \leq 2 \sum_{v \in V} \sum_{(u,J) \in R(I)} x(v)x(u)$$
$$\sum_{v \in V} \sum_{u \in N[v]} x(v)x(u) \leq 2 \sum_{v \in V} x(v) \sum_{(u,J) \in R(I)} x(u)$$

Since $R(I)$ defines an interval clique, $\sum_{(u,J) \in R(I)} x(u) \leq 1$.

$$\sum_{v \in V} \sum_{u \in N[v]} x(v)x(u) \leq 2 \sum_{v \in V} x(v) \sum_{I \in v} = 2t \sum_{v \in V} x(v)$$
$$\sum_{v \in V} x(v) \sum_{u \in N[v]} x(u) \leq 2t \sum_{v \in V} x(v)$$

Thus $\exists v \in V$ such that:

$$\sum_{u \in N[v]} x(u) \leq 2t$$

□

Definition 4.2 (Multicoloring) *Given graph $G = (V, E)$, where each vertex $v \in V$ has a color requirement $x(v) \in \mathbb{N}$. A multicoloring is an assignment of colors to vertices such that adjacent vertices cannot receive the same color, and the colors assigned to a vertex must satisfy the color requirement [44].*

We use an easier way to visualize the multicoloring, however, note that a large integer constant is not necessary as seen in the original demonstration. We multiply each $x(v)$ by a constant $L \in \mathbb{N}$, such that $\forall v \in V, x(v) \cdot L \in \mathbb{N}$. For every vertex v, set the color requirement to $x(v) \cdot L$. By using Lemma 4.2.1 on every vertex, we can map the vertex and its neighbors to the real line $[1, L(2t+1)]$, where the extra L is the space to allocate the vertex. It suffices to allocate every vertex to the smallest available values in the line without overlapping a vertex with its neighbors, this is possible since the sum of the values of a vertex neighbors is bounded by $2t$ ($2tL$ here). Notice that, if two or more vertices are neighbors their sum is at most 1, given this issue, we can ensure that, if we partition the $[1, L(2t+1)]$ in $2t$ parts, the neighboring vertices do not belong to the same part, thus, the vertices of each part form an independent set. Let the size of each part be $[z_{\mathrm{i}}, z_{\mathrm{i}+1})$, $\mathrm{i} = 1, \ldots, 2t+1$.

Consider the sets $S_{\mathrm{i}} = \{v \in V | x_{\mathrm{i}} \in \psi(v)\}$, where $x_{\mathrm{i}} \in [1, L(2t+1)]$. Every set is an independent set.

Theorem 4.2.2 $\max\{S_1, \ldots, S_{2t+1}\}$ is a $2t$-approximate independent set.

Proof. The amount of colors to which $x(v)$ is mapped is $x(v) \cdot L$. Additionally, notice that $\sum_{S_{\mathrm{i}} \ni v}(z_{\mathrm{i}} - z_{\mathrm{i}+1}) = x(v) \cdot L$.

$$\sum_{v \in V} x(v)w(v) = \sum_{v \in V} w(v) \sum_{S_{\mathrm{i}} \ni v} \frac{(z_{\mathrm{i}+1} - z_{\mathrm{i}})}{L} = \frac{1}{L} \sum_{S_{\mathrm{i}}} (z_{\mathrm{i}+1} - z_{\mathrm{i}}) \sum_{v \in S_{\mathrm{i}}} w(v)$$
$$= \frac{1}{L} \sum_{\mathrm{i}=1}^{2t+1} (z_{\mathrm{i}+1} - z_{\mathrm{i}}) w(S_{\mathrm{i}}) \leq \frac{1}{L} \sum_{\mathrm{i}=1}^{2t+1} (z_{\mathrm{i}+1} - z_{\mathrm{i}}) w(\mathcal{I}) = 2tw(\mathcal{I})$$

□

4.3 Additional results on MWIS over t-interval graphs

Section 4.2 is a longer explanation of the proof given in Bar-Yehuda et al [8]. However, we noticed that the algorithm has a better approximation conditioned on another parameter.

Corollary 4.3.0.1 The algorithm analysis can be improved to a τ-approximation. Where $\tau = \max_C |C|$ the maximum size of the cliques where vertices have $x(v) > 0$.

Proof. Notice that, if all neighbors of a vertex v cover the real line, then, there is a clique of size $2t$, otherwise, two vertices could be mapped to the same space. Moreover, notice that for this clique each interval has a clique with different neighboring intervals, otherwise, if two or more have the same neighbors they would sum 1. Therefore, the number of total space in the real line necessary to map all vertices is bounded by $\tau = \max_{\{C\}} \sum_{\{u \in C\}} x(u) \leq \max_C |C|$.

Notice that this also applies to the Local Ratio version of the algorithm in the original paper [8]. □

Given that we take the highest weight set S_{i}, an independent set, a good question is if we can find a subroutine that maps the vertices in such a way that S_{i} is the highest possible. Clearly, if for every time we take a vertex and its neighbors, we map them in all the orders possible, one of the final independent sets will be the optimal value, this does not give an FPT algorithm since the runtime depends on the amount and size of cliques, moreover, Fellows et al. showed that MWIS is W[1]-complete for 2-interval graphs [36]. However, following the same idea, randomizing the order of assignation may give a better approximation algorithm.

4.4 *t*-Interval Flexible Scheduling in Identical Machines

We focus on a general version of the problem in Section 4.2, here we may assign the segments of each job over multiple identical machines. We have n jobs J_j associated with at most t non-intersecting segments in the real line, $t \geq 1$ and weight w_j. The weight is only obtained when all segments of the job are scheduled. The jobs most be scheduled on m identical machines, where no machine can have conflicting jobs, jobs whose intervals intersect. We refer to this problem as the t-Interval Flexible Parallel Scheduling (t-IFPS)

We give an overview of the proof. We follow a similar sketch to the proof in Section 4.2, we solve the relaxation of a version of the problem where multiple machines are considered. We give a bound over the sum of the indicator variables corresponding to the neighbors of all the segments of a vertex on the same machine. Using the bound, we map the vertices to a real line with a size given by such value, we do it in a way that avoid intersecting segments to be mapped to the same colors if they are assigned to the same machine in the relaxation solution. After mapping the values, we have a number of independent sets from which we have to pick the ones with the largest weight, we prove that this weight is bounded by $2t$.

For explanation purposes, we take the initial optimization problem statement for t-interval scheduling in 4.1 and apply it to m machines. We have the following optimization problem for a graph $G = (V, E)$ in t-IFPS constructed as shown in Section 4.2.

$$
\begin{aligned}
\max \quad & \sum_{v \in V} x(v) w(v) \quad \text{(t-IFPS)} \\
\text{s.t} \quad & \\
& \text{for each clique } \mathcal{C} : \sum_{(u,I) \in \mathcal{C}(I)} y(u, I) \leq m \\
& \forall v \in V, \forall I \in v : y(v, I) - x(v) \geq 0 \\
& \forall v \in V, \forall I \in v : y(v, I), x(v) \in \{0, 1\}
\end{aligned}
$$

It is important to notice that [t-IFPS] lacks the assignation of segments to machines. This problem can be alternatively represented as a single machine with a determined amount of a resource, and jobs require to consume a resource to run. This version is equivalent to the one shown in [t-IFPS] if the machine has capacity of m units and the jobs require 1 unit of resource. This alternative variation has a $6t$-approximation algorithm given by Bar-Yehuda and Rawitz, and Chakaravarthy et al. give an $8t$-approximation for the version where jobs have variable resources bounded by half of the machine's capacity [9, 22]. Notice that we

can solve our problem by using the $6t$-approximation for this variation and processing the segments as different jobs. This way, we can use the polynomial time algorithm to obtain a $6t$ solution. In this thesis we improve on this result.

Allocation of segments on different machines should be allowed, this is not possible in our current graph G nor optimization problem. We construct a new graph, let $G' = (V', E')$ a graph where for every job J_j represented by v in G, we include all segments $(v, I) \in V'$. Moreover, $((v, I), (u, J)) \in E'$ if and only if segments I and J intersect.

We also require a variable to show that a job is assigned to a machine. Let $r(v, I, j) \in [0, 1]$, with $v \in V, I \in v, j \in [m]$, this variable is a relaxed version of an indicator that is 1 when segment I of job represented by v is assigned to machine j and 0 otherwise. With this new variable, we formalize the problem as the following linear program which, for the rest of the proof we will refer as P.

$$
\begin{aligned}
\max \quad & \sum_{v \in V} x(v)w(v) \quad \text{(P)} \\
\text{s.t} \quad & \\
\forall \mathcal{C} \in G', \forall j \in [m] : \quad & \sum_{(u,J) \in C(v,I)} r(u, J, j) \leq 1 \\
\forall (v, I) \in V' : \quad & \sum_{j \in [m]} r(v, I, j) \leq 1 \\
\forall (v, I) \in V' : \quad & \sum_{j \in [m]} r(v, I, j) - y(v, I) \geq 0 \\
\forall (v, I) \in V' : \quad & y(v, I) - x(v) \geq 0 \\
\forall (v, I) \in V', \forall j \in [m] : \quad & 0 \leq x(v), y(v, I), r(v, I, j) \leq 1
\end{aligned}
$$

We use the same approach as the single machine solution by using multicoloring. We obtain an upper bound on the value of the neighbors and map the vertices on the real line. We adapt the lemma made by Bar-Yehuda et al. to the parallel and flexible case.

Corollary 4.4.0.1 Let x be a feasible solution to P. Then, there is at least one vertex $v \in V$ satisfying:

$$
\sum_{(u,J) \in \bigcup\limits_{I \in v} \{N(v,I)\}} r(u, I, j) \leq 2t \tag{4.2}
$$

Proof. We follow the same scheme as Lemma 4.2.1 using *weighted averaging*.

$$\sum_{v\in V}\sum_{I\in v}\sum_{(u,J)\in N(v,I)} y(u,J)y(v,I) \leq 2\sum_{v\in V}\sum_{I\in v}\sum_{(u,J)\in R(v,I)} y(u,J)y(v,I)$$

$$\sum_{v\in V}\sum_{I\in v}\sum_{(u,J)\in N(v,I)} y(u,J)y(v,I) \leq 2\sum_{v\in V}\sum_{I\in v} y(v,I)\sum_{(u,J)\in R(v,I)}\sum_{j\in[m]} r(u,J,j)$$

$$\sum_{v\in V}\sum_{I\in v} x(v)\sum_{(u,J)\in N(v,I)}\sum_{j\in[m]} r(u,J,j) \leq 2\sum_{v\in V} x(v)\sum_{I\in v}\sum_{j\in[m]}\sum_{(u,J)\in R(v,I)} r(u,J,j)$$

$$\sum_{j\in[m]}\sum_{v\in V} x(v)\sum_{I\in v}\sum_{(u,J)\in N(v,I)} r(u,J,j) \leq 2\sum_{j\in[m]}\sum_{v\in V} x(v)\sum_{I\in v}\sum_{(u,J)\in R(v,I)} r(u,J,j)$$

It follows that there is a vertex $v \in V$ and $j \in J$ such that.

$$\sum_{I\in v}\sum_{(u,J)\in N(v,I)} r(u,J,j) \leq 2\sum_{I\in v}\sum_{(u,J)\in R(v,I)} r(u,J,j)$$

$$\sum_{I\in v}\sum_{(u,J)\in N(v,I)} r(u,J,j) \leq 2t$$

$$\sum_{(u,J)\in \bigcup_{I\in v}\{N(v,I)\}} r(u,I,j) \leq \sum_{I\in v}\sum_{(u,J)\in N(v,I)} r(u,J,j) \leq 2t$$

□

We use this bound to set the size of the space in the real line that we require to map the vertices as a multicolor like we did in Section 4.2. However, given a machine j, we only want to map the part of vertex v assigned to j, such that we do not count segments mapped to different machines as neighbors.

To visualize this, we create a graph where only intersecting segments that have non-zero values on the same machine intersect. Let $\bar{G} = (\bar{V}, \bar{E})$ a graph such that $\forall v \in V, \forall j \in [m]$, we take each I such that $r(v, I, j) > 0$ and create a vertex $(v, j) \in \bar{V}$. If two vertices $(u, j), (v, j)$ have an intersecting segment $I \in v$, $J \in u$ and $r(v, I, j), r(u, J, j) > 0$, then there is an edge $((u, j), (v, j)) \in \bar{E}$. Moreover, if $v \in V$, $I \in v$, then for $j, j' \in [m], j \neq j'$, if $r(v, I, j), r(v, I, j') > 0$ then there is an edge $((v, j), (v, j')) \in \bar{E}$. In other words, segments that intersect and have non-zero value in the solution x under the same machine, have an edge on the graph, also when a segment has a value assigned on more than one machine, those segments also have an edge on the graph.

Notice that for every vertex $v \in \bar{V}$, the sum of the values $r(u, J, j)$ of the tuples (u, J, j) that correspond to $N(v) \subseteq \bar{V}$ is bounded by $2t$. With this notion, we can map the values to the multicolor range. We use a space of size $(2t+1)L$ in the real line, and allocate $[1, (2t+1)L]$ to map the vertices. Remember that we require $2tL$ to map the neighbors and the extra L is for the vertex itself. Now, as we did in Section 4.2, we map all vertices to the real line using Corollary 4.4.0.1, such that the neighbors only corresponding to a machine j (like the ones of graph $\bar{G}$) do not intersect each other. We need to assign all values of segments from the same job to the same multicolor, such that, when we pick over the independent sets,

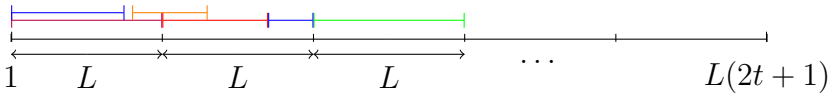

Figure 4.4: A representation of vertices from V mapped to the $[1, L(2t+1)]$ range. This is the same mapping of the algorithm in Section 4.2. Mappings of vertices that intersect such as purple with red, red with blue, blue with orange and green with all the other colors, are mapped to different spaces. As blue intersects with orange by the right, the rest of blue is mapped to another part of the line.

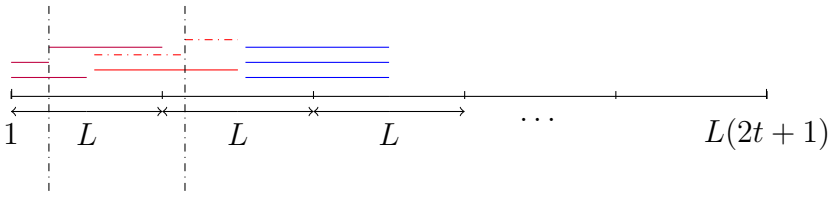

Figure 4.5: A representation of vertices from $\bar{V}$ mapped to the $[1, L(2t+1)]$ range. Here, segments assigned to different machines such as the red dotted segments, are mapped to non-intersecting spaces. This allows a set with intersecting segments that are mapped to different machines to be selected, like the interval between the vertical dotted lines.

all segments of a job are in the same independent set. We can do this in the same way we did in Section 4.2 by mapping the vertices of $\bar{V}$ to the minimum available value. This is similar to partitioning in j values the vertices of V we mapped in the single machine version. Figure 4.4 shows the mapping of V. On the other hand, Figure 4.5 shows the mapping we do with $\bar{V}$ which involves machine assignation. As the dotted red line represents the same tuple assigned to a different machine, this mapping allows different combinations of rooms.

We divide the real line in $2t$ parts $[z_{\mathrm{i}}, z_{\mathrm{i}+1})$, $\mathrm{i} = 1, \ldots, 2t+1$. Let $S_{\mathrm{i}} = \{v \in \bar{V} | x_{\mathrm{i}} \in \psi(v)\}$ where $x_{\mathrm{i}} \in [1, L(2t+1)]$, each S_{i} correspond to a part of the real line. Notice that this time, every set S_{i} is not an independent set in $\bar{V}$, since we may pick vertices (v, j) with the same segments but different machines assigned. We do not worry about this case, since we can use either combination of machine-segment in that case.

Lemma 4.4.1 *$max\{S_1, \ldots, S_{2t+1}\}$ is a $2t$-approximation to t-IPFS*

Proof. We use the same proof as we did for Theorem 4.2.2.

$$\sum_{v\in V} x(v)w(v) = \sum_{v\in V}\sum_{I\in v} y(v,I)\frac{w(v)}{|v|}$$

$$\sum_{v\in V} x(v)w(v) = \sum_{v\in V}\sum_{I\in v} \frac{w(v)}{|v|}\sum_{j\in[m]} r(v,I,j)$$

$$\sum_{v\in V} x(v)w(v) = \sum_{\bar{v}\in\bar{V}}\sum_{I\in\bar{v}} \frac{w(\bar{v})}{|\bar{v}|}\sum_{j\in[m]} r(\bar{v},I,j)$$

$$\sum_{v\in V} x(v)w(v) = \sum_{\bar{v}\in\bar{V}}\sum_{I\in\bar{v}} \frac{w(\bar{v})}{|\bar{v}|}\sum_{j\in[m]}\sum_{S_\mathrm{i}\ni(\bar{v},j)} \frac{(z_\mathrm{i}+1-z_\mathrm{i})}{L}$$

$$\sum_{v\in V} x(v)w(v) = \sum_{S_\mathrm{i}}\sum_{(\bar{v},j)\in S_\mathrm{i}} \frac{w(\bar{v})}{|\bar{v}|}\sum_{I\in v} \frac{(z_\mathrm{i}+1-z_\mathrm{i})}{L}$$

$$\sum_{v\in V} x(v)w(v) = \sum_{S_\mathrm{i}}\sum_{(\bar{v},j)\in S_\mathrm{i}} w(\bar{v})\frac{(z_\mathrm{i}+1-z_\mathrm{i})}{L}$$

$$\sum_{v\in V} x(v)w(v) = \sum_{\mathrm{i}=1}^{2t+1}\sum_{(\bar{v},j)\in S_\mathrm{i}} w(\bar{v})\frac{(z_\mathrm{i}+1-z_\mathrm{i})}{L}$$

$$\sum_{v\in V} x(v)w(v) = \sum_{\mathrm{i}=1}^{2t+1} w(S_\mathrm{i})\frac{(z_\mathrm{i}+1-z_\mathrm{i})}{L}$$

$$\sum_{v\in V} x(v)w(v) \le \sum_{\mathrm{i}=1}^{2t+1} \max\{w(S_k)\}\frac{(z_\mathrm{i}+1-z_\mathrm{i})}{L}, k = 1,\ldots,2t+1$$

$$\sum_{v\in V} x(v)w(v) \le 2t\cdot\max\{w(S_k)\}, k = 1,\ldots,2t+1$$

□

Corollary 4.4.1.1 When restricted to single bids, P_O is equivalent to t-IFPS $|R| \in \mathbb{N}$.

Proof. Analog to Lemma 4.1.1 but R corresponds to the number of machines. □

Theorem 4.4.2 When restricted to single bids, P_O has a $2t$-approximation. Where t is the maximum size bundle.

Proof. Direct from Corollary 4.4.1.1 and Lemma 4.4.1. □

Chapter 5

Simulation

We found an approximate solution to our operating room scheduling problem, however, the ratio in the worst case is not good in a practical setting. We require to test it in the context of chilean surgery rooms in order to test how it performs in a real life scenario. However, there are clear human issues on why testing is not possible. To check how such an algorithm perform we use the data of chilean surgery rooms and waiting lists to model and simulate how picking patients and scheduling would work in a real life setting.

In Section 5.1 we explain our assumptions and implementation on generating data to make the simulation and implementation. Section 5.2 explores the implementation that avoids the issue with the usage of the ellipsoid method and VCG. Finally, we present the results obtained in section 5.3.

5.1 Data and assumptions

We use anonymized data originated from the Ministry of Health (MINSAL). The data is composed from varios sources such as GES (patients with deadlines), NO-GES (no deadlines), DRG, SIRH and others. Data was collected during the 2014-2017 period. Data cleanse was done by removing patients with corrupted data, such as incongruent dates on its procedures or missing data. More information on the specific cases were data was removed is found in [60].

In Chile, during 2014 and 2017 the waiting list registered cases was 36.296.413 with 3.847.250 processed cases. Note that Chilean population is 19.12 millions. The health-care system may register multiple cases for a single procedure, i.e a broken hand may include five fingers registries. The number of individual patients shown in the waiting list during the 2014-2017 period is 10.357.685. Note that it may be the case that a person gives incorrect personal data due to varying reasons [11]. We use the data of 20 out of 62 institutions with reported surgery rooms in the DRG database. The chosen entities have different sizes and socio-economic situations that make them representative of Chilean hospitals. We give these results anynomized.

A main issue pointed in Section 3.3 is that until 2017 institutions were not obligated to

Table 5.1: Description of diagnosis used in the simulations. They compose more than 40% of the surgeries volume.

Diagnosis description
Hernia variations
Calculus variations
Varicose veins of lower limbs without ulceration or inflammation
Esophageal varices without bleeding
Rotator cuff syndrome
Ulcerative colitis
Specified forms of cataracts
Pterygium

register ambulatory surgeries under the DRG database. While there are institutions that already had this policy, we use data from 2017 to ensure anonymity.

Given the diverse amount of diagnosis, we choose the set of diagnosis that composes more than 40% of the total volume of surgeries [11]. These surgeries are mostly ambulatory, the reason why we have to process post 2016 data. Additionally, the Chilean system has shown that the simple high volume surgeries are the ones that clog the system, while complex cases are dealt with efficiency [11]. Chosen diagnosis are specified in Table 5.1.

We would like to compare the implementation of our mechanism against the current procedure. However, it is not possible to do this comparing with the data since there are many factors that would make this comparison unfair:

- Availability of resources at the time.
- Metrics in the situation where only the selected diagnosis are present.
- Availability of beds for non-ambulatory surgeries.

To address this issue, we use statistical learning to obtain an approximation on the surgeon's preferences through an Support Vector Machine (SVM) model. However, individual data of surgeons patient's history do not allow a preference inference. There are many diagnosis with few cases. To perform the simulations, we group surgeons with the same specialty and take them as a single agent. The model reaches a modest 82% accuracy on choosing the patients given the current waiting list. We add this error in the results.

In our mechanism, it is also necessary to obtain the preferences of the agents (surgeons), as a single value. Given the computational possibility of using the most common way, Multinomial Logistic Regression, we use number of cases over total cases to approximate the preference of each agent [41].

For the calculation of patients, we assume a Poisson distribution for every diagnosis used. We determine an initial amount of patients given that under any system implementation there will be patients in the waiting list. On scheduling hour preference for any agent, we use a gaussian distribution over the known preferred times (early and distant from lunch or

end of shift) [11]. The overall working of the simulation is the following:

1. Clean data.
2. Select hospital and separate by specialties.
3. Obtain preferences for agents.
4. Determine each agent's time preference.
5. For every agent, bid for the bundle that maximizes valuation.
6. Run the computational process and assign schedules.

Through data we simulate the preferences and interactions of surgeons. A centralized entity can also add value to certain prioritized patients. In the following section we specify the prioritization made.

In Section 3.3 we set QALY as a possible prioritization method. Given data on urgency of patients, we wanted to determine if this could be a proxy for a QALY measurement. However, we found no correlation between urgency and diagnosis for CIE10 and GRD codes. Figures [5.1,5.2,5.3] show the percentage of diagnosis where the Pearson coefficient between the urgentness index and waiting time has an absolute value over $0.1, 0.3 and 0.5$ respectively. Results show that there is no strong, or even weak correlation in most institutions.

There is no precise quantity under which the Pearson coefficient implies a meaningful correlation [27]. To create a value that represents urgency, we use a softmax function where the maximum value is achieved where an expert suggested deadline is reached. The experiment use a 90 days deadline as suggested for the Chilean system [11].

5.2 Practical issues

Usage of VCG and the ellipsoid method in the work of Lavi and Swamy make our $2t$-approximation mechanism unpractical [55]. Specifically, as pointed in Section 3.2.1, the exponential amount of extreme points make the algorithm slow in practice. Even if the theoretical algorithm is polynomial, it runs the already slow approximation algorithm iteratively until a solution is reached, ignoring the polynomial amount of constraints from the dual.

To address this issue, we must forgo truthfulness. Given that VCG requires an exact computation, and the technique requires calculating a convex decomposition of the extreme points with values equivalent to x^*/α. We avoid the exact calculation of this decomposition. We set an error on the decomposition, which we minimize using Minimum Squared Error (MSE) as objective function. This approximation is similar to how CORE-VCG auctions work, by minimizing the error to the VCG solution, however we do not require a minimum revenue value.

5.3 Results

Our experiments show that the implemented method ensures waiting times similar to *Highest Waiting First*(HWF). The difference between the current method and our algorithm is more explicit in the maximum waiting time.

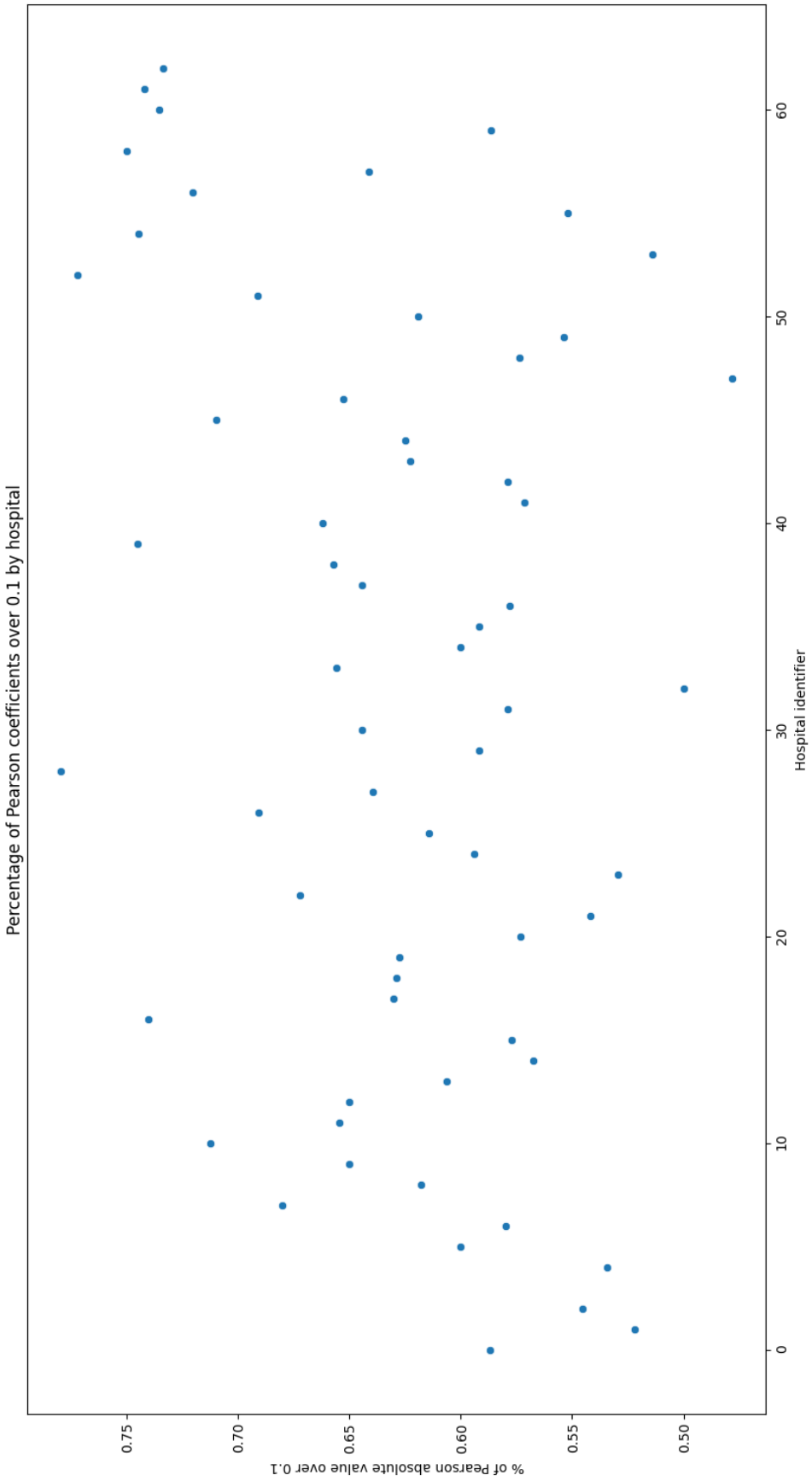

Figure 5.1: Percentage of CIE10 diagnosis that have correlation greater than 0.1 or smaller than -0.1.

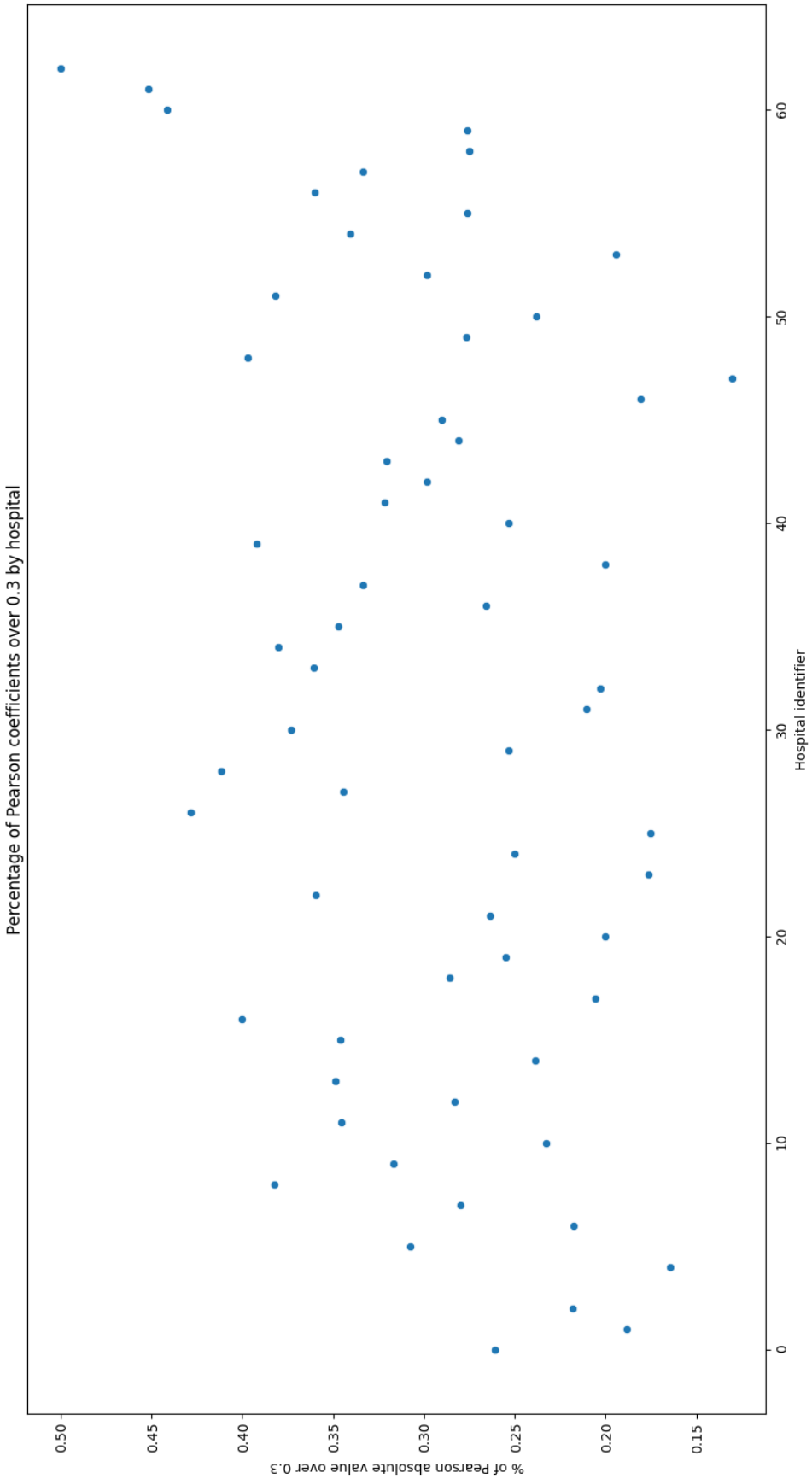

Figure 5.2: Percentage of CIE10 diagnosis that have correlation greater than 0.3 or smaller than -0.3.

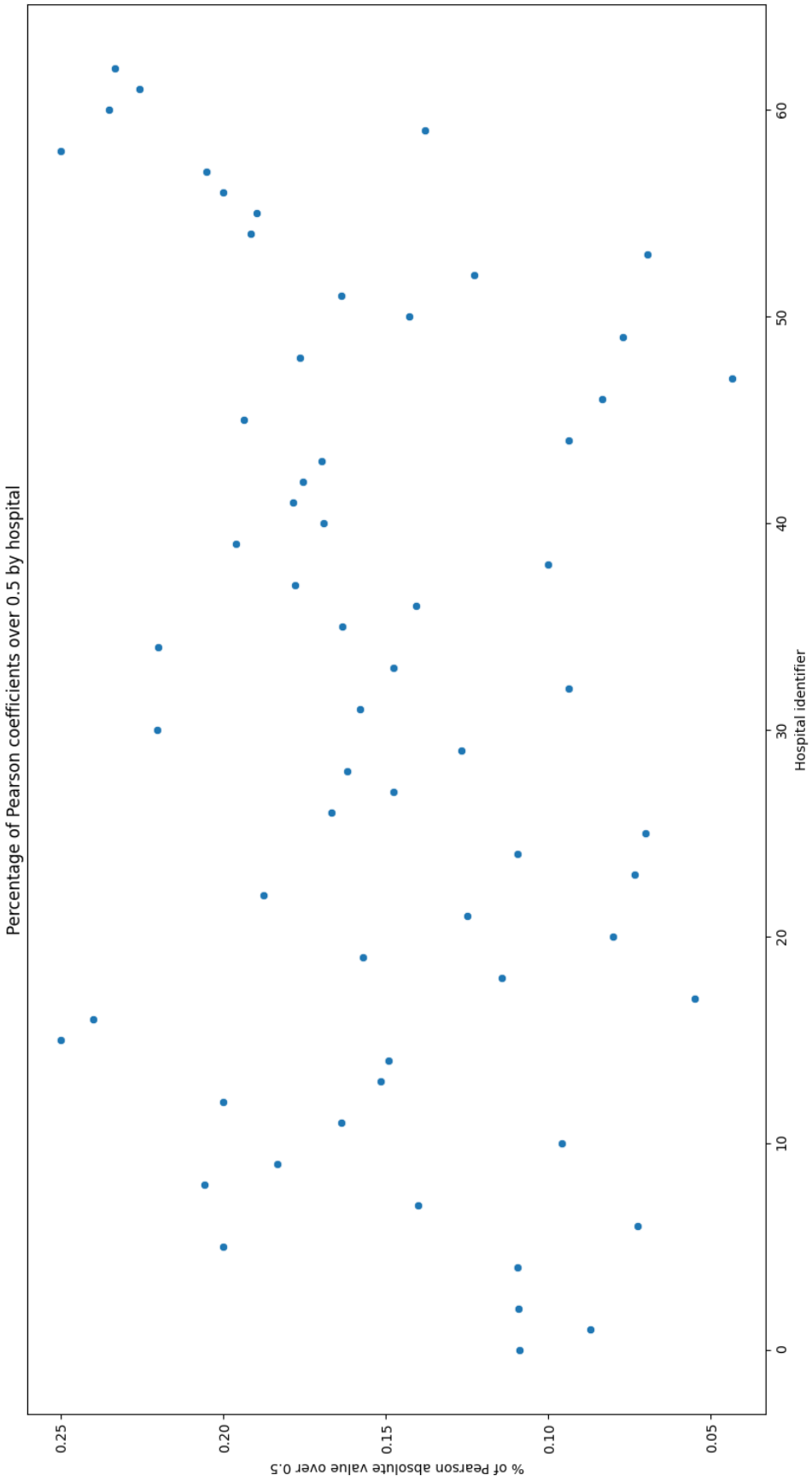

Figure 5.3: Percentage of CIE10 diagnosis that have correlation greater than 0.5 or smaller than -0.5.

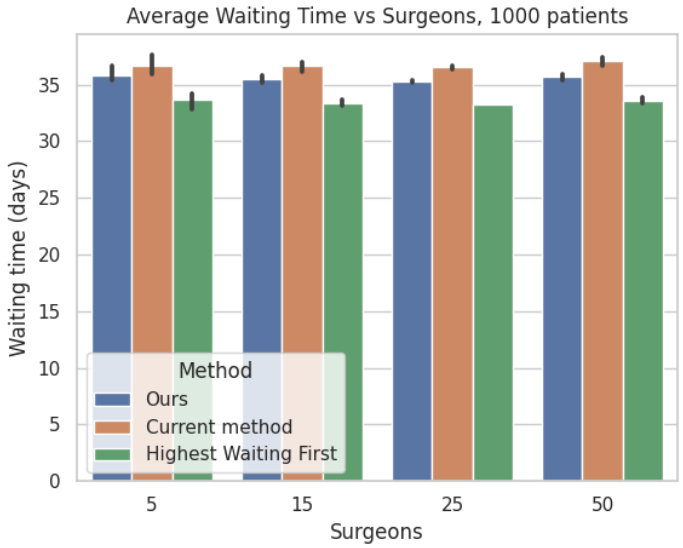

Figure 5.4: Comparison of average waiting time for Hospital 1. 1000 patients.

Given the impossibility of using the exact data of operating rooms to compare, since we do not allow every diagnosis in our experiments and we only know the surgeries performed. We trained a Support Vector Machine (SVM) in the data, achieving 0.81 F1 accuracy in predicting the patients chosen for surgery in the current system. Given the error chance, we compensate the error assuming the best case.

Due to legal restrictions, the experiments were performed in a computer with the following characteristics: 16 GB RAM, 256 GB SSD, Intel® Core™ i7-8550U CPU @ 1.80GHz × 8 with Ubuntu 20.04.03 LTS operating system. To obtain the relaxed optimal value we used the Gurobi Software with its Python package gurobipy [59].

We compared three methods: our algorithm with relaxed truthfulness, SVM on past data and HWF. Here, HWF represents a waiting time oriented optimization. The following plots corresponds to 1000 patients, with 5, 10 and 15 operating rooms, which are shown as error bars. The experiments were run for 5 weeks of scheduling. We assume that most patients are waiting up to 3 months before the start of the scheduling, and the rest shows up in a rate corresponding to the poisson distribution of the hospital.

The results shown in Figures [5.4,5.5,5.6] show that in the *Current method* case, the maximum waiting time goes up to 120 days, which implies that at least one patient had to wait until the last week for attention, even if it showed up on the list first. Our method has the same values than *Highest Waiting First* regarding the maximum value, which is positive for our objective. Notice that, given that the bottlenecks come from resources such as rooms, hours, surgeons, on average and median the waiting times do not differ by a large percentage. However, reducing waiting time is crucial to improve healthcare, our method results in waiting times between the lowest waiting time possible and the current method.

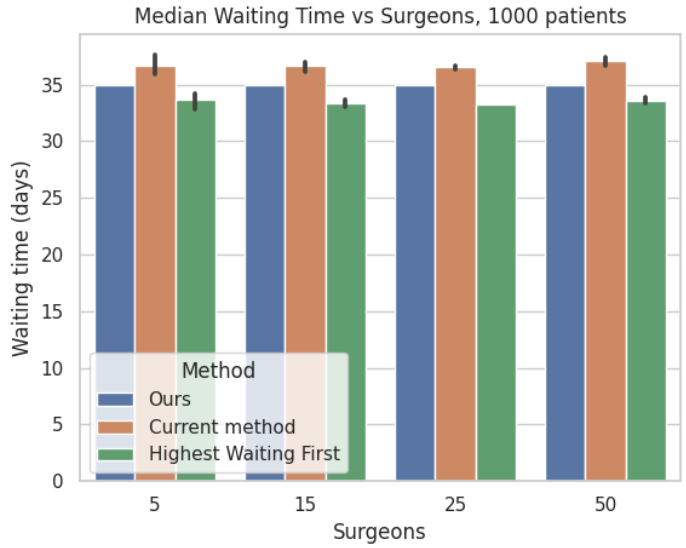

Figure 5.5: Comparison of median waiting time for Hospital 1. 1000 patients.

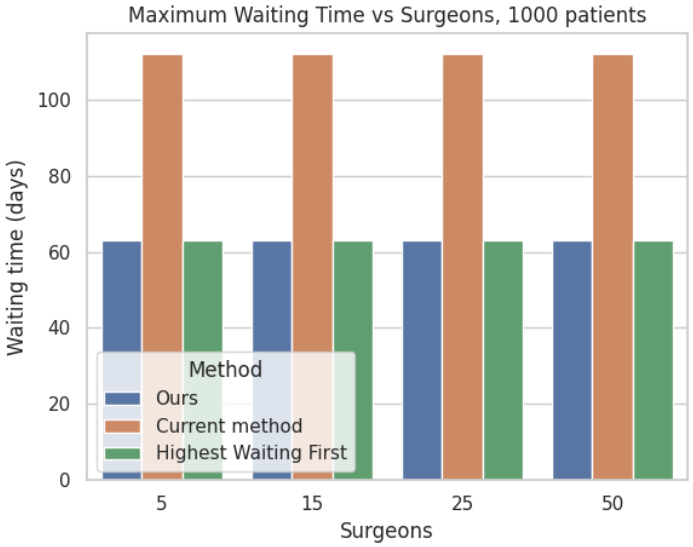

Figure 5.6: Comparison of maximum waiting time for Hospital 1. 1000 patients.

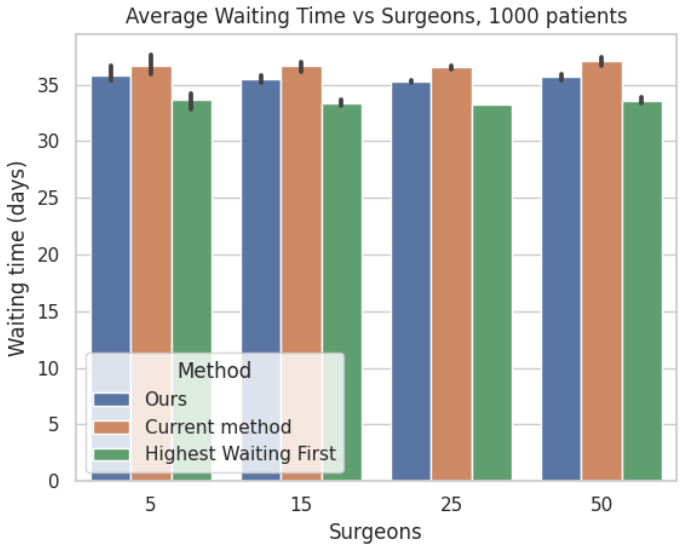

Figure 5.7: Comparison of average waiting time for Hospital 2. 1000 patients.

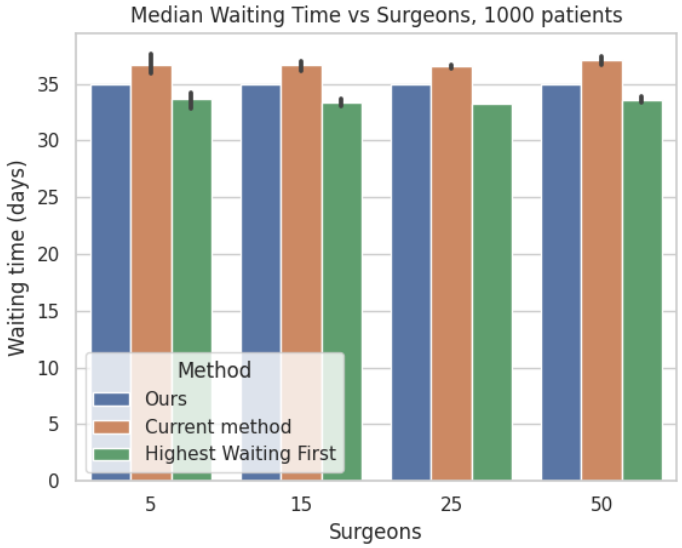

Figure 5.8: Comparison of median waiting time for Hospital 2. 1000 patients.

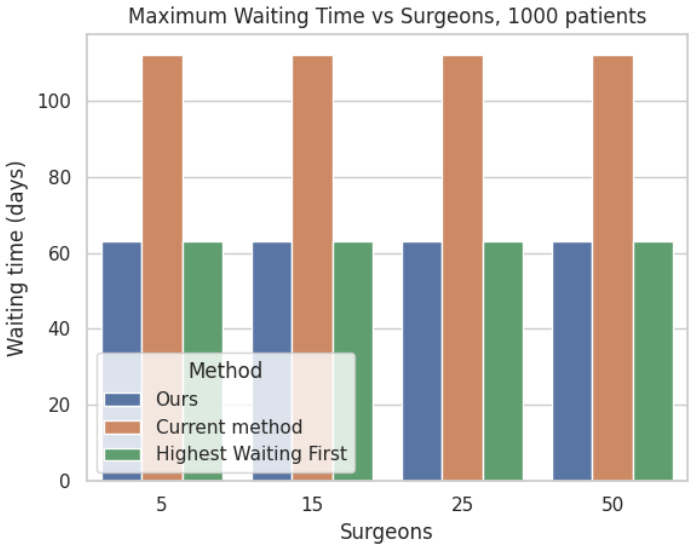

Figure 5.9: Comparison of maximum waiting time for Hospital 2. 1000 patients.

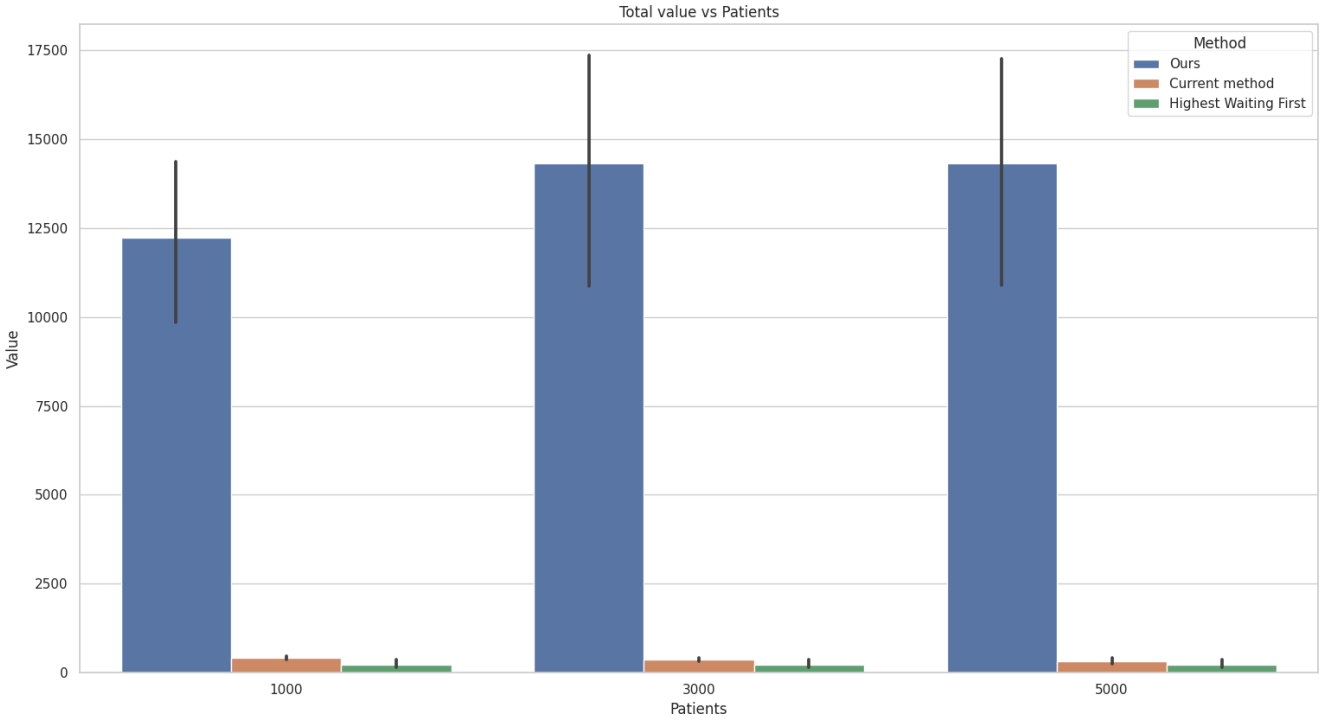

Figure 5.10: Sum of values over one month for Hospital 2

Results from Hospital 1 are similar to Hospital 2 as shown in Figures [5.7,5.8,5.9]. Figure 5.10 shows the main benefit of the mechanism, which is the maximization of perceived value of urgentness. The results show that the system behaves similar to a deadline avoid-

ing optimization algorithm but ensuring that the expert opinion is the conceptual value to maximize.

Chapter 6

Conclusions

In this chapter we shortly discuss the results and impact of this work, as well as future approaches to this problem. In Section 6.1 we give an overview of the thesis results and Section 6.2 discusses possible approaches to the problem and possible improvements to the results.

6.1 Overview

We provided a theoretical solution to the Operating Room Scheduling Problem (ORSP) in the case of Chilean hospitals focused on the equality of health access and surgeon acceptance of the proposed schedule. We tested a variation of the algorithm which does not hold the property of truthfulness but minimizes the distance of the approximation to the truthful solution, the tests were done with simulations generated from real world data.

We introduced a $2t$-approximated truthful in expectation mechanism for the ORSP within our set of assumptions and constraints. While literature is scarce on the intersection of mechanism design and ORSP, our mechanism requires solving an underlying NP-hard scheduling problem. The problem can be solved by a slight modification to the Maximum Weight t-Interval Scheduling with Demands on the uniform demand case, giving a $6t$-approximation algorithm for our problem, we improve on this by giving a $2t$-approximation [9]. Combining this result with work by Lavi and Swamy we give a $2t$-approximated truthful in expectation mechanism [55].

While our solution was originally built to allow surgeons and hospitals to decide a weekly schedule based on their perceptions of patients, our model is suitable to any kind of quasi-linear valuation. The central planner is allowed to set their valuations on any patient, which means that waiting time is not the only metric that could be considered. Moreover, a network of hospitals could be considered, with the cost of transport added as valuation, the transport added as an encoded special schedule time which can be decoded as an hour before the given surgery, solving issues with waiting list loads [11].

Due to practical constraints, our theoretically near optimal approximation algorithm cannot be implemented with high efficiency. While the algorithm is linear, the accompanying

constant is high. We take inspiration in CORE-VCG auctions and drop exact truthfulness, instead, we run an optimization algorithm minimizing the distance between the approximated solution and the VCG solution.

The implementation results effectively show an improvement in social welfare, in this instance, this is a metric on the likelihood of the schedule to be accepted by the surgeons and central planner. On the other hand, the mechanism results on waiting times are similar to waiting time minimizing algorithms and better than the current selection system. Moreover, deadline metrics are similar to deadline avoidance algorithms while still obtaining better acceptance likelihood score.

6.2 Discussion

Originally this work aimed to include the whole process from the entrance of the patient to the leave from the hospital, in this model the analyzed surgery room problem is a piece of a bigger queuing system. We decided to focus on the most important part of the problem given the difficulty of the task and lack of the necessary data.

Regarding the scheduling problem, it is not possible to obtain a $\mathcal{O}(t/\log t)$ approximation, a $\mathcal{O}(t)$ is near optimal in this case. A slight improvement alternative to explore is to add randomness to the multicoloring mapping in order to analyze the problem in a similar way to the bounds on maximum load in the balls and bins problem [75]. However, most balls and bins problems are related to minimizing load instead of maximizing it, which would require a different approach. Another improvement would be to extend the results for multiple bids, which solution should be a similar algorithm.

On the mechanism, given that the approximation is near-optimal and that Vickrey-Clarke-Groves solutions are ex-post, there is little room for improvement in our specific formulation [69]. A different unexplored approach is the usage of auctions without money [38, 35, 74].

However, the problem formulation is a simplified version of a mixed integer program with numerous restrictions as well as uncertainty. Our formulation does not take into account factors like uncertainty of operating room time, uncertainty on the patients arrival as well as taking for granted the presence of additional staff such as nurses and anesthesiologists. Although formulations on the ORSP that take these factors into account are cost or deadline avoiding focused, it is unfeasible to have theoretical guarantees over such an amount of factors, solutions to formulations with these constraints are mainly numerical and cannot be used to fulfill truthfulness or approximation guarantees.

On the other hand, regardless the approximation bound and truthfulness in our simplified problem, it is not of practical use. Even including workarounds on truthfulness we faced similar results as found on the literature, such as large profits obtained by non-truthful agents. To attain truthfulness in practice there is barely room for error. In our case, even if the solutions assigns almost the same bids that optimal case does, a selfish agent can always obtain a high profit from the system.

Further work on practical mechanism design for this problem requires either weaker prop-

erties or less understood tools. Regarding the later, recent work on deep learning applications on combinatorial auctions have given near optimal results in practice, even including the truthfulness property, however, usage in healthcare of such tools usually requires some understanding on how the program arrives to a solution [28, 33].

Our closing though is related to the data itself and the healthcare sector. The lack of technology in the Chilean public healthcare produces low quality data which not only require experts to process, but sometimes cannot be processed at all given the abuse of free text input. Moreover, we found no correlation between severity and waiting time for patients with similar characteristics, this points to either a bad classification or unfairness in the selection process. In both cases there is a lack of usage of tools and protocols that could solve these issues, which are on way to be solved in few institutions.

Chapter 7

Bibliography

[1] Zahraa A. Abdalkareem, Amiza Amir, Mohammed Azmi Al-Betar, Phaklen Ekhan, and Abdelaziz I. Hammouri. Healthcare scheduling in optimization context: a review. *Health and Technology*, 11(3):445–469, 2021-04.

[2] Zakaria Abdelrasol, Nermine Harraz, and Amr Eltawil. Operating room scheduling problems: A survey and a proposed solution framework. In *Transactions on Engineering Technologies*, pages 717–731. Springer Netherlands, 2014.

[3] Fidaa Abed, José R. Correa, and Chien-Chung Huang. Optimal coordination mechanisms for multi-job scheduling games. In *Algorithms - ESA 2014*, pages 13–24. Springer Berlin Heidelberg, 2014.

[4] Ellur Anand and Ramasamy Panneerselvam. Literature review of open shop scheduling problems. *Intelligent Information Management*, 07(01):33–52, 2015.

[5] Alaka Ananth and K. Chandra Sekaran. Game theoretic approaches for job scheduling in cloud computing: A survey. In *2014 International Conference on Computer and Communication Technology (ICCCT)*. IEEE, 2014-09.

[6] Per Austrin, Subhash Khot, and Muli Safra. Inapproximability of vertex cover and independent set in bounded degree graphs. In *2009 24th Annual IEEE Conference on Computational Complexity*. IEEE, 2009-07.

[7] Amotz Bar-Noy, Reuven Bar-Yehuda, Ari Freund, Joseph (Seffi) Naor, and Baruch Schieber. A unified approach to approximating resource allocation and scheduling. In *Journal of the ACM*, pages 735–744, 2000.

[8] Reuven Bar-Yehuda, Magnús Halldórsson, Joseph Naor, Hadas Shachnai, and Ira Rosen. Scheduling split intervals. *SIAM J. Comput.*, 36:1–15, 01 2006.

[9] Reuven Bar-Yehuda and Dror Rawitz. Using fractional primal–dual to schedule split intervals with demands. *Discrete Optimization*, 3(4):275–287, 2006-12.

[10] Piotr Berman. A d/2 approximation for maximum weight independent set in d-claw free graphs. In *Algorithm Theory - SWAT 2000*, pages 214–219. Springer Berlin Heidelberg, 2000.

[11] José Luis Contreras Biekert, Carla Tokman Ramos, Rodrigo Miranda Toledo, Raúl Aguilar Barrientos, Valeria Lobos Ossandón, Camila Arroyo From, Javiera Duarte Flores, Andres Olivares Arredondo, and Serge Jabouim Silva-Videla. *Uso Eficiente de Quirófanos Electivos y Gestión de Lista de Espera Quirúrgica NO GES*. Comisión Nacional de Productividad. Comisión Nacional de Productividad, editorial universitaria s.a edition, 2020.

[12] Heydenreich Birgit, Rudolf Müller, and Marc Uetz. Games and mechanism design in machine scheduling – an introduction. *Production and Operations Management*, 16, 02 2006.

[13] Craig Boutilier and Holger H. Hoos. Bidding languages for combinatorial auctions. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'01, page 1211–1217, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[14] Khalid I. Bouzina and Hamilton Emmons. Interval scheduling on identical machines. *Journal of Global Optimization*, 9(3-4):379–393, 1996-12.

[15] Eagen Brendan. *Ambulatory Clinic Scheduling*. PhD thesis, University of Toronto, 2017.

[16] Peter Brucker. *Scheduling Algorithms*. Springer-Verlag GmbH, 2007.

[17] Ayelet Butman, Danny Hermelin, Moshe Lewenstein, and Dror Rawitz. Optimization problems in multiple-interval graphs. *ACM Transactions on Algorithms*, 6(2):1–18, 2010-03.

[18] Martin C. Carlisle and Errol L. Lloyd. On the k-coloring of intervals. *Discrete Applied Mathematics*, 59(3):225–235, 1995-05.

[19] Robert Carr and Santosh Vempala. Randomized metarounding. *Random Structures & Algorithms*, 20:343 – 352, 05 2002.

[20] Tugba Cayirli and Emre Veral. Outpatient scheduling in health care: A review of literature. *Production and Operations Management*, 12:519 – 549, 01 2009.

[21] Sofia Ceppi and Nicola Gatti. An algorithmic game theory study of wholesale electricity markets based on central auction. *Integrated Computer-Aided Engineering*, 17:273–290, 01 2010.

[22] Venkatesan T. Chakaravarthy, Anamitra R. Choudhury, Sambuddha Roy, and Yogish Sabharwal. Scheduling split intervals with non-uniform demands. *Discrete Optimization*, 38:100611, 2020-11.

[23] Camilo Cid and Ibern Pere. Regulación del financiamiento a hospitales: "yardstick

competition" aplicada a los hospitales públicos en chile hospital financing regulation: yardstick competition in chilean public hospitals. *Cuadernos médicos-sociales*, ep2008, Vol. 48:p155–164., 01 2008.

[24] Vincent Conitzer and Tuomas Sandholm. Failures of the VCG mechanism in combinatorial auctions and exchanges. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems - AAMAS '06*. ACM Press, 2006.

[25] Andrea Curtis, Colin Russell, Johannes Stoelwinder, and John Mcneil. Waiting lists and elective surgery: Ordering the queue. *The Medical journal of Australia*, 192:217–20, 02 2010.

[26] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Daniel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer Publishing Company, Incorporated, 1st edition, 2015.

[27] Daniela-Emanuela Danacica. Methodological and Applicative Problems of using Pearson Correlation Coefficient in the Analysis of Socio-Economic Variables. *Romanian Statistical Review Supplement*, 65(2):148–163, February 2017.

[28] Constantinos Daskalakis. Technical perspective: The quest for optimal multi-item auctions. *Communications of the ACM*, 64(8):108–108, 2021-08.

[29] Reinhard Diestel. *Graph theory*. Springer Berlin Heidelberg Imprint Springer, 2017.

[30] DIPRES. Informe de finanzas pÚblicas tercer trimestre 2019.

[31] Shaddin Dughmi and Arpita Ghosh. Truthful assignment without money. In *Proceedings of the 11th ACM Conference on Electronic Commerce*, EC '10, page 325–334, New York, NY, USA, 2010. Association for Computing Machinery.

[32] Guillermo Durán, Pablo A. Rey, and Patricio Wolff. Solving the operating room scheduling problem with prioritized lists of patients. *Annals of Operations Research*, 258(2):395–414, Nov 2017.

[33] Paul Dütting, Zhe Feng, Harikrishna Narasimhan, David C. Parkes, and Sai S. Ravindranath. Optimal auctions through deep learning. *Communications of the ACM*, 64(8):109–116, 2021-08.

[34] Rhiannon Tudor Edwards and Judith Barlow. Rationing health care by waiting list: an extra-welfarist perspective. Working Papers 114chedp, Centre for Health Economics, University of York, January 1994.

[35] Salman Fadaei and Martin Bichler. Generalized assignment problem: Truthful mechanism design without money. *Operations Research Letters*, 45(1):72 – 76, 2017.

[36] Michael R. Fellows, Danny Hermelin, Frances Rosamond, and Stéphane Vialette. On the parameterized complexity of multiple-interval graph problems. *Theoretical Computer Science*, 410(1):53–61, 2009-01.

[37] Yann B. Ferrand, Michael J. Magazine, and Uday S. Rao. Managing operating room efficiency and responsiveness for emergency and elective surgeries—a literature survey. *IIE Transactions on Healthcare Systems Engineering*, 4(1):49–64, 2014.

[38] Dimitris Fotakis, Piotr Krysta, and Carmine Ventre. Combinatorial auctions without money. *13th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2014*, 2, 10 2013.

[39] Jugal Garg and Peter McGlaughlin. A truthful mechanism for interval scheduling. In *Algorithmic Game Theory*, pages 100–112. Springer International Publishing, 2018.

[40] Stanisław Gawiejnowicz. A review of four decades of time-dependent scheduling: main results, new topics, and open problems. *Journal of Scheduling*, 01 2020.

[41] William H. Greene. *Econometric Analysis*. Pearson Education, fifth edition, 2003.

[42] C. Gudex, A. Williams, M. Jourdan, R. Mason, J. Maynard, R. O'Flynn, and M. Rendall. Prioritising waiting lists. *Health Trends*, 22(3):103–108, 1990.

[43] M M Günal and M. Pidd. Discrete event simulation for performance modelling in health care: a review of the literature. *Journal of Simulation*, 4(1):42–51, Mar 2010.

[44] Magnús M. Halldórsson and Guy Kortsarz. Multicoloring: Problems and techniques. In *Lecture Notes in Computer Science*, pages 25–41. Springer Berlin Heidelberg, 2004.

[45] Arne Herzel, Michael Hopf, and Clemens Thielen. Multistage interval scheduling games. *Journal of Scheduling*, 22(3):359–377, 2018-05.

[46] Michael Hopf, Clemens Thielen, and Oliver Wendt. Competitive algorithms for multistage online scheduling. *European Journal of Operational Research*, 260(2):468–481, 2017-07.

[47] Akihisa Kako, Takao Ono, Tomio Hirata, and Magnús M. Halldórsson. Approximation algorithms for the weighted independent set problem. In *Graph-Theoretic Concepts in Computer Science*, pages 341–350. Springer Berlin Heidelberg, 2005.

[48] Dipak Kalra and David Ingram. Electronic health records. In *Health Informatics*, pages 135–181. Springer London, 2006.

[49] Thomas Kittsteiner, Jörg Nikutta, and Eyal Winter. Declining valuations in sequential auctions. *International Journal of Games Theory*, 33(1):89–106, 2004-12.

[50] V. Knight, Janet E. Williams, and I. Reynolds. Modelling patient choice in healthcare systems: development and application of a discrete event simulation with agent-based decision making. *Journal of Simulation*, 6:92–102, 2012.

[51] Antoon W.J. Kolen, Jan Karel Lenstra, Christos H. Papadimitriou, and Frits C.R. Spieksma. Interval scheduling: A survey. *Naval Research Logistics*, 54(5):530–543, 2007.

[52] Baig MM Koli NV, Mirza F and Ullah E. An agent-based modelling approach for scheduling and management of elective surgeries. *SM J Health Med Inform.*, 2018;.

[53] Mikhail Y. Kovalyov, C.T. Ng, and T.C. Edwin Cheng. Fixed interval scheduling: Models, applications, computational complexity and algorithms. *European Journal of Operational Research*, 178(2):331–342, 2007-04.

[54] Dominik Kress, Sebastian Meiswinkel, and Erwin Pesch. Mechanism design for machine scheduling problems: classification and literature overview. *OR Spectrum*, 40(3):583–611, 2018-02.

[55] Ron Lavi and Chaitanya Swamy. Truthful and near-optimal mechanism design via linear programming. *J. ACM*, 58(6), December 2011.

[56] Lorena Leiva. El 42% de pacientes fonasa inscritos debe esperar al menos un año. *La Tercera*, 2019-01-14.

[57] Fei Li, Diwakar Gupta, and Sandra Potthoff. Improving operating room schedules. *Health Care Management Science*, 19(3):261–278, 2015-02.

[58] Lu Liu, Chun Wang, and Jianjun Wang. A combinatorial auction mechanism for surgical scheduling considering surgeon's private availability information. *Journal of Combinatorial Optimization*, 37(1):405–417, 2018-01.

[59] Gurobi Optimization LLC. Gurobi optimizer reference manual, 2020.

[60] Valeria Lobos and Andrés Olivares. Calidad de datos y sistemas de información en salud pública. Technical report, Comisión Nacional de Productividad, Amunátegui 232, of. 401, Santiago, Chile., 2020 [Online].

[61] Mackarena Zapata M. Importancia del sistema grd para alcanzar la eficiencia hospitalaria. *Revista Médica Clínica Las Condes*, 29(3):347 – 352, 2018. Tema central: Enfermería.

[62] Robert Maidstone. Discrete event simulation, system dynamics and agent based simulation: Discussion and comparison, 03 2012.

[63] Ruta Mehta and Vijay Vazirani. An incentive compatible, efficient market for air traffic flow management. *Theoretical Computer Science*, 09 2018.

[64] Natasa Mihailovic, Sanja Kocic, and Mihajlo Jakovljevic. Review of diagnosis-related group-based financing of hospital care. *Health Services Research and Managerial Epidemiology*, 3:2333392816647892, 2016.

[65] MINSAL and FONASA. Base de datos de salud minsal, ugcq, fonasa. unpublished, 2017.

[66] Michael Mitzenmacher and Eli Upfal. *Probability and Computing*. Cambridge University Press, 2017.

[67] Penelope M. Mullen. Prioritising waiting lists: how and why? *European Journal of Operational Research*, 150(1):32–45, 2003-10.

[68] Thành Nguyen, Ahmad Peivandi, and Rakesh Vohra. Assignment problems with complementarities. *Journal of Economic Theory*, 165:209–241, 2016-09.

[69] Noam Nisan, Tim Roughgarden, Éva Tardos, and Vijay V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, 2007.

[70] OECD. *Better Ways to Pay for Health Care*. OECD, 2016.

[71] M. Pal. Intersection graphs: An introduction. *ArXiv*, abs/1404.5468, 2014.

[72] Ignacio Palacios-Huerta, David C. Parkes, and Richard Steinberg. Combinatorial auctions in practice. *SSRN Electronic Journal*, 2021.

[73] Ramtin Pedarsani, Jean Walrand, and Yuan Zhong. Scheduling tasks with precedence constraints on multiple servers. In *2014 52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2014-09.

[74] Ariel D. Procaccia and Moshe Tennenholtz. Approximate mechanism design without money. *ACM Trans. Econ. Comput.*, 1(4), December 2013.

[75] Martin Raab and Angelika Steger. "balls into bins" — a simple and tight analysis. In *Randomization and Approximation Techniques in Computer Science. RANDOM 1998. Lecture Notes in Computer Science, vol 1518.*, pages 159–170. Springer Berlin Heidelberg, 1998.

[76] Sandeep Rath, Kumar Rajaram, and Aman Mahajan. Integrated anesthesiologist and room scheduling for surgeries: Methodology and application. *Operations Research*, 65(6):1460–1478, 2017-12.

[77] Alvin E. Roth, Tayfun Sönmez, and M. Utku Ünver. Pairwise kidney exchange. *Journal of Economic Theory*, 125(2):151–188, 2005-12.

[78] Tim Roughgarden. *Twenty Lectures on Algorithmic Game Theory*. Cambridge University Press, 2016.

[79] Tim Roughgarden. *Beyond the Worst-Case Analysis of Algorithms*. Cambridge University Press, 2021.

[80] Masoumeh Saeedian, Mohammad Mehdi Sepehri, Ammar Jalalimanesh, and Pejman Shadpour. Operating room orchestration by using agent-based simulation. *Perioperative Care and Operating Room Management*, 15:100074, 2019-06.

[81] Shuichi Sakai, Mitsunori Togasaki, and Koichi Yamazaki. A note on greedy algorithms for the maximum weighted independent set problem. *Discrete Applied Mathematics*, 126(2-3):313–322, 2003-03.

[82] James Schummer and Azar Abizada. Incentives in landing slot problems. *Journal of Economic Theory*, 170:29–55, 2017-07.

[83] Andrei Shleifer. A theory of yardstick competition. *The RAND Journal of Economics*, 16(3):319–327, 1985.

[84] Luigi Siciliani and Jeremy Hurst. Tackling excessive waiting times for elective surgery: a comparative analysis of policies in 12 oecd countries. *Health Policy*, 72(2):201 – 215, 2005.

[85] Michael Sipser. *Introduction to the Theory of Computation*. Thomson, Jan 1997.

[86] Malgorzata Sterna, Jacek Juraszek, and Erwin Pesch. Revenue maximization on parallel machines. In *Operations Research Proceedings 2008*, pages 153–158. Springer Berlin Heidelberg, 2009.

[87] A. Testi, E. Tanfani, R. Valente, G. L. Ansaldo, and G. C. Torre. Prioritizing surgical waiting lists. *Journal of Evaluation in Clinical Practice*, 14(1):59–64, 2008-01.

[88] Clemens Thielen and Sven O. Krumke. A general scheme for designing monotone algorithms for scheduling problems with precedence constraints. In *Approximation and Online Algorithms*, pages 105–118. Springer Berlin Heidelberg, 2009.

[89] Vijay V. Vazirani. *Approximation Algorithms*. Springer-Verlag, Berlin, Heidelberg, 2001.

[90] Shuwan Zhu, Wenjuan Fan, Shanlin Yang, Jun Pei, and Panos M. Pardalos. Operating room planning and surgical case scheduling: a review of literature. *Journal of Combinatorial Optimization*, 37(3):757–805, 2018-07.