Resource Scheduling - Evaluating Solving Using Satisfiability Modulo Theory

Sol Marie Wallevik
Supervisor: Violet Ka i Pun

Master's thesis in Software Engineering at

Department of Computer science, Electrical engineering and Mathematical sciences,
Western Norway University of Applied Sciences

Department of Informatics,
University of Bergen

June 2023

Western Norway
University of
Applied Sciences

Acknowledgements

First and foremost, a special thank you to my supervisor, associate professor Violet Ka I Pun, from the Western Norway University of Applied Sciences. Her guidance and expert knowledge have been instrumental in the formulation and successful completion of this thesis. I would like to extend my heartfelt appreciation to my family for their unwavering encouragement and moral support throughout my academic journey. Their constant belief in my abilities, along with their willingness to lend an attentive ear and provide constructive feedback, has been immensely motivating and inspiring. Furthermore, I would like to express my gratitude to my friends, whose unwavering support and positivity played a pivotal role in keeping my spirits up during the challenging phases of completing this thesis. In addition, I wish to acknowledge the invaluable contributions of the pathology department at Haukeland Hospital. I am thankful for their generosity in granting me access to their data, which has been vital in conducting my research. I am particularly grateful to Patrick Stünkel for his role as the department's contact person, as he has been instrumental in facilitating my collaboration with the pathology department.

Abstract

Workflow constitutes a series of sequential or parallel activities necessary to accomplish a particular task [21]. The effective planning of workflow involves determining the precise order of activities while taking into account the available resources. Scheduling assumes a critical role in this process by outlining the procedures and overseeing the execution of interdependent tasks within the workflow. It entails the allocation of appropriate resources to various workflows, ensuring efficient and timely completion of tasks and activities [13].

Inadequate scheduling can give rise to adverse consequences that disrupt the workflow. Such as repercussions like bottlenecks, missed deadlines, inefficient time management, excessive or underutilized resources, heightened stress and frustration, compromised quality, and increased costs. Overall, scheduling enhances the planning, prioritization, allocation, sequencing, and time management of tasks, thereby facilitating efficient resource utilization, establishing clear milestones, and enabling effective progress monitoring. These consequences collectively contribute to improved task execution and overall project success. By providing structure and visibility, scheduling assists team members in working together more efficiently, reducing the risk of delays or errors.

Scheduling conveys structure and visibility by establishing a framework for organizing and structuring time, assigning specific time periods to tasks, activities, and milestones, thereby establishing a chronological order for their execution. This structure aids individuals and teams in effectively allocating time, ensuring that each task receives the necessary attention and is completed within the designated time frame. Moreover, scheduling facilitates resource allocation, encompassing human resources, equipment, and materials. By assigning specific tasks to the appropriate resources within the schedule, it provides visibility into resource availability and utilization. Stakeholders can thus ensure efficient resource allocation and proactively identify and resolve potential conflicts or bottlenecks.

The objective of this thesis is to enhance scheduling practices to improve workflows. The study investigates whether a designated solver can generate and optimize solutions to enhance workflow efficiency. Given the variability of workflows across different contexts, domain-specific knowledge assumes crucial significance. To validate the concept, this thesis focuses on the pathology department at Haukeland Hospital as a case study. Collaborating with the department and acquiring real-life data generated by them will yield practical insights into implementing and evaluating the proposed scheduling approach within the healthcare domain.

Contents

Acronyms **8**

1 Introduction **9**

2 Background **11**
2.1 Satisfiability Modulo Theory 11
2.1.1 The different SMT solvers and what was chosen 12
2.1.2 An example using Z3 14
2.2 Conformance checking 16
2.3 Optimization Modulo Theory 17
2.3.1 The different OMT solvers 18

3 Research questions and methods **20**
3.1 Creating the program for the resource scheduling problem 20
3.2 Optimizing the solution 21

4 Engineering method **22**
4.1 Z3 - An SMT Solver 22
4.2 Conformance checking 23
4.3 Z3 with Optimize - an OMT solver 24

5 The pathology department **25**
5.1 Point system 25
5.2 Special cases 26
5.3 Different subject groups 27
5.4 Sick days/Absence 28
5.5 Special areas of responsibility 29
5.6 Special requests 29

6 Creating the program **31**
6.1 The base case 31
6.2 Expanding the base case 34
6.3 Special cases 37
6.4 Processing time 40
6.5 Optimizing the program 41

7 Evaluation of the results **43**
7.1 Processing and analyzing the data/Data Collection 44

7.2 Define metrics and data alignment 44
7.3 Comparative analysis 45
7.4 Quantitative assessment and qualitative evaluation 47
7.5 Interpretation of results 47

8 Results **49**
8.1 Reflection over the solvers 49
8.2 Answering the research questions 50

9 Conclusion and future work **55**
9.1 Conclusion 55
9.2 Future work 56

A Source code **61**

B Tables of the comparison of point distribution. **62**

List of Figures

2.1 The three rules of Sudoku written in Z3. 14
2.2 Unsolved Sudoku board given to the solver in Z3. 15
2.3 Solved Sudoku board using Z3. 15
2.4 The solutions for the Sudoku example 16

5.1 The overview of the pathology departments point system 26
5.2 Overview of the different tags and groups at the pathology department 28

6.1 Example of a logical formula is written in Python for Z3 32
6.2 The converter was created to calculate the correct amount of points based on the number of slices. 33
6.3 Constraint that ensures that each sample is assigned to at most one pathologist. 33
6.4 The satisfiable solution produced for the base case. 34
6.5 The expanded case base case with two of the days, the remaining points, and the new maximum 35
6.6 Example of when a pathologist is sick and the updated deduction list. 36
6.7 Example f the two pathologists that have requested physical samples. 36
6.8 Example of the new max points the pathologists can have if they have different work routines. 37
6.9 The converter created that takes slices and converts them to the corresponding points for the special samples. 38
6.10 Method used for matching the pathologist's with the correct samples based on specializations. 38
6.11 The results produced with the addition of special cases. 39
6.12 Boolean list analyzed and empty list non_analyzed to keep track of which samples have been analyzed. 40
6.13 List of samples not analyzed one day. 40
6.14 The different processing times each pathologist has attained one day. 41
6.15 The results produced from the optimization. 42

7.1 Comparison of points Monday 06.09.23 46

B.1 Comparison of the point distributions on Tuesday 07.09.2021. . . 62

B.2 Comparison of the point distributions on Wednesday 08.09.2021. 63
B.3 Comparison of the point distributions on Thursday 09.09.2021. . 63
B.4 Comparison of the point distributions on Friday 10.09.2021. . . . 64

Acronyms

CC Conformance Checking.

OMT Optimisation Modulo Theory.

sat satisfiable.

SMT Satisfiability Modulo Theories.

Chapter 1

Introduction

Optimizing scheduling procedures becomes easier without of constraints, as these factors can restrict available options and impede the discovery of an optimal solution. However, one rarely encounters scenarios where scheduling can be accomplished without any limitations, due to the unique requirements and resources inherent in every system. Consequently, constraints become inevitable for successful scheduling. Numerous considerations must be taken into account when devising an effective schedule, with particular focus placed on resource scheduling within the context of this thesis. Physical resources, including equipment, materials, supplies, and infrastructure [5], play a pivotal role in scheduling. Questions can arise such as whether a meeting room can accommodate all attendees with sufficient chairs or whether there are enough screws available to build a specific appliance. The distinct capabilities of various resources need to be taken into consideration. These are the quantifiable constraints that are limitations or constraints that can be expressed numerically. Time, capacity, quality, skills, and cost are examples of quantitative constraints. Additionally, human resources introduce another layer of complexity, since humans are not machines. Every person possesses unique preferences which make them better suited for different tasks. Furthermore, personal preferences and requests must be taken into account, such as a pathologist desiring a reduced workload on a given day. These non-quantifiable constraints are limitations that cannot easily be measured or expressed in numerical terms. Skill level, cultural factors, ethical considerations, and interpersonal relationships are other examples of non-quantifiable constraints. All in all, the challenge lies in developing a schedule that optimally supports the process rather than impending its process. Achieving this objective requires the careful management of the aforementioned constraints and the navigation of their complexities.

This thesis aims to solve a scheduling problem with both quantifiable and non-quantifiable constraints. As a proof of concept, the results formulated in this thesis will be evaluated by comparing them with data collected from the pathology department at Haukeland Hospital. Pathology is a medical discipline that plays a crucial role in delivering diagnostic information to patients and healthcare professionals, impacting numerous aspects of patient care. From the diagnosis of cancer to the management of chronic illnesses through accurate laboratory

testing [2]. Ensuring efficiency and precision in the execution of tasks within the pathology department is important, as other departments rely on the timely completion of their diagnostic results. The scheduling of resource allocation at Haukeland Hospital has been identified as an area where there can be made some improvements with the current scheduling practice being time-consuming. In the pursuit of an optimal scheduling solution, it is essential to account for both quantifiable and non-quantifiable constraints inherent in the pathology department's operations. The quantifiable constraints encompass subject specializations and a point system designed to monitor task allocation and ensure the even distribution of workloads. Additionally, non-quantifiable constraints must be taken into consideration during the schedule creation process. These include factors such as meetings, complications of some special samples, staff absences due to illness, and variations in individual pathologists' workloads. By integrating both types of constraints into the scheduling framework, a more comprehensive and efficient scheduling system can be devised. The proposed research seeks to address these challenges and improve the scheduling practices within the pathology department at Haukeland Hospital. By developing a scheduling model that accommodates both quantifiable and non-quantifiable constraints, this study aims to enhance efficiency and overall performance, thus contributing to the seamless delivery of essential diagnostic services to the healthcare facility.

Chapter 2

Background

2.1 Satisfiability Modulo Theory

Satisfiability Modulo Theories (SMT) is a discipline within computer science that deals with the automated reasoning about logical formulas to determine if a scheduling problem is satisfiable. The primary objective of SMT is to establish the satisfiability of a given logical formula through the utilization of decision procedures for different theories, including arithmetic, arrays, and bit-vectors. SMT solvers are commonly used in software verification, automated theorem proving, and other computer science and engineering areas. Logical formulas are expressions that represent logical propositions or constraints that can be either true or false. An example of this is

$$(X > 5)AND(Y < 10)AND(X + Y = 15)$$

This formula states that X must be greater than 5, Y must be less than 10, and the sum of X and Y must be equal to 15. By passing this formula to an SMT solver, it can check if there exists any assignment of values to X and Y that satisfies all the constraints simultaneously. And what does it mean that a problem is satisfiable? In the context SMT solving, a problem is satisfiable if there exists a satisfying assignment of values to the variables that satisfy the logical constraints specified in the problem. Continuing with the example above, one possible solution that satisfies the constraints is when $\mathbf{X = 6}$ **and** $\mathbf{Y = 9}$**, making** $\mathbf{X + Y = 15}$. Building upon the foundation of Boolean satisfiability (SAT), SMT extends its scope by allowing for constraints expressed within more expressive theories. For example, taking the previous formula and involving both arithmetic and array theories:

$$(X > 5)AND(y < 10)AND(array[Z] = 42)$$

In this formula, the arithmetic constraints remain the same as in the previous example, while the addition of the array theory introduces a new constraint. The formula states that X must be greater than 5, Y must be less than 10, and the value stored at index Z in the array must be equal to 42. Consequently, an SMT problem comprises a collection of logical formulas, encompassing Boolean

formulas and formulas within specific theories. To determine whether an SMT problem is satisfiable, an SMT solver will attempt to find a model that satisfies all the logical formulas in the problem. If a model exists, then the problem is satisfiable. If no model exists, then the problem is unsatisfiable, and the SMT solver can report that the problem is unsatisfiable or provide proof of unsatisfiability [8]. Let's consider another example of an SMT problem involving the theory of arithmetic. Suppose we have the two following logical formulas:

$$Formula1 : X + Y = 10$$

$$Formula2 : X > 3$$

To determine the satisfiability of this problem, an SMT solver will attempt to find a model that satisfies both formulas simultaneously. If a model exists, it means that there is an assignment of values to variables (in this case, X and Y) that satisfies both Formula 1 and Formula 2. For example, if one assigns $\mathbf{X = 4}$ **and** $\mathbf{Y = 6}$, both formulas are satisfied: $\mathbf{4 + 6 = 10}$ (satisfying Formula 1) and $\mathbf{4 > 3}$ (satisfying Formula 2). In this case, the SMT solver concludes that the problem is satisfiable. On the other hand, if no model exists, it means that there is no assignment of values to variables that can satisfy both formulas simultaneously. For instance, if we cannot find any values for x and y that satisfy both Formula 1 and Formula 2, the SMT solver determines that the problem is unsatisfiable. The solver can either report the problem as unsatisfiable or provide proof of unsatisfiability, demonstrating that no valid assignment satisfies the formulas.

So how can SMT help with resource scheduling? SMT solvers provide automated reasoning capabilities to optimize scheduling decisions. The utilization of an SMT solver in the context of resource scheduling entails several aspects, including but not limited to the following: optimizing resource allocation, detecting scheduling conflicts, identifying optimal schedules, and incorporating intricate constraints. Overall, SMT solvers can be seen as a powerful tool for resource scheduling by automating the decision-making process and optimizing the allocation of resources to tasks. This approach facilitates scheduling decisions that can be made more quickly and accurately, resulting in improved efficiency and reduced costs.

2.1.1 The different SMT solvers and what was chosen

Within the realm of SMT solvers, numerous options exist, each with its strengths, weaknesses, and features. One such notable solver is CVC4, an open-source SMT solver developed by researchers at New York University and the University of Iowa [6]. Renowned for its capacity to handle quantifiers and other advanced logical features, and has been used in a wide range of applications, including program verification and automated theorem proving. MathSAT, on the other hand, represents a commercial SMT solver developed by SRI International, a prominent software company [14]. Distinguished by its support for a wide array of logical theories encompassing arithmetic, bit-vectors, and arrays, MathSAT is particularly recognized for its efficient and scalable algorithms. Yices, yet another SMT solver developed by SRI International [12], has garnered acclaim for its efficiency and ability to handle large, complex problems. Yices has been

successfully employed in diverse applications, spanning program verification and software synthesis.

For this thesis I will be using Z3 [3], the SMT solver developed by Microsoft Research. Z3 is recognized as a high-performance, cross-platform solver renowned for its versatility and extensive adoption. Its implementation is primarily in C++ supplemented with bindings for TypeScript (including JavaScript) and Python, enabling its utilization through APIs in various programming languages such as C++, C#, Java, and Python. The selection of Z3 as the preferred solver is justified by several factors that contribute to its widespread popularity and utility. Z3 is esteemed for its efficiency in solving SMT problems, accommodating a diverse range of logical theories. Furthermore, it is highly regarded for its user-friendly nature, offering ease of integration and employment. Notably, Z3 benefits from continuous development and maintenance by Microsoft Research, ensuring ongoing advancements and support for users [17]. Another advantage of utilizing Z3 lies in its large user community, facilitating knowledge exchange and assistance in resolving queries and challenges. However, it is important to acknowledge that Z3 has some limitations. These encompass relatively limited support for quantifiers, partially inadequate documentation, and non-standard error reporting. Despite these constraints, Z3 remains a powerful and efficient SMT solver that is widely used within the research and development communities. While it has some limitations, it is a good choice for numerous applications within these domains.

Z3 stands out as one of the most widely adopted SMT solvers; nevertheless, as mentioned previously, there exists a variety of other solvers in the field. It is fitting to compare Z3 to these alternative solvers across several key aspects:

1. Efficiency: Z3 is known for its efficiency, demonstrating a remarkable capability to handle large, complex problems. This strength is also shared by other leading SMT solvers, such as CVC4 and MathSAT, which possess similar proficiency in handling complex scenarios.

2. Logical theories supported: Z3 supports a wide range of logical theories, including arithmetic, bit-vectors, arrays, and uninterpreted functions. While this is a noteworthy strength shared by multiple SMT solvers, the precise set of supported theories may vary among different solvers.

3. Support for quantifiers: Z3's support for quantifiers is less robust when compared to other SMT solvers, such as CVC4. The latter is renowned for its efficient and effective handling of quantifiers, presenting a greater advantage in scenarios requiring quantifier-heavy reasoning.

4. Advanced features: Z3 lacks support for certain advanced features like proof reconstruction and interpolation, which is available in other SMT solvers like CVC4. These additional capabilities contribute to the versatility and enhanced reasoning power of the solvers that offer them.

5. User-friendliness: Z3 excels in providing a user-friendly interface and a simplified Software Programming Interface (SPI), facilitating seamless integration into diverse projects. This attribute is shared by various other SMT solvers, including MathSAT.

Considering the requirements of this thesis, Z3 seems like a suitable choice. The

availability of Z3Py, its Python-based API, contributes to its favorable selection due to the familiarity and convenience it affords. Additionally, the popularity and active maintenance of Z3 ensure that potential obstacles encountered during its utilization can be readily resolved with the assistance of an extensive user community.

2.1.2 An example using Z3

To illustrate SMT's application, this section presents an illustrative example employing the Z3 solver [15]. The example leverages Z3's Python bindings and develops the entire code within the Python environment, utilizing tools such as Microsoft's Visual Studio Community. The chosen example revolves around solving a Sudoku puzzle, which can be mathematically expressed through three fundamental constraints of Sudoku:

1. Uniqueness within rows: Each element of a row must be distinct from the others in that same row.
2. Uniqueness within columns: Each element of a column must be distinct from the others in that same column.
3. Uniqueness within subgrids: Each element within a subgrid (commonly 3x3 square) must be distinct from the others within that same subgrid.

```
# assure that every row covers every value:
for row in 'ABCDEFGHI':
    s.add(Distinct([symbols[row+col] for col in '123456789']))

# assure that every column covers every value:
for col in '123456789':
    s.add(Distinct([symbols[row + col] for row in 'ABCDEFGHI']))

# assure that every block covers every value:
for i in range(3):
    for j in range(3):
        s.add(Distinct([symbols['ABCDEFGHI'[m + i * 3] + '123456789'[n + j * 3]] for m in range(3) for n in range(3)]))
```

Figure 2.1: The three rules of Sudoku written in Z3.

In Figure 2.1, the Z3 formulation of the Sudoku rules is depicted, even though additional constraints were required for solving the puzzle. The first constraint ensures that every row covers every value from 1 to 9. The line of code s.add(Distinct([symbols[row+col] for col in '123456789'])) embodies the key mechanism. The term **Distinct** is widely used in mathematical logic and constraint programming, signifying that elements within a collection or set must be distinct from one another [9]. In the context of Z3, it serves to specify a constraint that mandates distinctness among a set of variables [4], specifically the numbers 1 to 9 within a row. This constraint is incorporated into the solver using the add() function. The same principle applies to the other two constraints demonstrated in Figure 2.1.

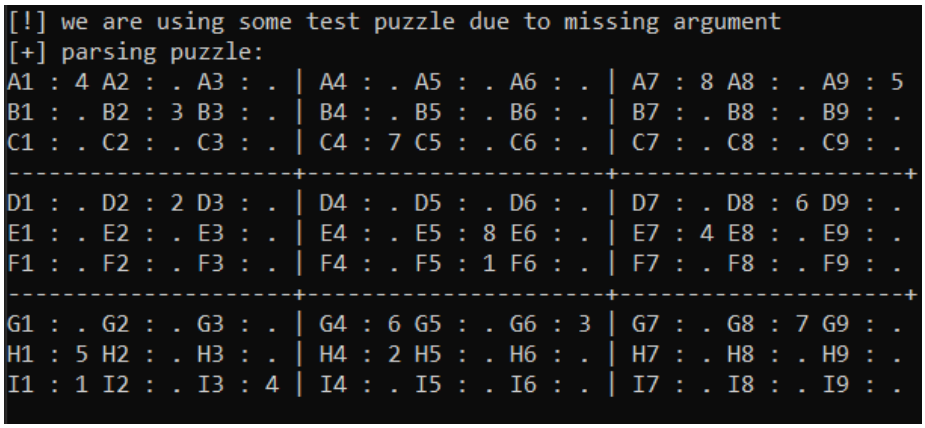

Figure 2.2: Unsolved Sudoku board given to the solver in Z3.

By developing a program using Z3 that incorporates these Sudoku rules, Z3 effectively solves a partially filled Sudoku board, as depicted in figure Fig 2.2. In this figure, the grid represents the unsolved puzzle provided to the SMT solver.

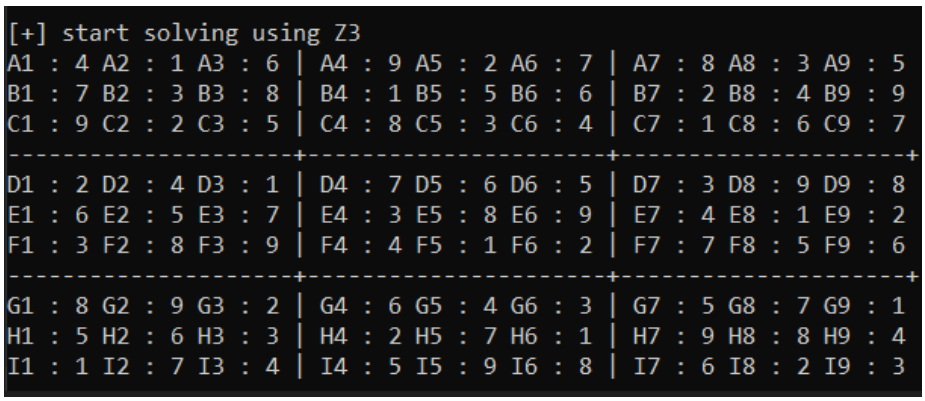

Figure 2.3: Solved Sudoku board using Z3.

Figure 2.3 depicts the grid containing the solved Sudoku puzzle. Columns are denoted by the letters A to I, where A corresponds to the first column and I correspond to the ninth column. Rows are labeled numerically from 1 to 9, with 1 representing the first row and 9 representing the ninth row. For instance, **[A1] : 4** indicates that the answer for the element located in the first row is 4.

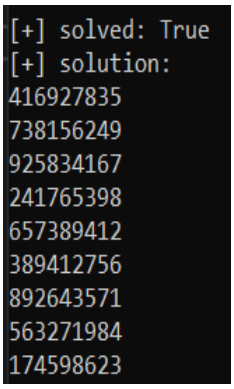

Figure 2.4: The solutions for the Sudoku example

If the Sudoku is solvable, the second grid representing the solved puzzle is printed. Then if the puzzle is unsolvable, the second grid does not appear. Figure Fig 2.4 illustrates the printed output when the puzzle is solvable. The output displays **[+] solved: True** to indicate a successful solution, accompanied by the solutions for the individual rows. Then again if the puzzle proves to be unsolvable, the output displays **solved: False**, and no solution gets printed.

2.2 Conformance checking

Conformance Checking (CC), derived from the field of process mining, employs data science techniques to discover, validate, and enhance workflows [11]. It aims to compare the actual behavior observed in a process with the behavior captured in a process model [16]. This comparison enables the identification of deviations, i.e., instances where the observed behavior differs from the expected or desired behavior. By uncovering such information, organizations can identify areas for improvement and ensure that all process deviations are appropriately addressed. While CC originates from process mining, its applicability extends beyond this domain. It can be effectively employed in various contexts depicted in the list below:

1. Software development: CC in software development verifies adherence to coding standards, functional requirements, and the absence of vulnerabilities or security weaknesses within software systems.
2. Quality assurance: CC is employed to assess products or services against predefined quality standards, identifying areas where they deviate or fail to meet the established criteria.
3. Regulatory compliance: CC ensures that businesses and organizations comply with legal and regulatory mandates, encompassing safety standards, environmental regulations, and financial reporting requirements.
4. Supply chain management: CC plays a vital role in verifying supplier and vendor compliance with stipulated quality and safety standards, as well as contractual obligations in the context of supply chain management.

Within each of these contexts, Conformance Checking entails the comparison of two systems or models to identify any areas of disparity or non-alignment. This process can involve the utilization of automated tools or manual inspection, and may require modifications or updates to either or both systems to enhance their conformance. In the scope of this thesis, the results obtained from the SMT solver will be evaluated through the application of CC techniques. Specifically, in the context of the pathology department at Haukeland Hospital, the actual behavior of the department will be compared to the solution derived from the SMT solver. This comparison aims to reveal any discrepancies and to see whether adjustments are necessary, either in the form of modifying constraints or refining the problem formulation.

2.3 Optimization Modulo Theory

Optimisation Modulo Theory (OMT) is a framework that combines optimization techniques with decision procedures and is an extension of SMT [19]. It is used in computer science and formal verification to solve optimization problems over variables that are subject to constraints defined by theories. In optimization problems, the goal is to find the best solution (maximize or minimize) within a given set of constraints. Suppose we have a transportation optimization problem where we need to determine the optimal routes and schedules for a fleet of delivery vehicles. The objective is to minimize the total cost while ensuring timely deliveries and adhering to various constraints. To solve this problem using OMT, the optimization model can be formulated by combining mathematical optimization techniques with theories from different domains.

Even though SMT and OMT are closely related, they do have distinct branches of formal verification and constraint solving. Below is a comparison of the two:

1. Purpose
    - SMT: The primary goal of SMT is to determine the satisfiability of logical formulas containing constraints from various theories. It focuses on determining if there exists an assignment of values to variables that satisfies all the constraints.
    - OMT: OMT extends the capabilities of SMT by incorporating optimization objectives in addition to satisfiability. OMT aims to find the optimal solution that satisfies both the logical constraints and the optimization criteria specified in the problem.
2. Optimization:
    - SMT: SMT solver do not handle optimization objectives directly. They focus solely on determining whether a given logical formula is satisfiable or unsatisfiable.
    - OMT: OMT solver are designed to handle optimization objectives alongside logical constraints.
3. Problem Formulation:

- SMT: In SMT, problems are formulated using logical formulas with constraints from different theories.
- OMT: OMT extends SMT by allowing the formulation of optimization problems. In addition to logical formulas, OMT problems include an objective function that expresses the optimization goal.

4. Solver Capabilities:
   - SMT: SMT solver are well-suited for problems that require determining the satisfiability of logical formulas and performing consistency checking across different theories.
   - OMT: OMT solvers encompass the capabilities of SMT solver but also handle optimization objectives. They are suitable for problems that involve both logical constraints and optimization criteria, allowing the search for optimal solutions.

In summary, while both SMT and OMT deal with logical formulas and theories, OMT extends the capabilities of SMT by incorporating optimization objectives. SMT focuses on satisfiability checking, while OMT addresses optimization problems by finding the optimal solution that satisfies both the logical constraints and the specified optimization criteria [19]. OMT has applications in various domains, including formal verification of hardware and software systems, program synthesis, planning, scheduling, and resource allocation. By automated reasoning about complex problems with both logical constraints and optimization objectives.

2.3.1 The different OMT solvers

There are several OMT solvers available that implement the Optimisation Modulo Theory framework, each with its own strengths and weaknesses. Here are a few commonly used solvers and a brief comparison:

1. OptiMathSAT: OptiMathSAT is an open-source solver developed at the University of Trento - Italy [22]. It is an OMT solver that integrates optimization objectives, including linear and non-linear objectives, as well as mixed-integer and continuous variables. OptiMathSAT is known for its efficiency and versatility, providing strong performance across various problem domains.
2. CVC4 with OMT: as mentioned in section 2.1.1 CVC4 is an open-source SMT solver. It includes an OMT module that enables optimization over theories. CVC4 with OMT supports linear and non-linear optimization objectives, as well as mixed-integer and continuous variables. It provides competitive performance and is actively developed and maintained.
3. Z3 with Optimize: as mentioned in section 2.1.1, Z3 is a powerful SMT solver, and it also includes an optimization module called Optimize. The Optimize module extends Z3's capabilities to handle optimization problems. Z3 with Optimize supports various optimization objectives, including linear and non-linear objectives, as well as mixed-integer and continuous variables. It offers good performance and is widely used in both academic and industrial settings.

In this thesis, the preferred solver for the resource scheduling problem is Z3 with the Optimize module. The selection of Z3 is primarily motivated by its convenience during the development process, although the Optimize module in Z3 offers additional factors that contribute to its popularity. The rationale for choosing Z3 has been elaborated upon in section 2.1.1. By opting for Z3 with OMT capabilities, the thesis benefits from the utilization of Z3's SMT solver while seamlessly integrating optimization objectives into the problem-solving process. This choice provides an efficient, expressive, and flexible framework for addressing complex problems that involve both logical constraints and optimization criteria. Leveraging the capabilities of Z3 with OMT ensures that the power of the solver is harnessed, enabling the formulation of the resource scheduling problem as an optimization problem. Consequently, the solver can guide the search for optimal or near-optimal solutions based on the specified objectives. From a personal standpoint, choosing Z3 with OMT is motivated by its effortless integration into the existing codebase developed for the SMT component of the thesis. The continuity provided by using the same solver streamlines the implementation process and ensures compatibility between the SMT and OMT aspects. By leveraging the OMT capabilities of Z3, the thesis can leverage the solver's optimization functionalities to tackle the resource scheduling problem effectively.

Chapter 3

Research questions and methods

This chapter will introduce the research questions being answered and the research methods used to answer this thesis.

RQ1: Can SMT find a solution for a resource scheduling problem with non-quantifiable constraints?
RQ2: What does the SMT solution with no restrictions compare to the SMT solution that takes into consideration non-specific constraints? And how can this impact the workflow at the pathology department?
RQ3: By considering non-quantifiable restrictions, how can OMT help optimize the resource scheduling?
RQ4: Is the SMT solution to the resource scheduling problem applicable to the pathology department at Haukeland Hospital?

3.1 Creating the program for the resource scheduling problem

Resource scheduling problems involve the allocation of a designated set of resources to a given set of tasks while considering a range of constraints, including resource availability, task dependencies, and deadlines [20]. As part of addressing **RQ1**, Z3's SMT solver is utilized to develop a program to tackle the resources scheduling problem. By encoding the problem as a logical formula, SMT serves to identify a viable resource allocation that adheres to the specified constraints. The utilization of various theories within SMT facilitates the representation of constraints. For instance, the theory of arithmetic can express constraints related to task duration and resource availability, while the theory of arrays can model task dependencies. The logical formula is formulated using variables representing the resources and tasks, employing logical connectives, functions, and predicates according to the chosen theory. Subsequently, Z3 is employed to search for a solution that satisfies the constraints by construct-

ing a model that adheres to the logical formula. In cases where no solution is found, it indicates that the problem is infeasible, and the constraints may need to be relaxed or reformulation of the problem. The approach to addressing the resource scheduling problem involves a progressive iterative process. It starts with the base case, representing the simplest scenario of scheduling without many constraints. Subsequently, the constraints, including both quantifiable and non-quantifiable factors, are progressively introduced. This iterative approach enables the comparison and evaluation of different program iterations at various stages. To partially answer **RQ2**, CC techniques are employed to assess and validate the results obtained from the program iterations.

To address **RQ2** and **RQ4**, the example case of the pathology department at Haukeland Hospital is applied to the results from **RQ1**. To evaluate these results, real-world workflow data collected from the department is utilized, allowing CC to be performed. At each step of the iterative process of development, one can observe and compare as to what happens when adding more constraints in order to answer **RQ2**. The CC evaluation focuses on assessing the degree of adherence between the obtained results and the actual behavior of the department. Several constraints come into play when considering the pathology department's scheduling process. Some of them include their point system to ensure workload distribution, the different specialists, and the occurrence of sick leave. These constraints are better explained in chapter 5. By conducting the evaluation using CC techniques, we can identify any potentially problematic areas of deviations from the expected behavior. This analysis will inform whether adjustments to the existing constraints or problem reformulation are necessary to improve the overall scheduling process. In summary, by applying the proposed solution to the pathology department's case and utilizing CC along with the collected data, the aim is to gain insight into the effectiveness and suitability of the constraints, thereby addressing **RQ4** and providing additional insights relevant to **RQ2**.

3.2 Optimizing the solution

Finally, to address **RQ3**: How can Optimization Modulo Theory (OMT) contribute to optimizing resource scheduling while considering non-quantifiable constraints? The application of OMT will be employed to identify potential areas for optimization and seek optimal solutions for the resource scheduling problem. Building upon the code developed for addressing **RQ1**, **RQ2**, and **RQ4**, further refinement will be carried out to achieve an optimized solution. This entails exploring the aspects of resource scheduling that demonstrate potential for optimization and determining the feasibility of optimizing those areas. To accomplish this, the Optimize() module from Z3 will be employed to facilitate the process of identifying and attaining optimized solutions. Assessing the module's capability to generate feasible solutions that satisfy the imposed constraints and identifying an optimal solution.

Chapter 4

Engineering method

This chapter outlines the engineering methods employed in this thesis for solving resource scheduling problems using Z3, an SMT solver, and Z3 with Optimize, an OMT (Optimization Modulo Theory) solver. The chapter begins with an overview of Z3 and its role in the engineering process. It then describes the steps involved in utilizing Z3 for problem solving, followed by an exploration of conformance checking as a complementary method. The chapter concludes with an introduction to Z3 with Optimize and its application in solving optimization problems.

4.1 Z3 - An SMT Solver

I use Z3 together with Visual Studio Community, which is a free, cross-platform integrated development environment (IDE) for developing software applications. By using Z3's API for Python the created program that uses Z3 for SMT solving will be written in Python. The written code then calls the Z3 API for Python to create logical formulas, add constraints, and solve them using an SMT solver. The engineering method of Z3 can be summarized in the following steps [18]:

1. Define the problem: The first step is to define the problem that needs to be solved. This may involve identifying the variables, constraints, and logical formulas that describe the problem.
2. Model the problem: This involves creating variables, defining constraints, and specifying the logical formulas that describe the problem.
3. Check the satisfiability of the problem: This involves calling the 'check()' method to determine whether there exists a solution that satisfies the constraints and logical formulas.
4. Analyze the results: If Z3 finds a solution that satisfies the constraints and logical formulas, it can be analyzed to determine if it meets the desired criteria. This may involve checking the values of the variables to ensure that they are within acceptable ranges or comparing them to a target or benchmark.

5. Refine the problem and repeat: If Z3 does not find a solution, the problem can be refined and the process can be repeated until a satisfactory solution is found.

6. Implement the solution: This may involve using the values of the variables to make decisions, perform calculations, or optimize a process. In some cases, the solution may need to be integrated with other systems or processes to achieve the desired results.

Overall, the engineering method of Z3 involves a systematic approach to defining, modeling checking, analyzing, and refining problems using Z3. By following this method, organizations can leverage the power of Z3 to solve complex problems and optimize their processes.

4.2 Conformance checking

Conformance checking will be used to compare the event log (actual events) with the results derived from the SMT solver (process model). To determine the degree of conformance between them. The engineering method of conformance checking is a systematic approach to conformance checking that involves the following steps [1]:

1. Define the process model: A process model is a formal representation of the process that is being analyzed. For this thesis, this will be the solution the SMT solver provides. So for our example, the solution from the SMT solver will be compared to the event log from the pathology department.

2. Collect the event log data: An event log is a record of the events that occurred during the execution of the process. The log contains information about the activities performed, the order in which they were performed, and the time at which they occurred. For our example, this will be the data collected from Haukeland Hospital.

3. Reprocess the event log: The event log data must be reprocessed to prepare it for analysis. This may involve filtering our irrelevant events, enriching the log with additional information, and transforming the log into a format that can be used for analysis.

4. Apply conformance checking techniques: identify deviations.

5. Analyze the results: The results of the conformance checking analysis are used to identify areas of the process that need improvement. This may involve identifying bottlenecks, inefficiencies, or other problems that can be addressed through process redesign or other process improvement initiatives.

6. Implement process changes: implement the recommended changes to the process. This may involve updating the process model, changing the way the process is executed, or introducing new tools or technologies to support the process.

4.3 Z3 with Optimize - an OMT solver

Z3 with Optimize is a powerful OMT solver that combines optimization techniques with the Z3 SMT solver. It provides an effective and flexible approach to solving optimization problems in various domains. The engineering method of Z3 with Optimize involves several key steps [18]:

1. Problem formulation: Start by formulating an optimization problem mathematically or logically. Define the objective function you want to optimize and any constraints on the variables involved.
2. Modeling the problem: Use the Z3 programming API to create a model of the optimization problem. This involves defining the variables, constraints, and objective functions within the Z3 solver environment.
3. Setting optimization parameters: Configure the optimization parameters based on your problem requirements. For example, one can specify the optimization direction (minimize or maximize), the optimization strategy, and any additional parameters that may affect the solver's behavior.
4. Solving the optimization problem: Invoke the optimization solver in Z3 with Optimize to find the optimal or near-optimal solution to your problem. The solver will explore this solution space guided by the defined objective function and constraints.
5. Extracting the solution: Once the optimization solver completes, extract the solution values for the variables of interest. One can use the Z3 APO to query and retrieve the optimal values assigned to the variables.
6. Analyzing the solution: Analyze the obtained solution to understand its implications and assess its quality. Evaluate whether it meets the desired optimization objectives and satisfies the given constraints.
7. Iterative refinement: If the obtained solution does not meet your expectations of it you need to explore alternative solutions, you can refine the problem formulation, adjust the constraints, or modify the objective function and repeat the solving process.

Throughout this engineering process, it is important to pay attention to the details of formulating the problem correctly, specifying the constraints accurately, and interpreting the results appropriately. Proper modeling and parameter tuning can significantly impact the performance and effectiveness of Z3 with Optimize in solving an optimization problem. By following this engineering method, you can leverage the capabilities of Z3 with Optimize to tackle a wide range of optimization problems, form linear objectives to non-linear objectives, with mixed-integer or continuous variables.

Chapter 5

The pathology department

This thesis utilizes the pathology department at Haukeland Hospital as a case study to illustrate the application of the proposed program. Before delving into the program's development, it is essential to provide an in-depth explanation of the department's specific rules and regulations. These rules and regulations play a vital role in ensuring accurate sample processing and maintaining a balanced workload distribution within the department. This chapter aims to explain the specific rules and regulations governing the pathology department's operations. By understanding these rules, a better comprehension of the rationale behind the program's design can be achieved.

5.1 Point system

The pathology department has implemented a unique point system to ensure the even distribution of workload and to accurately track the contributions of individual pathologists. Each pathologist can accumulate a daily range of at most 23 to 25 points. To comprehend how pathologists earn points, it is necessary to outline the procedures employed by the pathology department in handling incoming samples. The pathology department deals with a variety of tests, covering blood tests, urinalysis, fecal analysis, and examination of bodily tissues. These tests aim to identify abnormalities indicative of diseases, such as cancer and chronic illnesses, as well as potential health risks like pre-diabetes [10]. When bodily tissues are received by the department, they undergo a systematic process before analysis. Although a detailed account of this process is beyond the scope of this discussion, it involves slicing the tissues into millimeter-think sections, scanned and handed out as PDF files to the pathologists. The number of slices required for analysis depends on the size of the tissue sample. Consequently, the point system takes into consideration the number of slices associated with a sample to determine its corresponding point value. For instance, samples comprising of 1 to 5 slices are assigned a value of 1 point, while samples consisting of 66 to 70 slices are allocated 14 points.

Amount of points per case	
1-5 slices	**1** point
6-10 slices	**2** points
11-15 slices	**3** points
16-20 slices	**4** points
21-25 slices	**5** points
26-30 slices	**6** points
31-35 slices	**7** points
36-40 slices	**8** points
41-45 slices	**9** points
46-50 slices	**10** points
51-55 slices	**11** points
56-60 slices	**12** points
61-65 slices	**13** points
66-70 slices	**14** points
71-75 slices	**15** points
76-80 slices	**16** points
81-85 slices	**17** points
86-90 slices	**18** points
91-95 slices	**19** points
96-100 slices	**20** points
101-105 slices	**21** points
106-110 slices	**22** points

Figure 5.1: The overview of the pathology departments point system

Figure Fig 5.1 illustrates the aforementioned point system. Furthermore, certain special cases may generate additional points explained in section 5.2. Saying that this points system has been adopted specifically to address the resource scheduling problem within the pathology department, serving as an example within the context of this thesis.

5.2 Special cases

The point system previously explained incorporates certain exceptions and variations. One notable exception is the carry-over remaining points from one day to the next, in cases where a pathologist fails to accumulate the minimum requirement of 23 points on a given day. Consequently, the outstanding points are transferred to the following day, thereby increasing the target point accumulation. For instance, if a pathologist earns only 19 points on Monday, they would need to obtain 27 points on Tuesday to compensate for the shortfall. While the carry-over of points can occur within the same week, they cannot be carried over to subsequent weeks. Important to note is that the absolute maximum a pathologist can obtain is 30 points. Furthermore, there are specific sample types that deviate from the standard point system depicted in figure Fig 5.1. These distinct sample types, along with their corresponding point allocations, are outlined below:

1. CITO samples: These are samples that pathologists prioritize for urgent analysis. They carry a minimum of two points, after which they adhere to the regular point system.
2. Oral samples: Analysis of oral samples does not contribute any points towards the pathologists' point accumulation.
3. Needle biopsies: These samples also carry a minimum of two points, after which they adhere to the regular point system.
4. Bone marrow biopsies: These are samples that also carry a minimum of two points, after which they adhere to the regular point system.

These variations and exceptions to the standard point system are incorporated to accurately reflect the complexities and unique requirements of the pathology department's workflow.

5.3 Different subject groups

Each pathologist within the pathology department is typically assigned to one or two subject groups, representing specific fields in which they specialize. However, it is worth noting that some pathologists may be assigned to three subject groups, while others may not have any specific subject group designation. The samples received by the pathology department for analysis are marked or tagged to indicate the corresponding field to which they belong. For instance, a sample related to breast reduction would be tagged with **m** to signify the Mom group, while a biopsy from the kidneys would be tagged with **y** to indicate the Kidney group, and so forth. These tags serve the purpose of ensuring that each sample is appropriately paired with a pathologist who possesses the requisite specialization in the corresponding field.

Tag	Group	Examples
u	Urogruppen	Urologiske prøver f.eks.nyre sylinderbiopsier mtp tumor, nyreresektater, urinveier, prostata)
x	Gynogruppen	f.eks. uturus, conus, ovarie, cervixpolypper
p	Perinatalgruppen	placenta, mola
m	Mammagruppen	f.eks. mamma resektater, brystreduksjon, sylinderbiopsier, vakuumbiopsier
g	Gastrogruppen	f.eks. lever sylinderbiopsier, galleblære, appendix, tarmpreparater, pancreas, øsofagus, ventrikkel, duodenum
h	Hudgruppen	f.eks. inflammatoriske hudprøver, melanom, store hudpreparater, alle interne prøver fra hud avdeling og plastikkirurgisk avdeling
l	Lymfomgruppen	f.eks. prøver fra lymfeknuter med spørsmål om lymkom
s	Sarkomgruppen	f.eks. lipom, bløtdelsvev, prøver med spørsmål om sarkom
r	Øre-nese-hals-gruppen og Lungegruppen	nesepoltpper, bihuler, larynx, spyttkjertler, andre som kommer fra ØNH leger og lunge sylinderbiopsier, lungepreparater
y	Nyregruppen	sylinderbiopsier fra nyre mtp inflammatorisk sykdom
oral	oral	f.eks. Sjøgrens syndrom, tenner, alle prøver som kommer fra tannlegekontorer
nevro	nevro	f.eks. hjerne

Figure 5.2: Overview of the different tags and groups at the pathology department

Figure 5.2 provides an overview of the different tags, and their corresponding subject groups, and includes an example of a sample associated with each tag. The utilization of subject groups and tags enables the efficient allocation of samples to pathologists, ensuring that they are assigned tasks that align with their areas of expertise within the pathology department.

5.4 Sick days/Absence

Inevitably, there are instances when a pathologist falls ill during their scheduled work hours. However, even in such situations, there is a need to ensure that the analysis of samples continues without disruption. To address this, the pathology department has implemented measures to redistribute the workload evenly and ensure that the same number of samples are analyzed despite the absence of a pathologist. When a pathologist is unable to work due to illness, their allocated 23 points for the day are reallocated among the other pathologists who are present and able to work. Ideally, the additional points should be assigned to pathologists with the same specialization as the absent colleague. However, it is important to note that a pathologist cannot exceed a total of 30 points for a single day. Consequently, the pathologists who take on the extra workload must keep track of the additional points they acquire. These points are recorded in a **Deductionlist** and can be utilized later when the pathologists themselves are

unable to work due to illness, thereby compensating for the intended points they were expected to earn on that day. Moreover, pathologists need to prioritize their own intended samples before attending to the extra samples generated by the redistribution of workload. By following this procedure, the pathology department ensures a systematic approach to managing the impact of illness on the workflow while maintaining fairness and adherence to the predetermined point allocation system.

5.5 Special areas of responsibility

In addition, certain pathologists have designated areas of responsibility within the pathology department, which require their dedicated attention. These responsibilities often involve attending meetings, resulting in their unavailability for specific periods during the week. The distribution of these responsibilities takes place on Monday mornings, ensuring that the workload is appropriately allocated. The distinct areas of responsibility and their associated meetings are as follows:

1. Mom : Pathologists assigned to the Mom areas have meetings scheduled on Tuesday at 14:30. Their focus includes analyzing samples related to maternity cases.
2. Frozen slices/CITO: This responsibility involves handling frozen slices and conducting CITO tests that have not been specifically assigned to the appropriate subject group.T
3. Thorax: Pathologists responsible for the Thorax area participate in meetings held on Fridays at 09:30. Their tasks encompass the analysis of lung samples and related diagnostic procedures.
4. Ear-Nose-Throat: This responsibility involves conducting CITO tests and other relevant analyses related to ear, nose, and throat cases.
5. Gastric CITO: Pathologists assigned to the Gastric CITO responsibility attend a meeting every Monday at 08:15. They are responsible for processing all gastric CITO samples and small package samples, including liver specimens.
6. Lymphoma/hema: Pathologists assigned to this responsibility primarily focus on analyzing lymphoma and bone marrow.

By designating specific areas of responsibility and scheduling corresponding meetings, the pathology department ensures that pathologists can effectively manage their specialized tasks while maintaining a coordinated workflow.

5.6 Special requests

As previously acknowledged, human individuals possess their personal preferences and desires that necessitate consideration within a work environment. Within the context of the pathology department, during discussions with the pathologists, several requests were identified that hold significance. These requests primarily revolve around two aspects: the desired number of working

hours and the preference for physical copies of samples. Regarding the number of working hours, it was observed that certain pathologists adhere to specific weekly schedules, limiting their availability to work for a designated number of hours. Three distinct working patterns were identified within the pathology department. Firstly, some pathologists work full-time, accumulating 23 to 25 points every day of the week. Alternatively, some pathologists adopt a half-time schedule, where they can earn up to 12 points every day. Lastly, some pathologists opt for a one-third schedule, earning 8 points each day during the week. These varying work schedules cater to the pathologist's request to work less than one week. In terms of the second request, it was noted that pathologists are provided with samples in the form of PDF files when working with sliced specimens. However, despite the department's ongoing transition towards digitization, a subset of pathologists expressed a desire to still receive physical copies of the sample slices in addition to the digital format. Consequently, the department accommodates these preferences and ensures the provision of physical slices alongside the PDF versions, acknowledging the need to balance the evolving digital approach with the individual preferences of the pathologists.

Chapter 6

Creating the program

This chapter aims to provide an overview of the code development process employed to formulate the problem that is being addressed by the SMT solver with the context of our example for this thesis. Specifically, the focus lies on the resource scheduling problem with the application domain of the pathology department at Haukeland Hospital. Given the inherent complexity of resource scheduling, which predominantly stems from the numerous constraints involved, the initial step involved establishing the foundational framework. This entailed the task allocation process, taking into account factors such as points and specializations while excluding the consideration of quantifiable and non-quantifiable constraints. It is crucial to acknowledge that distinct systems exhibit unique constraints and requirements. Consequently, the availability of relevant data is essential for addressing the research questions at hand. It is worth noting that the pathology department adheres to its own set of constraints and rules governing its workflow. The department has graciously shared its data set, which will be used to utilized for evaluating the obtained results and determining the efficacy of the work conducted in this thesis.

6.1 The base case

In the initial stage of developing the program, the primary objective is to define the problem, laying the foundation for subsequent steps. In this regard, the task involves distributing a given number of samples (referred to as slices) among a specific set of pathologists who are scheduled to work on a given day. Each sample is associated with a particular specialization and is assigned a predetermined number of points. Moreover, each pathologist possesses their specialization and has the potential to earn a maximum of 25 points per day. Considering the absence of available data during the creation of the base case, data were randomly generated to facilitate the initial development. However, this fabricated data will subsequently be replaced with actual real-life data. The following step in the process entails modeling the behavior of the system, which involves defining the relevant variables, establishing constraints, and specifying logical formulas.

```
from z3 import *

x = Int('x')
even = lambda x: x % 2 == 0
s = Solver()
s.add(ForAll(x, Implies(even(x), even(x*x))))
print(s.check())
```

Figure 6.1: Example of a logical formula is written in Python for Z3

In the context of Z3, logical formulas are expressions that represent logical propositions or constraints, capable of being either true or false. These formulas are expressed using a first-order logic language, enabling reasoning about various types of variables and predicates. To illustrate, Figure 6.1 demonstrates an example of logical formula, **"For all x, if x is even, then x squared is even."**.

The base case involves several variables that play a crucial role in the resource scheduling program. These variables can be categorized as follows:

1. The number of pathologists:
    - The number of pathologists: This represents the total count of pathologists available on a given day.
    - Maximum points per pathologist: Defines the upper limit for the number of points a pathologist can accumulate in a day.
    - Tags for pathologists: Denotes the specializations or subject groups to which each pathologist is assigned.
2. Sample-related variables:
    - Number of slices in a sample: Indicates the number of slices that constitute a single sample.
    - Number of samples: Reflects the total number of samples that need to be distributed among the pathologists.
    - Tags for samples: Represents the specialization or subject group to which each sample belongs.
3. Assignment and tracking variables:
    - Sample-pathologist assignment: Establishes the relationship between each sample and the assigned pathologist.
    - Total points assigned to each pathologist: Track the cumulative points assigned to each pathologist throughout the allocation process.

These variables collectively form the foundation of the base case, enabling the formulation of logical constraints and the subsequent distribution of samples

among the available pathologists.

```
def slices_to_points():
    points_for_todays_slices = []
    for pt, semp in point_table.items():
        for s in semp:
            for slice in slices:
                if s == slice:
                    points_for_todays_slices.append(pt)
    return points_for_todays_slices
```

Figure 6.2: The converter was created to calculate the correct amount of points based on the number of slices.

To facilitate the resource scheduling process, a converter was developed to determine the corresponding number of points based on the given number of slices in a sample. The converter can be seen in Figure 6.2. This converter plays a vital role in establishing the point system utilized in the pathology department. During the development of the program, five key constraints were identified to ensure its proper functioning. These constraints are as follows:

1. Sample-pathologist assignment constraint: ensures that each sample is assigned to at most one pathologist
2. Maximum points constraint: limits the number of points accumulated by each pathologist, ensuring that it does not exceed the maximum allowed value.
3. Unique assignment constraint: guarantees that each sample is assigned to exactly one pathologist, avoiding unassigned samples.
4. Maximum points per pathologist constraint: ensures that each pathologist has at most the maximum number of points (e.g., 25 points).
5. Tag matching constraint: validates that each tagged sample is assigned to the corresponding tagged pathologist, preserving the specialization-based assignment.

A comprehensive explanation of these constraints can be found in Chapter 5.1 and Chapter 5.3.

```
# Add constraints to ensure each sample is assigned to at most one pathologist
for i in range(num_samples):
    solver.add(sum([If(assignments[i][j], 1,0) for j in range(num_pathologist)]))
```

Figure 6.3: Constraint that ensures that each sample is assigned to at most one pathologist.

For instance, Figure 6.3 illustrates an example constraint implementation. The code iterates through the length of the samples, and using the **solver.add()** function, the constraint is added to the solver object. In this specific case, the sum of the elements in the list comprehension must be less than or equal to 1, ensuring that each sample is assigned to at most one pathologist.

Following the implementation of the constraints, a validation step is performed to determine if a valid solution exists. If the **solver.check()** function returns a satisfiable result (i.e., **sat**), the program proceeds to print the assignments. On the other hand, if the result is unsatisfiable (i.e., **unsat**), the program outputs the messages **"Status: unsat"** and **"No solution found"**, indicating the absence of a valid solution.

```
Status: sat
Sample Assignments and Points:
Pathologist 0 is assigned samples: Sample_1, Sample_7, Sample_8 with a total of 19 points
Pathologist 1 is assigned samples: Sample_3, Sample_5, Sample_6 with a total of 20 points
Pathologist 2 is assigned samples: Sample_0, Sample_2, Sample_4 with a total of 11 points
```

Figure 6.4: The satisfiable solution produced for the base case.

An example illustrating this process can be observed in Figure 6.4, where a satisfiable solution is present, as indicated by the **"Status: sat"** output and the subsequent printing of the **solution**.

6.2 Expanding the base case

The subsequent step involves expanding the base case by introducing additional constraints, one at a time. At this point, the program only covers a single day, which limits the insights gained from the simulation. To address this limitation, the program is extended to simulate an entire week. By running the code for each day of the week, the results for each day are printed sequentially. An important aspect of simulating a week of workflow is the transfer of points. If a pathologist earns fewer than 23 points, the remaining points must be carried over to the next day. However, it is crucial to ensure that the accumulated points do not exceed 30 for a given day.

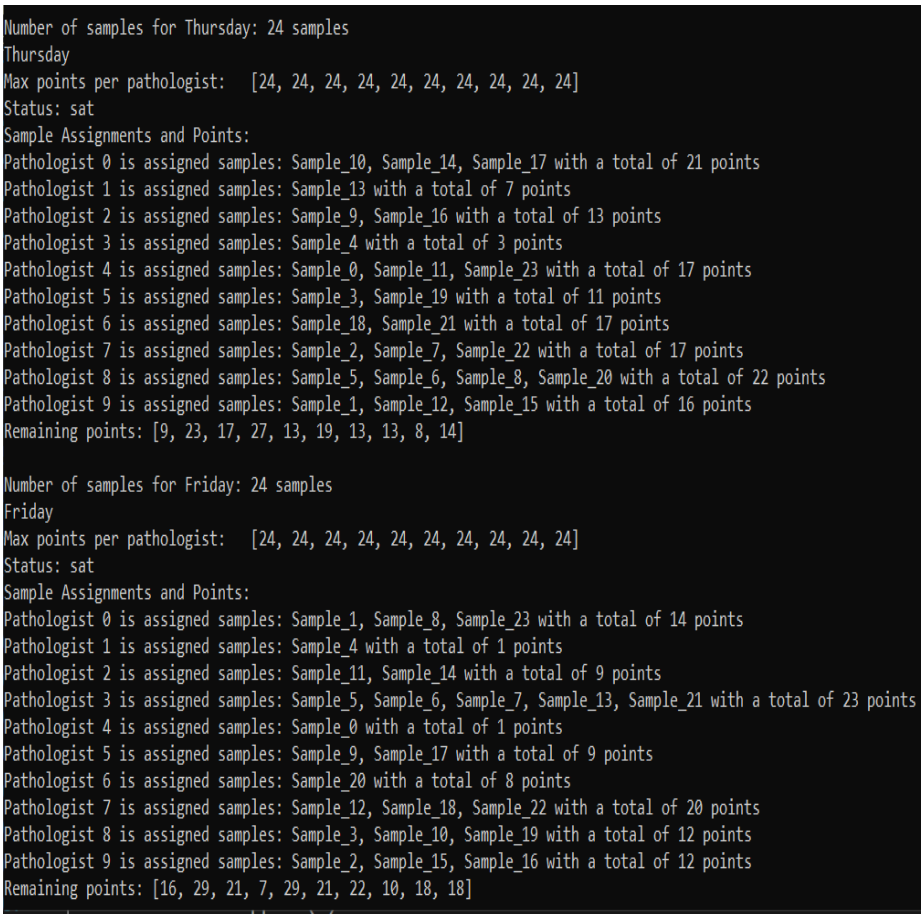

Figure 6.5: The expanded case base case with two of the days, the remaining points, and the new maximum

In Figure 6.5, the results from Tuesday and Wednesday, considering the newly added constraints, are depicted. The **"Max: [...]"** list represents the maximum number of points each pathologist can earn on a specific day. For example, on Tuesday, the maximum limit for each pathologist is 24 points, indicating that they earned at least 23 points the previous day. The results section displays the individual samples and the total points earned by each pathologist on that particular day. Additionally, the screenshot includes a list indicating the remaining points for each pathologist, representing the points carried over to the next day. In the case of Tuesday, the first pathologist has 9 remaining points. Since their total points for Wednesday would exceed 30 (24 + 9), their new maximum point limit for Wednesday is adjusted to 30. This adjustment is reflected in the updated maximum list shown for Wednesday. By simulating the workflow over a week, the program provides a more comprehensive understanding of resource allocation and point transfer dynamics.

The subsequent constraint considered was related to the special case of a pathologist falling ill, as explained in Chapter 5.4. In this case, if a pathologist is

marked as sick, their allocation of 23 points needs to be redistributed among the other pathologists who are working on that day. To handle this scenario, I created a Boolean list called **sick** that assigns True/False values to each pathologist, indicating whether they are sick or not. For example:

```
sick = [False for i in range(num_doctors)]
```

Then a constraint to ensure that if a pathologist is marked as sick, they should not be assigned any samples or points was implemented. When one pathologist is sick, the maximum points for the other working pathologists are increased to 30 points. Consequently, the 23 points originally allocated to the sick pathologist are redistributed among the other pathologists who are present that day. The additional points that pathologists earn when covering for their sick colleagues are recorded in a separate dictionary called the **Deductionlist**.

```
Pathologist 1 is sick
Deductionlist: {'Pathologist 0': 0, 'Pathologist 1': -24, 'Pathologist 2': 0, 'Pathologist 3': 0, 'Pathologist 4': 0,
'Pathologist 5': 0, 'Pathologist 6': 0, 'Pathologist 7': 0, 'Pathologist 8': 0, 'Pathologist 9': 0}
```

Figure 6.6: Example of when a pathologist is sick and the updated deduction list.

This information can be used to cover for absence at a later time. In Figure 6.6, an example is presented where one pathologist is marked as sick **"Pathologist 1"**. The updated deduction list indicates that 24 points have been deducted from the sick pathologist's allocation. By incorporating this constraint, the program accounts for the possibility of a pathologist falling ill and ensures a fair redistribution of the workload among the remaining working pathologists.

The next aspect to be incorporated is the consideration of special requests made by the pathologist, as explained in Chapter 5.6. Two specific requests are addressed: the desired amount of work hours and the preference for receiving a physical sample in addition to the PDF file copy. Regarding the request for a physical sample, I created a Boolean list that assigns True/False values to each pathologist, indicating whether they want a physical sample or not. Then a constraint is to ensure that if a pathologist desires a physical sample, it is included in the solver. For instance, the line of code

```
solver.add(request_physical_sample[2])
```

solver.add(request_physical_sample[2]) indicates that pathologist 3 has requested a physical sample.

```
Status: sat
Pathologist 2 - Request Physical Sample: True
Pathologist 6 - Request Physical Sample: True
```

Figure 6.7: Example f the two pathologists that have requested physical samples.

The output is shown in Figure 6.7 displays which pathologists have made this specific request. This information can guide the personnel working at the "table" (that is where samples are handed out) on what actions to take.

```
Friday
Max points per pathologist:   [24, 24, 24, 24, 24, 12, 8, 24, 24, 24]
Status: sat
Pathologist 2 - Request Physical Sample: True
Pathologist 6 - Request Physical Sample: True
```

Figure 6.8: Example of the new max points the pathologists can have if they have different work routines.

Finally, the other request pertains to the desired amount of work hours for a pathologist in a given week. To accommodate this, some adjustments to the simulation of a week's resource scheduling. The results displayed in Figure 6.8 showcase the new list of maximum points at the top, where all pathologists except pathologist 6 have a full-time routine with 24 points, while pathologist 6 has a part-time routine with 11 points. This allows for consideration of the pathologists' desired work hours in the scheduling process. By incorporating these special requests, the program takes into account the individual preferences of the pathologists, ensuring that their desired work hours and sample requirements are considered during the resource scheduling process.

6.3 Special cases

In addition to the main components, several operational details specific to the pathology department have been addressed in Section 5.2 and Section 5.5. These details were incorporated into the program after the core functionalities were established. Firstly, weekly meetings need to be assigned to different pathologists. Each week, five distinct meetings take place, and it is necessary to determine the pathologists responsible for each meeting. This was accomplished by introducing an additional function at the start of the simulated week where random doctors are picked. The results of these assignments can be observed in Figure 6.11, which displays the allocation of pathologists to specific meetings. Furthermore, certain pathologists have distinct areas of responsibility within the department. This means that when specific types of samples, such as CITO tests, are received, they should be assigned to the pathologists with corresponding responsibilities.

```python
#Convert the list of special samples to the correct amount of points
def special_slices_to_points():
    new_list = []
    for key, value in spes_samp_and_slice.items():
        if key.startswith('Oral'):
            new_list.append(0)
        elif key.startswith('PD-11'):
            new_list.append(1)
        elif value in point_table[1]:
            new_list.append(2)
        else:
            for ky, lst in point_table.items():
                if value in lst:
                    new_list.append(ky)
                    break
    return new_list
```

Figure 6.9: The converter created that takes slices and converts them to the corresponding points for the special samples.

Additionally, some samples differ from the previously established points system, requiring special considerations explained in section 5.2. These aspects were accommodated by incorporating a new converter that takes the special samples and calculates the points for them. This converter can be seen in Figure 6.9.

```python
special_sample_pathologist = {}
for sample, sample_groups in special_sample.items():
    m_patholog = []
    for patholog, patholog_groups in patholog_responsibility.items():
        if any(group in sample_groups for group in patholog_groups):
            m_patholog.append(patholog)
            special_sample_pathologist[sample] = random.choice(m_patholog)
```

Figure 6.10: Method used for matching the pathologist's with the correct samples based on specializations.

Figure 6.10 shows the method for matching each tagged special sample with the correct pathologist. This takes into consideration the special responsibilities that some pathologists have during that week. For the special cases to be taken into consideration some new constraints needed to be added. The constraint added were the same as for the regular samples but with a new list of slices, namely the special slices. The constraints were:

1. Ensure that each special sample is assigned to one pathologist.
2. Ensure that each special sample is assigned to at most one pathologist.
3. Alter previously made constraints to consider special points when limiting the number of points a pathologist can have. Making sure that one pathologist can only earn up to their max amount of points.

4. Ensure that each tagged special sample is assigned to the correct tagged pathologist.
5. Ensure that the total points are assigned to all pathologists. Must equal the sum of points for all the special samples.

```
Pathologist 6 : ['Mammamøte']
Pathologist 4 : ['ØNH møte']
Pathologist 3 : ['Gynmøte']
Pathologist 1 : ['Uromøte']

Responsibilities this week:
Pathologist 2 : ['Gastro CITO']
Pathologist 6 : ['ØNH CITO', 'Lymfom/hema']
Pathologist 9 : ['CITO']

Number of samples for Monday: 19 samples
Monday
Max points per pathologist:   [24, 24, 24, 24, 24, 12, 8, 24, 24, 24]
Todays special samples: ['Oral_0', 'Oral_1', 'Gastro CITO_2', 'ØNH CITO_3', 'ØNH CITO_4']
Status: sat
Pathologist 2 - Request Physical Sample: True
Pathologist 6 - Request Physical Sample: True
Sample Assignments and Points:
Pathologist 0 is assigned samples: Sample_1 with a total of 1 points
Pathologist 1 is assigned samples: Sample_3, Sample_17 with a total of 7 points
Pathologist 2 is assigned samples: Sample_10, Oral_0, Gastro CITO_2 with a total of 7 points
Pathologist 3 is assigned samples: Sample_2, Sample_7, ØNH CITO_3 with a total of 7 points
Pathologist 4 is assigned samples: Sample_4, Sample_8, Sample_16 with a total of 9 points
Pathologist 5 is assigned samples: Sample_9 with a total of 3 points
Pathologist 6 is assigned samples: Sample_5, Sample_6, Sample_12 with a total of 8 points
Pathologist 7 is assigned samples: Sample_0, Sample_13, Sample_18, ØNH CITO_4 with a total of 14 points
Pathologist 8 is assigned samples: Sample_14, Oral_1 with a total of 6 points
Pathologist 9 is assigned samples: Sample_11, Sample_15 with a total of 9 points
```

Figure 6.11: The results produced with the addition of special cases.

There were also constraints added to consider special samples when a doctor is sick and for even distribution. These are the same as explained previously but with special samples instead of regular samples. The outcome of these modifications can be seen in Figure 6.11, which illustrates the responsibilities assigned to each pathologist and showcases the matching of special samples to the appropriate pathologists.

One final aspect that was addressed related to the requirement of analyzing all samples each day. It was recognized that while this is the ideal scenario, it may not always be possible to complete the analysis in a single day. In such cases, the remaining samples can be carried over to the next day.

```
analyzed = [Bool(f"is_analyzed_{i}") for i in range(num_samples)]
not_analyzed = []
```

Figure 6.12: Boolean list analyzed and empty list non_analyzed to keep track of which samples have been analyzed.

To facilitate this, certain constraints were modified. A new Boolean list, **analyzed**, was created to indicate whether each sample has been assigned to a pathologist (True) or not (False). Additionally, an empty list, **non_analyzed**, was introduced to store the samples that have been assigned but, not analyzed and need to be carried over to the next day. This can be seen in Figure 6.12. The constraint that ensured each sample assigned to a pathologist was adjusted to allow for either assignment or marking as non-analyzed. After the **status.check()** operation, the **non_analyzed** list is compared to the original list to identify the specific samples, which are then transferred to the following day's samples.

```
Samples not analyzed today: []
```

Figure 6.13: List of samples not analyzed one day.

The results from this can be shown in Figure 6.13 where the list is empty. That means that all samples have been analyzed that day and there are no extra samples for the next day. By addressing these operational details, the program considers the specific requirements of the pathology department, including the assignment of meetings, area responsibilities, and the handling of non-analyzed samples.

6.4 Processing time

In addition to the aforementioned details, the processing time of each sample was considered an important factor in the scheduling process. The duration required for processing each sample was accounted for, ensuring that the total processing time for each pathologist did not exceed a regular workday. To simulate the processing time, two random generators were implemented - one for regular samples and another for special samples. The processing time was determined based on the size of the sample, with larger samples requiring more time to process. Subsequently, a constraint was added to ensure that the cumulative processing time of samples assigned to a pathologist did not exceed 400 minutes, equivalent to a typical workday of 7 hours.

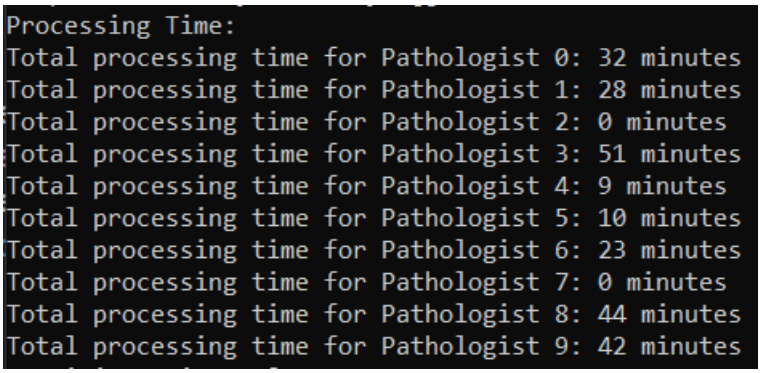

Figure 6.14: The different processing times each pathologist has attained one day.

The results of incorporating this processing time constraint can be observed in Figure 6.14. This aspect holds significance for further optimization of the program, as it allows for evaluating whether more samples can be analyzed within the available time frame. However, since there is currently no available data regarding the actual processing times at the pathology department, it is challenging to assess the potential improvements achievable through such optimizations.

6.5 Optimizing the program

The final step of the development is to optimize the program. In the optimization process, various aspects can be targeted, such as minimizing the makespan (the total time required to complete all tasks) or reducing resource idle time (minimizing the time when resources are not actively engaged in any task). However, for this particular study, my focus has been on optimizing two specific areas: throughput, which represents the number of samples processed each day, and the distribution of points among pathologists.

The rationale for selecting these two fields for optimization is twofold. Firstly, the primary objective is to maximize the number of samples analyzed each day, as it directly contributes to the efficiency and productivity of the process. By processing as many samples as possible, the overall workflow is improved. Secondly, it is crucial to ensure an equitable distribution of workload among pathologists. By optimizing the distribution of points, we aim to allocate samples in a balanced manner, preventing any pathologist from being overwhelmed with excessive work while ensuring that each pathologist contributes to the process.

These areas were chosen for optimization primarily because of the availability of sufficient data to support the optimization efforts. The optimization process will make use of the Optimized module from Z3 [7]. This module is specifically designed to handle optimization tasks and will facilitate the implementation of optimization techniques in the program. By leveraging this module, we can apply optimization methodologies to enhance the throughput and achieve a more balanced distribution of workload among pathologists.

To address the optimization of sample throughput, the Optimize module within Z3 was utilized. Implementing this module involved making a simple adjustment by changing the Solver() function to Optimize() after all the solving processes had been completed. Additionally, a constraint was introduced to track the number of samples that had been analyzed. This constraint was formulated as follows:

```
1    solver.maximize(num_samples_analyzed)
```

By including this line of code, the objective of the optimization process was specified: to maximize the value of the variable num_samples_analyzed, which represents the number of samples that have been analyzed by the pathologists. By formulating the optimization objective in this manner, the solver aimed to find an optimal assignment of samples to pathologists that maximized the number of analyzed samples.

```
Number of Samples Analyzed: 10
Optimized points Distribution: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

Figure 6.15: The results produced from the optimization.

During the optimization process, the solver would attempt to find a feasible solution that satisfied all the given constraints while simultaneously maximizing the number of analyzed samples. If a feasible solution was found, it would represent an assignment that optimized the throughput by maximizing the number of samples analyzed. Conversely, if the solver determined that no feasible solution existed, it indicated that there was no way to simultaneously satisfy all the constraints and maximize the number of analyzed samples. Figure 6.15 depict the results from the optimization. The line **Number of samples analyzed: 10** shows that a total of 10 samples have been analyzed. That is all of the samples that were to be processed that day. The second solution that the solver tried to find was the point distribution. Minimizing the distance between the maximum point a pathologist can have and the points they are assigned. The second list tries to show an optimized point distribution amongst the pathologist, but as seen in Figure 6.15 the list contains all zeros. Here is where the OMT solver failed to satisfy all the constraints and provide a solution to the optimization.

By incorporating the Optimize module and formulating the objective in terms of maximizing the number of samples analyzed, the optimization process aimed to improve the throughput of the sample analysis, ultimately enhancing the overall efficiency of the resource scheduling system.

Chapter 7

Evaluation of the results

This chapter outlines the evaluation methodology employed to assess the outcomes generated by the SMT solver within the context of resource scheduling. Specifically, the evaluation leverages Conformance Checking as a means of comparing the schedules generated by the solver against the actual execution of the scheduling process within the pathology department at Haukeland Hospital. The objective is to identify any discrepancies between the planned schedule and the real-world execution, thus estimating the feasibility of the solver's output. To facilitate this evaluation, event data obtained from the pathology department during the month of September 2021 is processed and analyzed. This data set encompasses resource utilization as well as any deviations from the intended schedule.

Subsequently, the expected behavior, represented by the devised SMT solver, is defined. This serves as the benchmark against which the actual execution of the resource scheduling process will be assessed. The CC techniques are then applied, involving a manual comparison between the observed data and the simulated results generated by the solver. The goal is to ascertain the degree of conformance between the two, thereby shedding light on the solver's efficiency in producing feasible schedules.

The resulting CC analysis is subjected to through examination. If the actual execution of the resource scheduling process aligns closely with the expected behavior defined by the SMT solver, it suggests that the solver has successfully generated a feasible solution. However, in cases where discrepancies arise between the planned schedule and the real-world execution, it may necessitate revisiting and refining the SMT solver to better account for these deviations. The ultimate aim of this evaluation process is to assess the performance of the SMT solver in addressing resource scheduling problems and identify areas for potential improvement.

By employing this evaluation methodology, a comprehensive understanding of the solver's performance in the context of resource scheduling is achieved. The findings contribute to the identification of potential enhancements to the solver and provide valuable insights for optimizing the resource allocation process within the pathology department.

7.1 Processing and analyzing the data/Data Collection

As previously highlighted in the introductory section of this chapter, a data set encompassing a wide array of information was obtained from the pathology department, specifically from the month of September 2021. While this data set proved to be a valuable resource, it contained both relevant and irrelevant information for the purposes of this study. Consequently, the initial step involved data cleansing, which entailed removing unnecessary data and computing missing values based on available information. Furthermore, the data required modification to ensure its suitability for utilization in CC analyses, which formed a crucial component of the evaluation process.

To commence the analysis, a comprehensive examination of the data set was conducted for a specific week, spanning from September 6th to September 10th, 2021. Mentioning that one week is Monday to Friday, that is a regular work week. Noteworthy findings emerged from this investigation, such as the presence of 20 to 21 pathologists working each day, with an average assignment of 237 samples in total. The majority of samples consisted of 1-2 slices, although there were instances where samples contained a greater number of slices. Additionally, the samples were identified solely by their respective IDs, necessitating translation to determine whether they were special. The working schedules of pathologists exhibit variations, with some pathologists working the entire week and others only on specific days. Establishing the specialization of each pathologist and their corresponding work routines also required careful consideration, incorporating certain assumptions as necessary. These preparatory steps aimed to facilitate the simulation of a realistic week of operations, enabling the SMT solver to generate a solution that aligns with real-world scenarios.

By conducting a meticulous analysis of the data set and addressing the intricacies surrounding pathologists, sample characteristics, and work routines, a comprehensive foundation was established for the subsequent execution of the SMT solver and the evaluation of its outcomes. This meticulous approach ensures that the simulation accurately reflects the operational realities within the pathology department, thereby enhancing the validity and relevance of the evaluation process.

7.2 Define metrics and data alignment

This section introduces the metrics employed to assess the conformance between simulated and real-life schedules, as well as the measures taken to ensure that both schedules cover the same data. Throughout the development phase, the program primarily operated with randomly generated data and limited data sets, as the precise number of pathologists and samples involved in daily operations was initially unknown. This approach allowed for more efficient code execution and iteration. However, as discussed in Section 7.1, it became evident that the number of pathologists and samples was significantly higher than initially assumed. Consequently, following the adjustments made to the data set, a decision was made to subject the program and the SMT solver to testing using the same variables as a specific week (from September 6th to September

10th, 2021) based on the data obtained from the pathology department. This evaluation aimed to ascertain the program's ability to produce a satisfactory solution for that particular week.

Given that the conformance checking was conducted manually, the solver's variables needed to be updated with the corresponding data from the pathology department. This included information on the exact number of pathologists, their specializations, and responsibilities, as well as the number of samples, the number of slices per sample, and their respective types (e.g., mom, gastric, gyno, etc.), along with the details of any special samples, such as the quantity, the number of slices per special sample, and the specific types (e.g., oral, needle biopsy, etc.).

During data processing, it was observed that certain results deviated from the predefined workflow rules outlined in the pathology department's documentation. That is the maximum point that a pathologist can earn in one day. The reasons behind these discrepancies remain unclear, as no definitive explanations were provided. Consequently, modifications were made to the code to accommodate the possibility of earning more than 25 points, with any additional workload recorded in the deduction list to provide a visual representation of the extra work performed by pathologists within a given day.

Moreover, the metrics used for comparing the two solutions need to be identified. These metrics are as follows:

1. Accuracy: This metric assesses the level of conformity between the two solutions and evaluated whether all constraints have been satisfied.
2. Throughput: This metric measures the number of samples analyzed each day, reflecting the efficiency of the scheduling process.
3. Point distribution: This metric evaluated the fairness and balance in the allocation of points among the pathologists, providing insights into the workload distribution within the department.

7.3 Comparative analysis

In this section, a comparative analysis between the simulated and real-life schedules is presented. As mentioned earlier, one of the chosen metrics for comparing the actual and simulated behavior is the point distribution among pathologists each day. During this stage of evaluation, it was observed that the solver's computation time increased, most likely due to the introduction of additional variables. Despite the increased time required to obtain results, the solver consistently produced feasible solutions. It is worth noting that the running time of the program also increased during this step, potentially influenced by the larger number of variables introduced. Moreover, the computational capabilities of the current machine, an HP laptop with a Windows operation system. The computer has 16 GB RAM and 2 cores. This may have contributed to the increased processing time. Employing a more powerful machine would be advisable for improved processing efficiency.

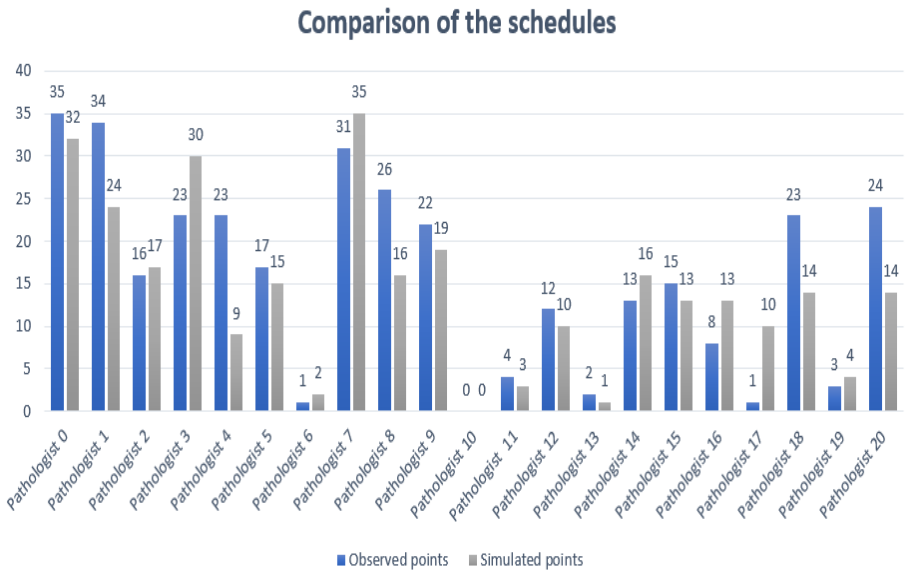

Figure 7.1: Comparison of points Monday 06.09.23

It is important to note that the SMT solver can provide multiple satisfiable solutions, and the solution depicted in Figure 7.1 represents one of many possible solutions. The figure illustrates the comparison between the solution provided by the SMT solver for Monday, September 6th, 2021, and the actual behavior of the pathology department on that same day. The y-axis represents the points, while the x-axis represents the different pathologists. The blue column represents the observed points, while the grey column represents the simulated points.

From Figure 7.1, it can be observed that the two distributions are not identical, which could be attributed to the lack of information in certain areas. Questions arise regarding why some pathologists can earn more than 25 points and why some pathologists have only one sample worth one point. There is also one pathologist that has zero points. That is because his specialization was oral samples, which are worth zero points. These unknown factors have led to certain assumptions being made, which could explain the differences observed. Another noteworthy observation is that the point distribution in the simulated process is relatively similar to the observed distribution, with some pathologists working less and others working more. The observed point distribution can also be attributed to the assignment of samples, which is influenced by the types of samples received each day and the availability of pathologists with corresponding specializations. It should be noted that certain types of samples, such as 100 skin samples, can only be allocated to pathologists specializing in dermatology rather than being distributed among all pathologists. Therefore, the point distribution is inherently influenced by the specific combinations of sample types and the available pathologists with the corresponding expertise.

Another metric used to compare the two solutions was accuracy, which ensured

that the constraints were satisfied during the evaluation. This involved verifying the correct matching of pathologists and samples based on their specializations, ensuring that special samples were assigned to the appropriate pathologists, and confirming adherence to the designated work routines. Despite certain assumptions made during this process, the matching and work routines were correctly implemented. The evaluation results for the remaining days of the week can be found in Appendix B.

The final metric employed in this comparison was throughput, which assessed the solver's ability to allocate all available samples by the department's capacity. The solver successfully assigned all samples, indicating a comparable level of throughput to that of the department.

7.4 Quantitative assessment and qualitative evaluation

This section focuses on the quantitative assessment of the schedules, examining the level of conformance between the actual behavior and the simulated behavior by calculating the previously defined performance metrics. Additionally, the qualitative evaluation of the results is discussed, considering factors such as the feasibility and practicality of the simulated schedules compared to the real-life schedules. Is the solution good enough? Is this way of creating a schedule more practical?

Upon analyzing the results of the simulated schedules and comparing them to the real-life schedules, it was found that the accuracy metric demonstrated a 100% rate of correctly assigned samples. No pathologists were assigned samples that did not align with their specialization. Regarding the point distribution metric, the simulated schedules showed similar results to the real-life schedules, neither significantly better nor worse. Furthermore, in terms of throughput, both solutions effectively scheduled the same number of samples each day.

Regarding the qualitative aspects of the schedules, the solver consistently produced feasible solutions. The evaluation results indicate that the solver accurately adheres to various constraints, both quantifiable and non-quantifiable. Moreover, employing the solver offers a more time-efficient approach to obtaining solutions for resource scheduling problems compared to manual creation.

Overall, the evaluation suggests that the solver's simulated schedules demonstrate high accuracy in conforming to the defined constraints and providing a feasible solution. Additionally, the solver offers a more efficient alternative to manual scheduling, making it a valuable tool for resource scheduling problems.

7.5 Interpretation of results

In this section, the alignment and discrepancies between the simulated and real-life schedules are examined. The strengths and limitations of the solver in accurately capturing real-world scheduling complexities are discussed, as well as potential areas for improvement or further investigation.

The solver demonstrates the capability to produce a solution that aligns with real-life schedules while satisfying all the specified constraints. It successfully generates different solutions to the same problem, offering flexibility in schedule generation. Another notable strength of the solver is its time efficiency. By running the code and waiting for a solution to be generated within minutes, the solver outperforms the current manual process of calculating and assigning samples. An example, when the data set had 250 samples, it took the solver approximately 5 minutes to produce a solution.

However, certain limitations exist in the solution produced by the solver. Specifically, the simulated schedules do not necessarily exhibit improvements over the observed schedule. The point distribution between the two solutions appears similar, suggesting that the solver may not optimize this aspect effectively. Further investigation and refinement in this area could prove beneficial for enhancing the solver's performance.

In conclusion, the solver demonstrates its ability to produce schedules that align with real-life schedules while adhering to constraints. It offers advantages in terms of time efficiency compared to manual scheduling processes. There is room for improvement in optimizing the point distribution aspect of the schedules, warranting further investigation.

Chapter 8

Results

In this chapter, I will present and discuss the outcomes of this research. This chapter aims to objectively and accurately report the findings that are relevant to answering the research questions defined in chapter 3.

8.1 Reflection over the solvers

The tools utilized in this thesis have shaped the solution in some way. Through reflection, insight into their limitations and strengths will unlock the path to innovation. This section justifies the choices made, evaluates their effectiveness, and offers insights for future investigation. The reason for choosing the different solvers is described in detail in section 2.1.1 and 2.3.1. A brief explanation of their well-known strength as well as their weaknesses are described, but this chapter is based on personal findings.

As previously described, the use of Satisfiability Modulo Theories (SMT) involves formulating logical formulas. Initially, translating problems into the context of logical formulas can be challenging. However, as Julius Caesar once said, "Experience is the teacher of all things." This new way of thinking may present difficulties but also holds significant benefits. It allows for the formalization and precise definition of problem statements, eliminating ambiguity or confusion that may arise from informal or vague descriptions. To fully leverage the power of SMT, the Z3 solver was employed. The documentation for Z3 proved to be extensive but also challenging to navigate, requiring considerable time and investment to find the desired information. Furthermore, it was observed that problem-solving became increasingly challenging as the complexity of the problem grew. While the solver can provide reasons for the unsatisfiability of a problem, in cases with numerous constraints, the solver was unable to produce results. Identifying the specific constraint leading to unsatisfiability could be valuable in understanding the underlying issues causing the failure. Due to the limitations encountered, certain assumptions had to be made regarding the points of failure within the schedule. Similar observations were made when using Z3 with Optimize. As the problem's complexity increased, the solver struggled to generate optimized solutions.

8.2 Answering the research questions

RQ1: Can SMT find a solution for a research scheduling problem with non-quantifiable constraints?

Stephen Covery said; "The key is not to prioritize what's on your schedule, but to schedule your priorities." And in a way that is exactly what the creation of constraints for the SMT solver is. Identifying and formulating constraints and telling the solver how to handle them. From the work done in this thesis, it is shown that the solver was able to find a solution for a research problem with non-quantifiable constraints. An observation made is that the solver can produce several solutions to the same problem, if there are any satisfiable solutions. Running the code over and over again the result was never the same. It is worth mentioning that random data was used during the development of the program, but during the evaluation of the program real-life data was used and it produced several solutions as well. The solver was not able to produce all of the solutions at once. This is because this is very time-consuming and hard to handle for the solver. Another observation made during the evaluation, is that it is not clear if a solution produced is necessarily better than any other or observed schedule.

In summary, the utilization of an SMT solver for handling constraints in resource scheduling offers promising results and potentially multiple solutions, considering non-quantifiable constraints. It is essential to approach the output of the solver with a critical mindset, considering the specific needs and objectives of the scheduling problem at hand. By combining the power of the solver with human experience, a comprehensive and well-informed decision-making process can be achieved, ensuring the most effective and suitable schedule for the given contexts.

RQ2: What does the SMT solution with no constraints compare to the SMT solution that takes into consideration non-specific constraints? And how can this impact the workflow at the pathology department?

The approach used in this thesis for developing the code for the SMT solver gives an answer to research question 2. This approach turned out to be a sensible approach. That is, step by step making the problem more complex. Identifying and adding more constraints. At each step observe if the solver was able to produce a solution. With this approach, further development becomes less complicated

if another demand/constraint occurs and the solver needs to consider it. An observation made during the development was that by adding more and more constraints it made the resource allocation more difficult, but still able to produce a solution. With no constraints, the solver was able to always return a satisfactory result, but by adding more and more constraints these instances became more infrequent. What I can deduce from this is that if there were no constraints at all there is always a solution, but by adding more and more constraints that may not always be the case.

In summary, the approach utilized in this thesis to develop the code for the SMT solver has provided valuable insights into the research question at hand. The step-by-step method of progressively introducing complexity and incorporating additional constraints has proven to be a practical and effective approach. By carefully monitoring the solver's ability to produce solutions at each stage, a deeper understanding of the impact of constraints on the resource allocation problem has been attained. Ultimately, the choice between the SMT solution with no constraints and the solution that incorporates non-specific constraints depends on the objectives and priorities of the resources allocation problem at hand. The former offers flexibility and a broader solution space, while the latter provides a more focused and refined set of solutions tailored to specific constraints. Understanding the trade-offs and implications of each solution variant is essential in determining the most suitable approach for addressing the unique requirements of the resource scheduling scenario.

RQ3: By taking non-quantifiable restrictions into consideration, how can OMT help optimize the resource scheduling?

There is not a clear result that can be deduced from this thesis. Because of the complexity of the program for the SMT solver, the OMT solver found it difficult to produce an optimized solution. After identifying areas were optimization could be useful the solver was not able to produce much result. In addition, there was limited data to work with and the reasons for a unsatisfiable result was not able to be presented. This lead to not that much development with regards to optimization. Saying that, areas were optimization could be useful have been identified and it is an area relevant for future work.

The findings of this thesis do not yield a definitive and conclusive outcome. The intricate nature of the program developed for the

SMT solver, coupled with the inherent complexity of the optimization process, presented challenges in attaining an optimized solution. Despite identifying potential areas where optimization could prove beneficial, the solver's capacity to generate substantial results was limited. Furthermore, the scarcity of available data and the inability to ascertain the reasons behind unsatisfiable outcomes impeded significant advancements in the realm of optimization.

It is important to acknowledge that the limitations encountered during the research process restricted the extent of progress achieved in the domain of optimization. However, the identification of areas where optimization could potentially enhance the resource scheduling problem represents a noteworthy contribution and establishes a foundation for future investigations. The relevance of optimization in resource scheduling remains a compelling and promising avenue for further exploration and inquiry. In conclusion, while this thesis does not offer conclusive findings, it sheds light on the challenges associated with implementing optimization within the SMT solver framework. Nevertheless, the identification of potential areas where optimization could provide a valuable springboard for future research and serves as a reminder of the relevance and potential of optimization in the context of resource scheduling.

RQ 4: Is the SMT solution to the resource scheduling problem applicable to the pathology department at Haukeland Hospital?

Addressing research question 4 entailed the application of the SMT solution to the resource scheduling problem specifically within the pathology department at Haukeland Hospital. To accomplish this, a comprehensive understanding of the department's operations, demands, rules, and constraints was crucial. This involved an in-depth exploration of the department's scheduling challenges and the expectations placed on the solver to effectively address them. The focal point of this thesis revolved around resource allocation, with the allocated resources being the daily influx of samples that require assignment to pathologists for analysis. It became evident that the pathology department operated under a multitude of diverse demands and rules, some relatively straightforward to implement, while others posed greater challenges. Notably, navigating the non-quantifiable constraints proved particularly intricate encompassing factors such as unexpected leave, complications associated with specialized samples or specimens, and specific requests for varying work-

loads on particular days.

After the development of the program capable of generating solutions to the resource scheduling problem within the pathology department, it became imperative to assess the outcomes. To achieve this, conformance checking was employed, involving a manual evaluation of the alignment between the simulated schedules generated by the SMT solver and the observed schedules derived from actual departmental data. This conformance analysis served as a means to gauge the solver's performance and efficacy. It is worth emphasizing that the available data served as a benchmark against which the solver's results were measured. However, it is important to note that the available data might be relatively limited, posing a potential disadvantage. Discrepancies or deviations identified between the simulated schedules and the observed schedules shed light on areas where the solver may not accurately capture the intricacies and nuances of the real-world scheduling processes within the pathology department. Consequently, this conformance analysis provides valuable insights into the effectiveness and reliability of the SMT solver in simulating the resource scheduling problem within the pathology department, facilitating evidence-based decision-making, and suggesting potential enhancements to the scheduling process.

The metrics chosen to compare the SMT solver's solution with the department's observed solution were point distribution and throughput, specifically the number of samples analyzed each day. The evaluation revealed that the SMT solver performs comparably to the observed departmental solution. All samples were successfully assigned and analyzed, mirroring the performance of the pathology department. The point distribution, while not identical, exhibited a similar pattern to that of the department. However, it remains inconclusive whether the SMT solver's solution is superior in terms of point distribution. This is where the optimization failed. Notably, a noteworthy observation during the development process was that each sample was assigned to the appropriate pathologist, satisfying the various work routines within the department. A reflection arising from the conformance checking was that as the solver processed larger quantities of data, i.e., variables, the time required to find a satisfiable solution increased. Nevertheless, it is important to highlight that despite this consideration, the solver demonstrated superior effectiveness compared to the observed departmental solution.

In conclusion, the application of the SMT solution to the resource

scheduling problem within the pathology department at Haukeland Hospital provides valuable insights. The comparison between the solver's solution and the observed departmental solution in terms of point distribution and throughput reveals comparable performance. While the solver's results align closely with the department's practices, it remains uncertain whether it offers better outcomes than the department's solution. Furthermore, the assignment of each sample to the appropriate pathologist and the satisfaction of distinct work routines underscore the solver's ability to address the complexities of the pathology department's scheduling requirements. The conformance-checking process exposed the potential impact of processing larger quantities of data on the solver's computational time. Nonetheless, the SMT solver proves to be an effective tool, surpassing the observed departmental solution in terms of creation efficiency and providing a solid foundation for further research and refinement in resource scheduling optimization within the pathology domain.

Chapter 9

Conclusion and future work

This thesis presents the creation and evaluation of the use of SMT solvers and the handling of non-quantifiable constraints. In addition the use of OMT solver in the hope of optimizing a solution. The results from the solvers presented in this thesis provide proof that a satisfiable solution is possible. In addition the potential for optimizing the solution. This section chapter aims to summarize the results and present suggestions for further research and development.

9.1 Conclusion

The primary objective of this thesis was to investigate the feasibility of using SMT to address a resource scheduling problem, particularly focusing on the incorporation of non-quantifiable constraints and striving for optimization. In conclusion, the SMT solver successfully considers non-quantifiable constraints when they can be formulated as logical formulas, resulting in a feasible solution that closely resembles the observed schedule and satisfies all constraints. It is important to note that the solution produced by the solver is not necessarily superior to the observed schedule.

One significant finding of this research is the evident need for optimization, which could yield valuable insights and improvements. Specific areas within the scheduling process have been identified as potential targets for optimization, although further investigation is required. Additionally, the scarcity of available data emerged as a prominent challenge, limiting the extent to which optimization could be explored. Obtaining more data would be beneficial in refining the scheduling process and achieving enhanced results.

Overall, the results of this thesis demonstrate that SMT is capable of generating a schedule that adheres to both quantifiable and non-quantifiable constraints. Additional measures like more extensive data collection and refinements about optimization are necessary to develop an improved solution that enhances the overall workflow. The positive outcome of this research is proof of the solver's ability to provide a satisfiable solution and the significant time efficiency gained. That is through the utilization of the SMT solver, as it streamlines the scheduling process compared to manual approaches.

9.2 Future work

The results of this research have provided valuable insights into potential avenues for further exploration within the scope of this thesis. One crucial aspect for future work is the development of a method to visualize the reasons for "unsat" (unsatisfiable) results. Understanding the specific scheduling issues causing unsatisfiability would enable targeted improvements to be made. Z3 does offer a built-in method, unsat.core(), for displaying the reason for unsatisfiability. When a problem becomes too complex with numerous constraints, it becomes challenging or even impossible for Z3 to generate the core of the problem. This realization came late in the program's development process, leading to frustration in further refining the code. By visualizing the reasons for unsat results, both the program's development and the quality of results can be enhanced.

The optimization aspect of the thesis presents another potential area for future development. As mentioned in Section 6.5, various aspects of resource scheduling can benefit from optimization. During the program's development, one idea that emerged was optimizing the time required for sample analysis to improve overall efficiency. Due to the lack of data from the Haukeland Hospital's pathology department, this optimization was not feasible. Although a mechanism for tracking the processing times of individual pathologists has been implemented, further optimization based on these times remains to be explored. Optimizing sample processing time has the potential to increase efficiency and reduce idle time, contributing to an improved workflow. Additionally, exploring other areas for optimization within the scheduling process may uncover further opportunities for improvement.

A third area of interest for future development is the user-friendliness of the program's results. That is if the solution is something that

the pathology department is interested in obtaining this solution. Currently, all the solver's output is printed in the terminal. While efforts have been made to present the results in a clear and readable format, it is not an optimal solution. This can lead to potential misunderstandings of the results. To address this, incorporating a front-end interface to display the solver's results could significantly enhance user-friendliness and facilitate better comprehension and interpretation of the outputs.

Bibliography

[1] Will van der Aalst. *Process Mining - Data Science in Action*. Springer Berlin Heidelberg, 2016.

[2] College of American Pathologists. *What is Pathology?* Definition of Pathology. URL: https://www.cap.org/member-resources/articles/what-is-pathology (visited on Oct. 3, 2022).

[3] Microsoft Corporation. *Documentation for Online Z3, Guide | Online*. Z3 homepage. URL: https://microsoft.github.io/z3guide (visited on June 23, 2023).

[4] Microsoft Corporation. *Z3: z3py Namespace Reference*. Z3 Distinct. URL: https://z3prover.github.io/api/html/namespacez3py.html#aac3e33e977e8037611916937caffa78e (visited on June 22, 2023).

[5] Ray Cruickshank. *Physical Resource Management*. Examples of physical resources. URL: https://www.pmeducation.com/post/2019/02/17/physical-resource-management (visited on Feb. 17, 2023).

[6] CVC4. *About CVC4*. CVC4 homepage. URL: https://cvc4.github.io/ (visited on Feb. 17, 2023).

[7] Doxygen. *Optimize Class Reference*. Optimize Class Explanation. URL: https://z3prover.github.io/api/html/classz3py_1_1_optimize.html (visited on June 22, 2023).

[8] Engati. *Satisfiability*. Definition of Satisfiability. URL: https://www.engati.com/glossary/satisfiability (visited on Oct. 3, 2022).

[9] HandWiki. *Distinct (mathematics)*. Distinct definition within mathematics. URL: https://handwiki.org/wiki/Distinct_(mathematics) (visited on June 22, 2023).

[10] Department of Health and Human Services. *Blood and pathology tests*. What are pathology test? URL: http://www.betterhealth.vic.gov.au/health/conditionsandtreatments/Blood-and-pathology-tests (visited on Mar. 13, 2023).

[11] IBM. *What is Process Mining?* Definition of process mining. URL: https://www.ibm.com/cloud/learn/process-mining (visited on Oct. 3, 2022).

[12] SRI International. *The Yices SMT Solver*. Yices homepage. URL: https://yices.csl.sri.com/ (visited on Feb. 17, 2023).

[13] Kissflow. *What is Workflow Scheduling? - How it Di ers from Task Scheduling*. Advantages of workflow scheduling. URL: https://kissflow.com/workflow/how-workflow-scheduling-optimizes-processes/ (visited on Feb. 17, 2023).

[14] MathSAT. *The MathSAT 5 SMT Solver*. MathSAT homepage. URL: https://mathsat.fbk.eu/ (visited on Feb. 17, 2023).

[15] ppmx. *Sudoku Solver using Z3*. GitHub Repository for Sudoku example. URL: https://github.com/ppmx/sudoku-solver (visited on Sept. 21, 2022).

[16] ProcessMining. *Conformance Checking - Process Mining*. Definition of conformance checking. URL: http://processmining.org/conformance.html (visited on Oct. 3, 2022).

[17] Microsoft Research. *Z3*. Z3 by Microsoft Research. URL: https://www.microsoft.com/en-us/research/project/z3-3/ (visited on Oct. 19, 2022).

[18] Microsoft Research. *Z3*. GitHub Repository for Z3. URL: https://github.com/Z3Prover/z3 (visited on Sept. 21, 2022).

[19] Patrick Trentin Roberto Sebastiani. *On Optimization Modulo Theories, MaxSMT and Sorting Networks*. Definition of OMT. URL: http://arxiv.org/abs/1702.02385 (visited on Oct. 4, 2022).

[20] Runn. *The Beginner's Guide to Resource Scheduling*. Definition of resource scheduling. URL: https://www.runn.io/blog/resource-scheduling (visited on Feb. 19, 2023).

[21] SearchCIO. *What is workflow? Definition and Examples*. Definition of worklfow. URL: https://www.techtarget.com/searchcio/definition/workflow (visited on Oct. 18, 2022).

[22] University of Trento - Italy. *The OptiMathSAT OMT Solver*. Homepage OptiMathSAT. URL: https://optimathsat.disi.unitn.it/ (visited on June 22, 2023).

Appendices

Appendix A

Source code

The source code for the plug-in is available at this URL:
`https://github.com/Z3Prover/z3`.

The source code for the SMT solver is available at this URL:
`https://github.com/solmariewallevik/Z3_pathology_scheduler`.

Appendix B

Tables of the comparison of point distribution.

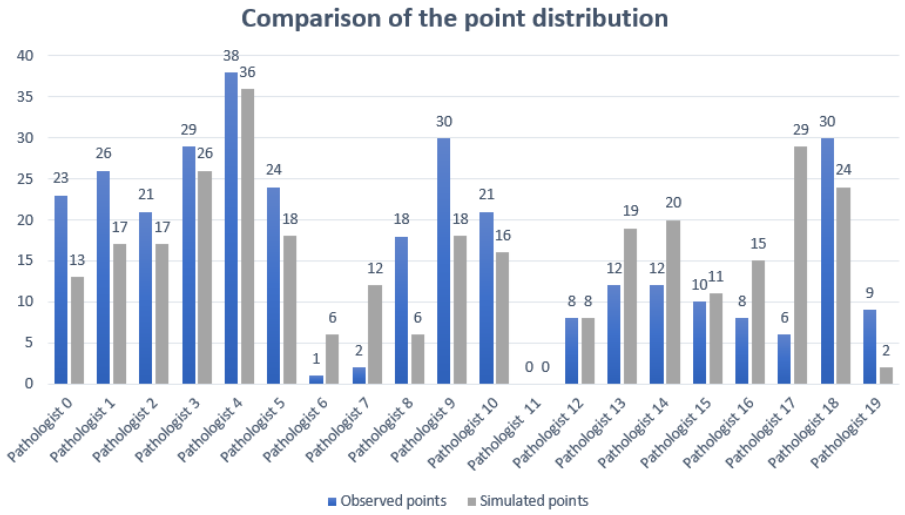

Figure B.1: Comparison of the point distributions on Tuesday 07.09.2021.

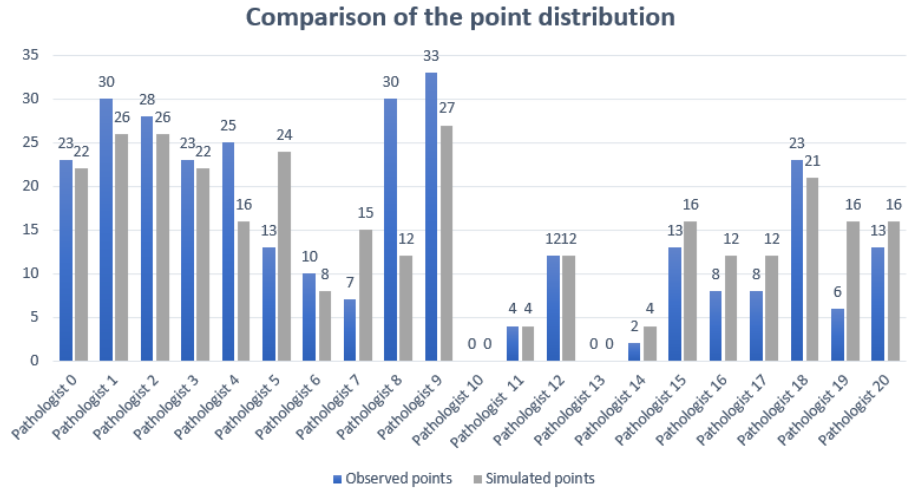

Figure B.2: Comparison of the point distributions on Wednesday 08.09.2021.

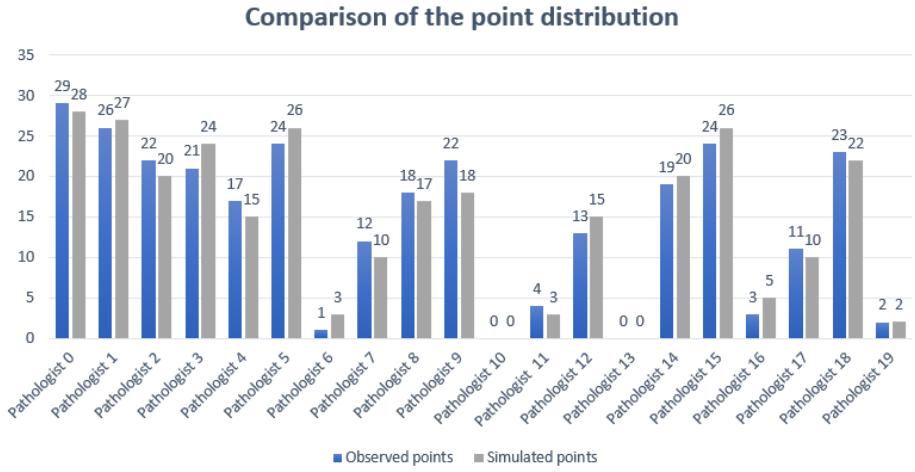

Figure B.3: Comparison of the point distributions on Thursday 09.09.2021.

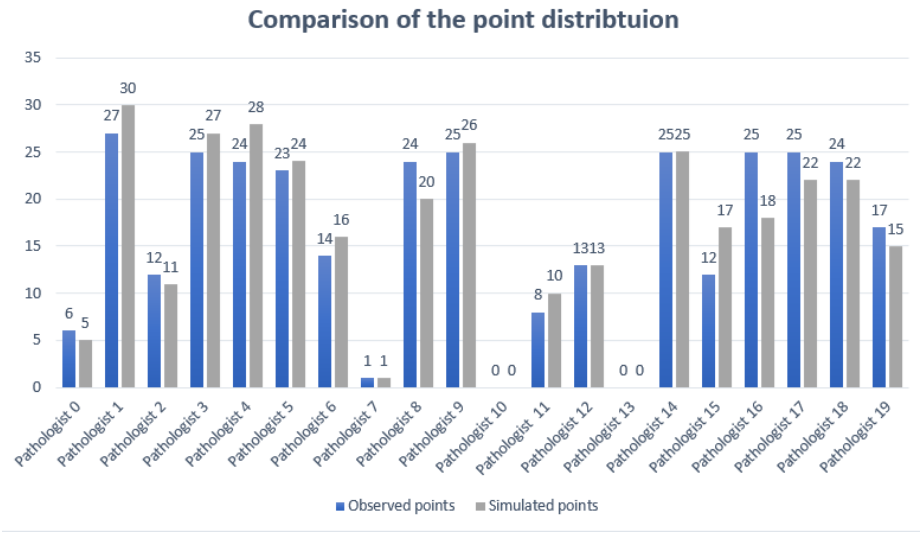

Figure B.4: Comparison of the point distributions on Friday 10.09.2021.