CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Scheduling Software for Improving Teaching Assistant Schedule Satisfaction

Bachelor's thesis in Computer Science and Engineering

Kusai Al Malt
Nova Alldén
Anna Blomberg
Mai Uy Vuong
Neo Zanders

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2026

Bachelor's thesis 2026

Scheduling Software for Improving Teaching Assistant Schedule Satisfaction

Kusai Al Malt
Nova Alldén
Anna Blomberg
Mai Uy Vuong
Neo Zanders

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2026

Scheduling Software for Improving Teaching Assistant Schedule Satisfaction
Kusai Al Malt Nova Alldén Anna Blomberg Mai Uy Vuong Neo Zanders

Supervisor: Niklas Broberg, Department of Computer Science and Engineering
Examiners: Patrik Jansson, Department of Computer Science and Engineering
Graded by teacher: Sandro Stucki, Department of Computer Science and Engineering

Bachelor's Thesis 2026
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2026

Scheduling Software for Improving Teaching Assistant Schedule Satisfaction
Kusai Al Malt, Nova Alldén, Anna Blomberg, Mai Uy Vuong, Neo Zanders

Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Teaching assistant scheduling at Chalmers University of Technology and the University of Gothenburg is often managed through manual and unstructured processes, such as spreadsheets and informal communication, leading to inefficiencies, scheduling conflicts, and unfair workload distribution. This bachelor's thesis investigates how an Automated Teaching Assistant Allocation System can improve teaching assistant schedule satisfaction by combining algorithmic optimization with user-defined constraints and preferences. The project focuses on both the technical problem of generating feasible schedules and the user-centered challenge of supporting teaching assistants in expressing availability and preferences in a clear and usable way.

To address this problem, a prototype web-based scheduling system was designed, implemented, and evaluated. The system allows course responsibles to configure courses and sessions, while teaching assistants can specify hard constraints, such as unavailable times, and soft constraints, such as preferred session types and scheduling preferences. Several scheduling algorithms were implemented and compared, including Constraint Programming models using Choco Solver with Large Neighborhood Search, and a hybrid approach combining Google Operations Research-Tools Constraint Programming-Satisfiability with a heuristic greedy algorithm. The algorithms aimed to satisfy all hard constraints while minimizing penalties associated with violated soft constraints.

A survey with 40 teaching assistants, an interview with a course responsible, and usability interviews with four teaching assistants were conducted to inform both system design and evaluation. The results indicate that current scheduling approaches are perceived as time-consuming, inconsistent, and lacking support for fairness and preference satisfaction. User evaluations of the prototype showed that participants found the constraint input process relativelt intuitive and expressed a preference for the system over current scheduling methods.

Benchmark testing demonstrated that Large Neighborhood Search-based approaches improved scheduling quality compared to a naive implementation, particularly for larger scheduling problems. The results suggest that combining constraint-based optimization with a user-centered interface can support both feasible scheduling and improved teaching assistant satisfaction.

Keywords: Teaching Assistant Scheduling, Constraint Programming, Scheduling Optimization, Large Neighborhood Search, CP-SAT, Soft Constraints, Workload Fairness, Automated Scheduling, User-Centered Design, Timetabling

Sammandrag

Schemaläggning av lärarassistenter vid Chalmers tekniska högskola och Göteborgs universitet hanteras ofta genom manuella och ostrukturerade processer, såsom kalkylblad och informell kommunikation, vilket leder till ineffektivitet, schemalagda konflikter och ojämn arbetsbelastning. Detta kandidatarbete undersöker hur ett automatiserat system för tilldelning av lärarassistenter (ATAAS) kan förbättra tillfredsställelsen med schemaläggningen genom att kombinera algoritmisk optimering med användardefinierade begränsningar och preferenser.

Ett prototypbaserat webbaserat schemaläggningssystem har utformats, implementerats och utvärderats. Systemet låter kursansvariga konfigurera kurser och sessioner, medan lärarassistenter kan ange hårda begränsningar, såsom tider de inte kan arbeta, och mjuka begränsningar, såsom föredragna sessionstyper och schemapreferenser. Flera schemaläggningsalgoritmer implementerades och jämfördes, däribland Constraint Programming-modeller med Choco Solver och Large Neighborhood Search, samt en hybridmetod som kombinerar Google OR-Tools CP-SAT med en heuristisk girig algoritm. Algoritmerna syftar till att uppfylla alla hårda begränsningar samtidigt som påföljder för brutna mjuka begränsningar minimeras.

En enkät med 40 lärarassistenter, en intervju med en kursansvarig samt användbarhetsintervjuer med fyra lärarassistenter genomfördes för att informera både systemdesign och utvärdering. Resultaten visar att nuvarande schemaläggningsmetoder upplevs som tidskrävande, inkonsistenta och bristfälliga vad gäller stöd för rättvisa och preferenstillfredsställelse. Användarutvärderingar visade att deltagarna fann inmatningen av begränsningar intuitiv och uppskattade funktioner som återkommande begränsningar, preferensrangordning och automatisk konflikthantering.

Benchmarktester visade att Large Neighborhood Search-baserade metoder konsekvent förbättrade schemaläggningskvaliteten jämfört med en naiv implementation, särskilt för större probleminstanser. Resultaten tyder på att en kombination av begränsningsbaserad optimering och ett användarcentrerat gränssnitt kan stödja både genomförbar schemaläggning och förbättrad tillfredsställelse hos lärarassistenter.

Nyckelord: Schemaläggning av lärarassistenter, Constraint Programming, Schemaläggningsoptimering, Large Neighborhood Search, CP-SAT, Mjuka begränsningar, Rättvisa arbetsbelastning, Automatiserad schemaläggning, Användarcentrerad design, Timetabling

Acknowledgements

We thank Niklas Broberg for his feedback, guidance, and support throughout this bachelor project.

We also thank Sandro Stucki for valuable feedback on the planning report and for insightful comments on the project as a whole.

Furthermore, we would like to thank Niklas and the teaching assistant department for assisting in distributing the survey, as well as all teaching assistants who put their time into participating in the survey.

Finally, we thank the course responsible and the teaching assistants who participated in the interviews, providing valuable insights for this bachelor project.

Kusai Al Malt, Nova Alldén, Anna Blomberg, Mai Uy Vuong, Neo Zanders, Gothenburg, June 2026

Contents

List of Acronyms

The following is a list of acronyms that have been used throughout this thesis.

API	Application Programming Interface
ATAAS	Automated Teaching Assistant Allocation System
CP	Constraint Programming
CR	Course Responsible
CSP	Constraint Satisfaction Problem
CSS	Cascading Style Sheets
LNS	Large Neighborhood Search
MRV	Minimum Remaining Values
NP	Nondeterministic Polynomial-Time
OR	Operations Research
REST	Representational State Transfer
SAT	Satisfiability
TA	Teaching Assistant

List of Figures

List of Tables

1
Introduction

This chapter introduces the background and motivation for the thesis, as well as the challenges related to teaching assistant (TA) scheduling. It also presents the aim, objectives, and delimitations of the project.

1.1 Background

Scheduling problems arise in many real-world contexts, where limited resources must be allocated under a set of constraints. At Chalmers University of Technology and the University of Gothenburg, one such problem is the allocation of TAs to course activities such as laboratory sessions and grading tasks.

TAs play a critical role in the educational structure at Chalmers University of Technology and the University of Gothenburg. They support course responsibles (CR) in delivering course content, for instance by assisting students during scheduled sessions or contributing to assessment-related tasks. Effective scheduling of TAs is therefore essential to ensure both the smooth operation of courses and a fair distribution of workload.

Despite their importance, current TA scheduling practices are often informal and lack a unified structure. In practice, scheduling is carried out using a variety of ad-hoc methods, such as shared spreadsheets, direct messaging, and manual coordination between participants. For example, availability may be communicated through spreadsheets or messaging platforms. Allocations, for example, are often performed manually or on a first-come, first-served basis.

These approaches vary significantly between courses and provide limited support for systematically handling constraints, ensuring fairness, or reducing administrative workload. This is particularly evident in courses with a large number of students and TAs, where manual allocation becomes time-consuming and difficult to manage and may lead to suboptimal or perceived unfair distributions of work. This lack of structure highlights the need for more systematic and automated approaches to TA scheduling in order to improve schedule satisfaction.

From an academic perspective, the problem belongs to a broader class of scheduling and resource allocation problems [1], which are known to be computationally complex and require methods capable of handling both hard constraints, such as availability, and soft constraints, such as individual preferences.

The primary users of the system are TAs, who are responsible for providing their availability and preferences. In addition, CRs are also users of the system. They interact with the system by creating courses, providing scheduling information,

inviting TAs, and initiating the allocation process.

1.2 Aim of Study

The aim of this bachelor's thesis is to design, implement, and evaluate an Automated Teaching Assistant Allocation System (ATAAS) that supports the scheduling of TAs in university courses. The main focus of the study is not only to create feasible schedules, but also to investigate how the system can support TA user satisfaction through both the input of constraints and the resulting allocation.

The project addresses the practical problem of TA scheduling, where CRs must configure courses and assign TAs to different course sessions while considering availability, workload limits, session requirements, and individual preferences. In the system, CRs are able to create and configure courses by adding sessions, such as labs, exercises, help sessions, or grading work. Each session contains information that affects the scheduling problem, including the time of the session and how many TAs are required. At the same time, the system should allow TAs to express relevant constraints in a clear and usable way. These constraints include hard constraints, such as times when a TA cannot work, and softer preferences, such as preferred session types or less preferred time periods.

The intended outcome of the project is a prototype system that can collect course and session information from CRs, collect TA constraints and preferences, use this information in an allocation algorithm, and produce schedules that are both valid and reasonable from the users' perspective. A valid schedule must satisfy hard requirements such as avoiding double bookings, respecting TA availability, meeting TA workload limits, and assigning the required number of TAs to each session. A reasonable schedule should also try to satisfy soft constraints when possible, for example by considering individual preferences and distributing work in a fair way.

The thesis is therefore guided by the following overall aim: to investigate how an ATAAS can be developed to support TA satisfaction with both the constraint input process and the generated scheduling outcome, while also supporting CRs in defining the course structure that the allocation is based on. This includes studying how courses, sessions, TA constraints, and preferences can be represented in the system, how this information can be used by the scheduling algorithm, and how the final schedules can be evaluated in terms of feasibility, preference satisfaction, fairness, and practical usefulness.

The scope of the thesis is limited to the development and evaluation of a prototype system for TA allocation. The web-based interface is included because it is necessary for collecting realistic course information, collecting TA input, and presenting scheduling results. However, the thesis does not focus on interaction design in detail. Instead, the main contribution is the combination of software development, constraint handling, and scheduling logic with a user-centered focus on TA satisfaction. Further details about delimitations of the thesis are provided in Section 1.4.

1.3 Objectives

TAs play a crucial role in ensuring smooth operation of university courses. However, current approaches to TA scheduling often lack structure, standardization, and systematic support, which leads to inefficiencies in planning, increased administrative effort, and potential inconsistencies in workload distribution.

The primary objective of this project is to design and evaluate a scheduling approach that improves efficiency and reliability of TA allocation. The proposed solution aims to handle essential constraints such as availability and maximum working hours, while also incorporating individual preferences and promoting fairness in workload distribution. To achieve this, the project addresses the following key challenges:

1. Information regarding TA availability and scheduling is often distributed across multiple tools and formats, requiring manual updates and increasing the risk of inconsistencies and errors.
2. Existing methods do not systematically ensure fair workload distribution or prevent scheduling conflicts, such as double bookings or exceeding allocated working hours.
3. The creation and maintenance of schedules require significant manual effort, particularly when adapting schedules across different course instances.
4. TAs often encounter different scheduling approaches depending on the course, which can lead to confusion and reduced efficiency.

Based on these challenges, the objective is not only to improve scheduling efficiency but also develop a solution that balances automation with flexibility, ensuring that it can be adapted to different course structures and user needs.

1.4 Delimitations

To ensure the feasibility of our product within the given time frame, several delimitations are defined.

The developed system focuses primarily on the scheduling algorithm for solving a constrained version of the general scheduling problem [2]. As a result, several supporting system features are either excluded or simplified. These include email integration, Canvas integration, and server hosting. While these features could improve usability and deployment in a real-world setting, they are considered outside the scope of this project, as they are not directly related to the research question.

The intended users of the system are employees at the Department of Computer Science and Engineering, a joint department between Chalmers University of Technology and the University of Gothenburg. The project is therefore limited to this context in order to ensure alignment with relevant stakeholder needs and to maintain a clear and well-defined application domain.

Additionally, the project focused on the technical and organizational aspects of TA scheduling rather than broader institutional or administrative policies. Although fairness was considered through workload balancing and preference handling, the system does not attempt to evaluate subjective notions of fairness beyond the defined

scheduling constraints and optimization criteria.

The project also considered ethical aspects related to privacy and transparency. Since scheduling systems may involve personal availability information, the system was designed to minimize the collection and storage of sensitive personal data. For example, the system focuses on whether a TA is unavailable rather than requiring detailed reasons for absence. Furthermore, the system was designed to support transparent scheduling decisions in order to reduce the risk of perceived unfairness or bias in the allocation process.

2
Theory

This chapter presents the theoretical background to the thesis, including Constraint-Based Scheduling, Greedy algorithms, Large Neighborhood Search (LNS), and previous work related to scheduling and resource allocation problems.

2.1 Constraint-Based Scheduling

Constraint Based Scheduling is a discipline that applies Constraint Programming (CP) to solve scheduling problems, problems where resources such as people, rooms, or time slots must be assigned under a set of rules. A CP system works through a sequence of steps to solve a scheduling problem. The first step is to define the problem. The problem for this system is to allocate TAs to course sessions and can be described as a combinatorial optimization problem. An optimization problem is a problem where the goal is to find the best possible solution based on some value.

For CP to be able to understand what we are trying to optimize, the problem must be expressed as a Constraint Satisfaction Problem (CSP), which is characterized with three elements: a set of variables, a set of possible values of each variable, and a set of constraints between the variables. For example, the variables are TAs and sessions, and the values they represent will be possible pairings between them.

To be able to pair them, a set of constraints will be applied to determine which pairings are valid. A TA and session pair must respect the following: a TA cannot be assigned to a session if the TA is unavailable, each session must be staffed by a minimum and maximum number of TAs, and no TA may be assigned to two sessions that overlap in time. A solution to a CSP is an assignment of a value to each variable such that all constraints are satisfied [3]. In this case, the solution is a schedule where each TA is paired with a session without breaking any constraints.

Since pairing TAs and sessions is a combinatorial problem, the time required to find a valid schedule grows exponentially with the number of TAs, sessions, and constraints. To decrease the time it takes to find a schedule, the constraints can be used to determine the availability of some TAs during certain sessions before searching for a schedule. This is the second step in the CP system known as constraint propagation, which uses the given constraints to deduce values, derive new constraints, and detect inconsistencies before performing searches. For example, a TA is unavailable on Monday mornings, and a lab session only takes place on Monday morning. By performing propagation, the values that cannot lead to a valid solution will be eliminated, directly reducing the search space. The search space in this case refers to the total number of TA and session combinations that the system

needs to explore [3].

With the search space reduced by propagation, the next step is to search for a valid solution using heuristic search. A heuristic search is a strategy that guides the system by deciding which TA-session pairing to attempt first, rather than trying every possible combination. The order in which pairings are attempted matters because the system may still reach a dead end where a chosen decision turns out to cause a conflict. When this happens, the system will backtrack by undoing the last decision and try a different option. For example, when the system starts with sessions that have many available TAs, it determines the assignments faster. However, the system may reach a session that only has one or two available TAs, since the others have already been assigned to earlier sessions. The system then has to undo many of those earlier decisions and try a new path, which is costly. A common strategy is to prioritize the most constrained variables first, since these have the fewest valid options remaining and are therefore harder to assign later, avoiding unnecessary backtracking.

An important property of CP that makes it practical for scheduling problems is that each constraint works on its own. For example, a constraint that blocks a TA from Monday morning does not need to know about the constraint that limits total working hours of the same TA. This means that new constraints can be added to the model without modifying the existing ones. This property makes CP flexible enough to handle scheduling problems where the rules can change or grow over time [3].

2.1.1 Constraint Propagation and Search

Constraint propagation plays a central role in a CP system. When the system assigns a value to a variable or updates the range of values it can take, every constraint involving that variable is reevaluated. If propagation reveals that some value in another variable's domain cannot lead to any valid solution, the value is then removed with a step called domain reduction [3].

A problem that is NP-complete does not have a standard method that can solve the problem in a reasonable time, as the problem gets larger the time to find a solution will significantly increase with it. Since the general CSP is nondeterministic polynomial time-complete (NP)[4], constraint propagation is usually incomplete, meaning that some but not all consequences of the constraints are deduced and not all inconsistencies can be detected [3]. The system must therefore combine propagation with search to decide if any solution exists. The search is most commonly performed using the tree search algorithm, where the system moves forward by selecting a variable and assigning it a value, then propagates the consequences of that decision. If no valid slots remain assignable to a TA, meaning no feasible assignment exists given the current assignment, the system backtracks. The common strategy is depth first chronological backtracking where the last decision is undone and a different value is tried[3].

2.2 Greedy Algorithm and Dispatching Rules

Dispatching rules are a set of heuristic methods for solving scheduling problems. Rather than searching through every possible combination of assignments, a dispatching rule builds the schedule one decision at a time. Dispatching rules work by assigning resources step by step, choosing the next task based on priority ranking. For example, the resources are TAs and sessions, and each step is a single decision on which TA and session will be assigned together based on the priority criteria, such as how much the assignment conflicts with TA preferred off time, their total assigned workload, and also by TA session type preference. The best candidate TA will then be assigned for that session.

Dispatching rules can be classified into static and dynamic rules; static rules are not time-dependent, they use fixed criteria that do not change as a schedule is built [5]. For example, a criterion such as a TA always preferring a specific session will not change after 10 assignments. Dynamic rules are time dependent and can change after each assignment based on the current information [5]. For example, at the beginning of the scheduling process, every TA has zero assigned sessions. After a TA is assigned to a session, their total working time increases, which changes their priority for following assignment decisions.

In practice, a single dispatching criterion will not be sufficient to produce a schedule that abides by all the hard constraints while minimizing soft-constraint violations. For example, two TAs might have the same soft-constraint penalty for a given session, so the algorithm needs a second criterion to break the tie. This is addressed through a composite dispatching rule, which combines multiple criteria in sequence, each breaking the ties left by the previous one [5]. A soft-constraint penalty is a score reflecting how much an assignment conflicts with a TA's soft constraints, a score of zero means no conflict, and a higher score indicates a stronger preference to avoid that assignment. For example, eligible TAs will be ranked by how much the assignment conflicts with their preferred-off windows. If two TAs have the same soft-constraint penalty, the tie is then broken by how many other sessions those two TAs can still cover. If that is also equal, the session type preference is then used to break the tie, and if that is still equal, the total assigned workload is the final tiebreaker.

A greedy algorithm is a framework that uses dispatching rules to make each decision, and is an example of what Pinedo [5] classifies as a constructive heuristic. It starts with no assignments and builds a schedule one decision at a time, where the dispatching rules define the priority logic used to make each decision [5]. A strategy widely used in greedy scheduling is addressing the most constrained decision first. In the context of constraint satisfaction, this is known as the Minimum Remaining Values (MRV) heuristic, which selects the variable with the fewest remaining valid assignments [3]. In our scheduling, the sessions that are hardest to staff are addressed before those with more flexibility, reducing the risk of impossible assignments.

A greedy algorithm based on dispatching rules will be useful as a second phase in a hybrid scheduling algorithm, since it operates on a feasible solution produced in the first phase and improves it by incorporating soft constraints without significantly increasing computational complexity [5].

2.3 Large Neighborhood Search

LNS is a method for solving optimization problems. LNS is often used for problems such as scheduling, routing, and assigning resources to different tasks.

The method starts with a solution that already works. After that, it tries to improve the solution step by step. It does this by selecting one part of the solution and allowing that part to change. The rest of the solution stays the same. The selected part is then solved again to see if a better solution can be found. Pisinger and Røpke describe LNS as a method where a solution is improved by repeatedly destroying and repairing parts of it [6]. They also describe it as part of Very Large-Scale Neighborhood Search, since the neighborhoods are too large to search through completely by hand. In this thesis, a solution is a complete TA schedule, where TAs are assigned to different course sessions. A neighborhood is a set of schedules that are similar to the current schedule. These schedules are created by allowing some selected TA assignments to change, while the rest of the schedule stays fixed. For example, one neighborhood could consist of all schedules where the assignments for one TA are changed. Another neighborhood could consist of all schedules where the assignments for one week or one session type are changed. The rest of the assignments are kept the same. The algorithm then searches inside this neighborhood to find a better schedule, for example a schedule with a lower penalty or fewer violated preferences.

LNS is different from normal local search. A normal local search method usually only makes small changes. For example, it might move one assignment or swap two assignments. LNS can instead change a larger part of the solution at the same time. This can be useful because small changes are not always enough to make a schedule better.

Sometimes a solution reaches a point where small changes do not improve it anymore. This is called a local optimum. The solution may look good compared to nearby solutions, but there may still be a better solution somewhere else. By allowing larger changes, LNS has a better chance of escaping this kind of situation.

The LNS process is usually described with two steps: destroy and repair.

In the destroy step, the algorithm chooses part of the current solution and relaxes it. This means that some decisions are allowed to change. For example, some TA-to-session assignments may be opened again. The rest of the schedule stays fixed.

In the repair step, the solver fills in the relaxed part again. It gives new values to the variables that were allowed to change. The new solution must still follow all hard constraints. If possible, it should also improve the objective value. In this thesis, the objective value represents the quality of a TA schedule. Since hard constraints must always be satisfied, the objective value is mainly based on soft constraints, such as TA time preferences, session type preferences, and preferences for a compact or spread-out schedule. Violating these preferences adds a penalty to the schedule. Therefore, a lower objective value means a better schedule.

This means that the neighborhood of a solution is made from all the solutions that can be reached by relaxing one part of the current solution and then repairing it [6].

The destroy step can also include some randomness. This means that the algo-

rithm does not always relax the same variables in each iteration. Instead, it can choose different parts of the solution to reconsider. This is useful because it helps the search explore more than one part of the solution space. It also reduces the risk of the algorithm getting stuck trying to improve the same part of the schedule repeatedly. A key decision in LNS is how large the relaxed part of the solution should be. If only a few variables are relaxed, the method becomes similar to ordinary local search. In that case, each change is small, which can make it difficult to move away from a local optimum. If too many variables are relaxed, the repair step becomes much harder. In the worst case, it can become almost the same as solving the full problem again. This can make each iteration slow.

Because of this, LNS needs a balance. The algorithm should relax enough variables to make meaningful improvements possible. At the same time, it should keep enough of the current solution fixed so that the repair step remains manageable. This balance is often described using the terms intensification and diversification. Intensification means focusing the search around a solution that is already good, in order to improve it further. Diversification means allowing the search to move to other parts of the solution space. A good LNS method needs both. It should use the current good solution, but it should also sometimes make larger changes so that better solutions are not missed.

LNS can also be used together with CP. Choco Solver, which is an open-source Java library for CP, provides built-in support for LNS [7]. According to the Choco documentation, LNS works by repeatedly relaxing part of a current solution and then using the CP solver to repair that part. In this thesis, a solution is a complete TA schedule. The variables describe whether a TA is assigned to a specific course session or not. When LNS is used, some of these variables are relaxed, meaning that they are allowed to change. For example, the algorithm may relax the assignments for one week, one TA, or one type of session. The relaxed variables are reset to their original domains. In this scheduling problem, this means that the solver can again decide whether a TA should or should not be assigned to the selected sessions. The remaining variables are fixed to the values they had in the current solution. This means that the rest of the schedule stays the same. The solver then searches for new values for the relaxed variables while still satisfying all hard constraints, such as TA availability, session staffing requirements, and working hour limits. If possible, it also tries to reduce the total penalty from soft-constraint violations. This continues until the search stops, either because an optimal solution has been proven within the neighborhood or because a search limit has been reached [7].

Choco also supports different kinds of neighborhoods. A random neighborhood can relax a random group of variables. Choco can also combine several neighborhoods. In a sequential neighborhood, different neighborhoods are used in a fixed order. In an adaptive neighborhood, the solver gives weights to the different neighborhoods. If one neighborhood often helps the solver find better solutions, its weight can increase. This makes that neighborhood more likely to be chosen again later [7]. In this way, the search can adapt based on which neighborhoods seem useful for the current problem.

In this thesis, LNS is useful because the TA scheduling problem contains both hard and soft constraints. Hard constraints must always be satisfied. If a hard

constraint is broken, the schedule is not valid. Examples of hard constraints are that a TA cannot be assigned to a session when they are unavailable, and that each session must have the required number of TAs.

Soft constraints are different. They describe preferences rather than strict rules. For example, a TA may prefer some session types over others or prefer not to work at certain times. Breaking a soft constraint does not make the schedule invalid. However, it makes the schedule worse because it increases the penalty value.

A complete schedule can contain many assignment variables. Changing only one assignment at a time may not be enough to lower the total penalty. This is because several assignments can depend on each other. Moving one TA from one session may only be useful if other assignments are also changed at the same time. LNS is useful here because it allows the solver to reconsider a larger part of the schedule, while still keeping the rest fixed.

For example, one neighborhood could relax all assignments for one TA. This would allow the solver to rebuild that TA's schedule. Another neighborhood could relax all assignments during one week. A random neighborhood could instead relax a selected group of assignment variables without following a fixed pattern. After this, Choco can repair the partial schedule. The relaxed variables are assigned new values, while the fixed variables keep their old values. During this step, the hard constraints still have to be satisfied. At the same time, the solver tries to find a solution with a lower penalty from the soft constraints.

LNS can therefore be seen as a practical middle ground. It does not restart the whole scheduling problem from the beginning in every iteration. It also does not only make very small changes. Instead, it repeatedly opens selected parts of the schedule and tries to improve them.

However, LNS does not always guarantee that the best solution found is the global optimum. This is only guaranteed if the solver can prove that no better solution exists before the search stops. In practice, the result depends on the chosen neighborhoods, the size of the relaxed part, and the stopping limit. Because of this, LNS should be understood as a heuristic optimization method. It can often improve difficult schedules, but it cannot always guarantee the best possible schedule within a limited amount of time.

2.4 Previous Work

Kaewchanid and Wangmaeteeku [8] investigated the problem of assigning TAs to course sessions using a constraint-based approach. In addition, the cluster method and the earliest deadline first technique are used to minimize the search space by grouping the resources and prioritizing scheduling the shortest deadline task. Their work showed that TA timetabling shares many structural characteristics with general employee scheduling problems, where the goal is to produce a valid assignment that satisfies a set of hard and soft constraints, such as TA availability and session coverage requirements, while minimizing violations of soft constraints, such as scheduling preferences.

Their study is closely related to the problem addressed in this thesis. Both problems involved assigning a set of TAs without violating any hard constraints

while accounting for soft constraints. The key distinction between this thesis and their prior study lies in scope and perspective. Kaewchanid and Wangmaeteeku's work focused primarily on an algorithmic perspective. The aim of this thesis extends beyond only ensuring feasible scheduling to also considering TA satisfaction with both the constraint input process and the resulting schedule. This provides an additional user-centered dimension to this thesis compared to prior work.

Kletzander and Musliu [2] investigated the general employee scheduling problem, which covers a broad range of workforce allocation challenges across industries such as healthcare, transportation, manufacturing, or call centers. They proposed a framework that allows various heuristic algorithms to be applied to a general employee scheduling problem through a unified constraint handling approach and reusable implementations for different algorithms. The study is related to the problem addressed in this thesis, as both involve assigning workers to tasks with different constraints while accounting for employee satisfaction. Kletzander and Musliu focused on developing a general framework for solving scheduling problems that is applicable across many work fields. This work provides a relevant background for the scheduling problem addressed in this thesis, as TA allocation is also classed as an employee scheduling problem that they studied.

Baptiste, Le Pape, and Nuijten [3] presented a framework for applying CP to scheduling problems. Their work covers concepts such as constraint propagation, domain reduction, and heuristic search strategies used to solve scheduling problems where resources must be assigned under a set of hard constraints.

Cambazard Demazeau, Jussien and David [9] investigated school timetabling problem using CP with explanation based techniques to handle the dynamic and over-constrained nature of such problems. Their approach combined CP for satisfying hard constraints with local search for optimization of soft constraints.

3
Methods

This chapter presents the methods used in the thesis, including the survey study, interviews with a CR and TAs, generation of test data, implementation and benchmarking of the algorithms, development of the user interface, and methods for data processing and validation.

3.1 Survey Study

To better understand current challenges in TAs scheduling and to identify desirable features for a new system, a survey was conducted. The survey was designed to collect both quantitative and qualitative data regarding TA's experiences, constraints, and preferences related to scheduling.

The questionnaire was designed using Google Forms (see Appendix A.3). It included a combination of multiple-choice and open-ended questions to capture both structured responses and more detailed feedback. The survey focused on identifying key factors affecting TA availability, preferences regarding scheduling, perceptions of fairness, and limitations of current scheduling methods.

The survey was distributed to 231 TAs at Chalmers University of Technology and the University of Gothenburg, and a total of 40 responses were collected. Participation was voluntary and responses were collected anonymously.

The questionnaire consisted of several sections. The first section gathered background information, including the respondent's role, prior experience as a TA, and what workload percentage they had.

Subsequent sections focused on availability constraints, where respondents identified factors affecting their availability, such as course schedules, exams, external work, and personal commitments. Participants were also asked how they currently communicate availability to CR and how precise their availability information typically is.

Further sections addressed preferences and fairness. Respondents indicated how important it is that their scheduling preferences are respected and selected relevant preferences, such as consistent time slots, avoiding certain times of the day, or working with specific colleagues. The survey also explored perceptions of fairness in scheduling, including aspects such as equal workload distribution and consideration of individual constraints.

In addition, participants evaluated current scheduling methods and identified common issues, such as reliance on spreadsheets, manual processes, and risk of scheduling conflicts. Finally, respondents were asked about desired features in a

potential scheduling system, including visual availability input, automatic conflict prevention, and fair allocation mechanisms. Open-ended questions were included to allow participants to provide additional feedback and suggestions. The collected data was used to inform the design and requirements of the proposed scheduling system.

3.2 Interview with Course Responsible

To complement the survey data and gain deeper insights into current scheduling practices, a semi-structured interview was conducted with a CR involved in TA scheduling. The purpose of the interview was to better understand the practical challenges of scheduling, the tools currently in use, and the requirements for a potential automated system.

The interview was conducted prior to the design and implementation phase of the system in order to inform both the system requirements and overall design decisions. A semi-structured interview format was chosen to ensure a consistent set of core questions while allowing flexibility for follow-up questions and discussion. The interview was conducted in Swedish and the questions presented in Appendix A.3 are translated into English.

The questions covered several key areas related to TA scheduling. These included the current scheduling process, tools used (e.g., spreadsheets), time consumption, and common issues encountered. Particular focus was placed on understanding how availability constraints and maximum working hours are handled, as well as how fairness in workload distribution is ensured. Additionally, the interview explored the extent of manual work involved in creating and maintaining schedules and whether issues such as double bookings, late changes, or miscommunication occur in practice.

In the later part of the interview, a preliminary sketch of the proposed system was presented to the participant. This allowed for feedback on the system design, including missing functionality, potential improvements, and overall usability. The participant was also asked whether they would consider using such system in practice and to motivate their response.

3.3 Interview with Teaching Assistants

A qualitative user study was conducted to evaluate the usability of the system and user satisfaction. The interviews took place after the application was developed to a certain degree (See section 3.9). The study used a semi-structured interview format, in which participants performed predefined tasks and were asked both task-specific and open-ended questions (See appendix A.4). The tasks performed during the interviews were based on the synthetic dataset described in Section 3.4, ensuring a consistent and controlled test scenario across participants.

The participants consisted of four TAs with prior experience in course-related TA activities, each interviewed separately, where each interview lasted approximately 15 to 30 minutes. The selection of participants was based on their relevance as end users of the system.

The purpose of the study was to evaluate how easily users could express their constraints, how intuitive the system interface was, and how satisfied participants would be with the allocation produced by the scheduling algorithm.

Data was collected through participant responses and observational notes taken during the interviews.

The interview was designed as a task-based evaluation, where participants interacted with the system through a sequence of realistic use-case scenarios. The tasks were constructed to reflect typical actions a TA would perform when using the system, such as specifying constraints after joining a course invitation.

The design was inspired by exploratory testing principles [10], encouraging participants to actively interact with the system while reflecting on their experience.

After completion of each task, participants were asked questions related to their experience, such as perceived difficulty, intuitiveness, and clarity. The tasks included logging into the system, accepting a course invitation, entering hard and soft constraints, and evaluating fairness of allocation. These tasks were selected to cover the core functionality of the application and assess how effectively users could interact with the system.

Following the tasks, the participants were allowed to freely explore the system and provide additional feedback. The interview concluded with open-ended questions aimed at capturing overall impressions, perceived strengths and weaknesses, and suggestions for improvement.

The data collected (See appendix A.5 for summary) from the interviews consisted of qualitative responses and observational notes. The analysis of the collected data focused on aspects such as ease of use, clarity of constraint input, and perceived fairness of schedule allocation. These insights were used to evaluate the system and to identify areas for improvement.

Since the scheduling algorithm relies on user-provided constraints, the clarity and correctness of user input were considered particularly important, as poorly defined constraints may lead to suboptimal scheduling outcomes.

3.4 Test Data Generation

To support the development of ATAAS, synthetic test data was generated. The generated data was also used to create consistent test scenarios for the TA interviews.

The datasets were synthetically generated and do not include any real TAs, CRs, or courses from Chalmers University of Technology or the University of Gothenburg. Instead, the data was manually constructed based on characteristics observed in previous courses and insights gathered from the conducted survey. This approach allowed for the creation of realistic yet non-identifiable scheduling scenarios suitable for system development.

Synthetic data was chosen for several reasons. First, it ensures that no personal or sensitive information is included, addressing privacy concerns. Second, access to complete real-world scheduling data is limited and may violate privacy concerns. Third, synthetic data allows for controlled and reproducible scenarios, which simplifies testing and debugging during development.

The data was informed by characteristics observed in previous course instances,

including the number of TAs required, the number and type of sessions, the number of students, and the distribution of workload among TAs. Based on these observations, assumptions were made to construct a plausible scheduling scenario. While the data does not correspond to any specific real course, it is designed to resemble realistic course structures. To avoid any potential linkage to real individuals, all personal identifiers, such as TA names, were generated using random name generators. The dataset was then manually constructed to reflect realistic, but entirely synthetic, scheduling conditions.

The dataset represents a single course instance with a predefined number of TAs, sessions, and students. Multiple session types are included, such as laboratory sessions and grading sessions, distributed over a fixed time period. Each TA is associated with a set of constraints, including both hard and soft constraints, as well as parameters such as maximum workload.

Constraints and preferences were generated based on insights from the survey. The results indicated that the primary source of unavailability for TAs is their own course schedules. To reflect this, each TA was assigned unavailable time slots corresponding to generated course schedules, following the scheduling block structure used at Chalmers University of Technology. Additional constraints were derived from common responses in the survey, such as exam preparation periods, personal commitments and other obligations. These were distributed across TAs to introduce variation. The assignment of constraints was partially randomized to avoid unintended patterns while maintaining realistic distributions.

The generated dataset has several limitations. It is based on a single scenario with a limited number of TAs and sessions, which restricts its representativeness. In addition, user behavior is simulated. All constraints are assumed to be predefined and static, whereas real-world constraints may change over time. Furthermore, although some TAs are assigned multiple concurrent courses, the dataset does not fully capture the complexity of real-world scheduling. In practice, TAs may have more dynamic and overlapping commitments, as well as varying priorities between obligations. These aspects are only partially represented in the synthetic data.

As a result, while the dataset provides a useful basis for development and testing, it does not fully reflect the complexity of real-world scheduling scenarios.

3.5 Algorithm Implementation

The development of a scheduling algorithm was a core part of the initial goals of the thesis. In order to create an algorithm that could satisfy all hard constraints while also trying to maximize TA satisfaction, the development process was broken down into two steps. Firstly, a common data-model for input and output data was developed. Secondly, multiple implementations of the algorithm satisfying an algorithm interface defining given inputs and expected outputs were implemented in parallel (see Sections 3.6 and 3.7).

These implementations had the same main goal, which was to create a valid schedule that followed all hard constraints and minimized the total penalty from violated soft constraints. Hard constraints were treated as rules that could not be broken, for example when a TA was unavailable or when a session required a certain

number of TAs. Soft constraints were instead handled with penalties. This meant that the schedule could still be valid even if some preferences were not fulfilled, but the quality of the schedule would become worse.

Developing several implementations in parallel made it possible to compare different approaches during the project. Since all implementations were based on the same data model, they could be tested using the same scheduling examples. This approach was chosen in order to simplify testing and comparing of the different implementations with regard to feasibility, soft-constraint satisfaction as well as determining which one would be most suitable for the final system. Further benefits which contributed to this approach are the fact that it made the development process more flexible and suited to the feature branch development style of the thesis group while also ensuring the project was not dependent on a single algorithm implementation from the beginning.

3.5.1 Data Structures and Models

To make the scheduling problem independent from a specific algorithm implementation, a shared input data model was defined. This model represented the information needed by the algorithms, such as sessions, TAs, hard constraints and soft constraints. A simplified overview of the model is shown in Figure 3.1.

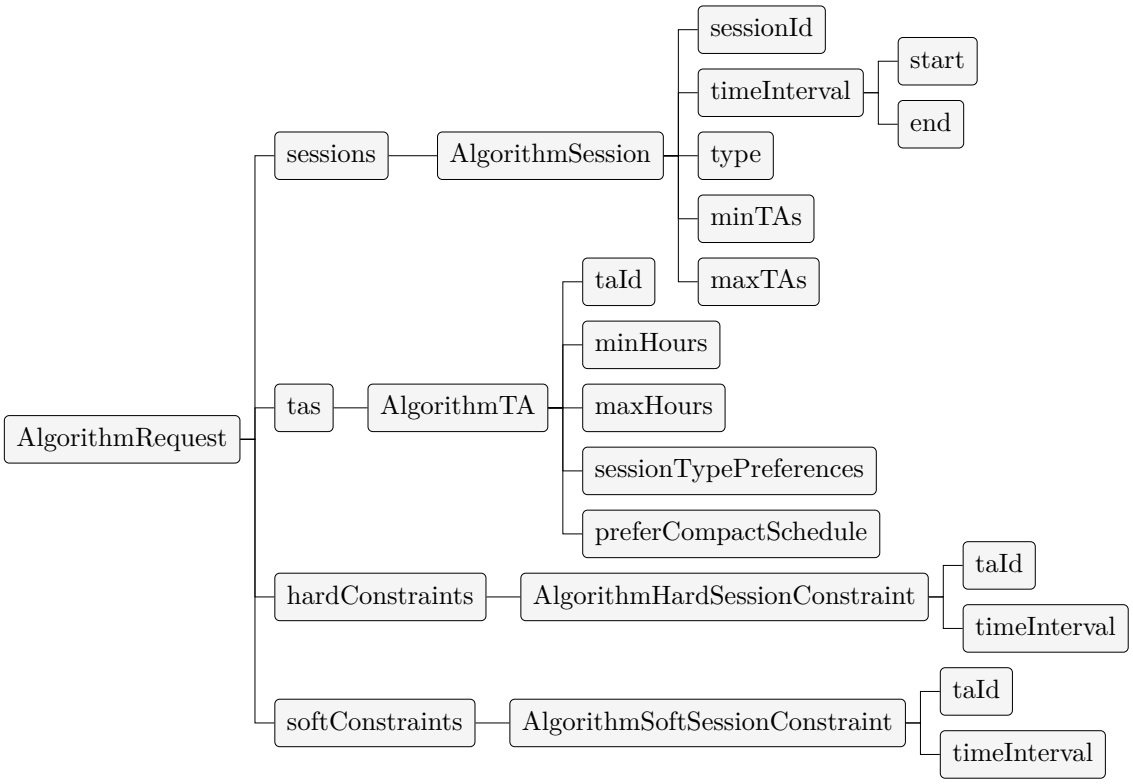


Figure 3.1: Simplified overview of the algorithm input data model.

The output of the scheduling algorithm was also represented using a shared data model. This output contained the generated allocations, the total penalty of the

schedule, and information about whether a feasible and proven optimal solution had been found. A simplified overview of this structure is shown in Figure 3.2.

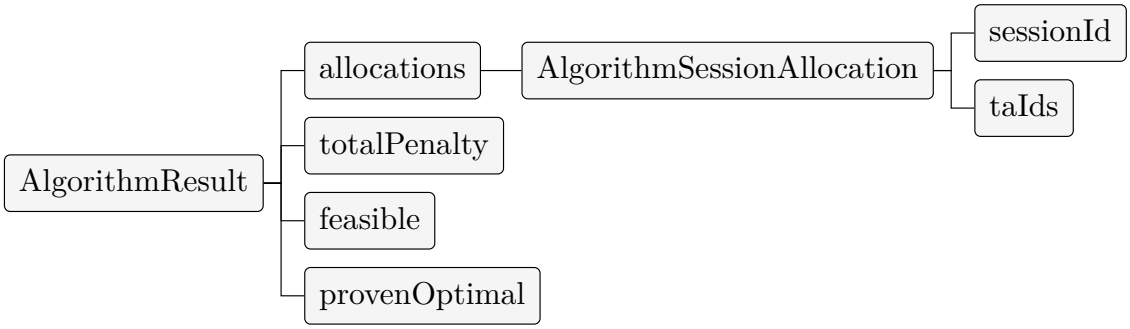


Figure 3.2: Simplified overview of the algorithm output data model.

3.6 Choco Solver

One implementation implemented the scheduling problem as a constraint optimization problem using Choco Solver. In the model, each possible assignment between a TA and a course session is represented by a Boolean decision variable. If the variable is true, the TA is assigned to that session. If it is false, the TA is not assigned.

The model contains several hard constraints. First, each session must be assigned at least its minimum number of TAs and at most its maximum number of TAs. Second, each TA must be assigned a total amount of work within their minimum and maximum hour limits. Third, a TA cannot be assigned to two sessions that overlap in time. Finally, hard availability constraints are posted by forcing assignment variables to false when a TA is unavailable during a session.

Soft constraints are handled by creating a penalty value. If a TA is assigned to a session that overlaps with one of their soft constraints, a penalty of 100 is added to the value. Session type preferences are also included. A session type ranked lower by a TA gets a larger penalty, using 10 penalty points per ranking step.

The compact or spread-out preference is also handled with penalties, but with a smaller weight of 1 per hour. If a TA wants a compact schedule, the solver adds a penalty when there are large time gaps between sessions. If a TA wants a spread-out schedule, the solver instead adds a penalty when sessions are too close together compared to the average time between sessions.

The solver then tries to minimize the total penalty while still satisfying all hard constraints. The weightings aim to make sure soft session constraints which relate to time slots are prioritized above all else, whereas the compact/spread-out preference acts as a tie-breaker in the model. The compact/spread-out tie-breaker in particular should improve performance by reducing the number of identical solutions which have to be considered before any algorithm implementation can assert that a given solution is optimal.

The naive implementation builds the full Choco model and then lets the solver search for a solution directly. The same hard constraints and penalty model are used as in the other implementations. The solver in this instance searches for a solution that minimizes the total penalty through classic constraint programming

techniques. A timeout and plateau limit are also used so that the search does not continue indefinitely if no better solution is found.

This implementation is simple because it does not add any special search strategy beyond the base Choco model. The advantage is that it is easy to understand and gives a useful baseline for comparison. However, as the problem grows, the solver may spend more time searching a very large solution space. Because of this, the naive implementation is mainly useful as a reference point when evaluating whether the LNS-based implementations improve the result.

The random neighborhood LNS implementation uses the same Choco model as the naive version, but adds LNS. Instead of only relying on the standard solver search, LNS repeatedly keeps parts of a previous solution fixed and relaxes other parts. The relaxed variables can then be re-optimized by the solver.

In this implementation, the neighborhood is chosen randomly from all decision variables. This means that different TA-session assignments can be relaxed in each search step. The purpose of this is to let the solver explore new parts of the solution space without restarting the full problem from the beginning. This can help the solver escape local optima where no small change improves the solution.

The random LNS implementation is useful as a comparison point because it uses LNS without much domain-specific knowledge. It shows how much improvement can be achieved by applying a general LNS strategy to the scheduling problem.

The custom neighborhood LNS implementation also uses LNS, but the neighborhoods are based more directly on the structure of the TA scheduling problem. In addition to a random neighborhood, it uses custom neighborhoods that relax variables related to selected TAs and selected weeks.

The TA-based neighborhood allows the solver to reconsider assignments for a small number of TAs. This is useful because improving one TA's schedule often requires moving several of that TA's assignments at once. The week-based neighborhood allows the solver to reconsider the schedule for a selected week. This is useful because scheduling conflicts and soft-constraint violations often occur within the same week.

The implementation combines these neighborhoods using an adaptive neighborhood strategy. This means that the solver can choose between several neighborhoods during the search. The goal is to keep the flexibility of random LNS while also using information about the structure of the scheduling problem. This makes the custom LNS implementation more tailored to the thesis problem than the random LNS version.

3.7 CP-SAT Solver and Heuristic Greedy Algorithm

The approach of this scheduling algorithm is a two phase hybrid algorithm. The first phase uses the CP-SAT solver provided by Google Operations Research (OR)-Tools to handle all hard constraints, systematically searching for feasible assignments and guaranteeing that no hard constraints are violated. However, optimizing soft constraints such as preferences and workload fairness within the CP-SAT solver alone

would significantly increase the complexity of the model. The greedy algorithm will address this by taking the complete assignment from CP-SAT as its starting point and filling any slots that have not yet reached their maximum staffing requirement, using soft-constraint optimization, without modifying the assignments already made by CP-SAT. If no feasible solution is found in first phase, the second phase will still execute using an empty assignment as its starting point.

There are four hard constraints in the model. A TA can only be assigned to a session if they are available at the time slots. Each TA's total assigned hours during the study period also have to stay within their stated minimum and maximum limits. Every session needs to have enough TAs assigned to meet its minimum and maximum staffing requirements. Finally, a TA cannot be scheduled for two sessions that overlap in time.

The CP-SAT solver treats each constraint as a binary assignment, where the decision for each TA and session paired is either assigned or not assigned.

The first is availability. For each TA and each session, if the TA is not available in the session's time slot, the corresponding binary variable is set to zero, meaning the TA will not be assigned for that session time slot.

The second constraint is about the workload limit for TA. For every TA, the total assigned workload is calculated by summing their assigned session's duration in hours. This total must then remain within the TA's minimum and maximum working hours for the study period.

The third constraint is to ensure that each session is staffed within the minimum and maximum requirements. For any given session, the total number of TAs assigned to it is computed as the sum of all binary variables across every TA for that session, and must fall within the session's minimum and maximum requirements.

The fourth is to prevent double booking for sessions that overlap. For every pair of sessions where the time overlaps, a TA cannot be assigned to both. Therefore, the sum of TA assignment variables must be at most one.

Once the model has been solved, the solver status is checked. If a feasible solution is found, the assignment is inserted into a map that links each session ID and the TAs assigned to it. If no feasible solution is found, the assignment map remains empty. The result is then returned as output that contains one scheduling result per session instance, along with a flag indicating whether all sessions have met their minimum staffing.

Once the CP-SAT phase is complete and its assignments are passed to the greedy phase, a candidate map is created and recomputed after each assignment to reflect the current state of eligibility. For any session that has not yet reached its maximum number of TAs, eligible candidates are those who are available, have not yet reached maximum working capacity, and have not already been assigned to that session.

All sessions are first grouped by overlapping time slots, so those competing for the same TAs are handled together. The groups are then sorted by size, with the largest groups, meaning those with the most competing sessions, handled first. Within each group, the sessions are ordered by how far they are from meeting their minimum staffing level, with the most understaffed ones assigned first. This is an MRV approach, where the most constrained cases are addressed early to reduce the risk of infeasible assignments later on.

TAs are ranked in a specific order for any session. Firstly, by their soft-constraint penalty, which reflect how much the assignment conflicts with their stated time preferences, where a lower penalty is preferred. Secondly, the constraint score, which counts how many other unfilled sessions the TA could still cover. TAs with lower scores are preferred since they are more constrained, meaning fewer options and harder to fill. Thirdly, whether the TA's session type preferences match the session type. Finally, if TAs are tied in the previous steps, the assigned one will be the one with fewer assigned hours. This is for fairness of the workload. The algorithms must also check if TAs have enough remaining capacity to cover the session's duration without exceeding their maximum working hours.

TAs are then assigned to the sessions, one at a time, in sorted order until each session reaches its minimum or maximum capacity. The algorithm iterates until all sessions have reached their minimum staffing requirement, or until no further assignments could be made in the last iteration, the algorithm then terminates.

Once this algorithm is completed, it returns a list of scheduling results for each session instance. The total penalty is then computed by iterating over every assignment made. For each assigned TA, if the session overlaps with any of their preferred off time, the soft-constraint weight is added to the total penalty. Additionally, if the session type does not match the TA's preferred session type, a penalty of 10 per ranking step is added. A feasibility flag indicates that every session has reached at least its minimum required TAs. In case the algorithm finishes without reaching the minimum TAs requirement for a session, that session will be flagged in the result as infeasible.

3.8 Benchmark Test Construction

To compare the algorithm implementations, benchmark tests were created for different problem sizes. The tested implementations were the naive Choco implementation, the random neighborhood LNS implementation, the custom neighborhood LNS implementation, and the hybrid CP and greedy implementation. The CP-SAT and greedy implementations were also benchmarked in order to provide a point of reference for the hybrid implementation. The purpose of the benchmarks was to measure both runtime and schedule quality.

The benchmark requests were generated programmatically. Each request contained a number of TAs, course sessions, hard constraints, and soft constraints, with specific details being influenced by a seeded random number. The quality of a generated schedule was measured using the total penalty, where a lower penalty represents a better schedule.

For every test scenario (see Table3.1), all four algorithms. Each scenario had a different seeded random number, although for the same scenario, the same seed was used regardless of the algorithm. The test checked that each algorithm found a feasible schedule. The benchmark also printed the runtime in milliseconds, the total penalty, and whether the algorithm could prove that the solution was optimal. Runtime was used to compare how fast the algorithms were, while the penalty was used to compare how well they handled soft constraints and TA preferences.

Scenario	*Weeks*	*TAs*	*Sessions/week*	*Total sessions*	*Min TAs/session*	*Max TAs/session*	*Hard/TA*	*Soft/TA*
Small generated	2	6	6	12	1	3	1	2
Medium generated	5	8	8	40	1	4	2	4
Large generated	8	12	12	96	1	4	3	8
Many soft constraints	4	8	8	32	1	4	1	10
Very many soft constraints	5	10	10	50	1	4	1	20
Many hard constraints	4	10	8	32	1	4	4	3
Higher staffing requirement	3	10	7	21	2	5	2	4
Dense sessions, few TAs	6	4	8	48	1	2	2	6
Sparse sessions, many TAs	4	12	5	20	1	3	2	8
Compact schedule stress	5	8	10	50	1	3	2	5
Session preference conflict	5	8	9	45	1	4	1	6
Tight hour budgets	4	7	8	32	1	3	2	5
Baseline no soft constraints	4	8	8	32	1	4	2	0

Table 3.1: Benchmark scenario details.

The *small*, *medium*, and *large* scenarios were used to see how the algorithms behave when the problem gets bigger. Some other scenarios were made to test more specific situations. For example, the *dense sessions, few TAs* scenario has many sessions but not many TAs, which makes the problem harder. The *sparse sessions, many TAs* scenario is easier in terms of staffing, because there are more TAs available.

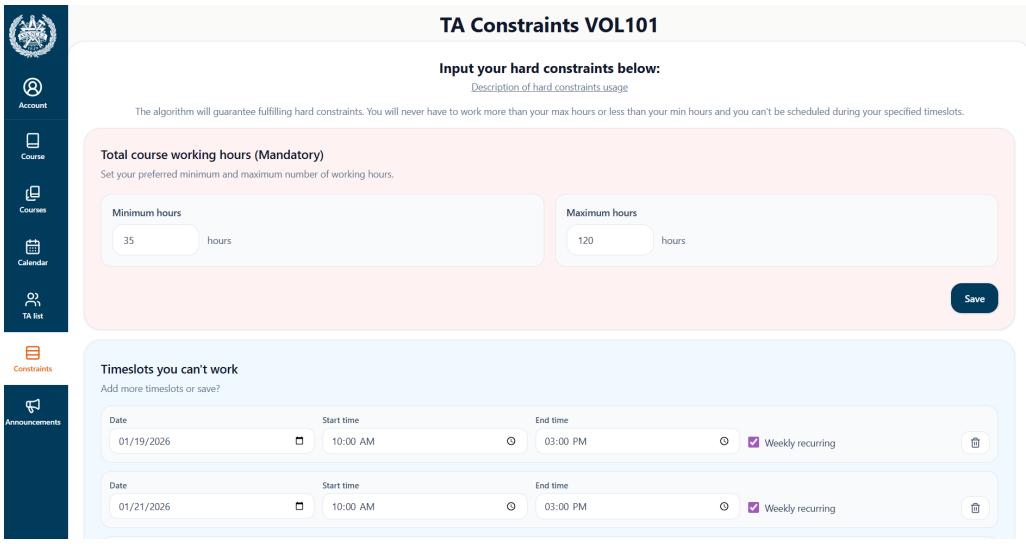
The *many soft constraints* and *very many soft constraints* scenarios were used to test how well the algorithms could reduce penalties from TA preferences. The *many hard constraints scenario* tested how the algorithms worked when many TAs were unavailable. The *higher staffing requirement* scenario made each session require more TAs. The *compact schedule stress* scenario was used to test the compact or spread-out schedule preference. The *baseline no soft constraints* scenario was included as a simpler case where no extra soft time constraints were added.

3.9 User Interface

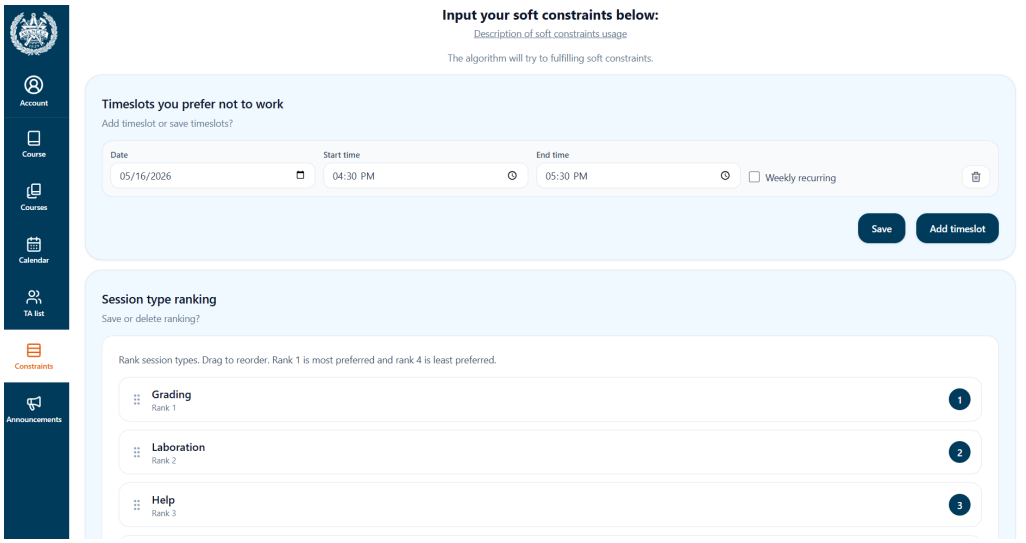
The user interface was developed to support the main aim of the thesis, which is to improve TA schedule satisfaction through scheduling software. Since the scheduling algorithm depends on information from both TAs and CRs, the interface was designed to make this information easy to enter, update and understand. A large focus was placed on the constraint system, because the constraints describe when TAs can work, when they prefer not to work, how many hours they want to work, and what types of sessions they prefer. In this way, the interface acts as the connection between the users and the scheduling algorithm.

The frontend was implemented using React, TypeScript and Tailwind Cascading Style Sheets (CSS). The system has two main user types: TA and CR. Both user types have access to the same main pages, but the available actions are different depending on the role. The pages can be reached through a sidebar, which makes the system easier to navigate. The sidebar also contains a course menu, where users can see the courses they are connected to. TAs can join courses, while CRs can create new courses. The pages are rendered based on both the current user and the selected course, which means that the information shown in the interface is connected to the correct course context.

The main pages in the interface are Account, Course, Calendar, TA List, Constraints and Announcements. The Course page shows general information about the selected course, such as the course description, the course owner, and the planned course sessions. CRs can create and delete course sessions from this page. When a session is created, the CR enters the minimum and maximum number of TAs needed for that session. These values are important because they are used as hard constraints in the scheduling algorithm. This means that the algorithm must try to assign enough TAs to each session without exceeding the maximum number of TAs allowed.



(a) Hard-constraint input.



(b) Soft-constraint input.

Figure 3.3: Final design of TA Constraints page.

The Calendar page shows the schedule after it has been generated by the algorithm. Only the CR can run the scheduling algorithm, because the CR is responsible for managing the final course schedule. After the algorithm has generated a schedule, the schedule cannot be edited manually in the current version of the system. This was done to keep the scope of the thesis focused on constraint-based scheduling and algorithmic schedule generation. However, manual editing of generated schedules could be added in future work.

The TA List page displays the TAs that have joined a course and their budget hours. In this system, the budget hours are based on the maximum number of hours that each TA has entered in their constraints. This helps the CR understand how much each TA is available to work. The CR can also invite TAs to a course by entering one or more email addresses. It is also possible to invite other CRs. This

feature was added after the user interviews, where a course owner explained that it would be useful to share some of the course administration with other users. This can reduce the workload for the main CR and can therefore improve the usability of the system.

The Constraints page is the most important part of the interface in relation to TA schedule satisfaction. The purpose of this page is to let TAs describe their availability and preferences in a structured way. These constraints are then used by the scheduling algorithm when creating the schedule. The first design of the page was implemented based on the needs expressed in the CR interview and TA survey (see Sections 4.2 & 4.1 respectively). The second design was created after conducting user interviews (see Section 4.3). During these user interviews, TA's expressed the desire for a simpler and more intuitive design.

The final design of the TA Constraints page was created to make the constraint input process clearer and more closely connected to the aim of improving TA schedule satisfaction (see Figure 3.3). In the first design many constraints were managed through an "Add, Delete or Update Constraints" button which opened a pop-up window. In this pop-up, the TA had to choose between different constraint types using a drop-down menu. This made the interaction less clear because the TA had to remember which constraints had already been added and move between the main page and the pop-up to update or delete them. The first design also did not clearly explain how the algorithm used hard and soft constraints. One TA stated in the interviews that this increased cognitive workload because the pop-up presented too much information at once. Another issue that all the TAs interviewed found with the old design was that the distinction between soft and hard constraints was not clear.

The final design therefore separates the page into two clear sections: "Input your hard constraints" and "Input your soft constraints". This separation was added to make it easier for TAs to understand which inputs the algorithm must follow and which inputs the algorithm should try to satisfy when possible. Each section also contains a description button. The hard-constraint section has a button called "Description of hard constraints usage", and the soft-constraints section has a button called "Description of soft constraints usage". When these buttons are clicked, the TA can read a short explanation of how the algorithm uses that type of constraint. This was added because TAs expressed during the interviews that they wanted to better understand how soft and hard constraints affect the generated schedule.

The first part of the final design focuses on the hard constraints. These are constraints that the algorithm needs to follow in order to create a valid schedule. The TA can enter their minimum and maximum weekly working hours, which are treated as hard constraints because the algorithm needs them to know how many hours the TA should be assigned. Since these values are required for the scheduling process, they are placed inside a light red box, and "Mandatory" is written next to "Total course working hours". This makes it clearer to the TA that these fields must be filled in. TAs can also add time slots when they cannot work. These time slots are also treated as hard constraints, meaning that the algorithm should not schedule the TA during those times. In the final design, these time slots can be added, viewed, deleted, and saved directly on the page.

The second part of the final design lets the users input soft constraints. These are preferences that the algorithm should try to satisfy, but which may be violated if necessary to create a feasible schedule. This part of the page lets TAs enter time slots they prefer not to work, rank different session types, and choose whether they prefer a compact or spread-out schedule. These preferences are important because they can affect how satisfied a TA is with the final schedule, even if they are not strict requirements.

TAs can enter "time slots you prefer not to work" as soft constraints. This means that the algorithm may still schedule the TA during those times if necessary, but should avoid them when possible. This distinction is important because it gives the scheduling algorithm more flexibility while still allowing it to prioritize TA satisfaction. TAs can also rank different session types, such as grading, laboration, help, and exercise sessions. This allows the algorithm to take into account what kind of work each TA prefers. The ranking is done through a drag-and-drop interface, where rank 1 represents the most preferred session type, and rank 4 represents the least preferred session type.

Another preference that can be entered is whether the TA prefers a compact schedule or a more spread-out schedule. This feature was added because the TAs expressed the need for it in the user survey. Some TAs may prefer to have their work placed close together to reduce the number of separate workdays, while others may prefer a more spread-out schedule.

The final design makes the TA Constraints page clearer and easier to use than the first design. Instead of hiding several actions inside a pop-up, the final design makes the constraints visible and editable directly on the page. This reduces the amount of information the TA has to remember while using the interface and makes it easier to understand, update, and delete constraints. By clearly separating mandatory hard constraints from preference-based soft constraints, the final design supports the main aim of the thesis: improving TA schedule satisfaction by helping the algorithm receive more accurate and understandable input from TAs.

For CRs, the Constraints page is more limited. The CR can view the TAs and their hard constraints, but does not need to see all soft preferences in detail. This design choice was made because hard constraints are the most important for determining whether a TA can be scheduled at a certain time. Soft constraints are mainly used by the algorithm to improve the quality of the schedule, and they are more personal to the TA. By separating the information shown to TAs and CRs, the interface avoids showing unnecessary details while still giving the CR the information needed to manage the course.

The Announcements page allows both TAs and CRs to create and view announcements. This feature was also influenced by the user interviews, where a CR expressed that TAs should be able to create announcements as well. This supports communication within the course and can make the system more useful in practice.

Overall, the user interface supports the thesis by making the constraint collection process clear and accessible. Since the algorithm can only generate a satisfying schedule if it receives useful information about TA availability and preferences, the interface is an important part of the scheduling system. By allowing TAs to enter hard constraints, soft constraints, session preferences and schedule preferences, the

frontend helps the system create schedules that are more aligned with the needs of the TAs. This makes the interface not only a way to use the application, but also a central part of improving TA schedule satisfaction.

3.10 Data Validation and Processing

A central part was the development of the backend logic used to validate, store and process the data needed for TA allocation. Since the scheduling algorithm depends on correct input data, the backend was designed to make sure that courses, sessions, TA assignments, and TA constraints follow a consistent structure before they are used to generate a schedule.

The backend was implemented using Spring Boot, and PostgreSQL was used for data persistence. The backend application and database were containerized and run together using Docker Compose for ease of development. The application exposes Representational State Transfer (REST) endpoints that allow CRs and TAs to interact with the system. CRs can create and configure courses, invite TAs and other CRs, and add course sessions. TAs can join courses and submit their scheduling constraints and preferences. This separation is important because the scheduling problem is built from data provided by both user groups.

Course data is validated before it is saved. A course must contain basic information such as a course code, description, start date, and end date. The dates are important because they define the period in which sessions and recurring constraints are meaningful. Sessions added by CRs must also contain valid information, such as session type, start time, end time, and the required number of TAs. For example, a session should not end before it starts and the minimum number of required TAs should not be greater than the maximum number of TAs. These checks reduce the risk of sending impossible or unclear data to the scheduling algorithm.

TA constraint data is also validated and processed in the backend. TAs can submit hard constraints, soft constraints, workload limits, session type preferences, and preferences related to schedule structure. Hard constraints represent times when a TA cannot work and must always be respected by the generated schedule. Soft constraints represent preferences that the algorithm should try to satisfy when possible. The backend checks that time intervals are valid, that submitted constraints belong to the correct course and TA assignment, and that the constraint dates are relevant for the course.

The backend also transforms application data into a format that can be used by the scheduling algorithm. Courses, sessions, TA assignments, and constraints are stored as domain objects in the application, but the algorithm uses a more focused input model. This model contains the sessions that need staffing, the available TAs, their workload limits, hard constraints, and soft constraints. By separating the application data model from the algorithm input model, the system becomes easier to maintain. Changes to the user interface or database structure do not necessarily require changes to the algorithm itself.

The processing stage can therefore be seen as a bridge between user input and schedule generation. Data entered by CRs defines the scheduling demand, while data entered by TAs defines the limitations and preferences of the available workers. The

backend validates this information, removes unnecessary duplication, and prepares it for the allocation algorithm. This helps ensure that the generated schedules are based on consistent and meaningful data.

This part of the system is important for the user-centered aim of the thesis. If the backend accepts unclear or invalid data, the resulting schedule may be infeasible or unsatisfactory even if the algorithm itself works correctly. By validating and processing the data before scheduling, the system increases the chance that the final allocation respects TA availability, reflects TA preferences, and satisfies the requirements set by the CRs.

3.10.1 TimeEdit Integration

The backend was also designed to support integration with TimeEdit. In this project, the purpose of the TimeEdit integration was to help TAs import hard constraints based on their existing schedule data. Instead of manually entering every time period when they are unavailable, a TA can use schedule information from TimeEdit as a basis for creating constraints in ATAAS.

The integration was handled through the backend rather than directly in the frontend. This is important because Application Programming Interface (API) keys and organisation-specific configuration should not be exposed to the client application. It also allows the backend to process the imported data before it is saved in the system. For example, events retrieved from TimeEdit can be converted into internal hard-constraint time slots, matched to the correct TA and course assignment, and stored in the same format as manually entered constraints.

In the context of the allocation problem, imported TimeEdit events are treated as hard constraints. This means that the scheduling algorithm must respect them when assigning TAs to course sessions. The integration therefore supports the user-centered aim of the system by reducing manual input for TAs and making it easier for them to provide accurate availability information.

The TimeEdit integration is not the main focus of the thesis, but it is relevant to the backend method because it affects how constraint data can enter the system. It also shows how the backend acts as a processing layer between external data sources, user input, and the scheduling algorithm.

4
Results

This chapter presents the results obtained from the methods and evaluations conducted in the thesis.

4.1 Survey Results

The survey received a total of 40 responses from TAs, providing insight into their experiences, constraints, and preferences related to scheduling. The majority of the students were bachelor students, followed by master students, while no PhD students participated (see Figure 4.1). Most respondents had prior experience as TAs, with 20 having worked 2-3 study periods and 14 having worked four or more periods (see Figure A.1). Additionally, a large majority worked at a 20% TA position, indicating that most participants had substantial involvement in TA activities (see Figure A.2).

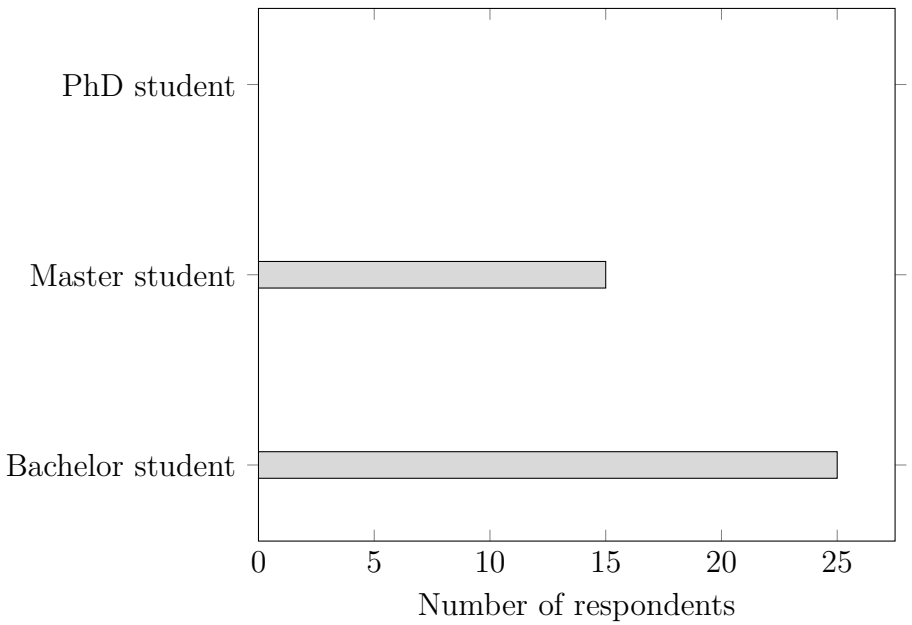


Figure 4.1: Distribution of respondents by role.

4.1.1 Availability Constraints

Availability was primarily influenced by the respondents' own course schedules, which was selected by 39 out of 40 participants (see Figure 4.2). Other common

factors included exams and family or personal commitments. In contrast, external work and health-related reasons were less frequently reported.

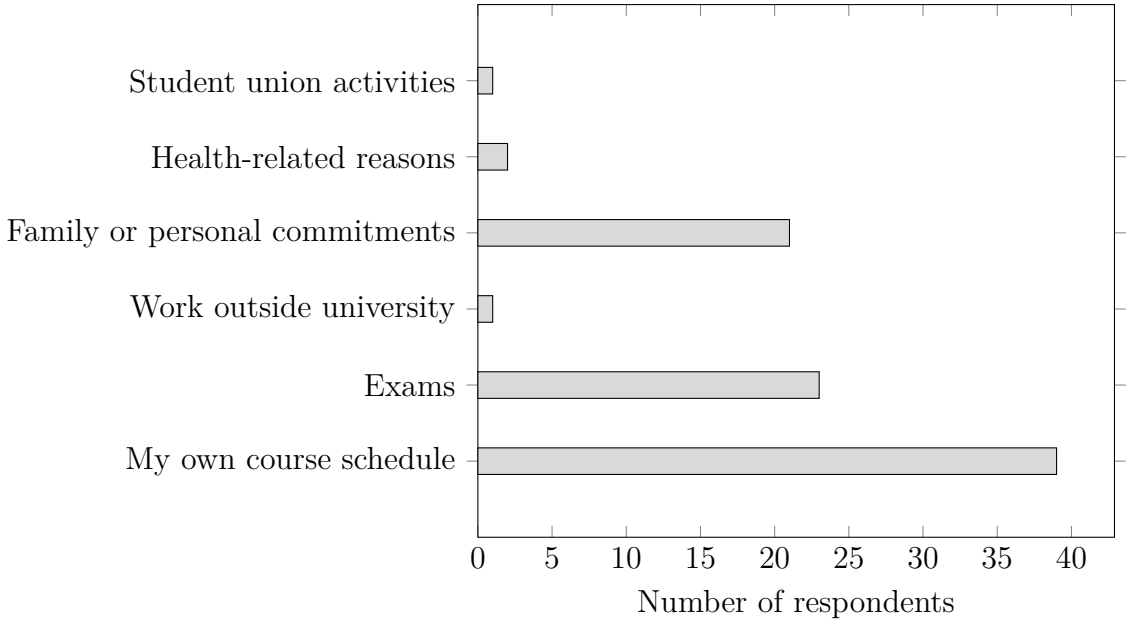


Figure 4.2: Factors affecting TA availability based on survey responses (multiple selections allowed).

Most TAs reported that they are able to specify exact time slots for their availability, although a notable portion indicated that their availability may change over time or is not known far in advance (see Figure A.4). Weekends were the most commonly unavailable time period, while a majority reported having no fixed time slots they could never work (see Figure A.5).

Communication of availability is currently fragmented. The most common methods are spreadsheets and messaging platforms such as Slack or Discord, followed by email and verbal agreements (see Figure A.3). This indicates a lack of standardized tools across courses.

4.1.2 Preferences

Preferences were found to be important for most respondents, with 37 out of 40 indicating that it is either very important or somewhat important that their preferences are respected (see Figure 4.3). Common preferences included having the same time slots each week, avoiding early mornings, and working the same type of sessions (see Figure A.7).

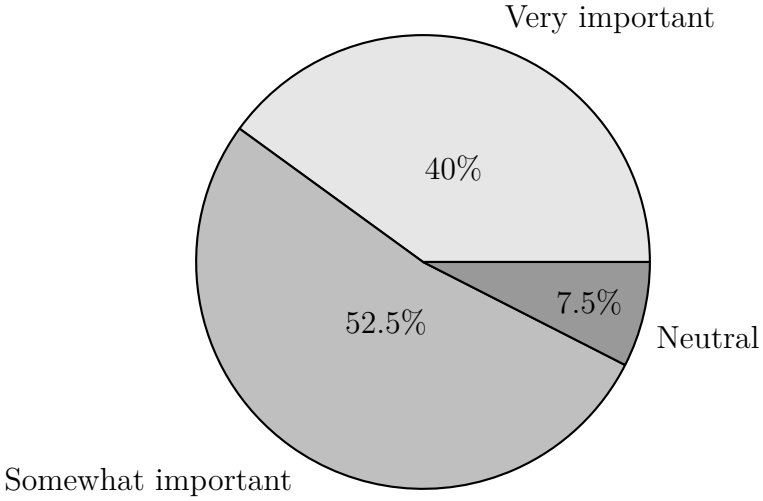


Figure 4.3: Importance of preference satisfaction in TA scheduling as reported by respondents.

4.1.3 Fairness and Workload

Fairness was most commonly associated with an equal number of working hours, selected by 34 respondents (see Figure 4.4). Other important aspects included taking unavailable times into account and considering personal preferences. Fewer respondents associated fairness with an equal distribution of undesirable time slots.

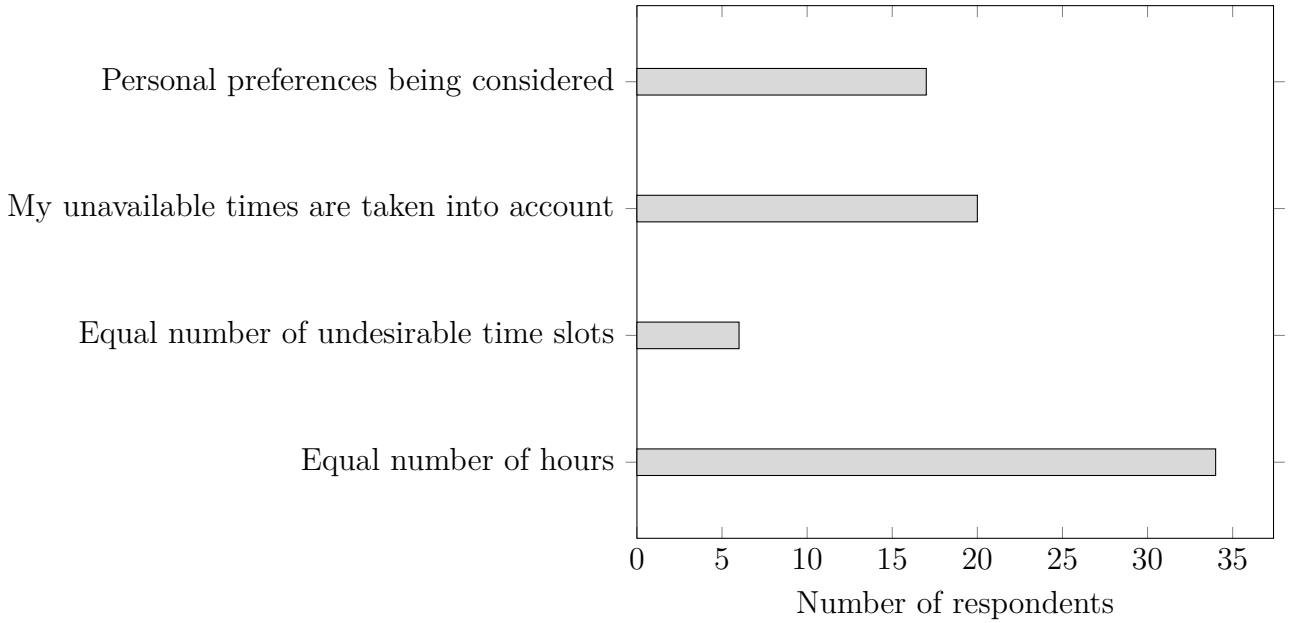


Figure 4.4: Respondents' views on what constitutes a fair TA schedule (multiple selections allowed).

Most respondents had not experienced unfair scheduling, although some reported issues such as uneven workload distribution and differences in allocated hours (see Figure A.8).

4.1.4 Current Scheduling Methods

Overall satisfaction with current scheduling methods was relatively high, with 30 respondents reporting that they were either very satisfied or somewhat satisfied (see Figure A.9). However, several issues were identified. The most common problems included manual and time-consuming processes, poor communication of changes, and confusing spreadsheets (see Figure A.10). Some respondents also highlighted inconsistencies between courses and lack of integration with personal calendars.

4.1.5 Desired Features

Respondents expressed strong interest in improvement scheduling functionality. The most desired features were visual availability input (calendar-style), notifications for schedule changes, and fair distribution of hours and sessions (see Figure A.11). Additionally, 21 respondents valued automatic conflict prevention, and 20 expressed interest in ranking preferred sessions.

A majority of the respondents indicated that they would be willing to regularly update their availability in such system, suggesting good potential for user adoption (see Figure 4.5).

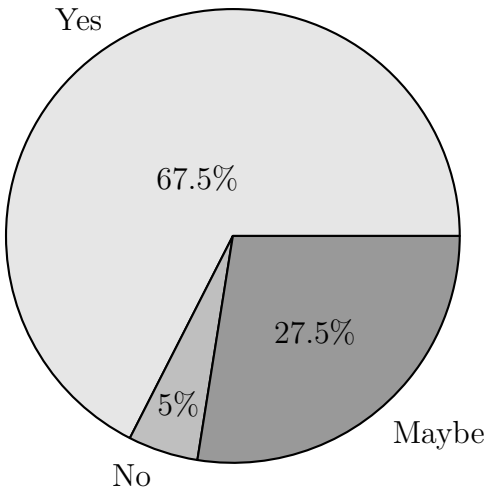


Figure 4.5: Willingness of respondents to regularly update their availability in a scheduling system.

4.1.6 Open Feedback

Open-ended responses emphasized the need for a more standardized and automated scheduling system. Several respondents highlighted the importance of integrating schedules with personal calendars, improving communication, and ensuring fairness in workload distribution. Others pointed out that current methods vary significantly between courses, leading to confusion and inefficiencies.

4.2 Interview with Course Responsible

An interview was conducted with a CR to gain insight into current TA scheduling practices and to identify challenges and requirements for a potential system.

The results show that TA scheduling is currently handled in a largely decentralized and semi-manual manner. In the most recent course, scheduling is managed using a spreadsheet where all sessions are listed, and TAs assign themselves by entering their names. The spreadsheet included automated calculations to track allocated hours, and TAs are expected to reach a target workload (e.g., approximately 60 hours). The CR generally avoids direct scheduling and instead relies on TAs to distribute the workload among themselves. Intervention only occurs when imbalances arise.

The scheduling process was described as iterative and partly time-consuming, particularly due to repeated manual work such as copying and adapting scheduling data between course instances. Despite this, the current approach is perceived to work relatively well in practice, largely due to cooperative behavior among TAs rather than the robustness of the system itself.

Several recurring challenges were identified. One issue is that certain sessions may become understaffed, especially later in the course when TA availability decreases due to exams. Additionally, differences in TA roles (e.g., student TAs versus PhD students) introduce constraints that are not explicitly handled in the current system. The scheduling process also depends heavily on voluntary participation, which can lead to uneven coverage in some cases.

Fairness was considered important, particularly since all TAs can view each other's workload in the spreadsheet. While no major complaints have been reported, there is a risk that more proactive TAs take on a disproportionate amount of work. Ensuring that TAs do not exceed their allocated work hours was also highlighted as a key requirement.

The interview further revealed that flexibility is a critical requirement for any new system. The CR emphasized the importance of being able to define different types of activities (e.g., lab sessions, grading, meetings) and to adapt the scheduling structure over time. A lack of flexibility was identified as a potential reason for not adopting a new system.

Regarding the proposed system design, several areas for improvement were noted. The distinction between "constraints" and "requests" was perceived as unclear, and the system was considered too centralized. The CR expressed a preference for delegation and collaborative scheduling, rather than a fully controlled, top-down approach. It was also suggested that TAs should be able to view each other's schedules to support transparency and coordination.

Additional desired features include the ability to export data (e.g., to CSV format), support for multiple access levels (e.g., read-only and full access), and minimizing the handling of sensitive personal information. The interview also highlighted the importance of limiting administrative overhead and ensuring that the system does not introduce more work compared to existing practices.

Overall, the interview confirms that while current scheduling methods can function adequately, they rely heavily on manual effort and informal coordination. This

highlights the need for a system that balances automation with flexibility and user control.

4.3 Interview with Teaching Assistants

In terms of TA interviews, participants reported mixed experiences with respect to the input process. Most participants found the interface relatively intuitive overall, although multiple participants reported confusion regarding specific navigation flows and constraint terminology.

One participant also reported that the interface presented too much information at once, leading to a sense of cognitive overload when interacting with the constraint input view.

A recurring issue observed among all participants was uncertainty regarding the distinction between hard and soft constraints. Three participants expressed difficulty in understanding how different types of constraints should be categorized, despite the presence of explanatory text in the interface.

In some cases, participants initially misinterpreted constraint types but were able to proceed after identifying the provided hints. However, confusion remained regarding hard and soft constraints with respect to more abstract constraint types, such as hourly workload-related settings, session preferences, and schedule compactness.

Participants expressed varying perceptions of fairness regarding algorithmic allocation of schedules. Although all participants understood that hard constraints would be satisfied by the algorithm, opinions differed regarding the importance of soft-constraint satisfaction.

Some participants indicated that unmet soft constraints were acceptable, given their nature as preferences. Others emphasized that fairness depends on how evenly such constraints are satisfied across different TAs, suggesting that unequal satisfaction of preferences could lead to perceptions of unfairness.

In general, the participants expressed a positive attitude towards the system and its potential use. All four participants indicated a preference for the system over current scheduling methods.

4.4 Choco Solver Benchmark Results

Table 4.1 shows the benchmark results for the three tested algorithm implementations. All algorithms found feasible schedules in all scenarios (for scenario details, see 3.1). This means that all implementations were able to satisfy the hard constraints, such as TA availability, session staffing requirements, and hour limits. The main difference between the algorithms was therefore the runtime and the penalty value.

Scenario	Naive runtime	Naive penalty	Random runtime	Random penalty	Custom runtime	Custom penalty
Higher staffing requirement	23415	820	10253	400	110	401
Very many soft constraints	27316	2440	12519	530	1961	530
Session preference conflict	26731	320	10088	40	153	40
Compact schedule stress	15552	420	10664	50	680	50
Baseline no soft constraints	2343	0	162	0	594	0
Small generated	407	10	735	10	61	70
Dense sessions, few TAs	18182	463	10669	143	87	143
Sparse sessions, many TAs	3417	180	10051	180	757	180
Tight hour budgets	28621	180	10049	70	78	70
Many soft constraints	47671	560	10127	330	195	330
Medium generated	11556	330	10041	40	250	40
Many hard constraints	16318	410	10085	40	137	40
Large generated	17865	1090	1674	0	1437	0

Table 4.1: Benchmark results for the three algorithm implementations. Runtime measured in milliseconds.

The naive algorithm often produced schedules with much higher penalties. This can be seen clearly in scenarios such as *very many soft constraints*, where the naive algorithm got a penalty of 2440, while both LNS algorithms got a penalty of 530. A similar pattern can be seen in the *medium generated*, *many hard constraints*, and *tight hour budgets* scenarios. This suggests that the naive algorithm can find feasible schedules, but it is worse at improving the schedule according to TA preferences.

The random LNS algorithm usually found much better penalties than the naive algorithm. In many scenarios, it found the same best penalty as the custom LNS algorithm. For example, both LNS algorithms got a penalty of 40 in the *medium generated*, *many hard constraints*, and *session preference conflict* scenarios. However, the random LNS algorithm often had a runtime around 10 seconds. This is probably because it spends more time searching for improvements.

The custom LNS algorithm gave the best overall balance between runtime and penalty. In most scenarios, it found the same penalty as the random LNS algorithm, but much faster. For example, in the *dense sessions, few TAs* scenario, both LNS algorithms got a penalty of 143, but the custom LNS algorithm finished in 87 ms compared to 10669 ms for random LNS. The same pattern can be seen in the *tight hour budgets* and *many soft constraints* scenarios.

One exception was the *small generated* scenario. In this case, the naive and random LNS algorithms got a penalty of 10, while the custom LNS algorithm got a penalty of 70. Since this scenario is small, the more advanced search strategy did not give a better result in this specific run. However, the custom LNS algorithm was still the fastest.

Overall, the results show that the LNS-based algorithms are better than the naive algorithm when soft constraints and preferences matter. The custom LNS algorithm performed especially well because it often reached the best penalty while using much less time than random LNS. This makes it the most suitable implementation for the scheduling system, since the goal is not only to create a feasible schedule, but also to create a schedule that better matches TA preferences.

4.5 CP-SAT and Greedy Solver Benchmark Results

Table 4.2 shows the benchmark result for the CP-SAT and greedy hybrid algorithm. The result shows that the hybrid algorithm found a feasible schedule that satisfied all the hard constraints that were mentioned before. To better understand the contribution of each phase, CP-SAT and the greedy algorithm were also evaluated independently alongside the hybrid algorithm, where the standalone greedy ran from an empty assignment without prior CP-SAT assignments.

Table 4.2: CP-SAT, Greedy, and Hybrid algorithm results across test scenarios.

Scenario	CP-SAT penalty	Greedy penalty	Hybrid penalty	Hybrid runtime (ms)
Higher staffing requirement	0	500	1840	7
Very many soft constraints	0	620	4950	21
Session preference conflict	0	280	1830	10
Baseline no soft constraints	0	420	470	6
Mock scenario	0	2080	3480	6
Small generated	0	0	350	1
Dense sessions, few TAs	0	520	1220	5
Sparse sessions, many TAs	0	270	1690	4
Tight hour budgets	0	340	1090	6
Many soft constraints	0	110	1950	8
Medium generated	0	210	1500	7
Many hard constraints	0	290	930	6
Large generated	0	960	2710	22

CP-SAT always produced a penalty of zero across all scenarios. This is expected since CP-SAT only optimizes hard constraints and does not consider soft-constraint violations.

The hybrid produced higher penalty scores than the greedy algorithm across all scenarios. For example, in *many soft constraints* scenario, the hybrid produced a penalty of 1950 compared to 110 for the greedy standalone algorithm. This is because the CP-SAT phase assigns TAs based solely on hard constraints without considering soft constraints. Since the greedy phase cannot modify the assignment already made by CP-SAT, any soft constraint that got violated in the first phase will also be counted in the total penalty score. This pattern can be seen across all test scenarios.

The hybrid completed all scenarios within 22 milliseconds, with the fastest one being 1 millisecond. This shows that the hybrid algorithm is computationally efficient regardless the problem size.

One scenario worth noticing in the *baseline no soft constraints* scenario, where the hybrid produced a penalty of 470 compared to 420 for greedy algorithm only. When there is no soft constraint, the CP-SAT assignments are less likely to conflict with TA preferences.

Overall, the results show that the greedy phase is the primary contributor to soft-constraint optimization, while CP-SAT provides the hard-constraint guarantee that the greedy phase builds upon. No scenario produced an infeasible CP-SAT result, so the empty assignment fallback described in Section 3.5.3 was not exercised in these benchmarks.

5
Discussion

This chapter discusses the results of the thesis, their implications and limitations, as well as possible directions for future work.

5.1 Discussion of Survey Results

The results of the survey provide valuable insights into current TA scheduling practices and highlight several key challenges that motivate the need for an improved system. Overall, the findings indicate that while existing scheduling methods are generally perceived as acceptable, there are clear inefficiencies and inconsistencies that can be addressed through automation and better system design.

A central observation is that TA availability is highly constrained by their own course schedules, as almost all respondents identified this as a primary factor. This emphasizes the importance of ensuring that any scheduling system strictly respects availability constraints. Furthermore, although many TAs are able to specify precise time slots, a significant portion reported that their availability may change over time or is not known far in advance. This suggests that the system should support flexible and updatable availability input rather than relying on static schedules.

The survey reveals that preferences play an important role for most TAs. A large majority indicated that it is important that their preferences are considered, with common preferences including consistent time slots and avoiding early mornings. This supports the inclusion of soft constraints in the scheduling algorithm, where preference satisfaction is taken into account alongside hard constraints. However, the variation in preferences also indicates that it may not always be possible to satisfy all preferences simultaneously, highlighting the need for a balanced and transparent prioritizing strategy.

Fairness emerged as another key aspect of scheduling. Most respondents associated fairness primarily with an equal distribution of working hours, while fewer emphasized equal distribution of undesirable time slots. This suggests that workload balance should be a primary objective in the scheduling algorithm. At the same time, the relatively low number of respondents reporting unfair scheduling indicates that current methods are not fundamentally flawed, but rather lack consistency and systemic support.

Although overall satisfaction with current scheduling methods was relatively high, several recurring issues were identified. In particular, respondents pointed out that current approaches are often manual, time-consuming, and fragmented across different tools such as spreadsheets, messaging platforms, and email. This lack of

standardization can lead to inefficiencies, miscommunication, and increased administrative workload for both TAs and CRs. Additionally, some respondents highlighted issues such as late scheduling, difficulty integrating schedules with personal calendars, and inconsistencies between courses.

The desired features identified in the survey strongly support the development of an automated scheduling system. Respondents expressed a clear preference for features such as visual availability input, automatic conflict prevention, and fair distribution of workload. The demand for notifications and calendar integration further indicates the importance of usability and seamless interaction with existing tools. Moreover, the majority of respondents indicated a willingness to regularly update their availability, suggesting that such a system would be practical to implement in a real-world setting.

The open-ended responses reinforce these findings by emphasizing the need for standardization, improved communication, and increased automation. Several respondents specifically highlighted the importance of having a unified system across courses, which would reduce confusion and improve the overall user experience.

In summary, the survey results validate the relevance of the problem in the thesis and provide clear guidance for the design of the proposed system. The findings highlight the importance of handling availability as a strict constraint, incorporating preferences as soft constraints, and ensuring fairness through balanced workload distribution. Several of these findings directly influenced the system design, including the addition of recurring constraints, drag-and-drop session type ranking, and a compact or spread-out schedule preference, all of which were motivated by preferences expressed in the survey.

5.2 Discussion of Interview with Course Responsible

The interview with the CR provides valuable insight into how TA scheduling is currently performed in practice and highlights several important considerations for the design of an automated scheduling system.

A key finding is that the existing approach relies heavily on decentralization and self-organization among TAs. Rather than centrally assigning schedules, the CR delegates much of the responsibility to the TAs themselves, who sign up for sessions in a shared spreadsheet. While this approach appears to function adequately in many cases, its success depends largely on the cooperation, availability, and initiative of the TAs rather than on a structured or robust system. This suggests that current methods may not scale well or remain reliable under less favorable conditions.

The interview also reveals a clear trade-off between automation and flexibility. While an automated scheduling system could reduce manual work and improve consistency, there is a risk that it becomes rigid. The CR emphasized that flexibility is essential, particularly in handling different types of tasks (e.g., lab sessions, gradings, meetings) and adapting scheduling structures across course instances.

Another important aspect is the preference for delegation rather than central control. The proposed system design initially assumed a more centralized structure,

where the CR manages most aspects of scheduling. However, the interview indicates that such an approach may not align with actual user needs. Instead, there is a demand for collaborative features that allow multiple actors to participate in the scheduling process. This highlights the importance of designing systems that support different organizational workflows rather than enforcing a single model.

The discussion also raises questions about the role of constraints and system complexity. The distinction between constraints and requests in the proposed design was perceived as unclear, suggesting that the conceptual model may need simplification.

Additionally, the interview points to the need for expressive yet manageable constraint handling, where users can define both strict requirements and more flexible preferences without excessive complexity.

Fairness and workload distribution were identified as important but currently handled implicitly through transparency rather than formal mechanisms. Since all TAs can view each other's workload, social pressure contributes to a relatively balanced distribution. However, this approach may not always guarantee fairness, particularly in cases where individuals differ in initiative or availability. This supports the motivation for incorporating fairness considerations into an automated scheduling algorithm.

Finally, the interview highlights practical concerns related to usability and adoption. Features such as data export, minimal administrative overhead, and avoidance of sensitive data handling were emphasized as important for real-world use. Notably, the CR expressed skepticism toward features that could increase workload, indicating that a new system must clearly demonstrate added value compared to existing tools.

Overall, the interview underscores that the challenge of TA scheduling is not purely technical but also organizational and social. A successful system must therefore balance automation, flexibility, transparency, and ease of use, while aligning with existing work practices. Several of these findings influenced concrete design decisions, such as adding support for multiple course responsibles and an announcements page where both TAs and CRs can post updates.

5.3 Discussion of Interview with Teaching Assistants

The mixed experiences related to constraint input suggest that, while the system is usable for some users, the interface may not sufficiently support all interaction styles. The reported cognitive overload indicates that presenting all constraint options simultaneously may hinder usability, particularly for first-time users.

This suggests that a more progressive or structured interface design could improve usability by allowing users to focus on relevant inputs without unnecessary complexity.

The consistent confusion regarding hard and soft constraints highlights a critical usability issue. Since the system relies on correctly specified constraints, misunderstandings at this stage may lead to unintended scheduling outcomes. This indicates that the system does not clearly communicate how constraints are interpreted by

the algorithm. Improving the clarity of constraint definitions, for example through better labeling or contextual explanations such as a help page, could reduce this ambiguity and improve the quality of user input.

The varying perceptions of fairness suggest that satisfying constraints alone is not sufficient, users also evaluate fairness in relation to other users. In particular, participants expressed concern over unequal satisfaction of soft constraints. This emphasizes the importance of transparency in how the algorithm prioritizes and balances constraints across users. Without a clear understanding of how decisions are made, users may perceive outcomes as unfair even if they are technically valid.

The findings demonstrate a strong dependency between system usability and algorithm allocation. Since the algorithm relies entirely on user-provided constraints, any ambiguity or difficulty in expressing these constraints may directly impact the quality of the generated schedules. This suggests that improving the user interface is not only a usability concern but also a critical factor in achieving effective and fair scheduling outcomes.

The generally positive reception of the system indicates that users recognize its potential value compared to existing ad-hoc scheduling methods. This suggests that structured, system-supported scheduling approaches may improve both user experience and perceived fairness in practice.

5.4 Choco Solver

The benchmark results (see 4.1) show that Choco Solver was useful for modeling the TA scheduling problem. The solver could handle hard constraints, such as TA unavailability and overlapping sessions, while also trying to minimize violations of soft constraints. This is important because the goal of the system is not only to create a possible schedule. The goal is also to create a schedule that respects TA preferences as much as possible.

One clear result is that the naive implementation worked well for simple instances such as the *small* scenario or *baseline no soft constraints*, but was less effective when the problem became larger. This means that the small problems were simple enough for the naive implementation to find a good solution, even outperforming the more sophisticated custom LNS implementation in some cases. However, in more complex scenarios such as the *very many soft constraints* scenario, the naive implementation produced much higher penalties than the LNS-based implementations. This suggests that the naive search strategy had more difficulty finding good schedules when there were more sessions, TAs, and constraints.

This is expected because the search space grows quickly. Each possible assignment between a TA and a session is represented by a decision variable. When there are more TAs and sessions, there are many more possible schedules. Choco can remove some impossible choices using constraints, but there can still be many valid schedules left. The naive implementation can find a feasible schedule, but it may not find a very good schedule within the available time.

The LNS implementations performed better because they improved an existing solution instead of completely searching from the beginning. LNS works by keeping parts of a solution fixed and allowing other parts to change. This fits the scheduling

problem well. For example, improving one TA's schedule may require changing several of that TA's assignments at the same time. A method that can change a larger part of the schedule can therefore be more useful than a method that only makes very small changes.

The random LNS implementation showed that LNS can improve the solution quality compared to the naive implementation. It was in all cases able to match or even outperform the penalty produced by the custom implementation. However, it was often slower than the implementation with custom defined neighborhoods. This is probably because the random neighborhood does not know which parts of the schedule are most important to change. It may spend time changing assignments that do not improve the penalty much. This makes random LNS flexible, but not always efficient.

The custom LNS implementation performed better overall because its neighborhoods were based on the structure of the scheduling problem. It could relax assignments connected to specific TAs or specific weeks. This is useful because bad schedules are often connected to one TA or one time period. For example, if one TA has many bad assignments, it makes sense to reconsider that TA's schedule. If one week contains many penalties, it makes sense to rebuild that week. This probably explains why custom LNS reached the same penalty as the random LNS implementation in almost all instances while also being faster on average.

The use of time limits and plateau limits is also important. In theory, the solver could continue searching for a very long time to prove that a solution is optimal. In practice, this is not always useful. A CR needs a schedule within a reasonable time. Therefore, it makes sense to stop the search after a time limit, or when the solver has not improved the solution for some time. This means that the result may not always be proven optimal, but it can still be feasible and useful.

Because of this, the value `provenOptimal` is important. A solution can be valid and have a low penalty even if the solver has not proven that it is the best possible solution. The system should therefore separate feasible solutions from proven optimal solutions. In the evaluation however, the penalty value is more useful for comparing schedule quality. This is because in many instances with a large search space, no model is able to prove optimality.

The soft-constraint weights are another important part of the model. A violating soft constraint adds a penalty to the objective value. A lower penalty means that the schedule respects more TA preferences. However, the chosen weights affect what the solver prioritizes. If all soft constraints have the same weight, then all preference violations are treated as equally important. This is simple and easy to understand. However, in reality, some preferences may be more important than others. For example, avoiding a certain time may be more important than avoiding a certain session type. This also introduces a certain subjectiveness to the benchmarking, as ultimately the way in which various soft-constraint violations (for Choco Solver penalty model, see 3.6) are penalized, though based on user feedback, is subjective. Future versions of the system could allow different weights for different kinds of preferences or even allow some degree of user input directly for weighting soft constraint. An example could be to allow users to rate how important it is that their session preferences are taken into account on a 10-point scale.

Fairness is also important to discuss. The current objective minimizes the total penalty. This improves the schedule overall, but it does not automatically mean that the schedule is fair for every TA. For example, one TA could receive many bad assignments while others receive good schedules, as long as the total penalty is still low. This may be unfair even if the total penalty is good. A future version could include fairness in the objective, for example by penalizing large differences in penalty between TAs.

The model also uses minimum and maximum hour limits for each TA. This helps distribute work more fairly because it prevents one TA from getting too many or too few hours. However, equal hours do not always mean equal satisfaction. Two TAs can have the same number of hours, but one TA may get preferred sessions while the other gets less preferred sessions. Therefore, fairness should include both workload and preference satisfaction.

One limitation of the benchmark is that the test scenarios were generated programmatically. This made the tests repeatable and made it possible to compare different problem sizes. However, generated scenarios may not fully represent real courses. Real courses may have more irregular sessions, different TA availability patterns, and more complex preferences. Therefore, the benchmark results should mainly be seen as controlled tests of the algorithms. More testing with real course data would be needed to evaluate practical performance more accurately.

Even with this limitation, the results support the use of Choco Solver and LNS for the thesis problem. Choco made it possible to model the scheduling rules as constraints and to minimize a penalty value. The LNS implementations, especially the custom LNS implementation, showed that better schedules can be found by using search strategies that fit the structure of the scheduling problem.

Overall, the results show a trade-off between runtime and schedule quality. The naive implementation can be fast and works well for small problems, but it gives worse penalties for larger problems. The LNS implementations take more time, but they produce schedules with much lower penalties. The custom LNS implementation gives the best balance in the benchmark results because it uses knowledge about TAs and weeks while still allowing the solver to search for better solutions.

5.5 CP-SAT and Greedy Hybrid Algorithm

This hybrid CP-SAT and greedy implementation was chosen because the CP-SAT solver is well-suited to handle hard constraints, as it can systematically search for a feasible solution while guaranteeing that no hard constraints are violated. Optimizing soft constraints such as preferences and workload fairness within the CP-SAT solver alone would significantly increase the complexity of the model. When CP-SAT handles soft constraints as well as hard constraints, it does not stop when a feasible solution is found. Instead, CP-SAT continues to search for a new feasible solution with a lower penalty score, stopping only when no better penalty score can be found or a time limit is reached. As the number of TAs and sessions grows, the number of feasible solutions grows exponentially, making this search increasingly expensive and impractical for larger scheduling problems. Since the goal of this implementation is not to find the most optimal schedule but a schedule that is good

enough while respecting TA preferences, the greedy algorithm is a better choice for handling soft constraints in larger scheduling problems. Unlike CP-SAT, the greedy algorithm commits to each assignment without backtracking, and produces reasonable soft-constraint quality through the composite dispatching rule.

The standalone greedy was able to get zero in the penalty score in the small generated test. This means that the small problem was simple enough for the standalone greedy to find a good solution. However, the hybrid algorithm still produced a penalty score of 350 in the same scenario compared to standalone greedy. The limitation of the current penalty calculation in the hybrid algorithm is that it does not distinguish soft-constraint violations introduced by the CP-SAT phase, and includes them in the total penalty score. This makes it difficult to accurately evaluate the greedy phase's performance, since part of the penalty was caused by CP-SAT that the greedy phase had no control over. A possible future improvement would be for the penalty calculation to be separated between two phases.

Another possible future improvement would be to add soft-constraint awareness to the CP-SAT phase, so that its assignments are already partially preference-aligned before the greedy phase begins. This could reduce the penalty gap between the hybrid and the standalone greedy.

These limitations suggest that the two phases should be more tightly integrated with each other. A possible future improvement would be to make the two phases work more collaboratively. The greedy phase should be allowed to swap and reassign TAs that were already placed by the CP-SAT phase instead of building on top of it.

5.6 Future Work

This chapter outlines future work for this thesis, including further investigation of algorithmic optimization and dynamic scheduling, usability, and the interaction between these aspects.

5.6.1 Algorithmic Optimization and Dynamic Scheduling

This thesis has primarily focused on the initial allocation of TAs to scheduled sessions. However, real-world scheduling environments are inherently dynamic, where constraints and availability can change over time. Future work could therefore investigate adaptive scheduling approaches that allow for incremental updates to existing schedules in response to such changes. For instance, if a TA becomes unavailable after an initial allocation, the system could employ localized re-optimization techniques to adjust only the affected parts of the schedule while minimizing disruption to other assignments.

Furthermore, the current implementation does not penalize unequal work distribution. It simply ensures TAs minimum and maximum hour constraints are respected. Therefore, a possible future improvement would be to expand the penalty model to penalize an unequal distribution, taking each TAs maximum working hours into account, so that TAs who should work more (according to contract of employment) actually do work proportionately more.

In addition, evaluating the system on larger and more diverse datasets, as well

as incorporating real-world data where feasible, would also strengthen the validity of the results. Finally, more sophisticated test data generation methods could be developed to better capture realistic distributions of constraints and preferences.

5.6.2 Usability

Future work should include larger and more diverse user studies to obtain results that are more generalizable beyond the small-scale qualitative evaluation conducted in this thesis, into how CRs and TAs interact with the system. In particular, further investigation is needed into how users interpret and express their constraints.

In addition, integration with existing educational platforms is another relevant direction for future work. For instance, fully connecting the system to the essential parts of the TimeEdit API and the Canvas API could be studied to evaluate its impact on usability for both CRs and TAs.

5.6.3 Interaction Between Usability and Algorithm

An important direction for future research lies in understanding the relationship between user interaction and algorithmic outcomes. The quality of the generated schedule is inherently dependent on the accuracy and completeness of the input provided by users. If constraints are poorly specified or misunderstood, even well-performing algorithms may produce suboptimal allocations.

Bibliography

[1] L. Joseph Y-T, *Handbook of Scheduling: Algorithms, Models, and Performance Analysis (1st ed.)*. Chapman and Hall/CRC, 2004. [Online]. Available: https://doi.org/10.1201/9780203489802

[2] L. Kletzander and N. Musliu, "Solving the general employee scheduling problem," *Computers & Operations Research*, 2020. [Online]. Available: https://doi.org/10.1016/j.cor.2019.104794

[3] B. Philippe, C. Pape, and N. Wim, *Constrain-Based scheduling*. Springer New York, NY, 2001.

[4] D. S. J. Michael R. Garey, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[5] M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, 6th ed. Springer, 2022.

[6] D. Pisinger and S. Ropke, *Large Neighborhood Search*. Springer, 09 2010, pp. 399–419.

[7] Choco Solver, "Large neighborhood search," https://choco-solver.org/docs/solving/lns/, 2024, accessed: 2026-04-24.

[8] P. Kaewchanid and P. Wangmaeteekul, "Solving a teaching assistant timetabling using constraint based approach," in *2016 International Conference on Computational Intelligence and Applications (ICCIA)*, 2016.

[9] H. Cambazard, F. Demazeau, N. Jussien, and P. David, "Interactively solving school timetabling problems using extensions of constraint programming," in *Practice and Theory of Automated Timetabling V*. Springer, Berlin, Heidelberg, 2005.

[10] J. A. Whittaker, *Exploratory software testing*. Addison-Wesley, 2009.

A

Appendix 1

A.1 Survey Questionnaire

The following questionnaire was distributed via Google Forms to 231 TAs.

Background Information

What is your current role?

- Bachelor student
- Master student
- PhD student
- Other:_____

How many study periods have you worked as a TA?

- 1
- 2–3
- 4–6
- More than 6

To what percentage do you work as a TA?

- 10%
- 15%
- 20%
- Other:_____

Availability Constraints

Which of the following affect your availability as a TA?
(Select all that apply)

- My own course schedule
- Exams
- Work outside university
- Family or personal commitments
- Health-related reasons
- Other:_____

How do you currently communicate unavailable times to course responsibles?
(Select all that apply)

- Spreadsheet (Google Sheets/Excel)
- Email
- Messaging platforms (Slack, Discord, etc.)

- Verbal agreement
- Other:____

How precise is your availability?

- I can specify exact time slots
- I can specify days but not exact times
- My availability changes week to week
- I often don't know far in advance
- Other:____

Are there time slots you can never work?
(Select all that apply)

- Early mornings
- Late afternoons/evenings
- Fridays
- Weekends
- No
- Other:____

Preferences

How important is it for you that your preferences are respected in scheduling?

- Very important
- Somewhat important
- Neutral
- Not important

Which preferences do you have as a TA?
(Select all that apply)

- Same time slots every week
- Same type of session (lab / exercise / help session)
- Working with the same TA(s)
- Avoiding certain sessions
- Compact schedule (few long days rather than many short ones)
- Avoiding early mornings
- Avoiding late afternoons
- Other:____

Fairness and Workload

What does "fair" TA schedule mean to you?
(Select all that apply)

- Equal number of hours
- Equal number of undesirable time slots
- My unavailable times are taken into account
- Personal preferences being considered
- Other:____

Have you experienced unfair TA scheduling?

- Yes

- No
- Not sure

If yes, what was the issue?

Current Scheduling Methods

How satisfied are you with current TA scheduling methods?

- Very satisfied
- Somewhat satisfied
- Neutral
- Somewhat dissatisfied
- Very dissatisfied

What are the biggest problems with current scheduling tools?
(Select all that apply)

- Confusing spreadsheets
- Manual and time-consuming
- Easy to double-book
- Poor communications of changes
- Preferences ignored
- Other:____

Desired Features

Which features would you value in a TA scheduling system?
(Select all that apply)

- Visual availability input (calendar-style)
- Ability to rank preferred sessions
- Prevents scheduling conflicts automatically
- Notifications for schedule changes
- Fair distribution of hours and sessions
- Other:____

Would you be willing to update your availability regularly in a system like this?

- Yes
- No
- Maybe

Open Feedback

What is one thing you would improve about TA scheduling at Chalmers/GU?
Any other comments or suggestions?

A.2 Survey Results

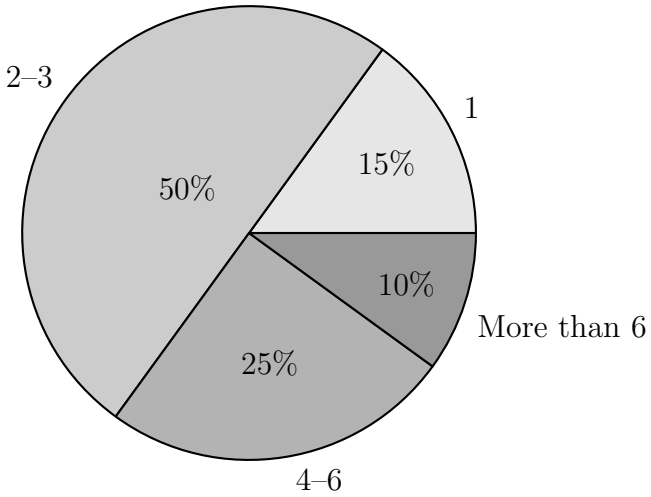


Figure A.1: Number of study periods respondents have worked as teaching assistants.

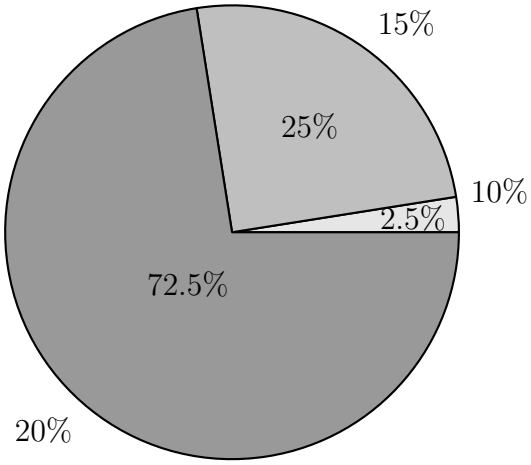


Figure A.2: Distribution of respondents by TA employment percentage.

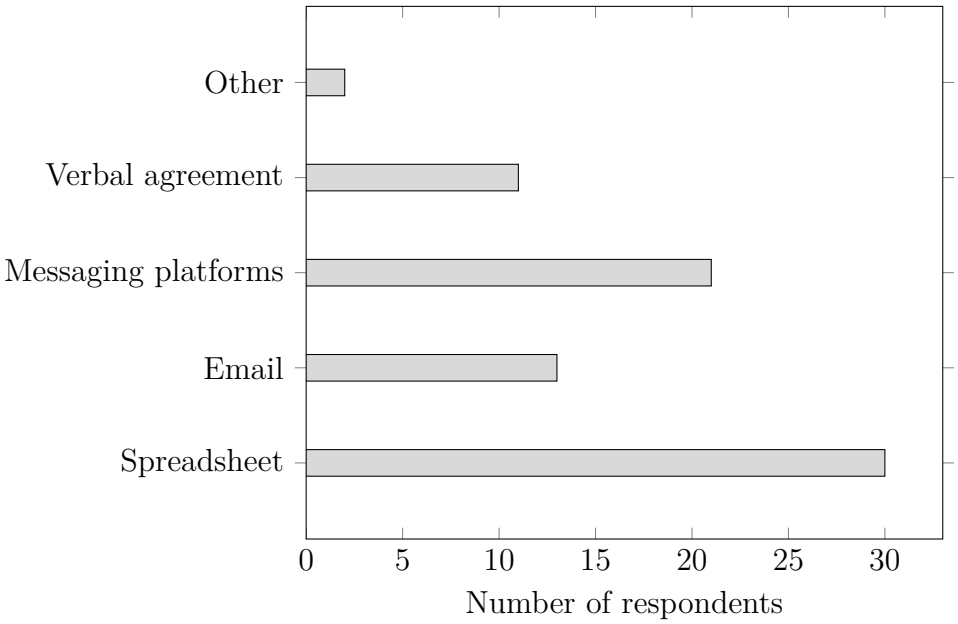


Figure A.3: Methods used by TAs to communicate unavailable times (multiple selections allowed).

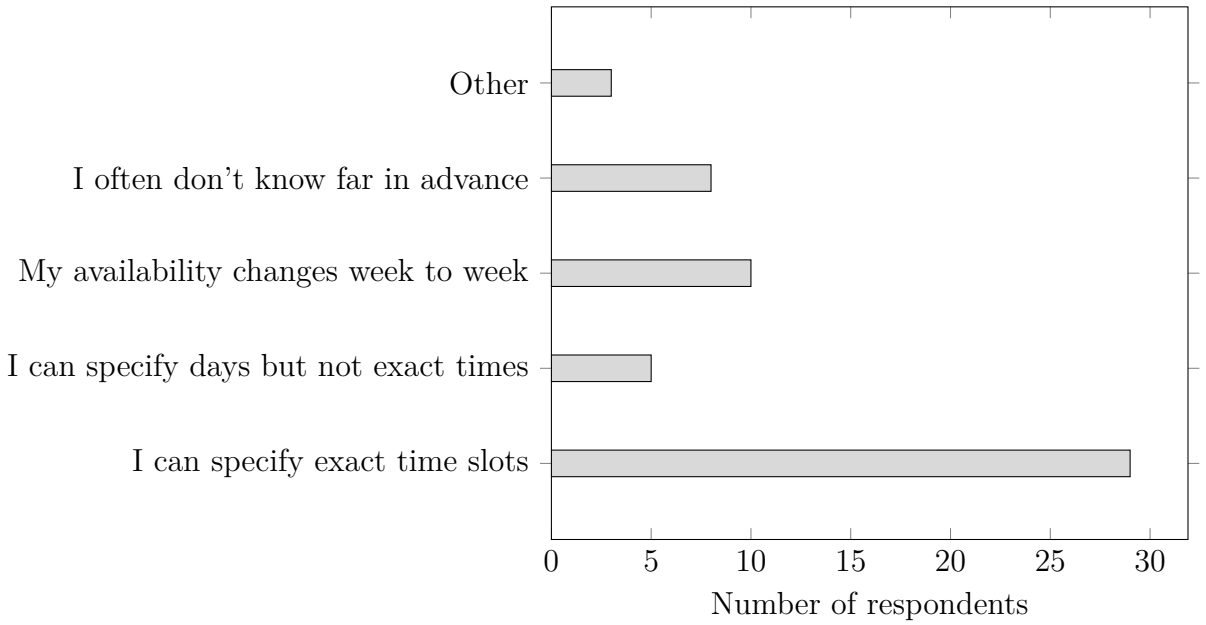


Figure A.4: Degree of precision with which respondents can specify their availability (multiple selections allowed).

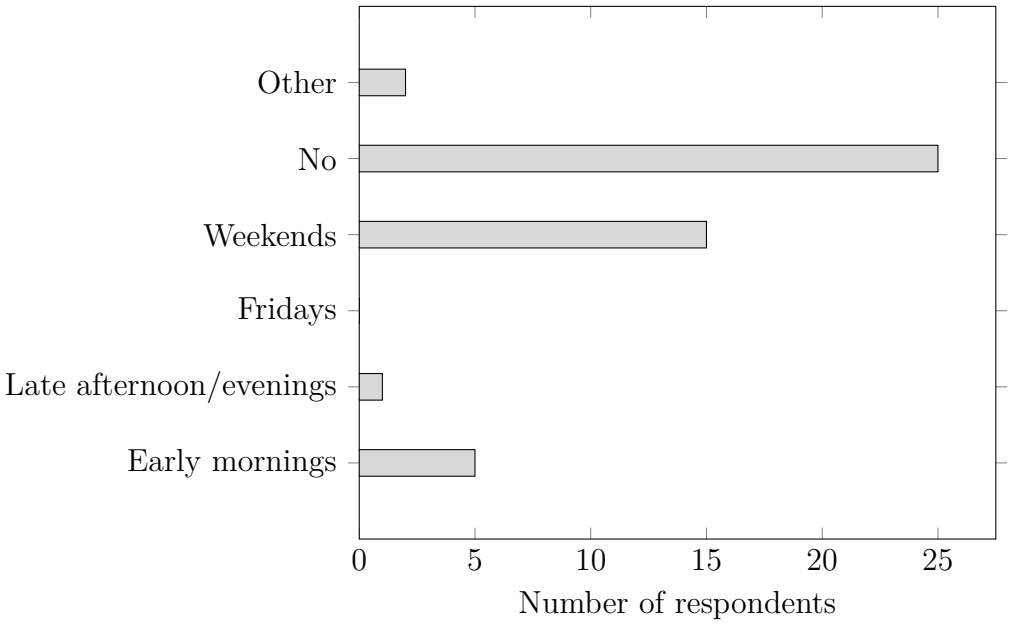


Figure A.5: Time slots during which respondents are unable to work (multiple selections allowed).

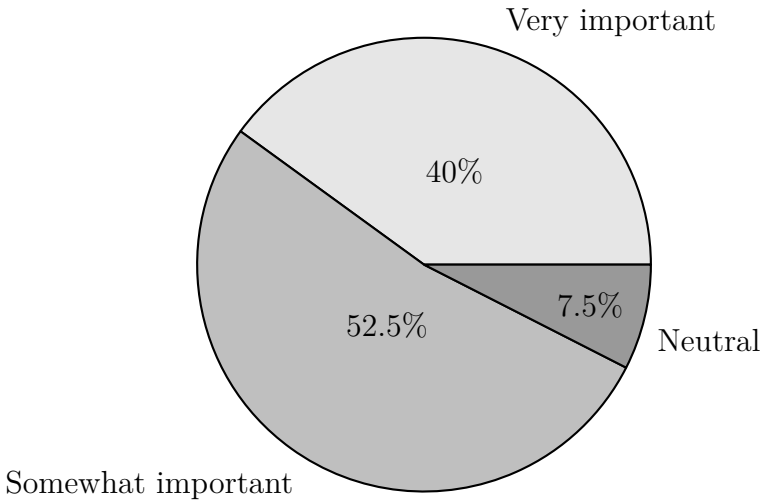


Figure A.6: Importance of preference satisfaction in TA scheduling as reported by respondents.

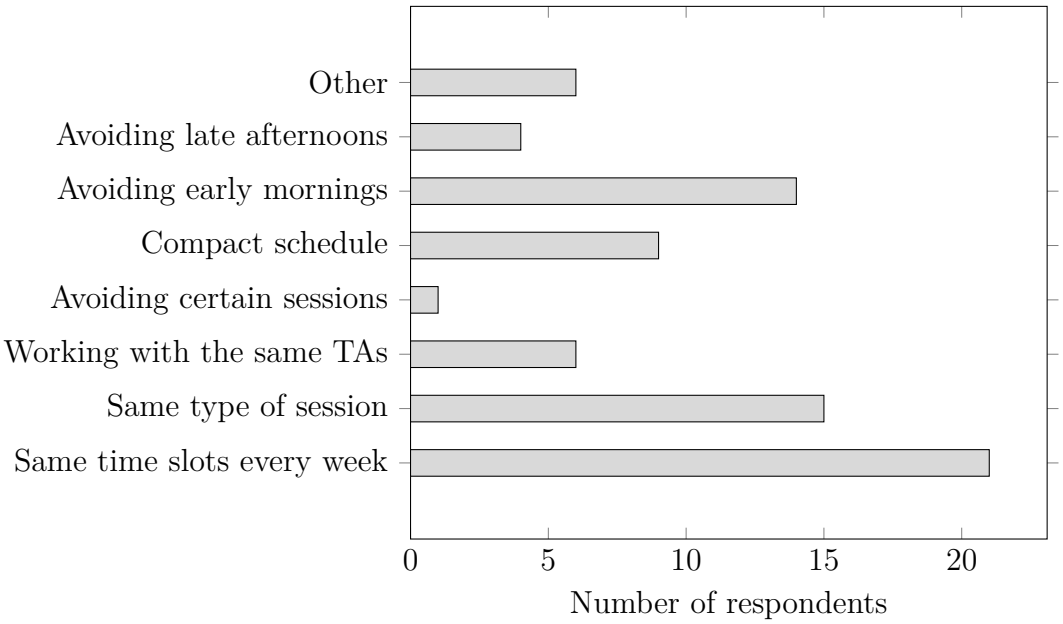


Figure A.7: Types of scheduling preferences expressed by respondents (multiple selections allowed).

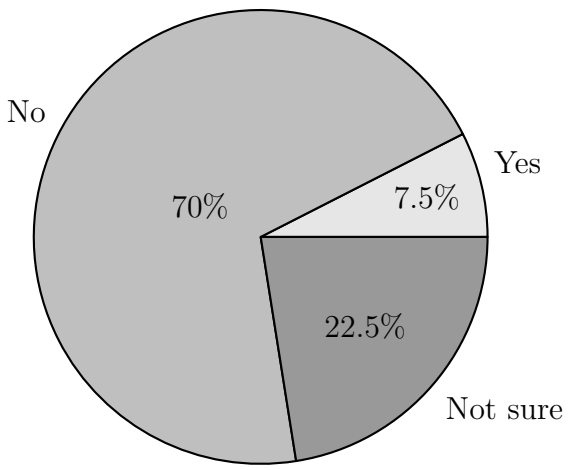


Figure A.8: Distribution of responses regarding whether respondents have experienced unfair TA scheduling.

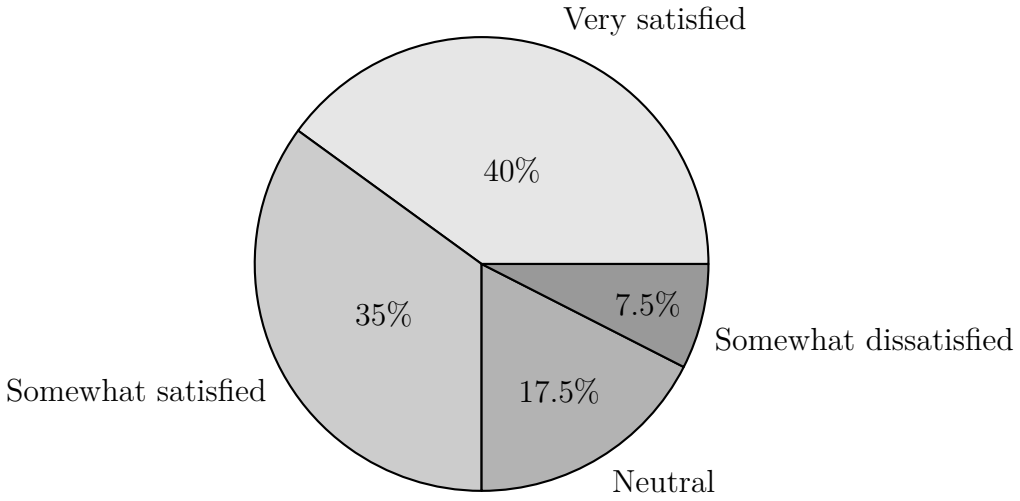


Figure A.9: Respondents' satisfaction with current TA scheduling methods.

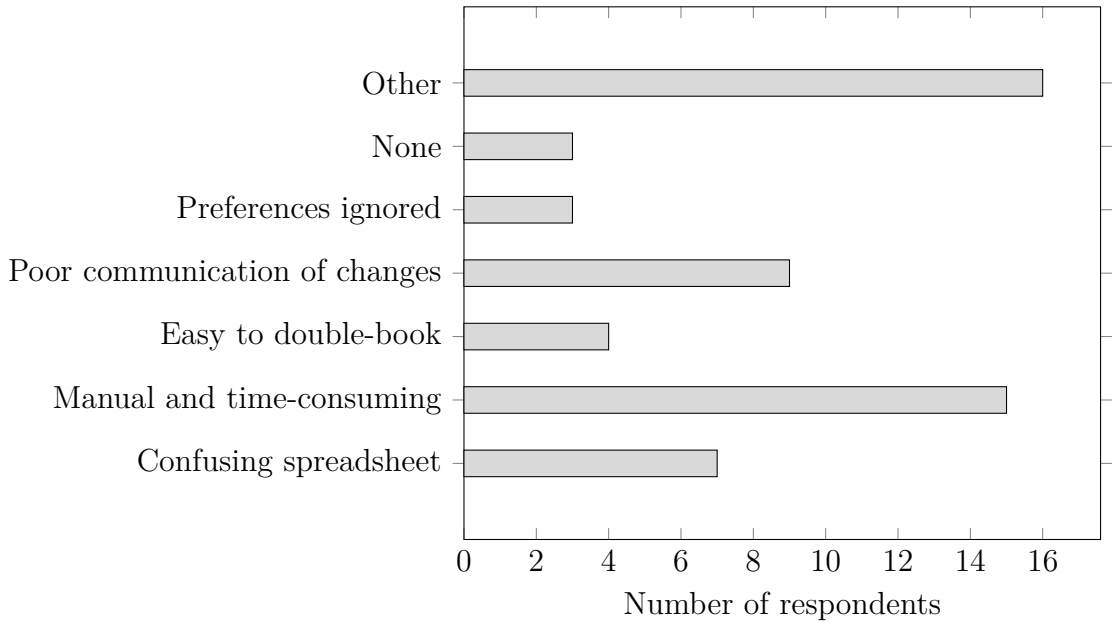


Figure A.10: Reported problems with current TA scheduling tools and methods (multiple selections allowed).

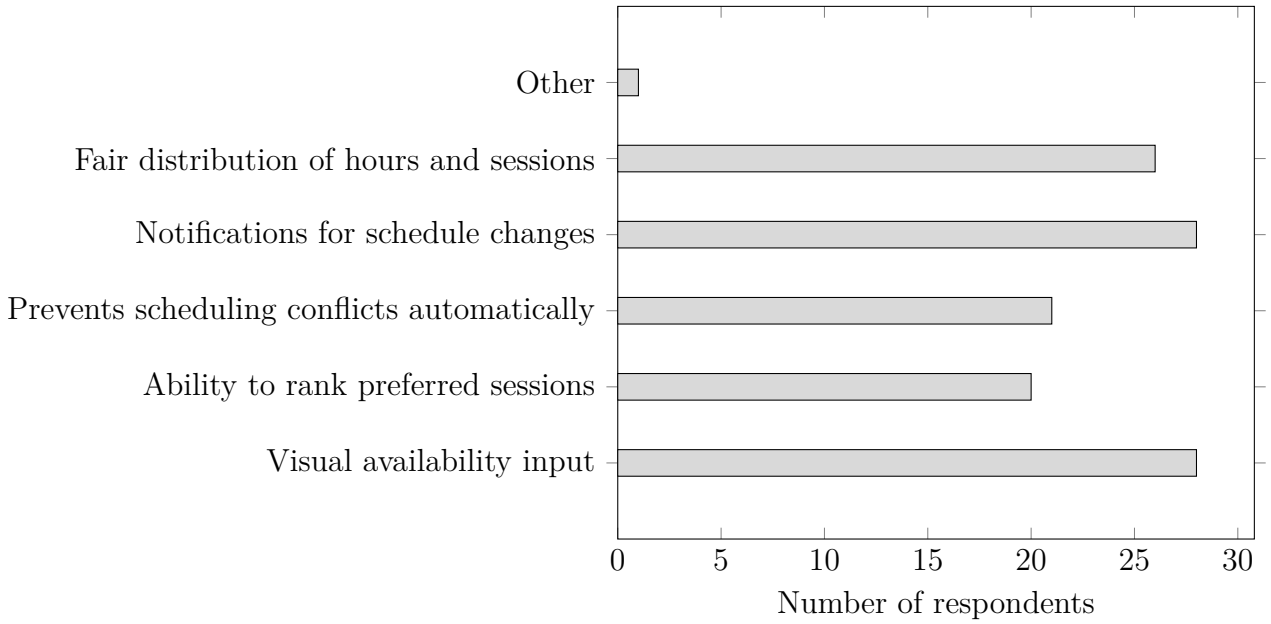


Figure A.11: Features desired in a TA scheduling system (multiple selections allowed).

A.3 Course Responsible Interview Questions

The interview was conducted in Swedish. The questions below are translated into English.

General Questions

1. Can you briefly describe your role in the course and your responsibilities regarding TA scheduling?
2. How is TA scheduling currently carried out in the course?
3. Which tools do you currently use (e.g., Google Sheets, Excel, email)?
4. Which parts of the scheduling process do you find most time-consuming?
5. Are there recurring issues, such as scheduling conflicts, last-minute changes, or misunderstandings?
6. How much manual work is required when creating a new schedule?
7. How are availability and maximum working hours handled today?
8. How do you ensure that a TA is not double-booked?
9. How important is it to distribute the workload fairly among TAs?
10. How do you currently determine what constitutes a fair distribution?
11. How much manual work is required when creating a new schedule?

Questions Related to System Design

1. Is there any functionality that you feel is missing?
2. Is there any part of the system that seems unnecessary or unclear?
3. Do you have any suggestions for how the system could be improved?

4. Is there anything that would make the system more useful for you?
5. Would you consider using such a system in practice? Why or why not?

A.4 Teaching Assistant Interview Questions

The interview was conducted in English. The following is a summary of the interview structure.

A.4.1 Introduction

Participants were asked to perform a series of tasks in the system while thinking aloud. Following each task, the participants answered questions about their experience. All responses were collected anonymously. Participants were provided with a test account representing a Teaching Assistant invited to a course.

A.4.2 Tasks and Questions

A.4.2.1 Task 1

Log into the system and accept the only course invitation you got.

Questions:

1. How easy or difficult was it to find and accept the course invitation?
2. Was the process intuitive?
3. Would you change anything?

A.4.2.2 Task 2

Navigate to the constraints section and add two hard constraints.

Questions:

1. How easy was it to find where to input constraints?
2. Was it easy to view existing constraints?

A.4.2.3 Task 3

Add two soft constraints and configure a compact schedule. Set “Laboration” as the highest preference.

Questions:

1. How easy was it to input soft constraints?
2. How did you interpret the difference between hard and soft constraints?
3. Was the process intuitive?

A.4.2.4 Task 4

The allocation algorithm will ensure that your hard constraints are satisfied, but may not satisfy your soft constraints.

Question:

1. How unsatisfying would it be if many of your soft constraints were not satisfied in your schedule compared to other TAs?

A.4.2.5 Post-interview

After the interview, the participants were asked open-ended questions.

Question:

1. Was there anything missing in the system? What would you improve?
2. What aspects of the system did you find intuitive or difficult?
3. What did you find confusing, if anything?
4. Would you prefer this system over current TA scheduling methods? Why or why not?
5. Do you have any additional comments?

A.5 Summary of Teaching Assistant Interview Results

The answers to the questions were provided in English by the participants, below is a summary of each participant's answers.

A.5.1 TA Participant 1

- Found the overall interface simple and intuitive.
- Reported difficulty understanding the meaning of hard and soft constraints.
- Expressed that unsatisfied soft constraints were acceptable.
- Suggested notifications for updated constraints and support for profile pictures.
- Requested the ability to export schedules as ICS calendar files.

A.5.2 TA Participant 2

- Suggested clearer notifications for course invitations.
- Initially found the distinction between hard and soft constraints confusing.
- Requested improved error handling and more descriptive validation feedback.
- Suggested usability improvements such as hover indicators and easier access to adding constraints.
- Expressed that fairness between TAs strongly affected satisfaction with generated schedules.
- Valued the standardized structure compared to existing scheduling approaches.

A.5.3 TA Participant 3

- Found the interface easy to navigate due to similarities with Canvas.
- Mentioned possible confusion between accepting invitations and joining courses.
- Suggested clearer confirmation when constraints are saved.
- Expressed concerns about fairness if soft constraints were ignored unevenly.

- Preferred the system over current scheduling methods because existing approaches vary between courses.
- Suggested limiting some views to Course Responsibles.

A.5.4 TA Participant 4

- Initially searched for constraints under the course page.
- Suggested adding tooltips/explanations for concepts such as “compact schedule”.
- Did not immediately understand the difference between hard and soft constraints.
- Considered fairness between TAs important when evaluating schedule satisfaction.
- Suggested investigating adversarial constraint input.
- Mentioned possible Canvas integration.
- Preferred the system over current methods if fairness could be ensured.