Czech Technical Univeristy in Prague
Faculty of Electrical Engineering
Department of Cybernetics

Bachelor's thesis

Optimization of Samples Processing in Medical Laboratories with Uncertain Release Times of Samples

David Procházka

Supervisor: doc. Ing. Přemysl Šůcha, Ph.D.

Study Program: Open Informatics
May, 2020

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Procházka David**	Personal ID number:	**474606**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Cybernetics**		
Study program:	**Open Informatics**		
Branch of study:	**Computer and Information Science**		

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Optimization of Samples Processing in Medical Laboratories with Uncertain Release Times of Samples

Bachelor’s thesis title in Czech:

Optimalizace vyhodnocování laboratorních vzorků při neurčité době připravenosti

Guidelines:

The goal of this thesis is to propose an algorithm minimizing the Turn Around Time (TAT) of the sample evaluation process in a biochemical laboratory. The work builds on the results of the master’s thesis of Karel Gavenčiak, who analyzed the laboratory process in FNKV. In this work, we will focus on the uncertain parameters of this process, specifically on the uncertain arrival time of the sample into the laboratory, i.e., uncertain release time in the scheduling terminology. This topic has the following goals:
1. Analyze related work in this domain.
2. Define a mathematical model of the robust scheduling problem with uncertain release times.
3. Propose and implement an algorithm for the defined model.
4. Benchmark the performance of the algorithm and compare the results with a non-robust version of the problem.

Bibliography / sources:

[1] Dimitris Bertsimas, Melvyn Sim: The Price of Robustness. Operations Research 52(1): 35-53 (2004)
[2] Karel Gavenčiak: Optimization of Samples Processing in Medical Laboratories: Problem Analysis Based on Data, Master’s Thesis CTU (2019)
[3] Dominique Feillet: A tutorial on column generation and branch-and-price for vehicle routing problems. 4OR 8(4): 407-424 (2010)

Name and workplace of bachelor’s thesis supervisor:

doc. Ing. Přemysl Šůcha, Ph.D.,   Department of Control Engineering,   FEE

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **10.01.2020**   Deadline for bachelor thesis submission: **22.05.2020**

Assignment valid until: **30.09.2021**

doc. Ing. Přemysl Šůcha, Ph.D.	doc. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

.

Date of assignment receipt   Student’s signature

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, 21. 5. 2020 . . . .

David Procházka

Acknowledgements

I would like to express my utmost thanks to my supervisor doc. Ing. Přemysl Šůcha, Ph.D. for all his time that he invested into this work, his great advice and constructive feedback.

Furthermore, I would like to thank my parents, family and my girlfriend for supporting me throughout my studies.

Abstract

This thesis aims to optimize one of the bottlenecks in the workflow of medical laboratories. We build on top of the data analysis performed in [1] while using the same data to create an efficient solution to the batching problem of the centrifuge. Unlike the existing approaches, we also consider an element of uncertainty on the release times of the samples, which stems from the workflow. Firstly, we investigate the state-of-the-art literature in the field of scheduling under uncertainty. Afterwards, we analyze and formally define our problem. Then, we examine the most promising approaches and based on the insights gained; we propose a proactive schedule generating model, whose robustness might be adjusted. Later, we incorporate this model into a proactive-reactive framework. Subsequently, we demonstrate the computational tractability of this approach, and explain its behavior in small instances of the problem. Finally, we run a simulation emulating the real processes in the laboratory. We conclude that by increasing the level of robustness of the proactive schedule, we get worse average in the total turnaround time of some samples, but with considerably decreased variance.

Keywords: scheduling, uncertainty, robustness, centrifuge

Abstrakt

Cílem této práce je optimalizace jednoho z úzkých hrdel pracovního postupu zdravotnické laboratoře. Budeme navazovat na datovou analýzu provedenou v [1], přičemž tato data také použijeme k vytvoření efektivního řešení problému obsazení dávek na odstředivce. Na rozdíl od předchozích přístupů budeme uvažovat vliv neurčitosti na čas uvolnění jednotlivých vzorků, který pramení z pracovního postupu. Nejdříve provedeme rešerši nejaktuálnější literatury v oboru rozvrhování s neurčitostí. Poté zanalyzujeme a formálně zadefinujeme náš problém. Dále prozkoumáme nejnadějnější přístupy řešení. Na základě získaných poznatků navrhneme model schopný vytvoření proaktvního rozvrhu, jehož robustnost jsme schopni ovládat. Potom jej využijeme k vytvoření proaktivně-reaktivního řešení celého problému, načež experimentálně ověříme, že je rychle vypočitatelné. Chování rozvrhovacího modelu je posléze osvětleno na malých instancích problému. V poslední řadě provádíme experiment, jehož cílem je věrně replikovat procesy v laboratoři. Docházíme k závěru, že zvyšováním robustnosti proaktivního rozvrhu dojde ke zhoršení průměrné doby odbavování, ale zároveň také k podstatnému snížení jejího rozptylu.

Klíčová slova: rozvrhování, neurčitost, robustnost, odstředivka

Contents

Chapter 1

Introduction

The goal of this thesis is to investigate one of the bottlenecks in the workflow of medical laboratories. Based on this analysis, we propose a scheduling framework which will reduce the time from sample collection until the results are available, which is known as turnaround time (TAT). This work aims to build on top of the analysis of anonymized data from Královské Vinohrady University Hospital, which was performed by Karel Gavenčiak [1]. In his thesis, he gives a detailed description of the laboratory workflow and also analyzes real data from their laboratory. He then proposes several improvements for another bottleneck, the medical analyzer, that performs the tests on the samples of biological material. However, most of the blood samples, which are the most often tested type of organic material, cannot be tested straight away as they come into the laboratory. They need to be prepared first. That is where our thesis comes in; we will be looking at ways to improve the first step in the process, the batching of samples into the centrifuge, which is a relatively standard optimization problem. However, unlike the existing approaches, we attempt to address an element of uncertainty on the release times of the samples, which stems from the way the material is delivered into the laboratory.

Firstly, we begin with an introduction and then examine state of the art literature in the fields relevant to this problem: scheduling under uncertainty and laboratory scheduling. In Chapter 2, we describe the setting we are modelling and formally declare our scheduling problem. In the subsequent Chapter 3, we discuss the advantages and disadvantages of the application of different methods of robust optimization on our problem. Following this analysis, we propose our own robust formulation of the problem. Afterwards, in Chapter 4, we evaluate this approach both on small synthetic instances as well as larger datasets and discuss its performance. Finally, concluding remarks are presented in Chapter 5.

1.1 Literature overview

The use of optimization in a hospital environment has a long history. Starting with the Nurse scheduling problem [2] in the seventies, researchers have since focused on optimizing many parts of the hospital, such as schedules of operating rooms [3], the effectiveness of intensive care units [4] and resident schedules [5]. We intend to contribute to this extensive field of research. In this work, we will focus on scheduling in a biochemical laboratory, and we will propose a solution which will speed up the sample evaluation process while providing a schedule which is to some degree protected against an unexpectedly delayed release of a task.

The main area of related works which we need to cover is scheduling under uncertainty. This field studies scheduling problems, where the input data are not precisely known and may be subject to change. Firstly, we will discuss the possible approaches towards the uncertainty, whether to plan for it in advance or react to it when it happens. Afterwards, we will examine the possible ways of generating the schedules and how to improve or repair them when they become suboptimal or infeasible due to the uncertainty. At last, we will survey the areas of laboratory scheduling, which might give us some insights when it comes to problems in a hospital setting.

1.1.1 Scheduling under uncertainty

When scheduling processes which are subject to uncertainty, it is very likely that as time progresses, the uncertainty manifests and reveals additional information. How these new realizations are taken into account divides the literature into several groups.

The authors in [6] discuss the three main categories: completely reactive scheduling, proactive scheduling and proactive-reactive scheduling. *Completely reactive scheduling* is used, when very little is known about the uncertainty, which prevents us from planning ahead. It requires constant knowledge about the problem domain in order to act quickly once the uncertainty is realized. On the other hand, *proactive scheduling* tries to build schedules, which are, to some degree, protected against possible realizations of the uncertainty. These methods can be used only when some information about the uncertainty is known when generating the schedule. The balance in between is represented by *proactive-reactive scheduling*, which builds a predictive schedule based on the known information about the uncertainty and later uses a reactive algorithm which modifies the schedule according to the revealed information.

To correctly choose which one of these approaches will be used is a decisive step, which must be analyzed thoroughly. Several factors need to be weighed in, such as the time it takes to react to the realization of the uncertainty and how often is a reaction needed. However, to cite the article [6] word for word "A scheduling system that is able to deal with uncertainty is very likely to employ both proactive and reactive scheduling". It is practically impossible even for proactive schedules to take into account all the possible realizations of uncertainty. Some degree of schedule modification will always be necessary to schedule long term processes. This reactive step might be elementary such as rerunning the scheduling algorithm with the updated data, or much more complex using a specific algorithm which repairs or improves the schedule.

In our work, we will focus on proactive scheduling, mainly because we have a lot of anonymized data, which will give us the possibility to analyze the uncertainty. However, as mentioned, when a sample arrives later than our proactive method expects, or when a new sample is registered, we will need to react, this will make our approach a proactive-reactive one.

Another important decision is how to generate the schedule. This can be done in a plethora of ways; a great overview is provided by [7]. The authors show a motivating example and then demonstrate how uncertainties may be modelled as well as discuss the ideas behind the methods which are used to create a schedule. A majority of proactive approaches use *linear programming* (LP), which will be our method of choice as well, mainly because it is capable of handling our computationally difficult problem while giving us truly optimal solutions. Moreover, LP is well suited for the structure of our task and most widely researched in the literature concerning our problem. The fields using LP in proactive scheduling that will be relevant the most to us and that we are going to examine next are robust optimization, stochastic programming and bilevel optimization.

Robust optimization

In this context, the word "robustness" means the extent to which the feasibility and the objective function value are susceptible to perturbations in the realizations of the uncertain parameters. The smaller the effect of changes in realizations, the more "robust" is the solution said to be. However, this protection almost always comes at a cost. In order to be prepared against the possible realizations of the uncertainty, one must assume bad outcomes, which will typically perform worse than the deterministic optimum.

The important aspect of this approach is that no assumptions about the distributions of the uncertainties are made. Instead, so-called "uncertainty sets" are considered. We are then looking for a solution that is feasible for every realization from the uncertainty set.

Paper [8] provides a great introduction to the practical ways of robust optimization and describes the two main ways to solve such problems; the first is to transform the problem into its "robust counterpart". Based on the original problem and the uncertainty set, hard constraints are added into the deterministic model in order to assure feasibility for realizations in the uncertainty set. The reformulated problem is then solved by the appropriate method such as LP, conic programming, or other, depending on the type of uncertainty set used. An example of this approach in practice is [9], where researchers use this approach to optimize the portfolio problem and provide us with numerical results.

The second method is called the *adversarial approach*, which is useful when we are unable to transform the problem into its robust counterpart, or when it is computationally demanding. An application of this approach can be seen in [10], where researchers use it to compute optimal basestock level based on uncertain demand. They use a finite set of realizations of the uncertain variables for a given constraint, so-called scenarios. Then, a min-max approach is used, the optimization problem is solved and then evaluated for the worst possible outcome of the uncertainty. If it is considered good enough, it is proclaimed as the solution. If not, the worst outcome of the uncertainty is added into the model, and the solution recalculated. This is repeated until a robust enough solution is acquired.

One of the important properties of robust optimization is the balance between the value of the criterion and the probability that the model will become unfeasible or strongly suboptimal. The first works in this field considered only strict feasibility for every realization in the uncertainty set; an example is [11]. However, these models tend to be very conservative and more recently, scientists have found ways, to control the equilibrium. This is researched in [12], where the authors allow control of the robustness on the level of individual constraints while introducing little computational overhead.

Article [13] focuses precisely on this trade-off between robustness and objective function value. The authors define a metric of the robustness of their model, called stability radius. Subsequently, they transform their single-objective scheduling problem into a bi-objective one, where the second objective is maximizing the stability radius. They then proceed to plot the Pareto frontier of their problem, which directly visualizes the relationship between optimality and robustness.

Stochastic programming

The main idea of these methods is that the probability distribution behind the uncertainty is either known or can be estimated, which makes it possible to introduce stochastic variables directly into the optimization model. The optimization is then performed with respect to the expected value of such variables.

However, analytical approaches are usually intractable in these cases. Instead, the random variables are discretized with real, or synthetically generated data based on the distribution, which are called scenarios. These methods yield standard LP models, which aim to optimize the expected value over all the scenarios and to some degree, enforce feasibility over them.

This leads to a technique known as *sample average approximation* (SAA). This approach is used by the researchers in [14] to schedule elective surgery procedures in a hospital. Firstly, several independent scenarios are sampled from the distributions we believe are governing the uncertainty. For each of these, a data-based model is solved to obtain a candidate solution. The mean and variance of objective values of these solutions are calculated and serve as an estimate of the true objective value. Based on this, the optimality gap is estimated for each of the candidate solutions, and one of them is chosen.

A possible drawback of this approach is that the data need to be included in the model, which yields a large number of constraints and variables. In order to get precise results, larger sample sizes need to be considered, thereby making the model more computationally demanding. Authors in [14] report computation times from one to twenty minutes for a problem concerning 203 patients. This seems to be too demanding for our problem.

Stochastic programming is also very often used to solve problems, which include two, or even more, stage decision process. The focus of the decisions made in the first stage is to optimize the expected value of the possible second stage decisions, which are performed after the uncertainty is revealed. This approach, together with SAA is used in [15] to design a supply chain network under uncertainty. Another illustrative example of these methods is [16], where the authors describe a two-stage approach towards operating room scheduling, which is subject to stochastic demand when an emergency occurs as well as deterministic planned surgery.

Bilevel optimization

This area studies situations, where there is one problem nested in another. A classic example of this is the *toll setting problem*, where in order for the government to know how much toll they will collect from the highway network, they need to calculate whether the optimal route for each traveller, given his constraints and toll price, uses the toll roads. Article [17] describes the formulation of this problem as well as provides an overview of the applications.

This method is used to address uncertainty via the min-max approach. A subproblem with the opposite objective is inserted into the main problem. The feasible region of the inner problem is only allowed to deviate from the feasible solutions of the outer problem to some predefined degree. These formulations then provide results, which have the best worst-case when considering the allowed deviations. This approach is used in [18], where the researchers aim to find a flow which has maximal remaining value after a given number of edges is removed from the graph.

Other methods

Of the rest of the methods used in proactive scheduling, *distributionally robust optimization* certainly deserves mention. This approach studies problems, where the uncertainty follows a certain distribution, which itself is subject to uncertainty. Since this is not our case, we have not investigated this area further. For a deeper explanation, see [19]. Another such field is *fuzzy programming.* These approaches are based on fuzzy mathematics, an extension of set theory and logic. Simple yes or no concepts such as truth and false and membership in a set are extended to the interval between zero and one, which allows for some novel applications in scheduling and optimization, but we consider them beyond the scope of this thesis. An example and starting point for further reading is the article [20], which describes a fuzzy approach to optimization of water resources usage.

Lastly, the field of *sensitivity analysis* studies how the uncertainties in the input affect the output of a given problem; it is useful for identifying which parameters of the model need to be carefully examined and which, even though uncertain, affect the output very little for nearly all of their realizations. However, few studies which focus on scheduling have been performed.

Reactive scheduling

As mentioned, no scheduling system can work in the long run without containing a reactive part. A deeper overview of the current state of literature in this field can be seen in [21], where the authors also come up with their own proactive-reactive approach for the *resource-constrained project scheduling problem*. As this area is not the primary focus of our work, for the reactive part, simply running our proactive schedule generating model again with updated inputs will suffice.

1.1.2 Laboratory scheduling

Since no other work so far has focused primarily on scheduling of the centrifuge, the related works in this field serve mostly as a way to familiarize ourselves with the environment where our problem is located.

In the thesis [22], the author focuses on the very same objective as we do, reducing turnaround time (TAT). The work explains the processes in a histopathology laboratory in great detail, and based on them puts several improvement techniques into practice in a hospital in Sweden, while measuring their effectivity. However, this thesis is written from more of a management and healthcare point of view, rather than a mathematical one.

Paper [23] gives insight into the preanalytic process, which is not our primary focus, but makes the interesting point that it is the phase, where most errors are made.

Another interesting recent article is [24], where the authors consider a very similar parallel batch scheduling problem with an online system. However, their focus is on the analyzers, not the centrifuge. They employ heuristics to obtain their schedules for models both with unbounded and bounded capacities and provide proof that their results lie within a given multiple of the optimal value of the offline version.

Article [25] gives a glimpse at where the field might look like in the future. Researchers study optimal schedules of mixture preparation for microfluidic biochips, devices which are able to integrate some functions of biochemical analyzers into a single integrated circuit.

Chapter 2

Problem Statement

In this chapter, we are going to familiarize ourselves with the environment our problem is set in as well as formally define the problem we will address.

In our thesis, we focus on optimizing the schedule of a medical laboratory, a place whose main function is to analyze samples received from various parts of the hospital and test them to provide key results for aiding the diagnostic process, selecting the appropriate treatment and prevention of disease.

Scientifically speaking, what a layman would call a ”test”, falls into the field of pathology [26], a branch of medicine which looks into the causes and effects of diseases and injuries, with the focus being on diseases in this case. Nowadays, there is a wide variety of materials that can be tested and tests that can be performed, which has led to a great degree of specialization in this field. The two main subdivisions are *anatomic pathology*, which focuses on the testing of tissue samples and *clinical pathology*, which analyzes bodily fluids. This area is further divided into:

- Clinical microbiology, which investigates infectious diseases, which can be caused by viruses, bacteria, fungi, parasites or prions. It studies how these infectious agents attack the body and how to diagnose, treat and prevent them.

- Chemical pathology, also called clinical biochemistry, focuses on diagnostics based on the testing of bodily fluids, mainly blood. It has grown from using simple chemical reactions to test various components of blood to specialized laboratories which can perform up to 700 different tests.

- Molecular genetics, whose main interest is to study the genetic code of an individual to detect possible risk factors and future diseases and to help choose the appropriate treatment.

- Hematology, which studies diseases directly linked to blood, such as blood cancers or bleeding disorders, as well as preparation and storage of blood to be used in transfusions.

As is often the case in medicine, all of these branches are closely related to each other, and certain procedures might be categorized under more than one of these subdivisions. In practice, medical laboratories range from smaller ones specializing in only a few testing methods, as is the case of reproductive clinics, or much larger institutions found in hospitals, which may even have several subdepartments where each is responsible for its own set of testing methods.

This is also the case of Královské Vinohrady University Hospital, whose Institue of Medical Laboratory Diagnostics consists of five departments. They have provided Karel Gavenčiak, the author of [1], with real anonymized data from their laboratory. As he describes in his thesis, their Clinical Biochemistry department is the one with one of the strictest requirements in terms of speed and quality of the testing, which is the reason why its data was analyzed. In our work, the insights Gavenčiak gained by performing his data analysis will be crucial for understanding the uncertainty we are addressing. Moreover, they will define the computation speed requirements of our approaches.

2.1 Laboratory workflow

Providing medical professionals and patients with fast and reliable test results is an important part of every health care facility. However, it is a complex process which has to take into account several factors such as the locations where the samples are collected, means of transport, the testing equipment itself, personal resources and cost-effectiveness.

In this section. we are going to describe this process to show what will be the focus of our work. As previously mentioned, for a more detailed description of the laboratory workflow see [1], in which the author also performs thorough data analysis as well as comes up with possible speedups for a specific biochemical analyzer machine.

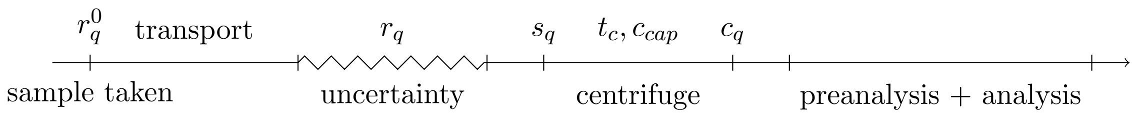


Figure 2.1: Simplified schema of the workflow

The very first step in this process is taken by a doctor, who decides based on the patient's symptoms and history, to perform a specific set of tests, which would then help with the diagnostic process. He fills out a testing request either in paper form or in an electronic system.

Next, as seen in Fig. 2.1, the sample has to be taken from the patient. This moment is represented by time r_q^0. In our data, we only have records associated with blood samples, which are always taken from the patients by nurses somewhere inside the hospital. However, other bodily fluids might be brought by the patients into the hospital anytime.

The samples then need to be transported to the laboratory for processing, which may be done by a person, who takes a larger group of samples in a bag or via a pneumatic tube system if there is one. A pneumatic tube system usually takes less than five minutes to deliver the sample to the laboratory and is usually able to send individual samples. Whereas when a member of staff is required to carry the samples, he usually performs only a few trips per day, transporting a large number of tubes at a time. The time when the sample arrives in the laboratory is represented by r_q.

It is this part of the workflow, as seen in Fig. 2.1, which creates the uncertainty we will attempt to model. When the sample is taken, the laboratory is notified via its information system. However, it takes some time for the sample to reach the laboratory. During the transport, delays may occur due to a wide variety of human factors: a sample might be forgotten, the staff likes to load the pneumatic tube capsule with more samples at a time or perhaps the person transporting the samples was needed to perform a different task.

Luckily, the data we received from the Clinical Biochemistry Department of the Institute of Laboratory Diagnostics contains precisely these times, that is when the sample was taken from the patient and when it reached the laboratory.

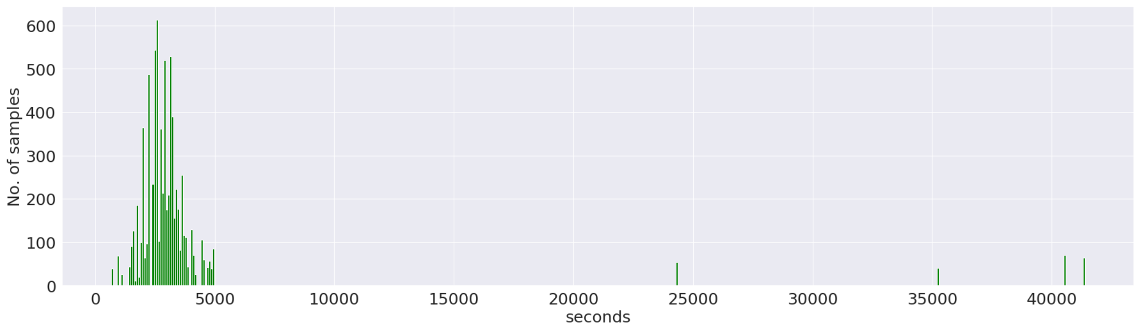


Figure 2.2: Sample transportations times for a given department

A specific example of the transport times during one month is shown in the following histograms in Fig. 2.2, and Fig. 2.3, the data is taken from samples originating from the same department of the hospital, whose transport times are likely to be linked.

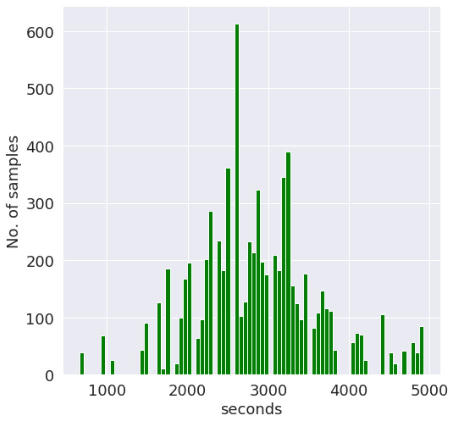


Figure 2.3: Transportation times without outliers

As we can observe, there are some outliers; these are of little importance to us as they are most likely the result of some human errors while working with the system. It is improbable, although possible that the samples were simply forgotten. When we plot a histogram of the data without outliers, we can see that the transportation times for this department can be estimated by a normal distribution with a mean of approximately 2700 seconds. We can also conclude that nearly all of the samples from this department were delivered by the time 5000 seconds have passed. Based on thorough analysis, samples from other departments also follow their own normal distributions, which is a key fact, which later allows us to model the uncertainty straightforwardly.

Once in the laboratory, a majority of the tubes have to go to the centrifuge, as seen in Fig. 2.1, where they are spun at high speed to separate the blood serum, the part used for most of the tests, from the clot. These machines usually take in batches of several test tubes, commonly 5 or 10, and the whole process typically takes 10-15 minutes. The latter parts of the testing process are unlikely to cause any significant delays, but if a tube is not assigned to the current batch, it will have to wait at least 10 minutes for the next one, or half that on average, if there are two centrifuges. This is the core of our problem and the process which we will be attempting to analyze and speed up.

The next step is called ”preanalysis”, see Fig. 2.1, and consists of simple manual labor tasks, such as pipetting a part of a sample into an empty test tube in order to duplicate it, lid removal and barcodes labelling. Some tubes, due to their physical build or test method requirements, skip this stage.

Finally, the samples enter the biochemical analyzer, in which they visit the chambers, where the specific chemical reaction needed to perform the desired test takes place, see Fig. 2.1. Only a tiny amount from the sample is used for each reaction, so there is always enough blood to perform all the necessary tests. This is the part of the process that has been thoroughly analyzed in [1].

Once the analyzer has finished, the laboratory personnel is notified and has to verify that the result is correct. All tests also require verification by a doctor. This is the final step of the whole process, after which the results are uploaded into the hospital system and may be viewed by the prescribing physician.

The laboratory staff which has provided us with the data performs this process on an average of 800 samples per day on workdays and 550 samples on weekends [1]. On these samples, the laboratory performs an average of 10800 tests per day. However, the load on the laboratory is not distributed evenly throughout the day, as can be seen in the following Fig. 2.4.

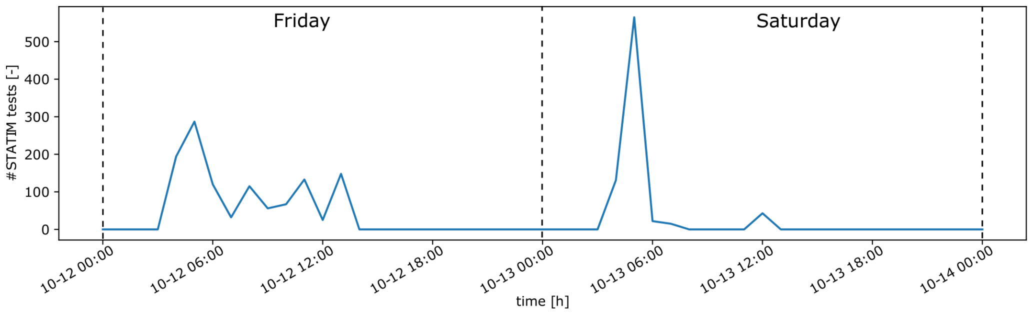


Figure 2.4: Usage of the laboratory

A peak which occurs during the morning hours is clearly visible. Fewer new samples are then received later throughout the day. The peak is even more pronounced during the weekend. The reason for this is that the data shown is only for the statim priority samples, which will be described in the following section. However, samples with less priority than these are not processed during the weekends, which forces the doctors who would like to know the results on the same day to issue a sample of higher priority than they normally would during the weekday. The overall load on the laboratory is very similar to the Fig. 2.4, with the morning peak, because this is the time most samples are taken by the nurses.

2.2 Scheduling problem

In the following section, we will attempt to formalize our problem and then describe it using Graham's notation.

In our problem, a task will be equivalent to one test tube passing through the centrifuge. The set of all tasks is $\mathcal{T}$. Each $q \in \mathcal{T}$ was taken at the time r_q^0 and arrives at the laboratory at time $\tilde{r}_q$, which is an unknown parameter. In our problem, the work on a task starts in the moment s_q, when it is assigned to a batch inside the centrifuge and starts spinning. The work on a task is completed at c_q, when it has exited the centrifuge. In our work, we do not consider the time it takes to perform the upcoming analysis, because the time a sample spends waiting on the centrifuge and then inside has negligible influence on the duration of the rest of its analytic process. This means that we can focus only on the optimization of this part of the workflow.

In our model, for the sake of simplicity, we will consider only one centrifuge. The centrifuge has capacity c_{cap} and its running time is t_c. The test tubes enter the centrifuge in batches. The set of all batches is $\mathcal{B}$. Each $b \in \mathcal{B}$ has a start time s_b^c and contains at most c_{cap} tubes.

One of the most important factors when working in a biochemical laboratory is prioritization, some tests are performed as a part of a routine checkup, whereas others are needed when treating a patient who is currently in life-threatening condition. In our problem, we will follow the setting of an actual laboratory, that we are trying to emulate. We will consider three levels of priority, called routine, statim and vital.

Routine samples are the least prioritized ones; these are the only type, that is not processed during weekends and holidays. According to [1], in our data, they represent about 54% of the samples tested. The other two types are processed continuously. Statim priority level is used when the test results are needed the same day. The laboratory sets even higher standards, its performance goal is to analyze 80% of these samples within 60 minutes of their arrival into the laboratory and 98.5% samples within 120 minutes after arrival. These samples form around 45% of the laboratory load. The last and the most prioritized sample type is vital, which is used for the most urgent tests, usually for patients who are in life-threatening situations. These samples have the highest priority, but are seldom used, as the circumstances which justify their usage are rare. They make up less than 1% of all total volume tested.

In order to take these priorities into account in our problem, we will multiply the contribution of each sample to the objective function by a constant w_q, which will be dependent on the priority level of the given sample. One for routine, two for statim and four for vital.

We have one machine that processes the tasks. The times $\tilde{r}_q$ are subject to uncertainty. Our objective is to minimize the total weighted turn around time, sometimes called total flow time or total lead time, which is defined as the difference between completion and release times. Our problem type in Graham's notation is:

$$1|\tilde{r}_q, batch|\sum_{q \in \mathcal{T}} w_q(c_q - r_q^0) \tag{2.1}$$

Chapter 3

Proposed solutions

In this chapter, we first examine how the uncertainty can be modelled. Afterwards, we propose a simple *integer linear programming* (ILP) model, which can schedule deterministic instances of our problem. We then add proactivity to this method with the help of state of the art approaches while discussing their possibilities and drawbacks. Finally, the reactive component of the solution is presented. As mentioned in Chapter 1, we focus on the methods used to generate the proactive schedule, while keeping the reactive part reasonably simple.

3.1 Modelling the uncertainty

The first step of addressing our problem is to decide how to use the knowledge we have gained so far to model the uncertainty. We are looking for an approach that adds very little complexity while providing a reasonable level of protection. Based on our literature review, three methods seem to be viable: replacing the uncertainty with estimated values, constructing uncertainty sets and a flow-based representation.

The simplest and most straightforward approach is to remove the uncertainty from the model altogether. Our unknown release times would be replaced by values we expect the samples to arrive. The advantage of this is that the uncertainty adds no overhead at all to the model. However, if what we have guessed is too optimistic, our model can often become infeasible, forcing a reaction. On the other hand, if our estimates are too pessimistic, suboptimal plans will be generated.

In our case, we might make use of the data and base our estimates on the fact presented in Chapter 2, where we explain that the transport and thereby arrival time of each request seems to be following a normal distribution: $r \sim \mathcal{N}(\mu, \sigma^2)$. We can then define the probability that the sample will arrive that we want to consider in our model. The quantile function of the given distribution can then give us the needed estimates. Of course, by using higher probabilities, our model will be more robust, but the values of the quantile function will be higher.

The uncertainty set approach builds on the assumption that the realizations of the uncertainty are constrained to some closed set; in our case, an interval is sufficient, i. e.: $\tilde{r}_q \in [r_q - \hat{r}_q, r_q + \hat{r}_q]$. However, similarly to removing the uncertainty, setting hard constraints on the uncertainty might create a schedule which will become infeasible often.

The final discussed approach is based on flows. Our problem may be viewed as a generalized assignment problem, where one batch can be assigned to multiple requests based on its capacity. We can then create a network for this problem, which contains the source, the sink and a bipartite graph, where one side corresponds to the tubes $\mathcal{T}$ and the other to the batches $\mathcal{B}$. This graph is depicted in Fig. 3.1.

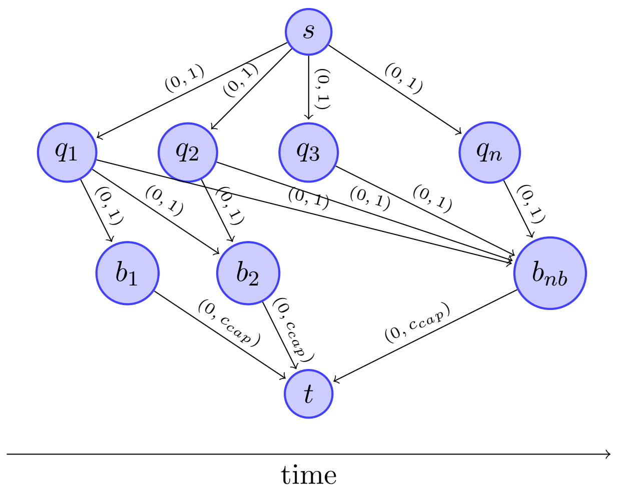


Figure 3.1:
Graph representation of our problem

The source connects to all the nodes representing the samples, which are then connected to the batches only if the given sample can be put into the given batch. That is only when its release time is later than the start time of the given batch. However, for reasons described later, the graph has to be defined before we attempt to calculate a solution. This forces us to assume that the centrifuge runs with no pauses in between batches. In this model, the approach to uncertainty is slightly different from the two previously discussed. When a sample arrives later than expected, an edge of the graph becomes unusable. This means that by looking for solutions which retain satisfying objective function value even in the worst case, when some number of edges is removed from the graph, we obtain robust solutions.

3.2 Schedule generation

The key part of designing a solution to our problem, which can run continuously is the generation of the schedule. In our work, we focus on building a schedule which includes protection against the uncertainty, called proactive or robust schedule. In this section, we examine several state-of-the-art approaches based on the models of uncertainty we have discussed and then use their ideas in generating our own proactive schedule.

3.2.1 Deterministic approach

The most direct approach for solving uncertain problems is to ignore the uncertainty. By solving the non-robust version, we get a glimpse at what would the real optimum look like under perfect conditions, when we would know the realization of the uncertainty beforehand. The downside of using this approach in practice is that even the slightest difference from what we are expecting can make the results suboptimal or infeasible. Upon analysis of the problem, we have come up with the following model to serve as a deterministic baseline.

The objective is straightforward; as mentioned previously, we minimize the turnaround time (TAT) weighted by the priorities of the samples w_q. Constraint (1) defines the completion time as the sum of start time and the duration of the centrifuge. Constraint (2) assures that the start of work on a sample will be scheduled after it is released. Constraints (3) and (4) bind the start time of the sample to the start time of the batch it has been assigned. Constraint (5) defines that a batch may only start after the previous one has finished. Constraints (6) specify that each sample has to be assigned to a batch, whereas constraint (7) ensures that the batch capacity will not be violated. Constraint (8) specifies that a sample is either assigned to a given batch or not. This is necessary because the nature of this model does not guarantee integral results if we used standard LP without integrality constraints. The final constraint (9) states that we want our timeline to start from zero.

$$\begin{aligned}
\min \quad & \sum_{q\in\mathcal{T}} && w_q(c_q - r_q^0) && && \\
\text{s. t.} \quad & c_q && = s_q + t_c && \forall q \in \mathcal{T} && (1)\\
& s_q && \geq r_q && \forall q \in \mathcal{T} && (2)\\
& s_q && \geq s_b^c - M(1-x_{qb}) && \forall q \in \mathcal{T}, \forall b \in \mathcal{B} && (3)\\
& s_q && \leq s_b^c + M(1-x_{qb}) && \forall q \in \mathcal{T}, \forall b \in \mathcal{B} && (4)\\
& s_{b+1}^c && \geq s_b^c + t_c && \forall b \in \mathcal{B} \setminus \{|\mathcal{B}|\} && (5)\\
& \sum_{b\in\mathcal{B}} x_{qb} && = 1 && \forall q \in \mathcal{T} && (6)\\
& \sum_{q\in\mathcal{T}} x_{qb} && \leq c_{cap} && \forall b \in \mathcal{B} && (7)\\
& x_{qb} && \in \{0,1\} && \forall q \in \mathcal{T}, \forall b \in \mathcal{B} && (8)\\
& r_q, s_q, c_q, s_b^c && \geq 0 && \forall q \in \mathcal{T} && (9)
\end{aligned}$$

Figure 3.2: The deterministic model

3.2.2 Stability radius

A relatively simple approach to protecting against uncertainty is to define a metric of how robust a given schedule is. For this, we define the stability radius Q of a schedule as the minimal difference between a task's start and release times, that is:

$$Q = \min_{q\in\mathcal{T}}(s_q - r_q)$$

The idea behind this is, that when we have a schedule with a stability radius of Q, we know that each task starts at least Q time units after we expect its delivery into the lab, which protects us against any number of delays smaller than Q.

At first glance, this might seem sound. However, the times r_q are still only estimates, so this approach is identical to the deterministic approach with the guesses incremented by Q. To increment the expected release times, of course, produces a more robust schedule, but at the cost of worse objective value. Such a realization, where all the samples would be very late rarely happens, which is why we would like to develop a model, which would retain most of this worst-case robustness while giving more optimal solutions.

3.2.3 Bertsimas, Sim robust counterpart

The idea of worst-case robustness has been studied for quite some time. The first notable approach is from 1973 by Soyster [11], whose approach has since become known as Soyster's method. In his work, he considers column-wise uncertainty on the constraint matrix. He defines convex sets K_j, which contain the possible realizations of columns $\mathbf{A_j}$ of matrix $\mathbf{A}$. He considers the following model:

$$\begin{aligned} \max \quad & \mathbf{c^T x} \\ \text{s. t.} \quad & \sum_{j=1}^{n} \mathbf{A_j} x_j \leq \mathbf{b} \quad \forall \mathbf{A_j} \in K_j, \forall j \\ & \mathbf{x} \qquad \geq \mathbf{0} \end{aligned} \tag{3.2}$$

The first constraint describes this worst-case robustness; it must hold for every possible combination of one column from each uncertainty set. The author then defines matrix

$$\bar{\mathbf{A}}, \bar{a}_{ij} = \sup_{\mathbf{A_j} \in K_j} A_{ij}$$

and proves that the program (3.2) is equivalent to the following one:

$$\begin{aligned} \max \quad & \mathbf{c^T x} \\ \text{s. t.} \quad & \bar{\mathbf{A}}\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{aligned}$$

Using this model in practice is elementary, the worst case of the uncertainty has to be established, and that is all that needs to be done. However, the drawback of this method is that it is over-conservative. To be protected against all considered realizations, a lot has to be ceded in terms of the optimal value. This has lead researchers to come up with methods, which would be able to precisely control the balance between these two objectives, which go against each other.

One such example is [12]. In this article, the authors propose a model which introduces very little computational complexity while allowing the control of the robustness on the level of individual constraints. Another promising property of this approach is that it is suitable for adding robustness to ILP models. However, as will be described later, due to the structure of our problem, this approach becomes identical with the previously discussed deterministic model. We are going to explain why that is the case. Let us will follow the author's derivation of the model. Firstly, they consider the nominal (deterministic) problem:

$$\begin{aligned} \max \quad & \mathbf{c^T x} \\ \text{s. t.} \quad & \mathbf{Ax} \leq \mathbf{b} \\ & \mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \end{aligned}$$

The researchers consider interval uncertainties, but prove, that even when the realizations are outside this interval, the model still gives good results. J_i is defined to be the set of coefficients a_{ij}, $j \in J_i$ which are subject to uncertainty. A symmetric distribution with mean equal to the nominal value is assumed, our data follow this assumption. Formally, the realization $\tilde{a}_{ij}$, $j \in J_i$ takes values in the interval $[a_{ij} - \hat{a}_{ij}, a_{ij} + \hat{a}_{ij}]$. For every constraint with uncertain parameters, a parameter $\Gamma_i \in [0, |J_i|]$ is introduced. This parameter allows us to control the level of robustness we want the model to have. However, as previously mentioned, more robustness inevitably leads to less optimistic results.

In order to ensure the feasibility of the model, the authors define $\beta_i(\mathbf{x}^*, \Gamma_i)$ as the protection needed to insert into constraint i to be feasible for all possible realizations, which leads us to the altered model.

$$\begin{aligned} \max \quad & \mathbf{c^T x} \\ \text{s. t.} \quad & \sum_j a_{ij}x_j + \beta_i(\mathbf{x}^*, \Gamma_i) \leq b_i \quad \forall i \\ & \mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \end{aligned} \tag{3.3}$$

The authors define their parameter Γ_i as the maximum sum of deviations $\frac{|\tilde{a}_{ij} - a_{ij}|}{\hat{a}_{ij}}$ from the nominal values, that can co-occur for the model to guarantee feasibility. This means that the protection can be expressed as:

$$\beta_i(\mathbf{x}^*, \Gamma_i) = \max_{\{S_i \cup t_i | S_i \subseteq J_i, |S| = \lfloor \Gamma_i \rfloor, t_i \in J_i \setminus S_i\}} \left\{ \sum_{j \in S_i} \hat{a}_{ij}|x^*| + (\Gamma_i - \lfloor \Gamma_i \rfloor)\hat{a}_{it}|x^*| \right\} \tag{3.4}$$

It can be seen, but the authors also provide a simple proof, that this protection is equal to the objective function of the following linear program:

$$\begin{aligned} \max \quad & \sum_{j \in J_i} \hat{a}_{ij}|x_j^*|z_{ij} \\ \text{s. t.} \quad & \sum_{j \in J_i} z_{ij} && \leq \Gamma_i \\ & 0 \leq z_{ij} && \leq 1 \quad \forall j \in J_i \end{aligned} \tag{3.5}$$

However, in our case, where the only constraints in the deterministic model affected by uncertainty are (2), the uncertainty concerns only the constant on the right-hand side. We can fix this by introducing a dummy variable, whose value will be constrained to 1 and multiplying it by the uncertain constant, instead of constraint (2) we would add

$$\begin{aligned} s_q - r_q y_q &\geq 0 \quad \forall q \in \mathcal{T} \\ y_q &= 1 \quad \forall q \in \mathcal{T}. \end{aligned}$$

The authors of the article then transform the model needed to calculate the protection (3.5) into its dual form. By strong duality, its objective function optimal value remains unchanged, still equal to the protection needed to be inserted into the constraint (3.4). Then they discuss the properties of the dual formulation and the original model (3.3), to prove, that they can incorporate this dual into the original problem to provide the necessary protection.

However, in our problem things simplify a bit more. By adding the dummy variable, whose value is constrained to one, we can replace the value of $|x_j^*|$ by one and given that we have only one uncertain parameter; the protection calculating model simplifies to:

$$\begin{aligned} \max \quad & \hat{r}_q z_{i1} \\ \text{s. t.} \quad & z_{i1} \leq \Gamma_i \\ & 0 \leq z_{i1} \leq 1 \end{aligned} \tag{3.6}$$

Which is trivial, and gives the necessary protection of $\hat{r}_q$ for $\Gamma_i > 1$ and $\Gamma_i \hat{r}_q$ for $\Gamma_i \leq 1$. This formulation would give us the same results as removing the uncertainty and guessing the release time, the only difference being that the guess is defined formally via the parameters $a_{ij}, \hat{a}_{ij}$ and Γ_i.

The main reason we believed that this method could be beneficial for our problem is by using the approach described in [27], where the authors sum some of the constraints with one uncertain parameter to obtain an aggregate constraint, which is redundant but allows them to control the robustness in a more sensibly. The dual variables from the aggregate constraint also need to be redistributed back to the constraints with one uncertain parameter in order to ensure feasibility.

However, in our case, even when adding such an aggregate constraint, the structure of our problem dictates, that when a sample is assigned to a batch, then that batch cannot start sooner than $r_q + \min(\Gamma, 1)\hat{r}_q$. This means that for any number and combination of aggregate constraints this formulation will be equivalent to the deterministic model with release times estimated as $r_q + \min(\Gamma_q, 1)\hat{r}_q$, where Γ_q is the Γ corresponding to the aggregate constraint in which request q is placed.

It seems that this approach is unfortunately not suited for our problem, where some variables are forced to the maximum of others, which are subject to uncertainty, which is precisely our case, where times s_q are forced to the maximum of all r_q that are assigned to it.

3.2.4 k-edges attack

This approach provides a slightly different perspective from the previous ones, we try to model the uncertainty based on the graph in Fig. 3.1. As mentioned, for this approach to work, we need to assume that the centrifuge runs continuously. Each edge has a cost equal to the waiting time of the sample before it is assigned to the batch. If we considered the deterministic version of this problem, we would search for the minimum cost flow, which saturates all the edges going out from the source.

With the uncertainty, our problem changes. We still need to find a flow with a rate equal to the number of requests, i.e. we want to schedule all of them. However, when a sample arrives later than the start time of a batch, it cannot be assigned to it, the corresponding edge from the graph in Fig. 3.1 becomes unusable. Therefore, we will be looking for a flow, which is to some degree immune to a removal of a given number of edges; a flow which has the best worst-case cost after removing all possible combinations of k edges. We can think of this as a problem with a nested subproblem which too has a subproblem nested in it.

In article [18], the authors solve a very similar problem, instead of minimum cost flow, they consider the maximum flow problem. Their approach is to transform the innermost problem into its dual form and then incorporate it into the middle problem, leading to a bilevel formulation. In our work, we will try to emulate their approach.

Formally, in the innermost level of our problem, we receive the initial flow f_{ij} and set S of the removed edges. We define its characteristic function $\chi_S(x)$, which returns one when $x \in S$ and zero otherwise. Let η_S^f be the minimum cost assignment of samples to batches in the uncertain graph, with the initial flow f, from which the edges in S were removed. This assignment must use the flow f on the edges which remained, but it must add flow to assign the samples, whose saturated edges were removed.

The second level represents the worst-case uncertainty. It attempts to find the set S with the worst value of η_S^f given the initial flow f_{ij}. The upper level aims to find the best initial flow among all the feasible flows. At last, the whole problem can be formally described as: $f^* \in \operatorname*{argmin}_{f \in \mathcal{F}} \max_{S, S \subseteq E, |S| = k} \eta_S^f$

To find this optimal initial flow, we have to start from the bottom. Let us consider the innermost problem. In order to simplify our approach, we can ignore the source and sink and focus only on the bipartite graph in between them. A linear program which calculates the value of η_S^f is presented in Fig. 3.3. Constraint (1) declares that each sample will be assigned. Constraint (2) assures, that batch capacity will not be violated. Constraints (3) and (4) define the new flow behavior. On removed edges, it is zero, whereas on the edges that remained, the flow cannot be removed if present, but it may be added, to compensate for an edge which was carrying flow and was removed. The final constraint (5) is redundant, based on the fact, that the original flow is between zero and one and so is $\chi_S(x)$, we only use it to for the construction of the dual problem. The constraint matrix of this problem is totally unimodular, which means that our results will always be integer even without strictly enforcing them.

$$
\begin{aligned}
\min \quad & \sum_{(i,j)\in E} && c_{ij} f'_{ij} && && \\
\text{s. t.} \quad & \sum_{(i,j)\in E} f'_{ij} && = 1 && \forall i \in \mathcal{Q} && (1)\\
& \sum_{(i,j)\in E} f'_{ij} && \leq c_{cap} && \forall j \in \mathcal{B} && (2)\\
& f'_{ij} && \leq 1 - \chi_S((i,j)) && \forall (i,j) \in E && (3)\\
& f'_{ij} && \geq (1 - \chi_S((i,j))) f_{ij} && \forall (i,j) \in E && (4)\\
& f'_{ij} && \geq 0 && \forall (i,j) \in E && (5)
\end{aligned}
$$

Figure 3.3: Minimum cost assignment

The next step is to incorporate this problem into the middle layer $\max\limits_{S, S\subseteq E, |S|=k} \eta^f_S$. This can be done by transforming the model in Fig. 3.3 which calculates η^f_S into its dual form, which is shown in Fig. 3.4.

$$
\begin{aligned}
\max \quad & \sum_{i\in\mathcal{Q}} \alpha_i + \sum_{j\in\mathcal{B}} c_{cap}\beta_j + \sum_{(i,j)\in E} (1 - \chi_S((i,j)))\gamma_{ij} + \sum_{(i,j)\in E} (1 - \chi_S((i,j))) f_{ij}\delta_{ij} && \\
\text{s. t.} \quad & \alpha_i + \beta_j + \gamma_{ij} + \delta_{ij} \ \leq c_{cap} \qquad \forall (i,j) \in E && (4)\\
& \alpha_i \in \mathbb{R} \qquad \forall i \in \mathcal{Q} && (5)\\
& \beta_i \leq 0 \qquad \forall j \in \mathcal{B} && (6)\\
& \gamma_{ij} \leq 0, \delta_{ij} \geq 0 \qquad \forall (i,j) \in E && (7)
\end{aligned}
$$

Figure 3.4: Dual of the inner minimum cost assignment

At the innermost level, both $\chi_S(x)$ and f_{ij} are constant, but to include this model into the middle one, where $\chi_S(x)$ is not constant, we first need to linearize the multiplications $\chi_S((i,j))\gamma_{ij}$ and $\chi_S((i,j))\gamma_{ij}$. We can do this by replacing the products with a new variable, which we then force to the result of the multiplication with additional constraints. Instead of $\chi_S((i,j))\gamma_{ij}$ and $\chi_S((i,j))\gamma_{ij}$ we would use ϵ_{ij} and θ_{ij} respectively and add the constraints in Fig. 3.5.

$$\begin{aligned}
&\epsilon_{ij} \leq \gamma_{ij} && \forall (i,j) \in E && (8)\\
&\epsilon_{ij} \leq \chi_S((i,j)) && \forall (i,j) \in E && (9)\\
&\epsilon_{ij} \geq 0 && \forall (i,j) \in E && (10)\\
&\theta_{ij} \leq \delta_{ij} && \forall (i,j) \in E && (11)\\
&\theta_{ij} \leq \chi_S((i,j)) && \forall (i,j) \in E && (12)\\
&\theta_{ij} \geq 0 && \forall (i,j) \in E && (13)
\end{aligned}$$

Figure 3.5: Constraints linearizing the multiplications

We can now construct the middle problem, with the innermost one incorporated into it.

$$\begin{aligned}
\max\ & \sum_{i \in \mathcal{Q}} \alpha_i + \sum_{j \in \mathcal{B}} c_{cap}\beta_j + \sum_{(i,j)\in E} \gamma_{ij} - \sum_{(i,j)\in E} \epsilon_{ij} + \sum_{(i,j)\in E} f_{ij}\delta_{ij} - \sum_{(i,j)\in E} f_{ij}\theta_{ij} \\
\text{s. t. }\ & (4) - (13) \\
& \sum_{(i,j)\in E} \chi_S((i,j)) \quad = k && (14)\\
& \chi_S((i,j)) \in 0, 1 \qquad \forall (i,j) \in E && (15)
\end{aligned}$$

Figure 3.6: The middle problem

Now, only the last step remains, constructing a bilevel problem, where the upper level selects a feasible flow which goes against the objective of the inner model in Fig. 3.6. However, the constraint (15) makes the inner problem an integer linear programming one. This area of bilevel optimization is called *bilevel mixed-integer linear programming* (BMILP) and is very difficult to solve. Although some approaches exist, such as [28], we consider beyond the scope of this thesis. Even though the idea behind this approach seems to be valid, the lack of a freely available BMILP solver makes the use of this method impossible.

3.3 Our approach

In this section, based on the knowledge described above, we propose a proactive-reactive scheduling framework, which provides a solution to our problem. Firstly, we compare the advantages and drawbacks of the methods of schedule generation discussed in this chapter in order to formulate our final proactive model. Afterwards, we define the reactive component of our framework, which handles all the possible manifestations of uncertainty in our model by reacting to them accordingly.

3.3.1 Proactive part

Considering the limitations of the approaches described in this chapter, mainly the computational intractability of the k-edges attack method and stochastic programming formulation; and the inapplicability of the Bertsimas, Sim approach, we have decided to use a modified deterministic approach to generate the schedule.

Due to a large number of integer variables, $|\mathcal{T}||\mathcal{B}|$, only an approach which adds little or no overhead to the deterministic model can provide the schedules quickly enough. However, even though our model is relatively simple, integer linear programming with this number of decision variables is computationally demanding for an average computer. We have conducted a series of experiments measuring the time our ILP solver needed to return optimal solutions for various sample sizes. Results can be seen in Chapter 4.

Because the laboratory receives the most considerable amount of samples in the morning hours, a few dozen of them might have to be kept waiting, while the centrifuge is running continuously. This could make the process of scheduling the next batches very demanding. However, a large proportion of these samples are either routine or have been taken very recently. When the laboratory is working at full capacity, no complicated algorithm is needed, as we would only load the centrifuge with the most priority samples first because those contribute to the objective function the most. With this in mind, we decided to limit the number of samples used in the calculation of the optimal schedule. We will call these *considered samples* and represent them by symbol $\mathcal{T}'$.

These samples, which we are taking into account in our model, can either be already physically present in the laboratory or on their way in transport with their exact arrival time unknown. We will refer to these groups as *available samples* and *expected samples*, symbolically $\mathcal{A}$ and $\mathcal{E}$ respectively.

Since the available samples are present, they are ready to be scheduled into a batch, but the question of how to estimate the release times of the expected samples remains. In this work, we have chosen to estimate the release time based on the fact, that it follows a normal distribution, whose mean and variance can be sampled from our data. This means that for each $q \in \mathcal{T}$, we will be able to sample $\mathcal{N}_q(\mu_q, \sigma_q^2)$, from the previous realizations of samples from the given department. Then, we will calculate our estimated value of r_q as the value of the quantile function for parameter $\Gamma_q \in (0,1)$, that is:

$$r_q = \Phi^{-1}_{\mathcal{N}_q}(\Gamma_q) \tag{3.11}$$

The parameters Γ_q allow us to define the degree of robustness we would like our model to have for each sample. However, fine-tuning the robustness of different types of samples is not our focus. Instead, we would like to examine how our model behaves for different parameters Γ, each of which will be applied to all the samples.

Based on these decisions, we propose an ILP model generating proactive schedules in Fig. 3.7. Constraint (1) remains the same as in the deterministic model in Fig. 3.2. It is not necessary to provide the optimal schedules, as the time it takes to process each sample is constant, and the optimal solution would remain the same if we subtracted t_c from the contribution of each sample to the objective. However, we need it to provide a comparison in results between models with different parameters, as we focus mainly on TAT as the performance indicator. Constraints (2) and (3) define the availability of the samples; work on the available samples might start immediately, whereas the expected samples' arrival time is estimated as described in Eq. (3.11). Constraints (4) and (5) remain the same as in Fig. 3.2 and force the start time of the sample to be equal to the start time of the batch it has been assigned. Constraint (6) states, that if there is a batch currently running, then the first batch of the new schedule has to be started after it has finished. Constraints (7) define this for the rest of the batches. Constraints (8) - (11) are unchanged from Fig. 3.2 and serve, in ascending order, to ensure, that every sample is scheduled; to ensure, that the batch capacity is not violated; to declare the decision variables as integer and to make our current timeline centered around zero. The final constraint (12) explains that we are limiting the number of samples considered when calculating the optimal schedule.

$$
\begin{aligned}
\min \quad & \sum_{q \in \mathcal{T}'} w_q(c_q - r_q^0) & & \\
\text{s. t.} \quad & c_q = s_q + t_c & \forall q \in \mathcal{T}' & \quad (1)\\
& s_q \geq \Phi^{-1}_{\mathcal{N}_q}(\Gamma_q) & \forall q \in \mathcal{E} & \quad (2)\\
& s_q \geq 0 & \forall q \in \mathcal{A} & \quad (3)\\
& s_q \geq s_b^c - M(1 - x_{qb}) & \forall q \in \mathcal{T}', \forall b \in \mathcal{B} & \quad (4)\\
& s_q \leq s_b^c + M(1 - x_{qb}) & \forall q \in \mathcal{T}', \forall b \in \mathcal{B} & \quad (5)\\
& s_0^c \geq s_{-1}^c + t_c & \text{if } \exists s_{-1}^c & \quad (6)\\
& s_{b+1}^c \geq s_b^c + t_c & \forall b \in \mathcal{B} \setminus \{|\mathcal{B}|\} & \quad (7)\\
& \sum_{b \in \mathcal{B}} x_{qb} = 1 & \forall q \in \mathcal{T}' & \quad (8)\\
& \sum_{q \in \mathcal{T}'} x_{qb} \leq c_{cap} & \forall b \in \mathcal{B} & \quad (9)\\
& x_{qb} \in \{0, 1\} & \forall q \in \mathcal{T}', \forall b \in \mathcal{B} & \quad (10)\\
& r_q, c_q, s_b^c \geq 0 & \forall q \in \mathcal{T}' & \quad (11)\\
& |\mathcal{T}'| \leq r_{max} & & \quad (12)
\end{aligned}
$$

Figure 3.7: Proactive schedule generating model

3.3.2 Reactive part

A proactive schedule is an essential part of the optimization of long-term processes, but as was discussed mainly in Chapter 1, it is rarely enough. So is the case with our problem, where we have to take into account several events, which will adversely affect our current schedule and require an appropriate reaction. The whole process is summed up in the flowchart in Fig. 3.8. When a schedule is generated, it remains unchanged, until one of four events which we are going to describe prompts a reaction.

First such possibility is that a sample arrives into the laboratory. When this happens, it is marked as arrived so that we know that it is ready to be assigned into a batch immediately. Another disruption which is fairly simple to handle is when a sample should have arrived according to our estimate, but it has not yet done so. In this case, we simply reestimate its arrival time, by incrementing the current value of its parameter Γ_q by $\frac{1}{2}(1 - \Gamma_q)$. Similarly, when a new sample is registered in the laboratory information system, its expected arrival time is estimated based on the default value of the parameter Γ. and stored in the model.

The last event which prompts a reaction is the planned start of a batch. When this happens, we need to make sure that all the samples that are planned to go into the batch are present. If not, the batch was waiting on the last sample(s), and it was late. In this case, we first reestimate the release time of the late sample and ignore the start of the batch. We do this because it is unclear what to do since the problem domain has changed due to the late sample. If all the samples are present, we simply start the batch and remove the samples it has been assigned from the model.

On all four of these occasions, the problem domain is changed. This means that the optimal schedule might have changed as well. In order to make sure that we are taking the best possible course of action, we need to rerun the scheduling model. As mentioned in Chapter 3, we consider only a predefined number of samples which are the most likely to influence the sum of weighted TATs when calculating a schedule. This is done to ensure the computational feasibility of our approach.

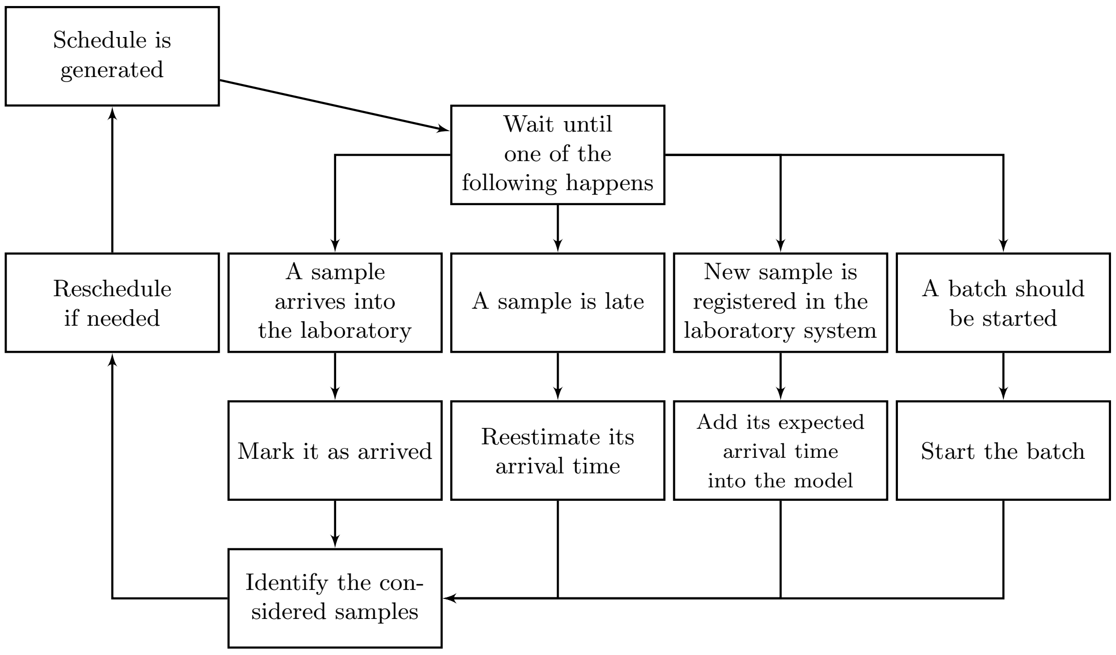

Schedule is generated
Wait until one of the following happens
Reschedule if needed
A sample arrives into the laboratory
A sample is late
New sample is registered in the laboratory system
A batch should be started
Mark it as arrived
Reestimate its arrival time
Add its expected arrival time into the model
Start the batch
Identify the considered samples

Figure 3.8: Flowchart of the proactive reactive scheduling framework

The considered samples $\mathcal{T}' \subseteq (\mathcal{A} \cup \mathcal{E})$ consist of the arrived and expected samples. A sample is added to expected samples when it is registered in the laboratory system, and its arrival time is estimated. When a sample actually arrives, it is removed from the expected samples and added into the arrived samples. We reconstruct the set $\mathcal{T}'$ before each rescheduling from the first r_{max} members of the set $\mathcal{A} \cup \mathcal{E}$ which we strictly order according to the operator $\prec$ defined as follows: $a \prec b$ if a is of higher priority than b, i. e. vital has the highest priority, and statim has a higher priority than routine. If there are two samples of the same priority, then $a \prec b$ if a has arrived and b has not. If this is not enough to determine the order, then $a \prec b$ if $r_a^0 < r_b^0$. Finally, if these three conditions cannot define the order of two samples, the one with smaller ID takes precedence. If an event has changed the problem domain, but the set $\mathcal{T}'$ has not changed after being reconstructed, then rescheduling is not needed.

Chapter 4

Results

In this chapter, we present the numerical results of our approach. Firstly, we discuss the runtime of the schedule generating ILP model. Afterwards, we describe the behavior of our model on small scale synthetic instances, where we can observe it nicely. Finally, we present the results of an experiment based on one day in a medical laboratory.

4.1 Computation time

To evaluate the speed of our model, we have used the Matlab R2019a program with the YALMIP toolbox [29] with Gurobi version 9.0.0. We have run the model described later 100 times on random data for each scenario and measured the time it took to solve it to optimality. We provide the mean (μ), standard deviation (σ) and the maximum of these times (max) in seconds. All the experiments have been performed on an Intel Core i5-8250U.

In each of the scenarios with defined $|\mathcal{T}|$, $|\mathcal{B}|$ and c_{cap}, we generate samples with expectations of arrival times distributed uniformly in the interval $[0, |\mathcal{T}|]$, the sample taken timers r_q^0 are defined as $r_q - t$, where t is taken from a uniform distribution between $[0, \frac{1}{2}|\mathcal{T}|]$. The centrifuge running time is defined as $\frac{|\mathcal{T}|}{|\mathcal{B}|}$. This setup ensures that the samples are neither far apart nor close together, which would make scheduling them easy. Priorities are also generated randomly based on the data analysis (1% vital, 45% statim and 54% routine).

scenario	$\lvert\mathcal{T}\rvert$ [-]	$\lvert\mathcal{B}\rvert$ [-]	c_{cap} [-]	$\mu[s]$	$\sigma[s]$	$\max[s]$
1	5	2	3	0.32	0.08	1.09
2	10	3	4	0.45	0.03	0.57
3	10	4	3	0.50	0.05	0.81
4	15	4	4	1.00	0.65	4.66
5	15	2	8	0.49	0.04	0.65
6	20	5	4	24.10	42.04	292.74
7	20	3	7	1.71	0.83	5.12
8	30	3	10	6.07	4.86	28.24
9	30	6	5	958.26	1302.12	3853.94
10	50	5	10	?	?	?

Table 4.1: Running times of our model

In scenario 10 in Table 4.1, during the evaluation of the first model, the solver managed to get to around 10% duality gap after 5500 seconds, after this point it was able to make very little progress, a model of this size is just intractable for an average computer in a reasonable time limit.

As can be observed in Table 4.1, the models which contain more samples need more time to be solved to optimality, as one would expect. However, it is intriguing too see the influence of the number of batches on the runtime. This is best seen in the drastic difference between scenarios 6 and 7. We have decided to investigate this further, by focusing only on the case, where we are considering 20 samples:

scenario	$\lvert\mathcal{T}\rvert$ [-]	$\lvert\mathcal{B}\rvert$ [-]	c_{cap} [-]	$\mu[s]$	$\sigma[s]$	max$[s]$
1	20	2	10	0.59	0.06	0.72
2	20	3	7	1.39	0.88	4.91
3	20	4	5	4.23	4.26	31.56
6	20	5	4	24.10	42.04	292.74
5	20	6	4	9.37	11.15	68.51
6	20	7	3	135.54	379.33	2221.73
9	20	10	2	697.36	512.58	7953.29
10	20	20	1	890.69	907.40	4905.70

Table 4.2: Running times of our model

It can be observed in Table 4.2 that increasing the number of batches greatly increases the time needed to solve the model to optimality. To use our model continuously throughout the day, we will have to limit the number of batches and requests it considers. However, scheduling only three or four batches ahead is more than enough in practice. As this corresponds to approximately 40-60 minutes of laboratory runtime.

4.2 Small instances

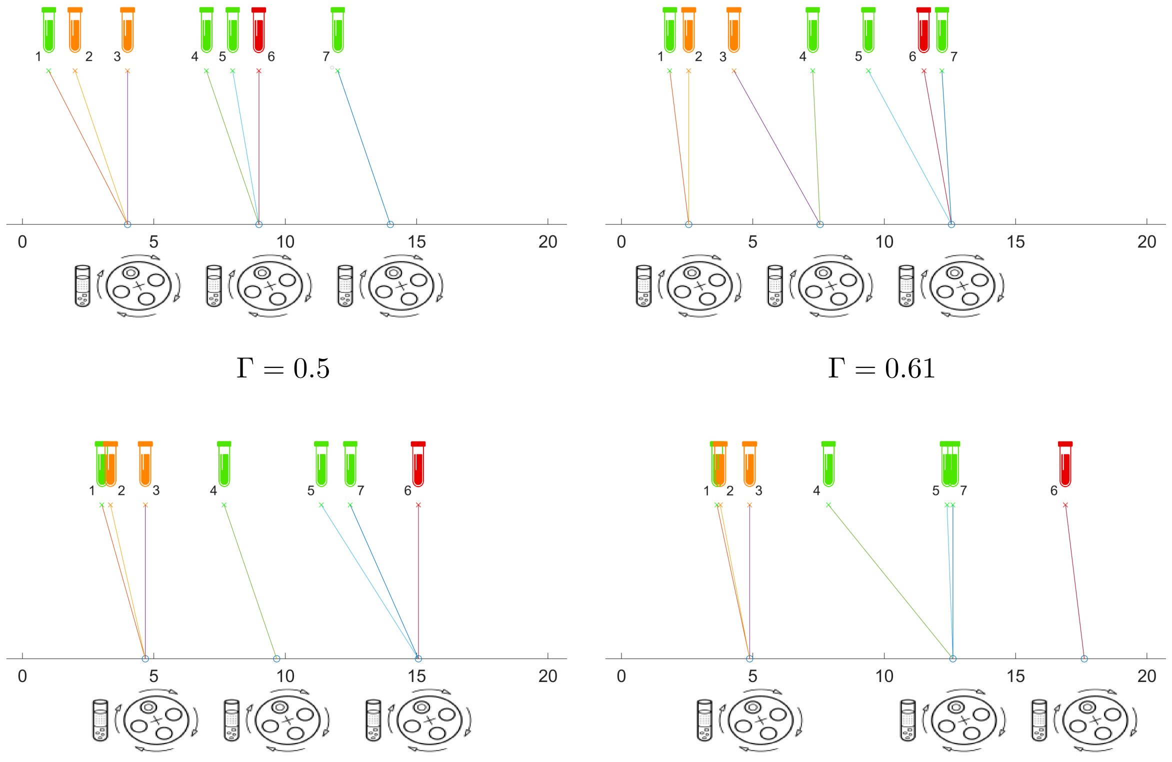
The behavior of the schedule generating algorithm might be best illustrated by optimal schedules of a small instance for different values of parameter Γ.

To create the schedules shown in Fig. 4.1, we have defined $|\mathcal{T}| = 7$, $|\mathcal{B}| = 3$, $c_{cap} = 3$ and $t_c = 5$. Routine samples are shown in green, statim and vital in orange and red, respectively. All samples have been taken and are in transport. The times they were taken as well as their expected arrival time distribution parameters are shown in the following Table 4.3.

#	priority	r_q^0	μ_q	σ_q
1	routine	-1	1	3
2	statim	-6	2	2
3	statim	-12	4	1
4	routine	-5	7	1
5	routine	-7	8	5
6	vital	-2	9	9
7	routine	-2	12	0.7

Table 4.3: Synthetic data

In Fig. 4.1 we present optimal schedules for four different values of the parameter Γ. We plot the expected release times of the samples, the times when the batches should be started and the optimal assignment. We calculated the schedule for all values of $\Gamma \in \{\frac{i}{100}\}, i = 50, 51, .., 99$ and decided to show these values, because they represent a change of assignment from the previously calculated schedule. The effect of robustness can be clearly seen. The larger the values of Γ, the larger the results of the normal quantile function and the later the samples are expected. This is visible the most on sample number six; whose expected arrival time has a large standard deviation. We can also observe the effect this sample has on the overall schedule. Because it contributes to the objective the most, it is scheduled very soon after its arrival. In the schedules for $\Gamma = 0.5$ and $\Gamma = 0.75$, the batch is scheduled immediately after its arrival. For the case of $\Gamma = 0.61$, it is scheduled to wait shortly for sample seven. Finally, in the case of $\Gamma = 0.81$, it has to wait until the centrifuge finishes the processing of the previous batch.



$\Gamma = 0.5$ $\Gamma = 0.61$

$\Gamma = 0.75$ $\Gamma = 0.81$

Figure 4.1: Proactive schedule for different values of Γ [30]

As mentioned throughout this thesis, greater robustness almost always leads to worse objective function value. To investigate this, we have calculated the optimal objective value of this instance for every $\Gamma \in \{\frac{i}{100}\}, i = 50, 51, .., 99$ and plotted the results in the following Fig. 4.2.

It is clearly visible that the function plotted in Fig. 4.2 is non-decreasing. This can be expected, as the quantile function of a normal distribution is increasing. For a greater Γ, we get later expectations of the release times, which lead to schedules with worse objective value.

However, the tradeoff is not constant. Up until around $\Gamma = 0.8$, we get a relatively linear growth. For $\Gamma = 0.8$, our objective has worsened by about 25%, which seems acceptable, considering the robustness this provides. After this point, the objective value starts to worsen more noticeably, which makes the tradeoff for increased robustness less attractive.

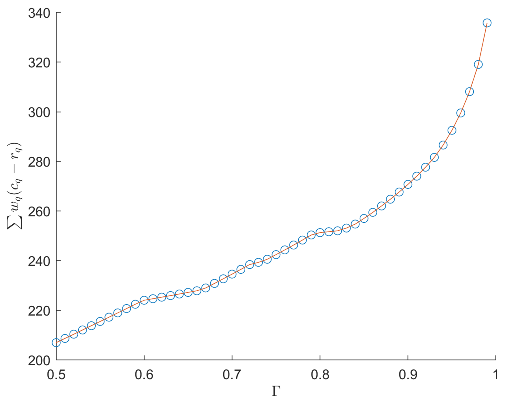


Figure 4.2: The objective value based on the parameter Γ

4.3 Simulations

This final experiment aims to evaluate the overall effectiveness of our proactive-reactive framework if it were deployed into an actual laboratory. We extracted the times r_q^0 when the samples were taken and r_q when they arrived into the laboratory from the data we have been provided, and we have restricted these only to samples that have come throughout one given afternoon.

This has been done based on the facts presented in Chapter 2, mainly in Fig. 2.4, where we can see that during the morning peak hours, the centrifuge is working continuously and scheduling it is not a complicated task, as we can load it with the most priority samples. On the other hand, during the afternoon hours, when there arrives approximately one sample per minute on average, it is much harder to decide, whether to start a batch or keep waiting for a higher priority sample.

Based on the results presented in Table 4.1 and Table 4.2, we have decided to limit the number of considered samples to 30. To stay true to the conditions in the laboratory, we define $c_{cap} = 10$ and $t_c = 600(s)$. In our experiment, we estimated the values μ_q and σ_q from the rest of the data. Subsequently, we have run our scheduling framework on 50 instances with the same r_q^0, but with the arrival times randomly sampled from their respective distributions in each scenario.

To evaluate our schedules, we have defined two metrics, called robust price (RP) and robust benefit (RB):

$$RP_p(\Gamma) = \frac{\mu_p(\Gamma) - \mu_p(0.5)}{\mu_p(0.5)}$$

$$RB_p(\Gamma) = \frac{\sigma_p(0.5) - \sigma_p(\Gamma)}{\sigma_p(0.5)}$$

We define the RP for samples of priority p and robustness Γ as the relative difference of their average TAT $\mu_p(\Gamma)$ over all the scenarios from the baseline determined by the average TAT for $\Gamma = 0.5$. Similarly, RB of priority p and robustness Γ is defined as the relative difference of the average TAT standard deviation $\sigma_p(\Gamma)$ from the baseline TAT standard deviation from $\Gamma = 0.5$. We provide the results for routine and statim samples in the following table for five different values of Γ. We represent routine and statim priorities by indices rut and st, respectively.

As can be seen in Table 4.4, the statim samples do not seem to be affected by the change in the parameter Γ. This is probably because their higher priority can manifest even through different levels of robustness. However, the routine samples TAT average and standard deviation change substantially. For larger values of Γ, we pay the robust price in the form of worse average TAT.

Γ	0.5	0.6	0.7	0.8	0.9
$\mu_{rut}[s]$	10747.2	11238.2	11982.8	12290.3	12514.4
$\sigma_{rut}[s]$	4386.5	3810.1	3606.7	3105.8	3082.8
$RP_{rut}[\%]$	0	4.6	11.5	14.4	16.5
$RB_{rut}[\%]$	0	13.1	17.8	29.2	29.7
$\mu_{st}[s]$	7671.3	7613.1	7677.3	7772.3	7826.7
$\sigma_{st}[s]$	7471.9	7423.6	7410.0	7333.8	7352.9
$RP_{st}[\%]$	0	-0.8	0.1	1.31	2.0
$RB_{st}[\%]$	0	0.7	0.8	1.9	1.6

Table 4.4: Robustness price and benefit

However, we get the robust benefit of smaller variation of these times. Moreover, the benefit-cost ratio seems to be favourable in our case, where we can decrease the TAT standard deviation by 13% or 29% by increasing the average routine TAT by only 5% or 14%, respectively. There were no vital samples in the data used for the experiments. However, given their rarity, we do not consider them necessary to evaluate the overall effectiveness of our approach.

One thing we would like to mention is that the laboratory goals mentioned in Chapter 2, mainly that 98.5% of samples should be processed in 120 minutes, do not take into account the time in transport, only the time spent in the laboratory. The statim samples in our instances spent on average 5825.9 seconds in transport. We did not measure the times the samples have spent in the laboratory, but it is likely, that these goals were fulfilled.

Chapter 5

Conclusion

In our thesis we build upon the previous data and workflow analysis of Karel Gavenčiak [1] and continue the efforts in the field of optimization within hospital environment and medical laboratories specifically. We also continue to make use of the data obtained by the cooperation with the Institute of Medical Biochemistry and Laboratory Diagnostic of the Královské Vinohrady University Hospital. The probelm we address is the batching of blood samples into a machine known as the centrifuge, which is one of the bottlenecks of the workflow needed to provide test results.

Firstly, we thoroughly examine the state-of-the-art methods used in literature to identify possible solutions. Afterwards, we describe the environment our problem is set in and analyze the problem domain. This reveals that the problem is relatively computationally demanding even when not considering the uncertainty, which makes certain approaches non-viable.

With all this in mind, we examine several sophisticated methods which show potential of providing proactive schedules, i.e., schedules which would be to some degree protected against adverse realizations of the uncertainty. Unfortunately, it turns out that some of these methods are either too computationally expensive or not suited for the structure of our task.

This has led us to come up with our own robust schedule generating model, which makes use of the data of previous realizations of the uncertainty while adding very little computational overhead. Subsequently, we define the reactive part of our framework which is responsible for calling the proactive schedule generating model when the problem domain changes.

Thereafter, we demonstrate that our model is solvable to optimality for reasonably large instances, which allows its use in practice. A behavior of our schedule generating model is then described for different levels of robustness. Finally, we evaluate our framework over a set of scenarios based on the actual data from the laboratory. We conclude that a greater level of robustness does not affect the statim samples, but leads to an increase in the average turnaround time of the routine samples, while considerably decreasing its average variance.

References

[1] Karel Gavenčiak. “Optimization of Samples Processing in Medical Laboratories: Problem Analysis Based on Data”. Czech Technical University in Prague, June 13, 2019. URL: http://hdl.handle.net/10467/83055.

[2] D. Michael Warner, Juan Prawda. “A Mathematical Programming Model for Scheduling Nursing Personnel in a Hospital”. In: *Management Science* 19.4-part-1 (Dec. 1972), pp. 411–422. DOI: 10.1287/mnsc.19.4.411. URL: https://doi.org/10.1287/mnsc.19.4.411.

[3] Franklin Dexter et al. “An Operating Room Scheduling Strategy to Maximize the Use of Operating Room Block Time”. In: *Anesthesia & Analgesia* 89.1 (July 1999), pp. 7–20. DOI: 10.1213/00000539-199907000-00003. URL: https://doi.org/10.1213/00000539-199907000-00003.

[4] Seung-Chul Kim, Ira Horowitz. “Scheduling hospital services: the efficacy of elective-surgery quotas”. In: *Omega* 30.5 (Oct. 2002), pp. 335–346. DOI: 10.1016/s0305-0483(02)00050-6. URL: https://doi.org/10.1016/s0305-0483(02)00050-6.

[5] Hanif D. Sherali, Muhannad H. Ramahi, and Quaid J. Saifee. “Hospital resident scheduling problem”. In: *Production Planning & Control* 13.2 (Jan. 2002), pp. 220–233. DOI: 10.1080/09537280110069667. URL: https://doi.org/10.1080/09537280110069667.

[6] Mohamed Ali Aloulou, Marie-claude Portmann. “An Efficient Proactive Reactive Scheduling Approach to Hedge against Shop Floor Disturbances”. In: *In Proceedings of the 1 st Multidisciplinary International Conference on Scheduling: Theory and Applications, MISTA 2003*. Pp. 337–362.

[7] Zukui Li, Marianthi Ierapetritou. “Process scheduling under uncertainty: Review and challenges”. In: *Computers & Chemical Engineering* 32.4-5 (Apr. 2008), pp. 715–727. DOI: 10.1016/j.compchemeng.2007.03.001. URL: https://doi.org/10.1016/j.compchemeng.2007.03.001.

[8] Bram L. Gorissen, İhsan Yanıkoğlu, and Dick den Hertog. “A practical guide to robust optimization”. In: *Omega* 53 (June 2015), pp. 124–137. DOI: 10.1016/j.omega.2014.12.006. URL: https://doi.org/10.1016/j.omega.2014.12.006.

[9] Aharon Ben-Tal, Tamar Margalit, and Arkadi Nemirovski. “Robust Modeling of Multi-Stage Portfolio Problems”. In: *Applied Optimization*. Springer US, 2000, pp. 303–328. DOI: 10.1007/978-1-4757-3216-0_12. URL: https://www2.isye.gatech.edu/~nemirovs/rob_portf.pdf.

[10] Daniel Bienstock, Nuri Özbay. “Computing robust basestock levels”. In: *Discrete Optimization* 5.2 (May 2008), pp. 389–414. DOI: 10.1016/j.disopt.2006.12.002. URL: https://doi.org/10.1016/j.disopt.2006.12.002.

[11] A. L. Soyster. "Technical Note—Convex Programming with Set-Inclusive Constraints and Applications to Inexact Linear Programming". In: *Operations Research* 21.5 (Oct. 1973), pp. 1154–1157. DOI: 10.1287/opre.21.5.1154. URL: https://doi.org/10.1287/opre.21.5.1154.

[12] Dimitris Bertsimas, Melvyn Sim. "The Price of Robustness". In: *Operations Research* 52.1 (2004), pp. 35–53. DOI: 10.1287/opre.1030.0065.

[13] Ada Che, Vladimir Kats, and Eugene Levner. "An efficient bicriteria algorithm for stable robotic flow shop scheduling". In: *European Journal of Operational Research* 260.3 (Aug. 2017), pp. 964–971. DOI: 10.1016/j.ejor.2017.01.033. URL: https://doi.org/10.1016/j.ejor.2017.01.033.

[14] Daiki Min, Yuehwern Yih. "Scheduling elective surgery under uncertainty and downstream capacity constraints". In: *European Journal of Operational Research* 206.3 (Nov. 2010), pp. 642–652. DOI: 10.1016/j.ejor.2010.03.014. URL: https://doi.org/10.1016/j.ejor.2010.03.014.

[15] Tjendera Santoso et al. "A stochastic programming approach for supply chain network design under uncertainty". In: *European Journal of Operational Research* 167.1 (Nov. 2005), pp. 96–115. DOI: 10.1016/j.ejor.2004.01.046. URL: https://doi.org/10.1016/j.ejor.2004.01.046.

[16] Mehdi Heydari, Asie Soudi. "Predictive / Reactive Planning and Scheduling of a Surgical Suite with Emergency Patient Arrival". In: *Journal of Medical Systems* 40.1 (Nov. 2015). DOI: 10.1007/s10916-015-0385-1. URL: https://doi.org/10.1007/s10916-015-0385-1.

[17] Benoît Colson, Patrice Marcotte, and Gilles Savard. "An overview of bilevel optimization". In: *Annals of Operations Research* 153.1 (Apr. 2007), pp. 235–256. DOI: 10.1007/s10479-007-0176-2. URL: https://doi.org/10.1007/s10479-007-0176-2.

[18] Jean-François Baffier, Pierre-Louis Poirion, and Vorapong Suppakitpaisarn. "Bilevel Model for Adaptive Network Flow Problem". In: *Electronic Notes in Discrete Mathematics* 64 (Feb. 2018), pp. 105–114. DOI: 10.1016/j.endm.2018.01.012. URL: https://doi.org/10.1016/j.endm.2018.01.012.

[19] Wolfram Wiesemann, Daniel Kuhn, and Melvyn Sim. "Distributionally Robust Convex Optimization". In: *Operations Research* 62.6 (Dec. 2014), pp. 1358–1376. DOI: 10.1287/opre.2014.1314. URL: https://doi.org/10.1287/opre.2014.1314.

[20] Chongfeng Ren, Hongbo Zhang. "A Fuzzy Max–Min Decision Bi-Level Fuzzy Programming Model for Water Resources Optimization Allocation under Uncertainty". In: *Water* 10.4 (Apr. 2018), p. 488. DOI: 10.3390/w10040488. URL: https://doi.org/10.3390/w10040488.

[21] Morteza Davari, Erik Demeulemeester. "The proactive and reactive resource-constrained project scheduling problem". In: *Journal of Scheduling* 22.2 (Dec. 2017), pp. 211–237. DOI: 10.1007/s10951-017-0553-x. URL: https://doi.org/10.1007/s10951-017-0553-x.

[22] Jenny Thureson. "Reducing the turnaround time in the histopathology service - Experiences of an improvement process". Jönköping University, URL: https://pdfs.semanticscholar.org/1362/346a776d8622725f843c4169888d41162afb.pdf.

[23] Jéssica M. Martins, Elayne Cristina M. Rateke, and Flávia Martinello. "Assessment of the pre-analytical phase of a clinical analyses laboratory". In: *Jornal Brasileiro de Patologia e Medicina Laboratorial* 54.4 (2018). DOI: `10.5935/1676-2444.20180040`. URL: `https://doi.org/10.5935/1676-2444.20180040`.

[24] Wenhua Li, Xing Chai. "The medical laboratory scheduling for weighted flow-time". In: *Journal of Combinatorial Optimization* 37.1 (2019), pp. 83–94. DOI: `10.1007/s10878-017-0211-4`.

[25] Varsha Agarwal et al. "Scheduling algorithms for reservoir- and mixer-aware sample preparation with microfluidic biochips". In: *Integration* 65 (Mar. 2019), pp. 428–443. DOI: `10.1016/j.vlsi.2018.01.002`. URL: `https://doi.org/10.1016/j.vlsi.2018.01.002`.

[26] Wikipedia contributors. *Pathology — Wikipedia, The Free Encyclopedia*. [Online; accessed 11-May-2020]. 2020. URL: `https://en.wikipedia.org/w/index.php?title=Pathology&oldid=956047556`.

[27] Carlos Bohle, Sergio Maturana, and Jorge Vera. "A robust optimization approach to wine grape harvesting scheduling". In: *European Journal of Operational Research* 200.1 (Jan. 2010), pp. 245–252. DOI: `10.1016/j.ejor.2008.12.003`. URL: `https://doi.org/10.1016/j.ejor.2008.12.003`.

[28] Pierre-Louis Poirion et al. "Algorithms and applications for a class of bilevel MILPs". In: *Discrete Applied Mathematics* 272 (Jan. 2020), pp. 75–89. DOI: `10.1016/j.dam.2018.02.015`. URL: `https://doi.org/10.1016/j.dam.2018.02.015`.

[29] J. Lofberg. "YALMIP : a toolbox for modeling and optimization in MATLAB". In: *2004 IEEE International Conference on Robotics and Automation (IEEE Cat. No.04CH37508)*. IEEE. DOI: `10.1109/cacsd.2004.1393890`. URL: `https://doi.org/10.1109/cacsd.2004.1393890`.

[30] Pictogram used to represent the centrifuge is plasma, blood by Olena Panasovska from the Noun Project. [Online; accessed 17-May-2020]. 2020. URL: `https://thenounproject.com/search/?q=centrifuge&i=2554295`.