Privacy-Preserved Meeting Organization

M.M.A.S.T. Akmeemana
A.I. Vidanage
K.P.G.K. Jayathilake

2025

Privacy-Preserved Meeting Organization

M.M.A.S.T. Akmeemana
Index No: 20020015
A.I. Vidanage
Index No: 20021089
K.P.G.K. Jayathilake
Index No: 20020521

Supervisor : Dr. C.I. Keppitiyagama
Co-supervisor : Mr. Tharindu Wijethilake

May 2025

Submitted in partial fulfillment of the requirements
of the
B.Sc. (Honours) Bachelor of Science in Information
Systems Final Year Project

Declaration

We, M.M.A.S.T. Akmeemana (2020/IS/001), A.I. Vidanage (2020/IS/108), K.P.G.K. Jayathilake (2020/IS/052) certify that this dissertation does not incorporate, without acknowledgement, any material previously submitted for a degree or diploma in any university and to the best of our knowledge and belief, it does not contain any material previously published or written by another person or ourselves except where due reference is made in the text. We also hereby give consent for our dissertation, if accepted, be made available for photocopying and for interlibrary loans, and for the title and abstract to be made available to outside organizations.

[illegible]

....
Signature of Candidate   Date : 20/06/2025

[illegible]

....
Signature of Candidate   Date : 20/06/2025

[illegible]

....
Signature of Candidate   Date : 20/06/2025

I, Dr. C.I.Keppitiyagama, certify that I supervised this dissertation entitled Privacy-Preserved Meeting Organization conducted by M.M.A.S.T. Akmeemana, A.I. Vidanage, K.P.G.K. Jayathilake in partial fulfillment of the requirements for the degree of Bachelor of Science Honours in Information Systems.

[illegible]

. . . .
Signature of Supervisor Date : 20/06/2025

I, Mr. Tharindu Wijethilaka, certify that I supervised this dissertation entitled Privacy-Preserved Meeting Organization conducted by M.M.A.S.T. Akmeemana, A.I. Vidanage, K.P.G.K. Jayathilake in partial fulfillment of the requirements for the degree of Bachelor of Science Honours in Information Systems.

. . . .
Signature of Co-Supervisor Date : 20/06/2025

Abstract

The modern organizations increasingly depend on virtual and hybrid meetings to coordinate remote and distributed teams. While these meeting formats offer flexibility and accessibility, they also introduce significant meeting privacy challenges specially when sensitive documents and diverse participant roles are involved. Ad-hoc scheduling practices often result in unauthorized access to confidential content and the selection of inappropriate meeting modes, thereby compromising meeting privacy. This study investigates whether enforcing document-based access control policies (Access Control Lists or ACLs) and considering participant locations can lead to a more privacy-preserving meeting scheduling process. The core hypothesis assumes that meeting participant selection should be governed by the ACLs of documents to be discussed, and that the meeting mode—onsite, online, or hybrid—should be determined based on participants' joining locations to mitigate privacy risks. To address this, we formulate a scheduling problem that integrates these constraints and analyze its computational complexity. A prototype system was developed using widely adopted tools like Google Calendar and Google Drive to demonstrate the feasibility of the proposed approach. Experimental analysis confirms that the problem can be solved in polynomial time using constraint-checking and filtering algorithms, validating the practicality of the hypothesis. The study offers both theoretical and practical contributions: a novel, privacy-aware scheduling framework and an efficient algorithmic approach to enforce access-based participant selection and context-aware meeting modes.

Acknowledgement

First and foremost, we would like to express our deepest gratitude to our supervisors, Dr. Chamath Keeppetiyagama and Mr. Tharindu Wijethilake. Despite their busy schedules, they generously dedicated time to discuss ideas and provide insightful feedback that kept us aligned with our research objectives. Their expert guidance, constructive critiques, and tremendous support were instrumental in shaping this work.

Our thanks extend to the faculty and staff of the University of Colombo School of Computing, whose encouragement made our research possible. We also appreciate the discussions shared with our colleagues and peers which greatly enriched our understanding of the research.

Finally, our deepest appreciation goes to our parents, siblings, and close friends, for their encouragement, patience, and moral support throughout this journey. Their faith in our abilities and constant motivation sustained us through the challenges of research and writing. Our heartfelt thanks go to all who have contributed in ways both large and small to the successful completion of this research.

Table of Contents

List of Figures

List of Tables

List of Acronyms

ACL Access Control List

API Application Programming Interface

CP Constraint Programming

DCOP Distributed Constraint Optimization Problem

DVCSP Distributed Valued Constraint Satisfaction Problem

GA Genetic algorithms

LP Lenear Programming

MaxSAT Maximum Satisfiability

MIP Mixed Integer Programming

ML Machine Learning

MULBS MUltiple Local Bounded Search

NP Non-deterministic Polynomial

SPB Soft Conflict Pseudo Boolean

TOC Theory of Constraints

Chapter 1

Introduction

1.1 Background

When meetings are organized, it must be well planned ahead of what is being discussed and what needs to be achieved at the end of the meetings. If the right participants are not included or if inappropriate participants are included in the meeting, that can lead to privacy concerns and even to potential breaches of confidential information.

Before the COVID-19 pandemic, meeting security and privacy concerns were relatively minimal, as most discussions took place in controlled, onsite environments. But the pandemic reshaped how organizations conduct meetings increasing the reliance on online and hybrid formats. Since 2020, the adoption of virtual meeting platforms has raised to accommodate remote work and collaboration [1]. While this shift has improved accessibility and flexibility, it has also introduced significant privacy challenges.

One of the major problems is in allowing only authorized participants to attend the meeting. Human cognitive biases and organizational pressures can lead to oversights in meeting privacy, which can result in financial losses, reputational damage, and unauthorized disclosure of sensitive information. Selecting the right participants and ensuring that meeting-related documents remain accessible only to authorized individuals are critical aspects of maintaining privacy and security. Existing tools lack the ability to cross-validate document permissions with participant roles and their joining locations. This leads to potential meeting privacy breaches. For instance, if participants are joining from insecure public locations, sensitive discussions may unintentionally become exposed. The inability to determine whether a meeting should be onsite, online, or hybrid based on such parameters increase organizational inefficiencies and privacy risks.

Research on meeting scheduling and efficiency has been conducted from a long time but no attention has been given to privacy-preserving meeting organization. Existing scheduling tools focus on preventing participant overlap or optimizing time slots but fail to consider how document permissions and participant roles influence meeting security.

Our research aims to explore the relationship between meeting privacy factors such as document access, participant roles, and meeting mode and develop a structured framework to ensure meeting privacy. The significance of this research is in its dual contribution. At first, the theoretical understanding of meeting scheduling complexity is studied in-depth, and next, a practical implementation using widely adopted tools is demonstrated. By combining algorithmic analysis with a working prototype, this study provides insights into the computational nature of meeting organization and offers a scalable, privacy-preserving approach to organize meetings efficiently. This thesis contributes to both academic and practical understanding of organizing privacy-preserved meeting organization.

1.2 Problem Statement

Effective and privacy-preserved communication has received a significant concern in the world. As remote work, hybrid teams, and global collaboration have risen, meetings are playing an important role in organizations. The shift to virtual and hybrid environments has created new and unique challenges when ensuring that meetings are privacy-preserved and inclusive. Meeting organizers face difficulties in allowing accessibility to participants while maintaining privacy of the meeting especially when sensitive meetings take place with sensitive documents and diverse participant roles. This brings our attention to the need for a better framework that can enhance the meeting scheduling process without compromising meeting privacy and efficiency.

When multiple topics are to be discussed in a meeting, it is challenging to determine the right participants for the meeting, and meeting organizers can include the wrong participants in the meeting. This can lead to two primary meeting privacy risks which we will be addressing when developing our framework:

Unauthorized Access to Confidential Information: When the meeting organizer has invited unauthorized participants to the meeting, they may gain access to restricted documents and be exposed to sensitive discussions that they are not authorized to view or hear. Here we can see the need for a systematic approach to invite eligible participants or participants with access rights to the meeting.

Privacy-Violating Meeting Modes: We often see how meetings are organized by meeting organizers based on their convenience or on the convenience of people in the upper hierarchy. This can lead sensitive discussions to be exposed to unauthorized individuals. The meeting mode decision-making process should be systematic based on the participants' locations and the sensitivity of the meeting agenda.

Based on the above privacy concerns, a need for a more structured approach to meeting organization can be identified as a crucial need.

In this research, we explore the privacy-preserved meeting scheduling problem by formulating it around a central assumption: meeting participant selection must be governed by the access control lists (ACLs) of the documents associated with the meeting agenda. Once the agenda is created, it becomes essential to validate each proposed participant against the role-based access permissions defined for those documents. Only individuals with the appropriate access rights should be allowed to participate in the meeting.

In addition to verifying access eligibility, the physical or virtual location from which a participant intends to join must also be evaluated. This ensures that the confidentiality of the meeting is not compromised due to exposure in insecure environments, especially when sensitive information is involved. For instance, a participant joining from a public or shared location may not meet the privacy requirements, even if they hold the correct access permissions.

If any participant fails to satisfy either the access control requirements or the contextual privacy constraints related to their location, the meeting, in its originally proposed configuration, cannot be scheduled. This framing acknowledges the multidimensional nature of privacy in meetings and drives the need for a policy-aware scheduling mechanism that can assess both access and other factors during scheduling.

The research problem is therefore to determine the algorithms that schedule meetings while validating the participants and taking privacy-aware mode into consideration.

1.3 Research Questions

Reseach question: What is the algorithm to schedule the meeting under the hypothesis ”Meeting participant selection is primarily dominated by the access control lists of the documents presented including meeting agenda, and the choice of meeting mode depends on the participants’ locations.”?

1.4 Goals and Objectives

1.4.1 Goals

- Resolving constraints associated with privacy-preserved meeting scheduling.
- Deciding the possibility of a privacy-preserved meeting to ensure the privacy of meeting content discussed, concerning the selected meeting participants.
- Selecting a privacy-preserving meeting mode, for conducting the particular meeting.
- Evaluating the solution in an acceptable, credible approach for ensuring its validity

1.4.2 Objectives

- Resolving constraints associated with privacy-preserved meeting organization.
- Deciding the possibility of a privacy-preserved meeting to ensure the privacy of meeting content discussed, concerning the selected meeting participants.
- Selecting a privacy-preserved meeting mode, based on availability and locations of selected meeting participants.

1.5 Research Approach

This research aims to investigate the type of the problem of this specific meeting scheduling variant—the privacy-preserved meeting scheduling problem under constraints, with the objective of determining its classification within the standard complexity classes—specifically, whether it falls under NP-complete, NP-hard, or P. The study uses the literature survey to initially begin the research and then move towards mapping the problem against NP-complete problems like MaxSAT, Knapsack, Subset Sum, and Circuit-SAT. The Boolean circuit approach helps to represent the logic involved in determining feasible meeting configurations. The research reveals that the problem no longer retains NP-completeness or NP-hardness, and can be classified under class P. The research is grounded in formal problem analysis, algorithmic modeling, and computational complexity theory, with practical verification through Python-based prototypes and real-world integration with Google APIs.

1.6 Scope and Assumptions

1.6.1 Scope

In scope

This research is focused on scheduling meetings under privacy preserved meeting settings. The following are within the scope of this study:

- Complexity Analysis: Identifying whether the meeting scheduling problem falls under known complexity classes such as NP-complete, NP-hard, or P, by comparing it with well-established NP-complete problems.
- Problem Definition and Mapping: Formally defining the scheduling problem with precise constraints such as participant availability, group dependencies, and quorum requirements, and attempting to map it onto logical and mathematical constructs like Boolean circuits.
- Boolean Circuit Modeling: Designing algorithms to model the scheduling problem as a Boolean circuit, enabling the breakdown of the problem into verifiable sub-problems for computational tractability.

- Algorithm Development: Constructing algorithms for both Boolean circuit generation and a polynomial-time solvable version of the problem to explore alternatives to exponential-time approaches.
- Prototype Implementation: Developing Python-based proofs of concept to demonstrate both theoretical mappings and practical feasibility, including the handling of group constraints and quorum conditions.
- Tool Integration: Extending the algorithmic solution into a usable tool by integrating it with existing technologies, specifically Google APIs, to validate the real-world applicability of the solution in organizational meeting setups.

1.6.2 Assumptions

The following assumptions are made in the context of this research:

- Availability data is known and static during scheduling: It is assumed that all participant availability information is known at the time of scheduling and does not change dynamically during the scheduling process.
- The locations of the participants are limited to only three known places—Remote Public, Remote Private, and Onsite: The exact geographical location is not considered when considering the location
- Pre-meeting requirement: We assume that documents containing the agenda, and information to be discussed in the meeting will be created with different access control levels.

1.6.3 Hypothesis

Meeting participant selection is primarily dominated by the access control lists of the documents presented including meeting agenda, and the choice of meeting mode depends on the participants' locations.

1.7 Contribution

This research addresses a common and critical challenge in the field of information systems: safe and effective meeting organization, especially when handling sensitive or confidential information. By identifying the nature of the privacy-preserved meeting scheduling problem, this study identifies efficient and secure methods for automating meeting arrangements—an increasingly essential task in both virtual and hybrid work environments. The research helps to reduce the cognitive load and manual work associated with scheduling, offering a framework that aids organizers in selecting the most suitable meeting mode based on the participants' locations, access control level and the sensitivity of the shared information. The outcomes contribute to society in several impactful ways:

- Enhanced Meeting Security: In a world of increasing data protection needs, the privacy-preserving nature of this approach ensures that critical information is protected during meetings and participation, strengthening trust in digital communication.
- Support for Remote and Hybrid Work: As remote work becomes a central feature of modern professional life, this work provides a tool that enhances the efficiency and safety of digital collaboration, supporting flexible work practices.
- Accessibility for Non-Experts: The approach is designed to assist individuals with limited technical expertise by automating complex decisions related to participant categorization and scheduling constraints.
- Efficiency and Productivity Gains: By classifying participants based on access levels and information sensitivity, this solution ensures optimal use of time and resources, minimizing unnecessary communication and maximizing focused collaboration. This can lead to increased organizational productivity and job satisfaction.
- Feasibility in Real-World Settings: The integrated implementation using Google APIs demonstrates the practicality of applying this approach within the existing tools and technologies commonly used in workplaces today.

In essence, this research bridges a theoretical understanding of complexity with a practical solution to an ever-present coordination problem. It enhances the field of information systems by introducing a novel, privacy-aware approach to scheduling that aligns with both privacy needs and usability needs—making it a valuable contribution to both academia and society at large.

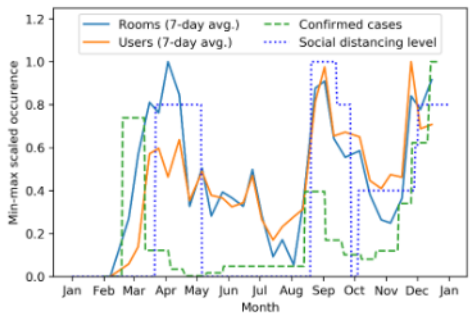
Chapter 2

Literature Review

2.1 Introduction

Meetings are vital for any organization as they enable collaboration, decision-making, and the exchange of crucial information. However, to ensure their success, meetings must be carefully planned and organized. This involves addressing key questions: Do we truly need to hold a meeting? Who should be invited? And what is the best format for the meeting — online, onsite, or hybrid ?

The Changing Landscape of Meetings: Before the COVID-19 pandemic, discussions about meeting security and privacy were less prevalent, as most meetings were conducted onsite with the physical presence of attendees. The pandemic has significantly altered this landscape, expanding meeting modes to include online and hybrid formats, in addition to traditional onsite meetings. Since 2020, the usage of online meeting platforms has surged due to quarantine restrictions and the need for remote collaboration [1].



(a)

Figure 2.1: Change in meeting landscape with quarantine restrictions, [1]

Security and Privacy Concerns: The shift towards online and hybrid meetings has brought security and privacy issues to the forefront. In the Information Technology sector, many participants possess a sound understanding of information security [3]. However, even knowledgeable individuals can make errors, especially as the complexity of the scenarios increases [4]. Human decision-making is often influenced by emotions and biases, which can lead to mistakes. This is equally applicable in the context of organizing meetings, where improper use of online platforms can result in financial losses, reputational losses, and losses of proprietary information, capital and corporate value.

Selecting the Right Participants: One of the most critical aspects of organizing a meeting is inviting the right participants. Although this may seem straightforward, it is often a rushed or overlooked step. The success and productivity of a meeting largely depend on having the appropriate individuals present, whether in person or virtually. Selecting the right attendees is crucial because their background and capabilities can influence the meeting's mode and effectiveness. Moreover, having the right participants is essential for maintaining the confidentiality of the information shared during the meeting. When only the necessary and relevant individuals are present, the risk of information leakage is minimized, and sensitive discussions can be conducted securely. Cognitive and organizational bias undermines good decision-making. To ensure a successful and secure meeting, consider the following principles for selecting participants:

1. Need: Identify who is essential based on their relevance to the meeting's topic and their ability to contribute significantly to the discussions. These are the individuals whose presence is crucial for the meeting's success and whose involvement ensures that confidentiality is preserved.

2. Want: Differentiate between those whose presence is desired but not critical. While these individuals might add value, they are not essential unless they can provide meaningful contributions to the meeting's agenda and help to maintain the meeting's confidentiality.

3. Value: Assess the potential value each attendee brings. The goal is to ensure that all participants can contribute to achieving the desired outcomes, whether that involves making decisions, offering insights, or advancing solutions. Their involvement should also align with the need to safeguard the information discussed.

By carefully selecting participants based on need, want, and value, you align the attendees with the meeting's goals, ensuring that each meeting is productive and purposeful. Clear objectives will guide you in choosing the right people, leading to more effective and efficient meetings. Moreover, this thoughtful selection process enhances the confidentiality of the information shared, ensuring that sensitive topics remain secure and are only accessible to those who are truly relevant. Accordingly, during our research project, we will explore whether it is possible to identify a relationship among privacy-related factors of a meeting including participants, capabilities of participants, documents, and data shared in the meeting and meeting mode. After identifying such a relationship, we expect to evaluate conformance of the relationship with real world scenarios accurately. Then future researchers will be able to use our findings with confidence, to proceed further in our research path associated with meeting security and privacy.

2.2 Background of literature survey

Meetings are an integral part of organizational operations, serving as platforms for collaboration, decision-making, and the exchange of essential information. Traditionally, most meetings were conducted onsite, with participants physically present in a designated location. However, the COVID-19 pandemic has transformed the landscape of how meetings are held, leading to a significant rise in online and hybrid meeting formats. This shift has brought new challenges and considerations, particularly concerning the security and confidentiality of information shared during meetings.

Prior to the pandemic, meetings were mainly held in person, and security concerns were largely physical — ensuring that meeting rooms were secure and that unauthorized persons were not present. The rapid adoption of online meeting platforms since 2020 has necessitated a re-evaluation of these security paradigms [1]. The ”new normal” of remote work has highlighted the vulnerabilities inherent in digital communications, with security breaches and privacy issues becoming more frequent and complex [5].

The increase in remote work and the reliance on online meeting platforms have led to a spike in reported security incidents [1][6]. These platforms, while convenient, have been found to have various security flaws, ranging from phishing attacks to unauthorized access and data breaches [5]. Early in the pandemic, many popular meeting platforms faced inspection over their inadequate security measures, which were quickly exploited as their usage increased [7]. Although improvements have been made, the challenge of securing online meetings remains prominent. However, irrespective of the meeting mode, any opportunities given to the wrong attendees for the meeting can give room for the insecurity of the meeting and the attack surface can be increased.

Research into the security issues associated with the ”Work From Home” culture has shown a marked increase in vulnerabilities, particularly in digital communication tools. Studies indicate that phishing and other cyber attacks have targeted the expanded digital workspace, exploiting both technological and human weaknesses [8]. Furthermore, while the security of online meetings has received significant attention, there is a noticeable lack of comprehensive strategies addressing the secure organization and execution of meetings based on the sensitivity of the information discussed and the selection of participants.

A critical aspect of meeting organization is the selection of participants. Ensuring that the right people are invited to a meeting is not only a matter of operational efficiency but also of maintaining the confidentiality and security of the information shared. Inappropriate or unnecessary attendees can increase the risk of information leakage and compromise the meeting's objectives. Therefore, a systematic approach to selecting participants, based on their relevance and the nature of the information being discussed, is essential for maintaining security [9].

2.3 Literature survey and identified research gap

2.3.1 Introduction to literature survey

Meeting scheduling becomes increasingly complex when incorporating privacy controls to restrict meeting access to authorized participants only. While many scheduling models address efficiency and availability, fewer approaches ensure access is controlled to prevent unauthorized participants from joining. This type of access-controlled privacy preservation is critical in sensitive or confidential meeting environments, such as corporate or client meetings, where unauthorized access could lead to security breaches. Despite notable advances, the field lacks a comprehensive, privacy-preserved scheduling model that fully addresses these needs. This literature review critically examines existing approaches, compares their methodologies and limitations, and identifies the gap in privacy-preserved meeting scheduling.

Meeting Scheduling Challenges

Meeting scheduling has so many challenges. Many researchers have conducted research to minimize the bottlenecks that arise during scheduling meetings. There are certain meetings which integrate access controls and that require decentralized management of permissions, which can be achieved through distributed models like Distributed Valued Constraint Satisfaction Problem (DVCSP) and Distributed Constraint Optimization Problem (DCOP). Reference [10] reveals that these models have agents that independently manage access for every participant while minimizing data sharing and reducing the risk of unauthorized access.

According to [10], both DVCSP and DCOP provide semi-optimal solutions with limited data exchange between agents. This makes them suitable for environments where meeting access needs to be restricted.

Although DVCSP allows flexible constraint relaxation, it may not work in instances where strict participant authorization is necessary. But, according to [11], there are algorithms within DCOP that provide a more secure way by focusing on minimal communication. This reduces the risk of unauthorized access through limited information exposure. DVCSP's main advantage is its flexibility which allows agents to relax lower-priority constraints to achieve a balance between ideal scheduling and participant privacy. Contrary to that DCOP uses independent agents to manage individual schedules with minimal data sharing, achieving semi-optimal solutions with lower computational costs. In reference

[11], researchers have found MULBS, a specific DCOP algorithm that emphasizes minimal communication requirements and privacy preservation, making it highly suitable for privacy-sensitive environments.

Another distinct approach comes from [12], which frames the scheduling problem within computational geometry, focusing on optimizing both time and spatial constraints through algorithms with O(n log n) complexity [12]. While DVCSP and DCOP both excel in preserving privacy through decentralized scheduling, they differ significantly in their computational approaches and the quality of solutions. DVCSP's semi-optimal solutions are sufficient for low-density environments but may be inadequate in high-density scheduling contexts where optimization is critical. The findings of [11] that led to the discovery of the MULBS algorithm achieve greater computational efficiency by minimizing communication, yet it also sacrifices some scheduling accuracy due to its semi-optimal focus. Computational geometry-based model in [12] achieves excellent efficiency for time-space scheduling, but does not address privacy directly, making it limited in privacy-sensitive scheduling environments.

Constraint-Based Optimization Techniques

Meetings can be arranged by considering different limits. These limits are referred to as Constraints. Many researchers have focused on arranging meetings by enforcing different constraints and thereby Constraint-Based Optimization Techniques have evolved. Constraint-based optimization techniques like Constraint Programming (CP), Mixed Integer Programming (MIP), and Maximum Satisfiability (MaxSAT) have been applied to scheduling. The mainly focused constraints in these researches are around availability, priority, and location. Reference [2] reveals how CP allows for precise encoding of scheduling constraints. But according to [2] research findings it lacks scalability in larger, more dynamic settings. Peteghem and Vanhoucke in 2010 had conducted a research [13] on how MIP could be used for effective scheduling. They found that MIP, with its ability to rapidly identify infeasible solutions, is effective for structured scheduling tasks with rigid constraints but is too inflexible for applications with dynamic privacy needs . Recent advances in MaxSAT, such as the Soft Conflict Pseudo Boolean (SPB) constraints, have introduced adaptive clause weighting strategies to improve local search efficiency in scheduling contexts.

instance	MaxSAT		Chuffed		MIP	
forum-13	2.1	0	1190.4	0	38.9	0
forum-13crafb	1.4	-	7.2	-	0.4	-
forum-13crafc	1.6	-	TO		106.1	-
forum-14	1.6	-	10.8	-	0.4	-
forumt-14	0.2	-	5.5	-	0.1	-
forumt-14crafc	0.2	-	4.6	-	0.1	-
forumt-14crafd	0.2	-	5.0	-	0.1	-
forumt-14crafe	0.2	-	4.5	-	0.2	-
tic-12	0.2	0	2.3	0	0.2	0
tic-12crafc	0.2	0	2.3	0	0.1	0
tic-13	0.3	0	6.1	0	2.8	0
tic-13crafb	0.3	0	5.8	0	0.2	0
tic-13crafc	0.2	-	4.2	-	1.1	-
tic-14crafa	0.6	0	5.6	0	0.6	0
tic-14crafc	0.6	0	4.5	0	2.6	0
tic-14crafd	0.3	0	5.5	0	0.8	0
ticf-13crafa	2.3	0	3595.5	0	38.4	0
ticf-13crafb	1.5	-	8.4	-	0.6	-
ticf-13crafc	2.2	-	TO		914.8	-
ticf-14crafa	6.3	0	67.2	0	3736.4	0
#solved	20		18		20	

Table 2.1: Results of MaxSAT, CP and MIP approaches in aspect of time spent in seconds [2]

In 2024, reference [14] revealed that SPB-MaxSAT algorithm, specifically, provides a high-performing solution for complex scheduling problems by adjusting clause weights to satisfy "soft" constraints, which are more flexible and adaptable. CP and MIP works well in traditional scheduling scenarios, but both fall short in contexts requiring real-time adaptability and privacy prioritization. CP offers flexibility with strict constraints. The drawback is that its lack of scalability makes that makes it unsuitable for larger, privacy-sensitive scheduling environments. MIP provides scalable solutions for structured schedules but lacks the adaptability needed to handle frequent privacy-related adjustments.

In comparison, MaxSAT—particularly the SPB-MaxSAT algorithm—introduces [2] soft constraints that enable adaptability, positioning it as a more flexible alternative to CP and MIP. However, although we see these improvements, MaxSAT lacks privacy-focused mechanisms. It primarily supports flexibility rather than privacy preservation. So in overall, constraint-based techniques each contribute a significant strength to the scheduling field. CP's precise encoding, MIP's efficiency in identifying infeasible constraints, and MaxSAT's adaptive clause weighting all provide valuable solutions to specific scheduling problems. But, the lack of real-time adaptability and privacy-preserving mechanisms across these methods shows their limitations in handling complex, privacy-focused meeting scheduling scenarios. MaxSAT's introduction of soft constraints offers the most promise approach for dynamic scheduling, but further integration with privacy-preservation techniques is essential to address privacy-sensitive needs fully.

In constraint optimization research [11], that have mapped meeting scheduling problem into a set of constraints, a common feature can be observed. These all researchers have identified meeting scheduling problem as an NP-complete problem. An NP-complete problem involves a decision problem and an optimization problem both. NP-complete problem domain is depicted in the figure below, with other related problem domains. In the figure, NP-hard problems involve optimization problems, NP problems involve decision problems and P problems involve polynomial time solvable problems.

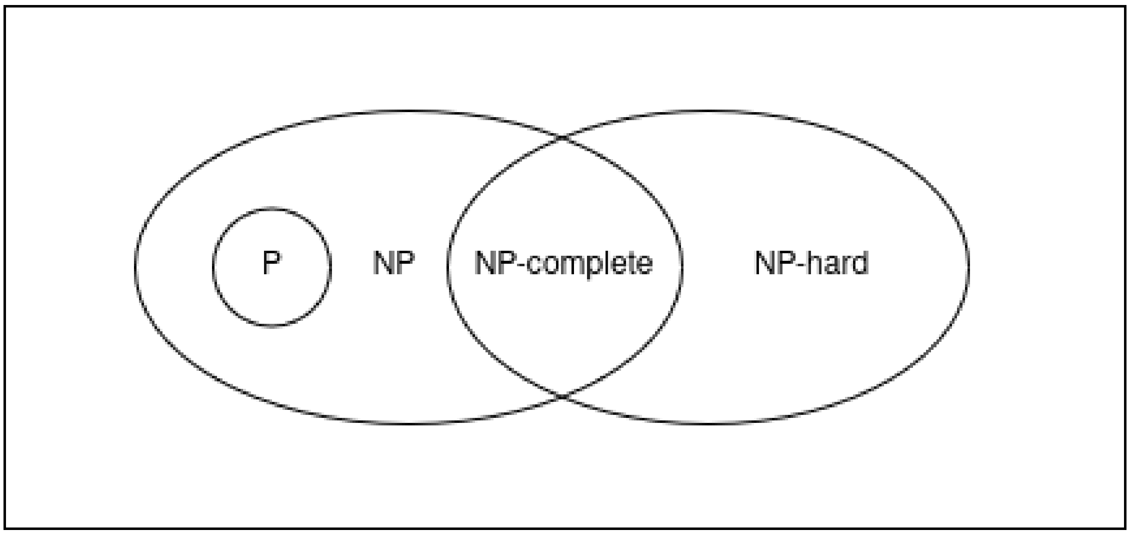


Figure 2.2: NP-complete problem domain, with other related problem domains

Genetic Algorithms and Adaptive Scheduling

Iterative optimization is a main approach in the solving of complex problems, especially in the case of large search spaces, dynamic environments, or time-dependent constraints. Instead of finding a solution in one direct way, iterative methods refine solutions incrementally through repeated improvements. This leads to better accuracy, efficiency, and flexibility. If meeting schedules also can evolve based on changing constraints or resource availability, that would add value to the scheduling process in a greater scale. Genetic algorithms (GAs) [13] introduce adaptability through iterative optimization. This allows schedules to evolve based on changing constraints or resource availability.

According to [13] in 2010, GAs use a bi-population model that supports continuous adjustments. This is advantageous in settings where participant availability and privacy needs change frequently [13]. GAs are more adaptable than constraint-based methods like CP and MIP. This is due to their iterative nature and ability to consider multiple solutions. This adaptability makes GAs suitable for environments with dynamic constraints, where immediate schedule adjustments are required. Although there are several advantages, GAs' computational intensity creates a challenge for real time applications. Specially in privacy-preserved contexts where participant selection may require immediate reallocation of timeslots based on privacy needs.

Compared to MaxSAT's adaptive clause weighting, GAs offer greater flexibility but are less efficient computationally. This requires careful tuning of parameters to achieve optimal performance. The bi-population structure of GAs provides an advantage in dynamic environments. GAs offer an adaptable alternative to more rigid constraint-based models. But GAs are obstructed by computational demands, which can limit their real-time responsiveness in privacy-preserved meeting scheduling. This indicates a need for hybrid approaches that incorporate GA's adaptability with computational efficiency to better work with privacy-preserved meeting scheduling needs.

Theory of Constraints (TOC) and Bottleneck Management

Eliyahu M. Goldratt developed a management and problem-solving approach known as the Theory of Constraints (TOC) in the 1980s [15]. This method focuses on maximizing the output by managing the constraints. Research has been conducted on how this concept can be applied in scheduling. The Theory of Constraints (TOC) focuses on identifying and reducing bottlenecks within scheduling systems, improving throughput in resource-constrained contexts.

According to [16], TOC is valuable in production-oriented scheduling, Here, specific bottlenecks are often seen as restricting scheduling efficiency and resource allocation. TOC is effective in managing resources and prioritizing critical constraints. But it does not incorporate privacy considerations or adaptability for dynamic participant needs. When we compare models like DVCSP and DCOP, they offer finer and better scheduling management because there, the main focus is on privacy-preserving through decentralized data management. TOC targets throughput and bottleneck reduction. But its limited adaptability and lack of privacy considerations restrict its effectiveness in privacy-preserved meeting scheduling contexts.

TOC's bottleneck prioritization method is beneficial for contexts where resource constraints drive scheduling but falls short in privacy-preserving meeting contexts. As TOC mainly focus on throughput limit, its applicability to meeting scheduling is limited, which often requires flexible participant selection based on confidentiality needs [16]. TOC's ability in optimizing resource allocation could be enhanced with additional privacy and adaptability mechanisms to better address modern scheduling requirements.

Machine Learning and Hybrid Approaches in Constraint Solving

By employing supervised learning to forecast participant preferences and optimize scheduling algorithms based on historical data, machine learning (ML) improves adaptive scheduling. By combining machine learning (ML) and conventional SAT solvers, hybrid models like Cube-and-Conquer increase scalability and efficiency in challenging scheduling tasks [17][13]. ML is a useful supplement to adaptive scheduling because of its capacity to forecast participant preferences, providing more flexibility than CP, MIP, and TOC. Unlike previous models, machine learning (ML) allows for real-time adaptability to changing situations, which is ideal for meeting scheduling that protects privacy. However, the lack of established privacy measures and evaluation tools limits ML's ability to properly handle confidentiality alongside scheduling, limiting its usefulness in privacy-sensitive scenarios. Through predictive capabilities, machine learning and hybrid models offer substantial potential for increasing schedule flexibility.

However, machine learning's current function in privacy-preserved scheduling is restricted due to the absence of privacy-specific applications and consistent measurements. Creating uniform assessment criteria and adding privacy safeguards might improve ML's applicability in scheduling situations where privacy is a concern.

Privacy-Centric Distributed Models for Scheduling

Distributed methods like DCOP and DVCSP, which decentralize data management to safeguard participant confidentiality, have advanced participant privacy-preserving scheduling. In order to preserve privacy, the MULBS algorithm introduced in [10] minimizes communication between agents, which is an example of the DCOP technique [11]. LP-type (Linear Programming) problems and graph models are used in [12], where computational geometry-based method is used to solve scheduling efficiently. However, privacy-preserving features are absent there as well. In contrast to centralized systems that are more vulnerable to data exposure, the DCOP and DVCSP models prioritize privacy through minimum data interchange. While time-space optimization techniques in [12] are efficient but do not explicitly handle privacy, MULBS's low communication requirements make it ideal for privacy-sensitive contexts.

These discussed approaches provide a balanced solution for privacy-sensitive contexts when compared to alternative scheduling models, however because of the trade-off with computational performance, they only produce semi-optimal results. Effective privacy protection is provided by distributed models, although at the expense of some optimization. Although the decentralized approach of DCOP and DVCSP better meets privacy concerns than centralized approaches, it is not able to provide fully optimum solutions. Models that can strike a compromise between privacy, efficiency, and scheduling quality are necessary because there is still a gap in attaining optimal scheduling without sacrificing privacy.

2.3.2 Research Gap

Despite significant advancements, privacy-preserving meeting scheduling remains largely unaddressed. Constraint-based methods excel in managing structured constraints but lack real-time adaptability and privacy-preserving mechanisms. Adaptive scheduling techniques, such as GAs and TOC, offer flexibility but fall short of the privacy requirements essential for modern scheduling contexts. Privacy-centric distributed models partially address privacy needs but achieve only semi-optimal outcomes, highlighting a clear research gap. Future research must focus on developing hybrid scheduling models that integrate constraint-based optimization, machine learning, and distributed frameworks to create secure, responsive, and privacy-focused scheduling solutions. Addressing this gap would advance the field, providing compliant, adaptable scheduling frameworks tailored to modern privacy-sensitive environments.

2.4 Key Findings

From the aforementioned studies we understood how computational geometry [12] was used to optimize scheduling complexity and achieve O(n log n) efficiency and was first shown in early meeting scheduling research. However, this effort did not prioritize privacy considerations. In 2012, DVCSP-based scheduling was examined by Enembreck and André Barthès [11], who emphasized its adaptability in applying constraints but also pointed out its drawbacks in requiring tight participant consent. Reference [10] extended their work by introducing the MULBS algorithm within DCOP, which greatly enhanced privacy by lowering communication exposure. MIP is quite good in optimizing large-scale scheduling issues, according to [13] analysis of its function in structured scheduling. They did point out that it is challenging to adjust to changing privacy requirements. At about the same time, genetic algorithms became more popular due to their versatility; [13] showed how well GAs worked for real-time scheduling. However, their high computing costs made them inapplicable in situations where privacy was a concern.

In their re-visitation of constraint-based optimization, researchers of [2] acknowledged the limitations of scalability of CP while reiterating its strength in directly embedding scheduling restrictions. References [17] and [18] introduced the Cube-and-Conquer technique, which improved scalability while preserving constraint precision, as part of the hybridization of machine learning with SAT solvers. More recently, [14] suggested Soft Conflict Pseudo Boolean (SPB) constraints for MaxSAT, which leverage adaptive clause weighting to increase the efficiency of local searches.

In [11], it emphasized the value of privacy in decentralized scheduling by pointing out how the MULBS algorithm reduces communication exposure in privacy-centric scheduling. But their method needed further work because it was computationally inefficient. According to [12], computational geometry-based approach showed great efficiency but lacked clear privacy safeguards, suggesting a trade-off between speed and confidentiality.

2.5 Limitations, and Conflicts

Despite extensive research, scheduling methodologies exhibit several limitations. In decentralized scheduling, DVCSP and DCOP provide privacy advantages but struggle with computational feasibility. DVCSP performs well in low-density environments but becomes inefficient as participant numbers grow. DCOP, while offering enhanced security, suffers from high communication overhead, reducing its suitability for large-scale scheduling. Constraint-based optimization techniques like CP and MIP excel in structured scheduling but lack flexibility in dynamic scenarios. While MaxSAT's soft constraints improve adaptability, they do not inherently address privacy concerns, requiring additional security mechanisms. Genetic Algorithms, although effective for adaptive scheduling, impose significant computational costs, limiting their practical application in real-time privacy-sensitive scheduling.

TOC's deterministic approach is valuable for bottleneck management but lacks the flexibility to accommodate evolving privacy requirements. Machine Learning, despite its predictive power, faces challenges in standardization and interpretability. ML-based scheduling models often operate as black-box solutions, making it difficult to verify their decision-making processes. Additionally, privacy risks associated with ML require further exploration, particularly in data-sharing scenarios. Privacy-centric distributed models, such as DCOP and DVCSP, enhance confidentiality but struggle with achieving optimal results efficiently. Many privacy-preserving scheduling approaches prioritize security at the expense of computational speed, making them less viable for real-time applications. GeoInformatica, while efficient, does not incorporate privacy mechanisms, highlighting an ongoing trade-off between performance and security.

2.6 How the Literature Review impacts our research

The findings illustrate a fundamental trade-off in scheduling research: optimizing for privacy often comes at the cost of efficiency. Current models either prioritize speed (e.g., GeoInformatica) or focus on privacy (e.g., DCOP, DVCSP), but few successfully integrate both aspects. Traditional constraint-based techniques offer structured scheduling but remain rigid in privacy-sensitive applications. Hybrid approaches that combine constraint-based optimization with adaptive scheduling methods, such as ML-enhanced SAT solvers, show promise but require further refinement to integrate privacy considerations effectively. To bridge these gaps, future research should explore novel frameworks that balance scheduling efficiency with privacy protection, contributing to a more comprehensive and adaptable meeting scheduling system.

Chapter 3

Methodology

3.1 Overview on methodology

The methodology used in this research is grounded in a systematic exploration of the complexity of meeting scheduling problems. The research started with a comprehensive literature survey, which revealed that meeting scheduling is naturally NP-complete, as stated by multiple prior studies. These works highlighted that meeting scheduling problems often involve combinatorial search, agent coordination, and constrained resource allocation, all of which contribute to their classification as NP-complete.

Following this, we proceeded to formally define our specific problem we aimed to solve in this research. This included identifying the relevant entities such as participants, time slots, access controls, and quorum requirements, along with the constraints governing the interaction. To better understand the nature of the problem, we analyzed a set of well-known NP-complete problems such as MaxSAT, Knapsack, Subset Sum, and Circuit-SAT. These problems were selected based on their structural and conceptual similarities to the meeting scheduling domain. Through this comparative analysis, we wanted to determine whether the meeting organizing problem could be heuristically solved by mapping onto one of these NP-complete problems.

Among these, Circuit-SAT showed initial promise due to its formulation as a boolean satisfiability problem. We mapped the meeting scheduling problem onto a boolean circuit representation and developed algorithms capable of generating these boolean circuits based on scheduling inputs and constraints. However, after deeper analysis, we identified that the boolean circuit constructed did not follow the standard formulation of the Circuit-SAT problem. As such, the mapping was insufficient to establish our problem as an NP-complete problem via Circuit-SAT or any other NP-complete problem form. Since the circuit verification process itself is deterministic and verifiable in polynomial time, the problem does not fall under the NP-hard category either.

Within the Boolean circuit, we recognized that all sub-problems except one could be solved in polynomial time. This single problematic sub-problem exhibited exponential time complexity. Therefore we focused our efforts on analyzing whether this segment could also be transformed into a polynomial-time computable procedure. Finally, we concluded that it is not possible to redesign this segment into a polynomial-time circuit using boolean logic alone. However, we discovered that this segment could be resolved using a non-circuit-based polynomial-time algorithm.

This breakthrough led to the significant conclusion that the overall problem is not NP-complete but solvable in polynomial time, classifying it as a P-type problem. To validate our theoretical findings, we implemented proof-of-concept prototypes in Python. We further extended our models to support real-world meeting scenarios as well. Finally, in order to demonstrate the feasibility and real-world applicability of our approach, we developed an integrated tool that utilizes Google APIs. This prototype shows how the proposed polynomial-time algorithm can be practically deployed using existing tools and technologies commonly available in organizational settings.

3.2 Formal definition of entities

3.2.1 Basic definitions

Following finite sets are defined:

- $\mathcal{D}$: The set of all documents.
- $\mathcal{R}$: The set of all roles.
- $\mathcal{I}$: The set of all individuals
- $\mathcal{L}$: The set of all locations.
- $\mathcal{T}$: The set of all time slots.

3.2.2 Access Control List

We define following relationships, using above definitions.

$$d = \{d \in \mathcal{D} \mid \text{d is a document}\}$$

$$i = \{i \in \mathcal{I} \mid i \text{ is an individual } \}$$

$$g = \{g \subseteq \mathcal{I} \mid g \text{ is a subset of one or more individuals in } \mathcal{I}\}$$

Above relationships mean that d is an element of set $\mathcal{D}$, and i is an element of set $\mathcal{I}$. Further, g is a group of one or more individuals (i), where $i \in \mathcal{I}$, such that $g \neq \emptyset$.

Consider that following finite set is also defined:

- $\mathcal{G}$: Set of all possible not-null subsets of $\mathcal{I}$

Based on above all sets, we define following relationship.

$$access(d) = \{g \in \mathcal{G} \mid g \text{ has access to } d\}$$

Above relationship means that g is an element of set $\mathcal{G}$, and that $access(d)$ is the set of groups (g) having access permission to document d. Here we note that, $access(d) = \mathcal{G}$ converts d to a **public document**.

By above last two relationships, since any element g of $access(d)$ is also a subset of $\mathcal{I}$, such that $g \subseteq \mathcal{I}$, we have the relationship $access(d) \subseteq \mathcal{I}$, when $access(d)$ is defined in form of **singleton subsets** of $\mathcal{I}$. It implies also that $|access(d)| \leq |\mathcal{I}|$, when $access(d)$ is defined in the form of singleton subsets of $\mathcal{I}$. Simply, a singleton subset of $\mathcal{I}$ includes an individual (i).

Regarding that inequality, $|access(d)| = |\mathcal{I}|$ is the situation when every i in $\mathcal{I}$ is present in at least one group (g), such that $g \subseteq access(d)$. At such a situation, both relationships $access(d) = \mathcal{G}$ and $|access(d)| = |\mathcal{I}|$ imply the same meaning that, document is a **public** document.

3.2.3 Meeting agenda

Agenda of a meeting is the document that defines the set of groups (g) required to attend the meeting, where **group** has same meaning as defined above. When we consider agenda as document d, those groups (g) are elements of set $access(agenda)$.

Theoretically it is possible to require all individuals of set $\mathcal{I}$ or all available not-null subsets of set $\mathcal{I}$, to attend a single meeting. But, in practical scenario, probability of organizing such a meeting is low.

However, there are both private meetings and public meetings, in our scope. If $access(d) = \mathcal{G}$ is used for meeting agenda of public meetings, it's impossible to distinguish the intended participant groups explicitly. Therefore, in agenda document of public meetings, we include a group labeled as **public** group, in addition to the actual intended participant groups of meeting, to state that agenda is **public**. So, on the other hand, absence of group labeled as **public** in $access(agenda)$ means that, meeting is **private**.

- If there is at least one document in meeting, such that $access(d) \neq \mathcal{G}$, **public** group shouldn't have access to meeting agenda.
- If every documents in meeting has $access(d)$ such that $access(d) = \mathcal{G}$, **public** group can have access to meeting agenda.

- If agenda is the only document in meeting, **public** label can be used by meeting organizer to define whether agenda document is **private** or **public** (i.e. whether meeting is private or public).

Following flow chart depicts the process of identifying whether a document is **private** or **public**.

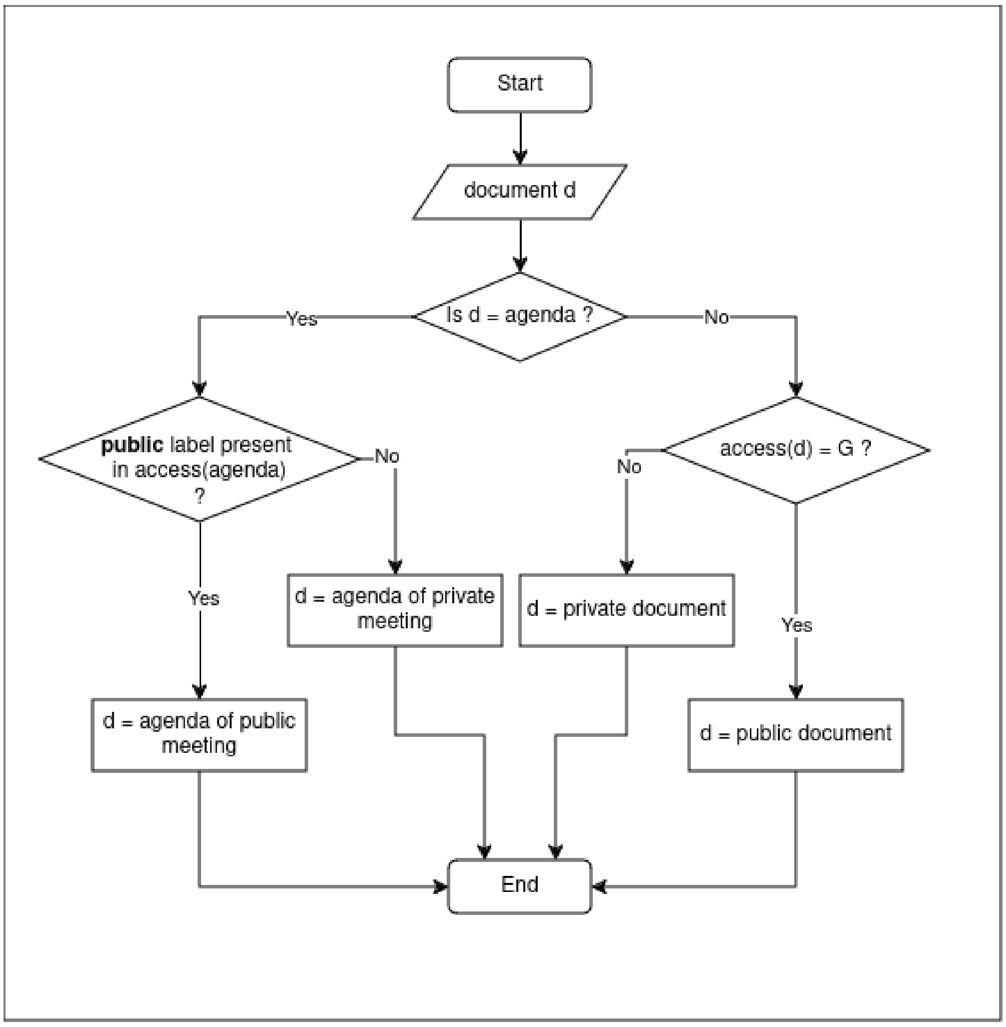


Figure 3.1: Process to identify whether a document is private or public

Regarding other documents except agenda, for simplicity of implementation, we can include only **public** label in access control list, without including any other group of $\mathcal{G}$, to mean that $access(d) = \mathcal{G}$ or that document is a public document. Because, we do not need to include any other group (g) in a public non-agenda document, though we required them in public agenda documents for identifying meeting participants.

In addition to presence or absence of **public** label in $access(agenda)$, meeting agenda should define the **meeting quorum**, for the meeting. This theme will be discussed later at 3.2.12 section.

3.2.4 Definition of a meeting

We assume that every meeting has an agenda associated with it, to define the set of groups(g) required to attend the meeting. Agenda of a particular meeting M is a document, belonging to set $\mathcal{D}$.

When we consider the agenda document of meeting M, for every group g invited to meeting M; $g \in access(agenda)$. Also consider that, D represents set of documents discussed in M, including agenda, such that $D \subseteq \mathcal{D}$. Hence, according to the assumption emphasized above, for any meeting M; $|D| \geq 1$.

For conducting a meeting, at least 2 individuals are required. Consider that I represents the set of individuals attending meeting M, such that $I \subseteq \mathcal{I}$, when groups (g) of $access(agenda)$ are converted to corresponding elementary individuals (i). Here we note that, for any meeting M; $|I| \geq 2$. Accordingly, when $access(agenda)$ is defined in terms of singleton subsets of $\mathcal{G}$, and those groups (singleton subsets) in $access(agenda)$ are represented by G, such that $G \subseteq \mathcal{G}$, it can be observed that $|G| \geq 2$.

Consider set of locations of individuals in M as L (in other words, set of locations of individuals in set I, during meeting time), such that $L \subseteq \mathcal{L}$. Every individual attends meeting from a particular location l, such that $l \in L$. We note that, if meeting is online or hybrid, $|L| > 1$. If meeting is onsite, $|L| = 1$, since every individual is at same location. Every meeting should be in one mode, out of online, hybrid, onsite modes. Therefore, for any meeting M; $|L| \geq 1$.

Since a **meeting** is a **synchronous** communication, every individual in meeting M should attend the meeting during the same time slot t (Assuming that all individuals are in same time zone).

Based on these definitions, we define meeting M as a 4-tuple,

$$M =< D, I, L, t >$$

such that,

$$D \subseteq \mathcal{D}$$

$$L \subseteq \mathcal{L}$$

$$I \subseteq \mathcal{I}$$

$$t \in \mathcal{T}$$

3.2.5 Transformation of individual into role

Consider that same sets defined above will be used in explanations below, in same notations:

Consider g and g' as subsets of $\mathcal{G}$ such that $g, g' \subseteq \mathcal{G}$. And consider d as a **private** document , l as a location and t as a time slot such that $d \in \mathcal{D}$, $l \in \mathcal{L}$ and $t \in \mathcal{T}$. Further consider that $g \in access(d)$ and $g' \notin access(d)$, for restricting access of document d, where $|access(d)| = n$, given that $access(d)$ is defined as a set of singleton subsets of $\mathcal{G}$. A singleton subset of $\mathcal{G}$ means an elementary subset g, in which only one element (i.e. only one individual i) is present.

Also note that, i and i' are two individuals representing subsets g and g', respectively.

Assume that at scenario 1, i attends a **meeting** at location l during time slot t to discuss document d, where i' has no access to location l during same time slot t.
Here we state that privacy of meeting discussing document d was preserved at context $l \times t$

Now assume that at scenario 2, i attends a **meeting** at location l during time slot t to discuss document d, where i' also has access to location l during same time slot t.
Here we state that privacy of meeting discussing document d was violated at context $l \times t$, because $n+1$ individuals including i' have got access to content of document d. But actually $|access(d)| = n$, when $access(d)$ is defined as a set of singleton subsets of $\mathcal{G}$, as mentioned above. We observe that $(n+1) > |access(d)| = n$

When above 2 scenarios are compared, we observe that role of same individual i, representing subset g such that $g \in access(d)$, has experienced a variation. Context of i has changed, depending on location and time.

Therefore we define that presence of i at context $l \times t$ transforms i to role r.

$$transform(i, l, t) = r : r \text{ is role of } i \text{ at location } l \text{ at time slot } t$$

If $g \in access(d)$, $g' \notin access(d)$ and d is a **private** document, i representing g should attend a meeting to discuss d at context $l \times t$, only if i' representing g' has no access to $l \times t$. Accordingly, to identify the privacy preserving context for discussing document d, combination of i, l, t is required.

3.2.6 Difference between public and private roles

When we consider a **private** document d, we cannot exactly predict the time, at which

i', representing g', such that $g' \notin access(d)$, will get access to location l. Therefore, meeting organizer has the responsibility of defining location l as a **private** location or a **public** location, considering whether access of i' has been strictly restricted, during all potential meeting time slots (represented by set $\mathcal{T}$).

Using this definition and above formula, we can show that, i representing g, such that $g \in access(d)$ where d is a **private** document, is transformed to role g itself, at a **private** location. Here, location should be defined as a **private** location, by same entity, that defined the set $access(d)$ for document d.

$$transform(i, l, t) = r$$

$$transform(i, (private\_location), t) = r$$

$$transform(i, (private\_location), t) = g$$

On the other hand, any location l is defined as a **public** location, if access of i' has **not** been strictly restricted, during any potential meeting time slot in set of time slots $\mathcal{T}$.

Using this definition and above formula, we can show that, i representing g, such that $g \in access(d)$, is transformed to **public** role, at a **public** location. Location should be defined as a **public** location, by same entity, that defined the set $access(d)$ for document d.

$$transform(i, l, t) = r$$

$$transform(i, (public\_location), t) = r$$

$$transform(i, (public\_location), t) = public$$

Based on these derivations, we have identified a constraint relevant to i, for discussing d in a privacy preserved meeting.

Constraint: When d is a **private** document, every i representing g, such that $g \in access(d)$, that attends a meeting to discuss document d, must represent role g in the meeting.

When d is a **public** document, every i that attends a meeting to discuss document d, is allowed to represent **public** role in the meeting.

3.2.7 Roles in meeting agenda

If meeting agenda document does not include the group labeled as **public** in $access(agenda)$, it means that **public** $\notin access(agenda)$. Then i' representing g' such that $g' \notin access(d)$, should be strictly prevented from accessing the meeting, by conducting meeting at a **private** location, defined by relevant meeting organizing entity.

On the other hand, if meeting agenda includes group labeled as **public** in $access(agenda)$, it means that **public** $\in access(agenda)$. Then it is **not** mandatory to prevent access of i' representing g' such that $g' \notin access(d)$, for the meeting. Therefore, meeting can be conducted at a **private** location or **public** location, based on locations defined by relevant meeting organizing entity.

3.2.8 Variation of role

Now consider a situation where individual i representing g, such that $g \in access(d)$ has x number of locations, out of which any one can be selected for attending a meeting to discuss d. And assume that i has y number of time slots, out of which any one can be selected for attending the meeting.

We can depict the possible variations of $transform(i, l, t)$ function as below, for individual i, depending on locations defined by the entity, assuming that i does not change location during middle of a time slot.

(i)	t_1	t_2	...	t_{y-1}	t_y
l_1	x	x		x	x
l_2	x	x		x	x
...					
l_{x-1}	x	x		x	x
l_x	x	x		x	x

Table 3.1: Possibilities in variation of $transform(i, l, t)$ for individual i

Note that l_x represents the x^{th} location, while t_y represents the y^{th} time slot. Meanwhile x represents the role of i at the corresponding l and t (based on formula $transform(i, l, t) = r$). According to this representation, we observe that i has $x \times y$ number of possibilities at maximum, to attain the role.

Here we emphasize that some x roles can be categorized as **public**, with respect to **public** locations defined by an entity. According to the constraint identified, if d is a **private** document, i should attend the meeting only when $r = g$, such that $g \in access(d)$. When r = **public** role, individual i should strictly avoid discussing **private** documents. By following this constraint, access of i' representing g', such that $g' \notin aceess(d)$, into this meeting can be prevented.

3.2.9 Participants in access control lists of non-agenda documents

It is possible to discuss one or more documents in a meeting. Further, there can be both public documents and private documents among these documents. We do not need to follow any constraint to protect the privacy of public documents.

But when private documents are considered, it is needed to follow some constraints to protect the privacy. For example, consider d_1 and d_2 as 2 private documents. An individual i representing r, such that $r = g$ and $g \in access(d_1)$, can be absent in access control list of d_2. In other words, $g \notin access(d_2)$ relationship can exist.

In this situation, discussing both d_1 and d_2 in same meeting violates the privacy of d_2, when above mentioned individual i participates in that meeting. It means that, for discussing both d_1 and d_2 in same meeting, roles of all meeting participants should mandatorily be present in both $access(d_1)$ and $access(d_2)$. This relationship is graphically depicted in diagram below.

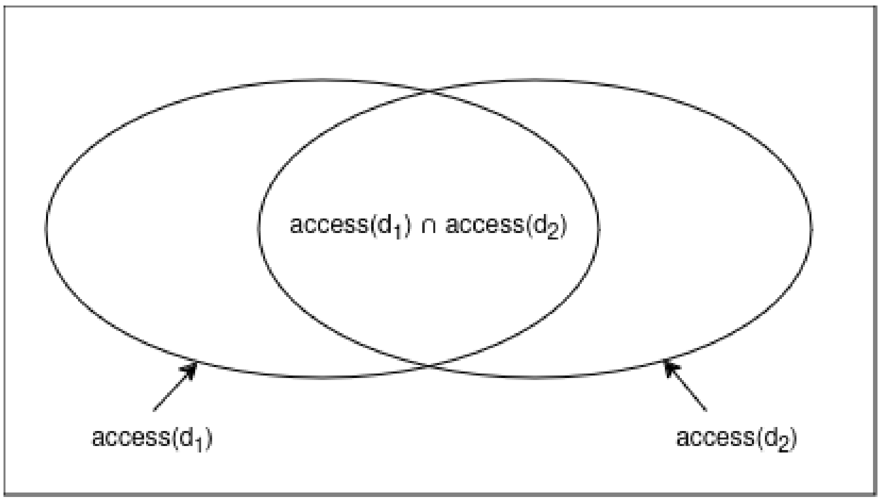


Figure 3.2: Intersection of access control lists of 2 private documents

This concept is applicable not only for 2 private documents, but also for any number of private documents discussed in same meeting. When there are n number of private documents to discuss in a meeting, intersection of access control lists of all those documents should be considered. Any individual i representing r, such that $r = g$, and $g \notin (access(d_1) \cap access(d_2) \cap ... \cap access(d_{n-1}) \cap access(d_n))$ should be prevented from accessing the meeting. If same individual represents multiple roles in different documents, it is recommended to convert all groups except the *public* group in those $access(d)$ sets to singleton sets, such that each individual participant can be uniquely identified.

3.2.10 Meeting participant validation

When there are private documents to discuss in a meeting, it is required to validate intersection of participants identified as above, with participants included in access control list of meeting agenda (“Meeting agenda” section at 3.2.3 discusses more details on meeting agenda).

For protecting privacy of n number of private documents defined as $d_1, ..., d_n$, following relationship must be satisfied for the meeting.

$$access(agenda) \subseteq \{access(d_1) \cap access(d_2) \cap ... \cap access(d_{n-1}) \cap access(d_n)\}$$

Simply, above relationship means that each and every individual i representing role r, such that $r = g$ and $g \in access(agenda)$, is also an element of the intersection of $access(d)$ of all private documents discussed in meeting. A situation in which above relationship is violated can be explained by using following relationship.

$$\{access(d_1) \cap access(d_2) \cap ... \cap access(d_{n-1}) \cap access(d_n)\} \subset access(agenda)$$

In simple terms, above relationship means that there exists at least one i representing r, such that $r = g$ and $g \in access(agenda)$, where g is not an element in the intersection of $access(d)$ of all private documents discussed in meeting.

However, for discussing public documents in meeting, it is not required to perform any participant validation. In other words, any individual i can discuss public documents, in any private or public meeting.

3.2.11 Privacy-preserved meeting

Based on above descriptions and definitions, we define privacy-preserved meeting as below;

A privacy-preserved meeting is a meeting in which, individual i representing role r has no access to the meeting, when $r = g$ and $g \notin access(agenda)$, where access(agenda) satisfies,

$$access(agenda) \subseteq \{access(d_1) \cap access(d_2) \cap ... \cap access(d_{n-1}) \cap access(d_n)\}$$

for all private documents $d_1 ... d_n$, discussed in the meeting.

3.2.12 Meeting quorum

In a **privacy-preserved meeting** of our research context, we define **meeting quorum** as the minimum number of individuals (i) representing participant groups (g), required to attend a meeting, such that $g \in access(agenda)$.

In privacy preserved meeting context, if a specific **meeting quorum** rule is not defined in the agenda, other than $access(agenda)$ set, we assume that every i such that,

- $i \in g$, and
- $g \in access(agenda)$,

is required for the meeting. But, it is not applicable for $g = \textbf{public}$, since "public" group means that meeting is a public meeting, but does not represent an invited group of participants. In meeting agenda document, when a numeric meeting quorum rule is not defined specifically, and $access(agenda)$ is defined in form of singleton subsets of $\mathcal{G}$, $|meeting\ quorum| = |access(agenda)|$ relationship exists.

In addition, it is possible that $|meeting\ quorum| < |access(agenda)|$, if a numeric **meeting quorum** rule is defined in meeting agenda. Therefore in overall, $|meeting\ quorum| \leq |access(agenda)|$, when $access(agenda)$ is defined in form of singleton subsets of $\mathcal{G}$.

Since at least 2 individuals (i) are required for any meeting, $2 \leq |meeting\ quorum|$.

Accordingly, $2 \leq |meeting\ quorum| \leq |access(agenda)|$.

When $access(agenda)$ is defined in form of singleton subsets of $\mathcal{G}$, as we have already depicted earlier, $|access(agenda)| \leq |\mathcal{I}|$. By merging this inequality with above expression, we obtain following expression theoretically.

$2 \leq |meeting\ quorum| \leq |access(agenda)| \leq |\mathcal{I}|$

3.3 Problem mapping and analysis

Our problem focused in organizing privacy-preserved meetings has few distinct steps, that can be clearly identified within it. Based on decision making points identified within the problem, this problem was mapped into a boolean circuit, following a union operation. Here, union operation can be introduced as a set related pre-processing operation, applied on the $access(d)$ sets of all documents of the meeting, including the *agenda*. Before applying this union operation, it is needed to make sure that all groups except the *public* group in those $access(d)$ sets are converted to singleton sets, such that each individual participant can be uniquely identified. Distinct sections of the problem are analyzed below, considering the circuit diagrams produced for them.

Following algorithms have been designed to align with the definitions defined by us, earlier in this research, regarding concepts such as documents, groups of individuals, access control lists, classification of locations into three groups, time slots, meeting quorum etc. Therefore, it is recommended to be familiar with those definitions, before analyzing the algorithms.

3.3.1 Participant validation based on documents

After calculating the union of $access(d)$ sets of all documents to be discussed in the meeting, as $\{access(doc\_1) \cup ... \cup access(doc\_n)\} \cup access(agenda)$, *public* group is subtracted, since our initial requirement is to identify all the individuals having access to at least one document. In addition, we identify all documents associated with meeting as $doc\_1, ..., doc\_n, agenda$, since we need to check whether each individual in the union of individuals identified, has access to all the documents, for discussing them in a meeting. In example depicted in the diagram, consider that there are only 3 individuals as i_1, i_2, i_3 and only 4 documents as $doc\_1, doc\_2, doc\_3, agenda$.

Algorithm for participant validation section is explained below.

Algorithm 1 Participant validation based on documents

```
 1: Input: Access control lists (access(d)) of documents $doc\_1$, ... , $doc\_n$, $agenda$
 2: Output: Eligibility of each individual for meeting by document analysis
 3:
 4: union of $access(d) = access(doc\_1) \cup \cdots \cup access(doc\_n) \cup access(agenda)$
 5: union of individuals = union of $access(d) - public$
 6: set of documents = $doc\_1, \ldots, doc\_n, agenda$
 7: for each individual $i\_n$ in (union of individuals) do
 8:     validity of $i\_n$ by doc analysis = true
 9:     for each $doc\_x$ in (set of documents) do
10:         if $doc\_x \neq agenda$ then
11:             validity of $i\_n$ for $doc\_x = (i\_n \in access(doc\_x))$ OR $(public \in access(doc\_x))$
12:         else
13:             validity of $i\_n$ for $doc\_x = i\_n \in access(agenda)$
14:         end if
15:         validity of $i\_n$ by doc analysis = validity of $i\_n$ by doc analysis
              AND validity of $i\_n$ for $doc\_n$
16:     end for
17:     Return validity of $i\_n$ by doc analysis
18: end for
```

Figure 3.3: Participant validation based on documents

Above algorithm explains the process depicted by logical circuit diagram below. In the algorithm,

$$|union\ of\ access(d)| = |access(doc\_1)| + ... + |access(doc\_n)| + |access(agenda)|$$

relationship means that time complexity of the *union of access*(d) operation is, $O(|access(doc\_1)| + ... + |access(doc\_n)| + |access(agenda)|)$. It is a linearly increasing time complexity. In addition, each iteration in outer loop outputs whether an individual of the union of individuals is valid by $access(d)$ analysis of all documents. Inner loop checks whether particular individual has access to all documents including *agenda*. Operations within the inner loop are considered as operations of constant time complexity, $O(1)$. Accordingly, since there is a nested loop in above algorithm, time complexity of this section of problem can be $O(n^2)$, at maximum. Because number of individuals can be any positive integer, and number of documents including *agenda* also can be any positive integer.

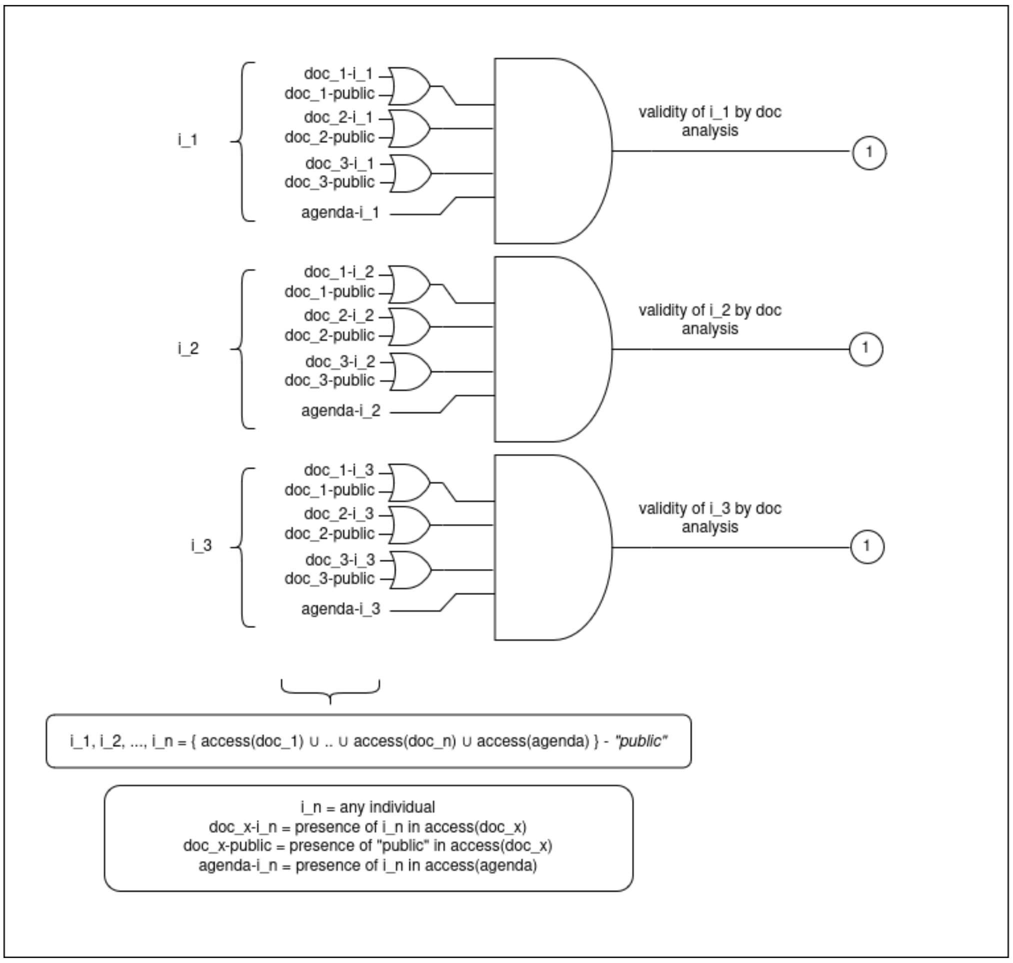

i_1
doc_1-i_1
doc_1-public
doc_2-i_1
doc_2-public
doc_3-i_1
doc_3-public
agenda-i_1
validity of i_1 by doc analysis
1
i_2
doc_1-i_2
doc_1-public
doc_2-i_2
doc_2-public
doc_3-i_2
doc_3-public
agenda-i_2
validity of i_2 by doc analysis
1
i_3
doc_1-i_3
doc_1-public
doc_2-i_3
doc_2-public
doc_3-i_3
doc_3-public
agenda-i_3
validity of i_3 by doc analysis
1
i_1, i_2, ..., i_n = { access(doc_1) ∪ .. ∪ access(doc_n) ∪ access(agenda) } - "public"
i_n = any individual
doc_x-i_n = presence of i_n in access(doc_x)
doc_x-public = presence of "public" in access(doc_x)
agenda-i_n = presence of i_n in access(agenda)

Figure 3.4: Participant validation based on $access(d)$ of documents

3.3.2 Eligibility of each individual, in each time slot, for meeting

After checking the validity of each participant by document analysis, it is needed to check their locations in different time slots. Based on the location, eligibility of each individual to discuss the set of documents changes. Because, as we have described earlier, *private* documents should be discussed at *private* locations only, while *public* documents can be discussed at any location. If there is at least one *private* document in the set of documents discussed in meeting, then each individual *i_n* validated by document analysis should attend this meeting from a *private* location. Following circuit diagram consists of the location analysis of each individual, in every time slot. Here, these time slots mean the time slots belonging to the union of available time slots of all individuals, present in the union of individuals, mentioned in previous algorithm (figure 3.3).

For deciding the eligibility of each individual, to discuss set of documents, in different time slots, we need inputs included in the following table. In it, each individual of the union of individuals, and his/her available time slots with locations should be present. Please note that following table consists of some sample data only, as individuals, time slots and locations. It is supposed to contain the real data, regarding the respective scenario. For simplicity, we consider only 3 locations as onsite, remote_private and remote_public, since every location can be categorized into one of those 3 categories, by a meeting organizing entity.

Individual (i_x)	Time slot(slot$y$)	Location
i_1	slot1	onsite
i_1	slot2	remote_private
i_1	slot3	remote_public
...	...	...
i_n	slot$m$	location of i_n in slot$m$

Table 3.2: Available time slots and locations of individuals, in union of individuals

Algorithm for deciding the eligibility of each individual, to discuss the set of documents of meeting, in different time slots, is described below.

Algorithm 2 Deciding eligibility of each individual for meeting, in each time slot

```
Input 1: Validity of each individual $i\_1$, ... , $i\_n$ by doc analysis
Note: Individuals $i\_1$, ... , $i\_n$ mean union of individuals of previous algorithm, and input 1 is obtained from output of previous algorithm
Input 2: Availability of each individual in each time slot (at any location)
Input 3: Location of each individual in each time slot (whether remote_public or no)
Note: Input 2 and input 3 are obtained from the table containing available time slots and locations of individuals.
Input 4: Presence of $public$ group in $access(d)$ of each document, in set of documents, as $(doc\_1 - public), ..., (doc\_n - public), (agenda - public)$
Output: Eligibility of each individual, to discuss the set of documents, in each time slot

union of time slots = $availability(i\_1) \cup \cdots \cup availability(i\_n)$
for each slot_$x$ in (union of time slots) do
    for each individual $i\_y$ in (union of individuals) do
        publicity of meeting = $(doc\_1 - public)$ AND ... AND $(doc\_n - public)$ AND $(agenda - public)$
        slot_$x - i\_y$ = availability of $i\_y$ in slot_$x$ at any location
        slot_$x - i\_y$_remote_public = whether $i\_y$ is at remote_public location in slot_$x$
        availability of slot_$x - i\_y$ for public meeting = (publicity of meeting) AND (slot_$x - i\_y$)
        availability of slot_$x - i\_y$ for private meeting = (slot_$x - i\_y$) AND NOT(slot_$x - i\_y$_remote_public)
        availability of slot_$x - i\_y$ for meeting by time slot analysis = (availability of slot_$x - i\_y$ for public meeting) OR (availability of slot_$x - i\_y$ for private meeting)
        slot_$x - i\_y$ eligibility for meeting = (Validity of $i\_y$ by doc analysis) AND (availability of slot_$x - i\_y$ for meeting by time slot analysis)
    end for
    Return slot_$x - i\_y$ eligibility for meeting
end for
```

Figure 3.5: Deciding eligibility of each individual for meeting, in each time slot

In above algorithm, time complexity of *union of time slots* operation is, $O(|availability(i\_1)| + \cdots + |availability(i\_n)|)$, at maximum. It is a linear time complexity. When considering the loops, outer loop iterates through all time slots, in the union of time slots, calculated by considering available time slots of all individuals, in union of individuals. Meanwhile, inner loop iterates through each individual in the union of individuals. In addition, number of documents discussed in the meeting can be any positive integer. Number of documents is considered when deciding the meeting publicity, within the inner loop. Further, validity of the individual by document analysis, obtained from output of previous algorithm, is utilized in in inner loop. Number of documents is concerned for that as well. But, it is in parallel level, with publicity calculation of the meeting. Therefore, we can consider that number of documents has linear time complexity as $O(n)$, within inner loop. Accordingly, it is observed that, time complexity of above algorithm can be $O(n^3)$, at maximum, due to outer loop, inner loop, and number of documents within inner loop.

Circuit diagram corresponding to above algorithm is depicted below. The following boolean circuit is supposed to be repeated for each individual, in each time slot, as explained in algorithm.

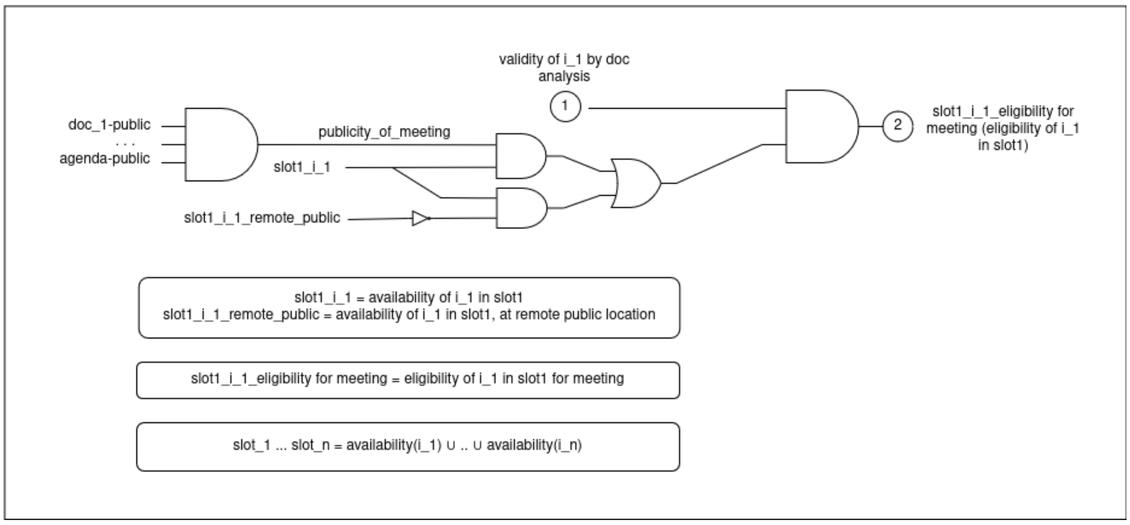

validity of i_1 by doc analysis
1
doc_1-public
. . .
agenda-public
publicity_of_meeting
slot1_i_1
slot1_i_1_remote_public
2
slot1_i_1_eligibility for meeting (eligibility of i_1 in slot1)
slot1_i_1 = availability of i_1 in slot1
slot1_i_1_remote_public = availability of i_1 in slot1, at remote public location
slot1_i_1_eligibility for meeting = eligibility of i_1 in slot1 for meeting
slot_1 ... slot_n = availability(i_1) ∪ .. ∪ availability(i_n)

Figure 3.6: Eligibility of each individual for meeting, in each time slot

3.3.3 Meeting quorum satisfiability

Eligibility of each individual to discuss the set of documents, in each time slot, is considered for deciding the quorum satisfiability of each time slot, in the union of time slots. Then it is checked whether there is at least one quorum satisfying time slot. Because, if there is no at least one quorum satisfying time slot, then it is not possible to conduct the meeting.

Algorithm for checking the quorum satisfiability of time slots is explained below.

Algorithm 3 Identifying meeting quorum satisfiability of time slots

```
1: Input 1: Eligibility of each individual for meeting, in each time slot
2: Note: Input 1 is obtained from output of previous algorithm
3: Input 2: Numerical meeting quorum value
4: Input 3: Availability of each individual in each time slot (at any location)
5: Output: Existence of quorum satisfying time slot/s, for the meeting
6:
7: union of time slots = $availability(i\_1) \cup \cdots \cup availability(i\_n)$
8: availability of a meeting quorum satisfying time slot = false
9: quorum satisfiability of slots = initiate an empty dictionary structure
10:
11: for each slot_x in (union of time slots) do
12:     meeting quorum satisfiability of slot_x = false
13:     combinations of individuals in slot_x = combinations of individuals, with magnitude
        of meeting quorum size
14:     for each combination in (combinations of individuals in slot_x) do
15:         meeting quorum satisfiability of combination = AND operation (for all individuals
            in the combination)
16:         meeting quorum satisfiability of slot_x = (meeting quorum satisfiability of slot_x)
            OR (meeting quorum satisfiability of combination)
17:     end for
18:     Store (meeting quorum satisfiability of slot_x) in (quorum satisfiability of slots)
        dictionary
19:     (availability of a meeting quorum satisfying time slot) = (availability of a meeting
        quorum satisfying time slot) OR (meeting quorum satisfiability of slot_x)
20: end for
21: Return (quorum satisfiability of slots) dictionary, (availability of a meeting quorum
    satisfying time slot) value
```

Figure 3.7: Identifying meeting quorum satisfiability of time slots

Similar to previous algorithm (figure 3.5), in above algorithm also, time complexity of *union of time slots* operation is, $O(|availability(i\_1)| + \cdots + |availability(i\_n)|)$, at maximum. It is observed that, in loop structure, outer loop iterates through all time slots in union of time slots. That number of time slots can be any positive integer. Inner loop contains a *combination* operation, which is intended to create combinations of magnitude of meeting quorum size, out of individuals of the union of individuals. Mathematically, number of such possible combinations can be calculated as,

$$n\text{C}r = \binom{n}{r} = \frac{n!}{r!(n-r)!}$$

when n = number of individuals in union of individuals, and r = meeting quorum size.

Regarding *combination* operation, highest number of combinations is obtained when $r = \frac{n}{2}$ or $r \approx \frac{n}{2}$ (appendix A). Number of possible combinations reduces, when r is considerably small ($r \approx 0$), or when r is close to n ($r \approx n$). Due to this nature of *combination* operation, maximum possible time complexity for creating $n\mathrm{C}r$ number of combinations is $O(\binom{n}{\frac{n}{2}})$ or $O(\binom{n}{\frac{n-1}{2}})$, depending on whether n is even or odd.

In the inner loop of above algorithm, since AND operation is applied for all individuals, in each *combination* created, maximum time complexity associated with inner loop is $O(n \times \binom{n}{\frac{n}{2}})$ or $O(n \times \binom{n}{\frac{n-1}{2}})$. Because, the number of participants in each combination created can be any positive integer greater than or equal to 2. When considering a particular scenario, that number is equal to the meeting quorum size. In addition, since inner loop is repeated by outer loop, where number of time slots also can be any positive integer, maximum possible time complexity of above algorithm is,

$$O(n \times n \times \binom{n}{\frac{n}{2}}) = O(n^2 \times \binom{n}{\frac{n}{2}}) \text{ ,when union has even number of individuals}$$

or

$$O(n \times n \times \binom{n}{\frac{n-1}{2}}) = O(n^2 \times \binom{n}{\frac{n-1}{2}}) \text{ ,when union has odd number of individuals.}$$

Time complexity of $O(n^2 \times \binom{n}{\frac{n}{2}})$ or $O(n^2 \times \binom{n}{\frac{n-1}{2}})$ has polynomial component of $O(n^2)$. Further, when considering $O(\binom{n}{\frac{n}{2}})$ or $O(\binom{n}{\frac{n-1}{2}})$ component, it is observed that, for large n values,

$$O(\binom{n}{\frac{n}{2}}) \approx O(\frac{2^n}{\sqrt{n}}) \qquad (\textit{appendix B})$$

or

$$O(\binom{n}{\frac{n-1}{2}}) \approx O(\frac{2^n}{\sqrt{n}}) \qquad (\textit{appendix C})$$

depending on whether union has an even number of individuals, or an odd number of individuals. Using that complexity value, time complexity for *algorithm* 3 can be calculated as below.

$$O(n^2 \times \frac{2^n}{\sqrt{n}}) = O(n^{\frac{3}{2}} \times 2^n)$$

$$O(n^{\frac{3}{2}} \times 2^n) = O(\sqrt{n^3} \times 2^n)$$

Time complexity of $O(\sqrt{n^3} \times 2^n)$ has $O(2^n)$ as the dominant term, for large values of n. Time complexity $O(2^n)$ is considered as an exponential time complexity, because n is not a constant value ([19]). Since time complexity of $O(2^n)$ increases exponentially with increase of n, it is eligible to use a heuristic or an existing library, when implementing above algorithm for execution with inputs. Because, there is no specific standard methodology defined, for creating combinations, as included in above algorithm.

Figure below depicts the corresponding boolean circuit for above algorithm.

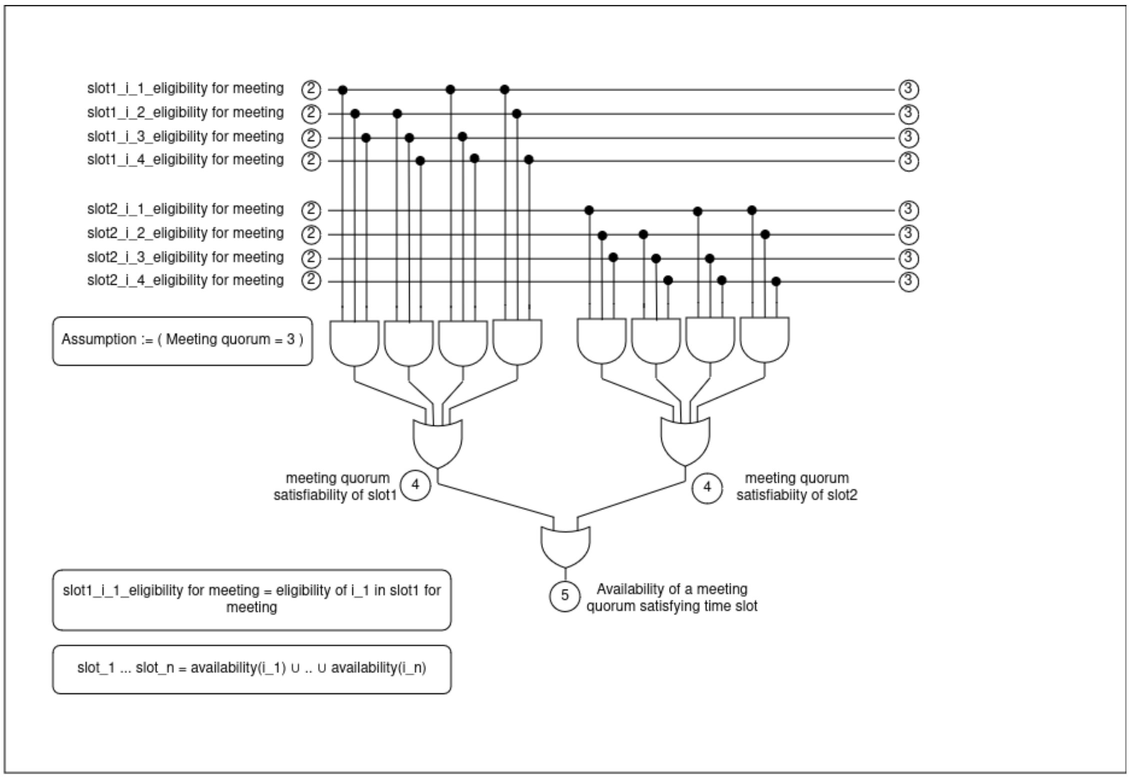

slot1_i_1_eligibility for meeting
slot1_i_2_eligibility for meeting
slot1_i_3_eligibility for meeting
slot1_i_4_eligibility for meeting
slot2_i_1_eligibility for meeting
slot2_i_2_eligibility for meeting
slot2_i_3_eligibility for meeting
slot2_i_4_eligibility for meeting
2
3
Assumption := (Meeting quorum = 3)
meeting quorum satisfiability of slot1
4
meeting quorum satisfiabilty of slot2
5
Availability of a meeting quorum satisfying time slot
slot1_i_1_eligibility for meeting = eligibility of i_1 in slot1 for meeting
slot_1 ... slot_n = availability(i_1) ∪ .. ∪ availability(i_n)

Figure 3.8: Meeting quorum satisfiability of time slots

3.3.4 Selection of earliest, meeting quorum satisfying time slot

Selecting the earliest, meeting quorum satisfying time slot is not required to find a solution for our research problem. Because, after identifying the quorum satisfying time slots by above algorithm, our next objective associated with research problem is to, analyze the privacy-preserving meeting mode for each meeting quorum satisfying time slot. Therefore, we introduce this step of selecting the earliest, meeting quorum satisfying time slot, as an additional step provided by us. Since privacy-preserved meeting organization is a real world problem, when implementing a usable system for that purpose, this additional step can be helpful to identify the earliest, eligible time slot for conducting the privacy-preserved meeting, rather than suggesting numerous eligible time slots.

Algorithm for selecting the earliest, meeting quorum satisfying time slot is explained below.

Algorithm 4 Selection of earliest, meeting quorum satisfying time slot

```
1: Input: Meeting quorum satisfiability of time slots, in union of time slots
2: Note: Input is produced by processing previous algorithm
3: Output: Earliest, meeting quorum satisfying time slot
4:
5: union of time slots with quorum satisfiability = quorum satisfiability of each time slot,
   in ($availability(i\_1) \cup \cdots \cup availability(i\_n)$), in the order of precedence
6: earliest eligible time slot = initialization of an empty list type structure, to store whether
   each slot is earliest eligible slot or no
7: for each slot_$x$ and $meeting\_quorum\_satisfiability$ in (union of time slots with quorum
   satisfiability) do
8:     if slot_$x$ is earliest slot in (union of time slots with quorum satisfiability) then
9:         store slot_$x$ and its $meeting\_quorum\_satisfiability$, in (earliest eligible time slot)
   list
10:    else
11:        previous condition = list to store result of, NOT operation (for each slot currently
   stored in (union of time slots with quorum satisfiability) list)
12:        previous condition integrated = AND operation (for all slots in (previous
   condition) list)
13:        whether slot_$x$ is earliest meeting quorum satisfying time slot =
   (($meeting\_quorum\_satisfiability$ of slot_$x$) AND (previous condition integrated))
14:        store slot_$x$ and (whether slot_$x$ is earliest meeting quorum satisfying time slot),
   in (earliest eligible time slot) list
15:    end if
16: end for
17: Return (earliest eligible time slot) list
18: Note: At Return point, (earliest eligible time slot) list contains $true$ for earliest,
   meeting quorum satisfying time slot, and $false$ for all other time slots
```

Figure 3.9: Selection of earliest, meeting quorum satisfying time slot

In above algorithm, there is a reason to apply a different process on the earliest time slot in union of time slots, apart from all other time slots in the union. If earliest time slot in union of time slots satisfies the meeting quorum, then without considering any pre-condition, it becomes the earliest meeting quorum satisfying time slot. It can be analyzed by extracting it as a separate condition, like below.

```
If earliest slot in union of time slots satisfies the meeting quorum, Then
    earliest slot is the earliest, meeting quorum satisfying time slot
Else
    earliest slot is not the earliest, meeting quorum satisfying time slot
End If
```

Above condition block is the meaning implied by “If” block, in the condition of the algorithm. But, if any time slot other than the earliest time slot of union, satisfies the meeting quorum, then it is needed to check an additional condition as well, to identify whether it is the **earliest** meeting quorum satisfying time slot or no. Accordingly, following condition block elaborates the meaning implied by "Else" block of the condition, in above algorithm.

```
If any slot$x$ except the earliest slot in union of time slots, satisfies the meeting quorum,
Then
    If any earlier slot than slot$x$ does not satisfy the meeting quorum, Then
        slot$x$ is the earliest, meeting quorum satisfying time slot
    Else
        slot$x$ is not the earliest, meeting quorum satisfying time slot
    End If
Else
    slot$x$ is not the earliest, meeting quorum satisfying time slot
End If
```

It is observed that, there is a nested "If" condition in "If" block, in above condition block. Requirement to check this additional condition is the reason for treating earliest time slot in union of time slots in a different manner, than other time slots, when identifying the earliest meeting quorum satisfying time slot.

Above algorithm utilizes the meeting quorum satisfiability of each time slot, obtained by processing previous algorithm. Therefore, When we calculate the time complexity of above algorithm, we consider that meeting quorum satisfiability information is stored in an eligible data structure after processing previous algorithm. Otherwise, if we consider these both algorithms together, time complexity of above algorithm (figure 3.9) and previous algorithm (figure 3.7) collectively becomes, $O(\sqrt{n^3} \times 2^n)$ at maximum, as explained in time complexity description of previous algorithm.

But, when meeting quorum satisfiability information obtained from previous algorithm is stored in an eligible data structure and provided to above algorithm, time complexity of above algorithm is dominated by "for" loop, and its content. Number of iteration in this loop can be any positive integer. There is no inner loop, in the "for" loop. But, in "Else" block of each iteration, "NOT" operation is applied on each time slot, until the slot which is currently being iterated by "for" loop. Therefore, it is observed that, time complexity of above algorithm can be $O(n^2)$, at maximum. Even though there is an "AND" operation applied on all time slots, in the "Else" block, it can be neglected, when calculating time complexity. Because, it is in the parallel level to "NOT" operation, which is already considered. All other operations within above algorithm are considered as operations having constant time complexity.

Circuit diagram corresponding to above algorithm is depicted below.

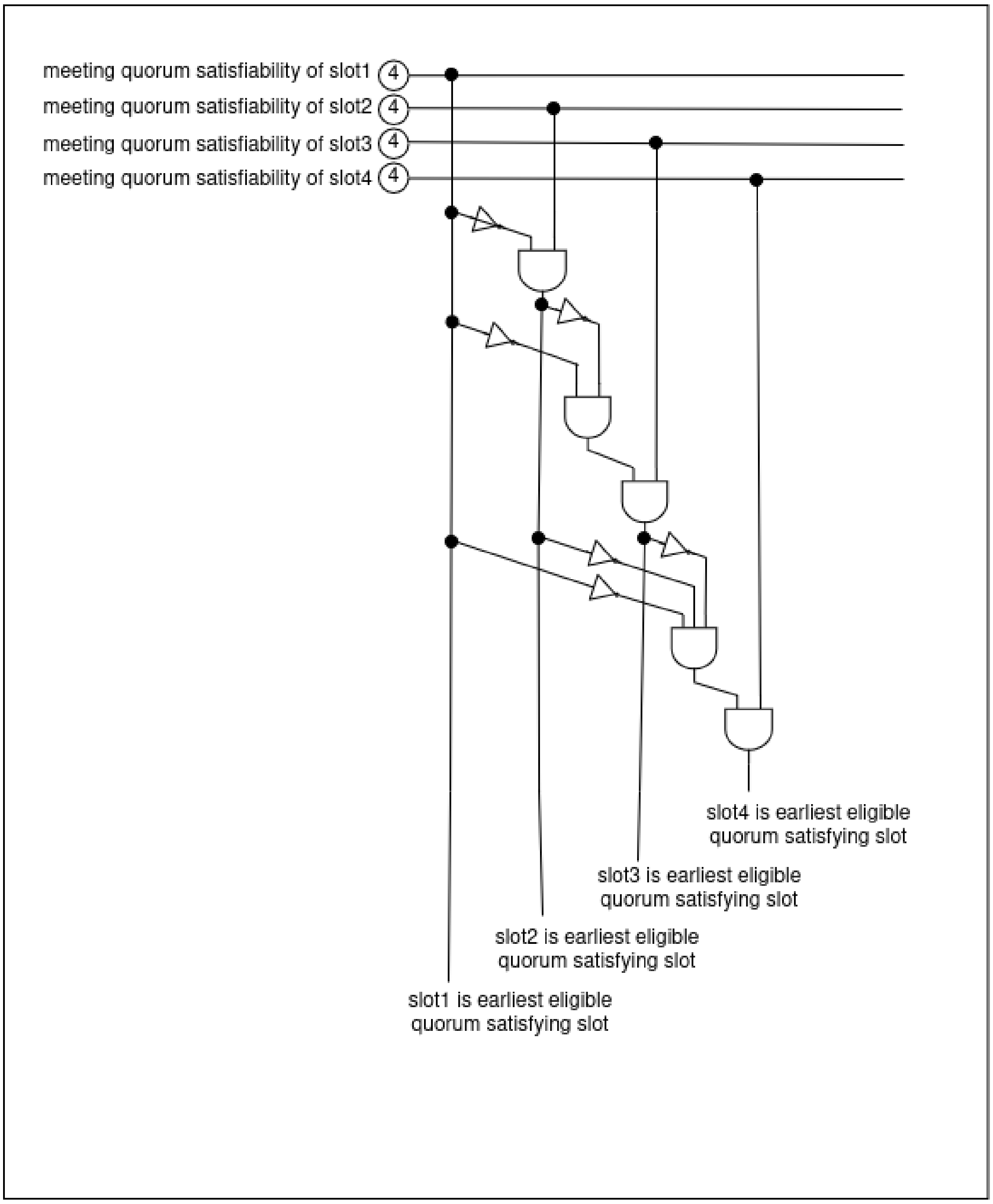

meeting quorum satisfiability of slot1
4
meeting quorum satisfiability of slot2
4
meeting quorum satisfiability of slot3
4
meeting quorum satisfiability of slot4
4
slot4 is earliest eligible quorum satisfying slot
slot3 is earliest eligible quorum satisfying slot
slot2 is earliest eligible quorum satisfying slot
slot1 is earliest eligible quorum satisfying slot

Figure 3.10: Selection of earliest, meeting quorum satisfying time slot

3.3.5 Meeting mode selection

After identifying the quorum satisfying time slots, meeting mode of each time slot should be decided, based on locations of eligible participants. In real world scenario, participants can be present at numerous locations. However in our research, as it is already mentioned earlier, only three location types are considered (onsite, remote_private and remote_public locations). It is observed that three meeting modes are possible, based on those three locations, as onsite mode, hybrid mode and online mode. Logic for deciding the meeting mode is depicted by the flow chart below.

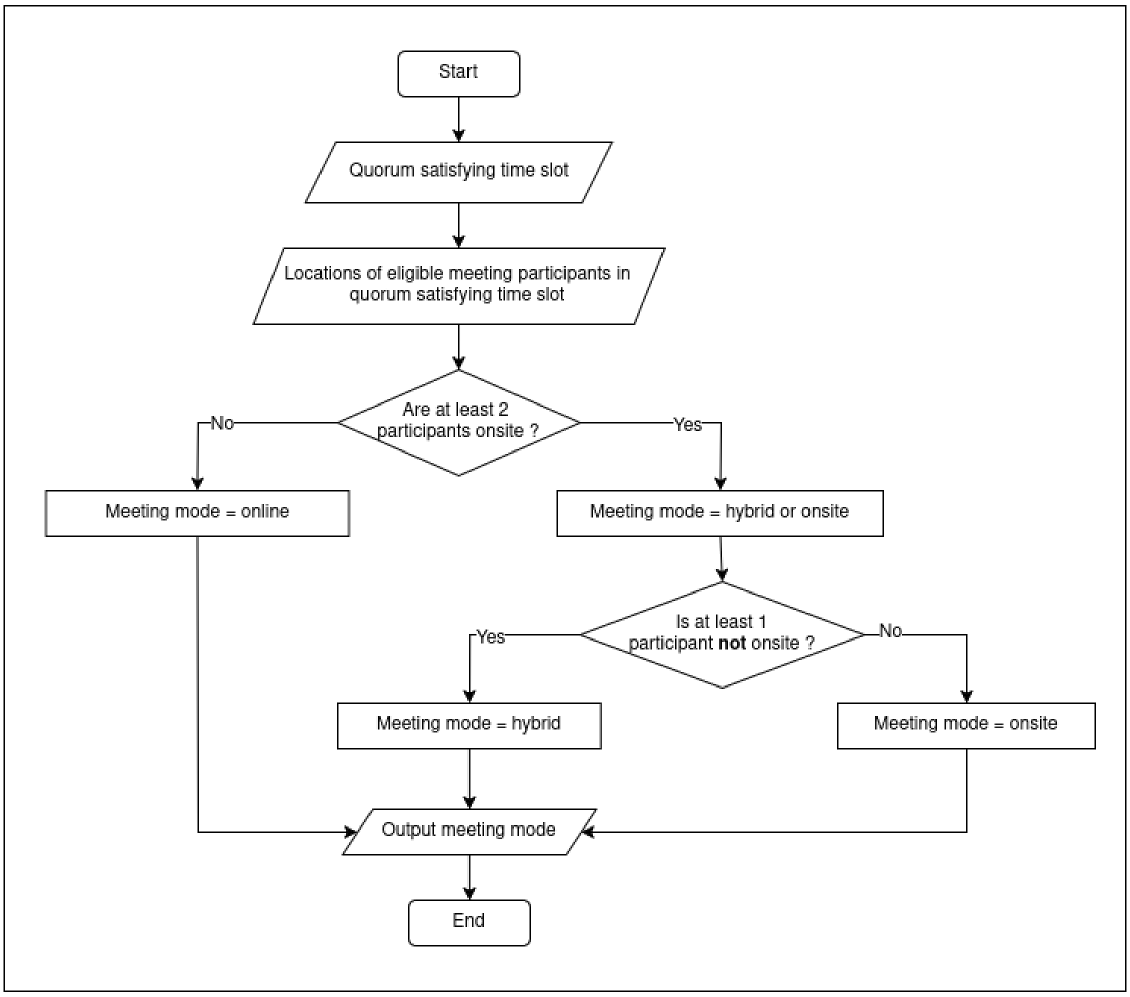

Start
Quorum satisfying time slot
Locations of eligible meeting participants in quorum satisfying time slot
Are at least 2 participants onsite ?
No
Yes
Meeting mode = online
Meeting mode = hybrid or onsite
Is at least 1 participant **not** onsite ?
Yes
No
Meeting mode = hybrid
Meeting mode = onsite
Output meeting mode
End

Figure 3.11: Logic for selecting the meeting mode

According to the logic, initially we check whether at least two eligible participants are at onsite locations, during the quorum satisfying time slot. If only one participant is at onsite location (i.e. onsite office location of meeting organization), or no one is at onsite location, it means that obviously meeting will be in online mode. But, if at least two participants are available at onsite location, it is required to check whether at least one participant is not at the onsite location. If at least one participant is at a remote location in the considered time slot, meeting will obviously be in hybrid mode. Unless, if at least two participants are onsite, and at least one participant is **not** at a remote location (remote_private or remote_public), it means that all available participants are at onsite location. It implies that meeting is in onsite mode.

The algorithm for deciding the meeting mode of a quorum satisfying time slot is explained below.

Algorithm 5 Meeting mode selection

```
1: Input 1: Eligibility of each participant for meeting, in each time slot
2: Note: Input 1 is produced by processing algorithm 1
3: Input 2: Meeting quorum satisfiability of each time slot, in union of time slots
4: Note: Input 2 is produced by processing algorithm 3
5: Input 3: Whether each meeting participant is at onsite location, in each time slot
6: Output: Whether meeting mode is onsite or hybrid or online
7:
8: for each slot_$x$ in (union of time slots) do
9:     slot_$x$ onsite = initialize an empty list
10:    slot_$x$ online = initialize an empty list
11:    for each participant_$y$ with $eligibility\ for\ meeting$ status in slot_$x$ do
12:        participant_$y$_slot_$x$_onsite = whether participant_$y$ is onsite in slot_$x$
13:        append to (slot_$x$ onsite) list: ($eligibility\ for\ meeting$ for participant_$y$) AND (participant_$y$_slot_$x$_onsite)
14:        append to (slot_$x$ online) list: ($eligibility\ for\ meeting$ for participant_$y$) AND NOT(participant_$y$_slot_$x$_onsite)
15:    end for
16:    slot_$x$ is onsite or hybrid = false
17:    onsite combinations of slot_$x$ = create all possible combinations of size 2, with items in (slot_$x$ onsite) list
18:    for each combination_$z$ in (onsite combinations of slot_$x$) do
19:        combination_$z$ is onsite = apply AND operation between 2 items in combination_$z$
20:        slot_$x$ is onsite or hybrid = (slot_$x$ is onsite or hybrid) OR (combination_$z$ is onsite)
21:    end for
22:    eligibility of slot_$x$ for online mode = (NOT(slot_$x$ is onsite or hybrid)) AND (meeting quorum satisfiability of slot_$x$)
23:    slot_$x$ is hybrid = false
24:    for each participant $p$ in (slot_$x$ online) list do
25:        slot_$x$ is hybrid = (slot_$x$ is hybrid) OR (whether participant $p$ is online)
26:    end for
27:    eligibility of slot_$x$ for hybrid mode = (slot_$x$ is hybrid) AND {(meeting quorum satisfiability of slot_$x$) AND (slot_$x$ is onsite or hybrid)}
28:    eligibility of slot_$x$ for onsite mode = (NOT(slot_$x$ is hybrid)) AND {(meeting quorum satisfiability of slot_$x$) AND (slot_$x$ is onsite or hybrid)}
29:    Return eligibility of slot_$x$ for online mode, eligibility of slot_$x$ for hybrid mode, eligibility of slot_$x$ for onsite mode
30:    Note: At Return point, depending on whether eligible meeting mode is online or hybrid or onsite, respective variable will be true, and other two variables will be false.
31: end for
```

Figure 3.12: Meeting mode selection

When considering the algorithm above, it is observed that it obtains multiple outputs from previous algorithms. In other words, it is impossible to provide inputs required by this algorithm, without processing previous algorithms. However, since it is not mandatory to execute this algorithm in parallel with previous algorithms, it is possible to calculate the the time complexity of this algorithm, as below.

In above algorithm, it is observed that there is a nested "for" loop structure. There are some separate"for" loops, inside the outer loop. Among those "for" loops, the loop iterating through all possible combinations of two individuals has nCr number of iterations, where n = number of individuals in union of individuals, and r = 2. Therefore, maximum possible number of iterations in that particular inner loop can be mathematically calculated as below.

$$n\text{C}2 = \binom{n}{2} = \frac{n!}{2!(n-2)!}$$

$$\frac{n!}{2!(n-2)!} = \frac{n!}{2 \times (n-2)!}$$

$$\frac{n!}{2 \times (n-2)!} = \frac{(n-2)! \times (n-1) \times n}{2 \times (n-2)!}$$

$$\frac{(n-2)! \times (n-1) \times n}{2 \times (n-2)!} = \frac{(n-1) \times n}{2}$$

$$\frac{(n-1) \times n}{2} = \frac{n^2}{2} - \frac{n}{2} = \frac{1}{2} \times (n^2 - n)$$

$$\therefore n\text{C}2 = \frac{1}{2} \times (n^2 - n)$$

Maximum possible time complexity of "for" loop, which iterates through all possible combinations of two individuals is $O(n\text{C}r)$. Since value of r = 2, and depending on above calculation, time complexity of this "for" loop can be depicted as below.

$$O(n\text{C}2) = O(\binom{n}{2}) = O(n^2 - n)$$

$$O(n^2 - n) = O(n(n-1)) = O(n(n))$$

$$\therefore O(n\text{C}2) = O(n^2)$$

In addition, when considering the outer "for" loop together with the inner "for" loop, that iterates through all possible combinations of two individuals, total time complexity can be calculated as below. Because, outer loop can iterate through n number of iterations, where n can be any positive integer.

$$O(n) \times O(n\text{C}2) = O(n \times (n\text{C}2))$$

$$O(n \times (n\text{C}2)) = O(n \times n^2)$$

$$O(n \times n^2) = O(n^3)$$

Except above discussed inner "for" loop that iterates through combinations, every other inner "for" loop iterates through n number of iterations, where n is any positive integer. When such a "for" loop is considered together with the outer "for" loop, which also can iterate thorugh n number of iterations, combined time complexity becomes $O(n \times n)$. It can be simplified as below.

$$O(n \times n) = O(n^2)$$

Accordingly, when those inner "for" loops are considered, since $O(n^3) > O(n^2)$, maximum possible time complexity of above algorithm is $O(n^3)$. When calculating the time complexity, data retrieval operations which can be implemented using efficient methods like dictionaries are considered to have constant time complexities.

The circuit diagram corresponding to the above algorithm is depicted by figure below.

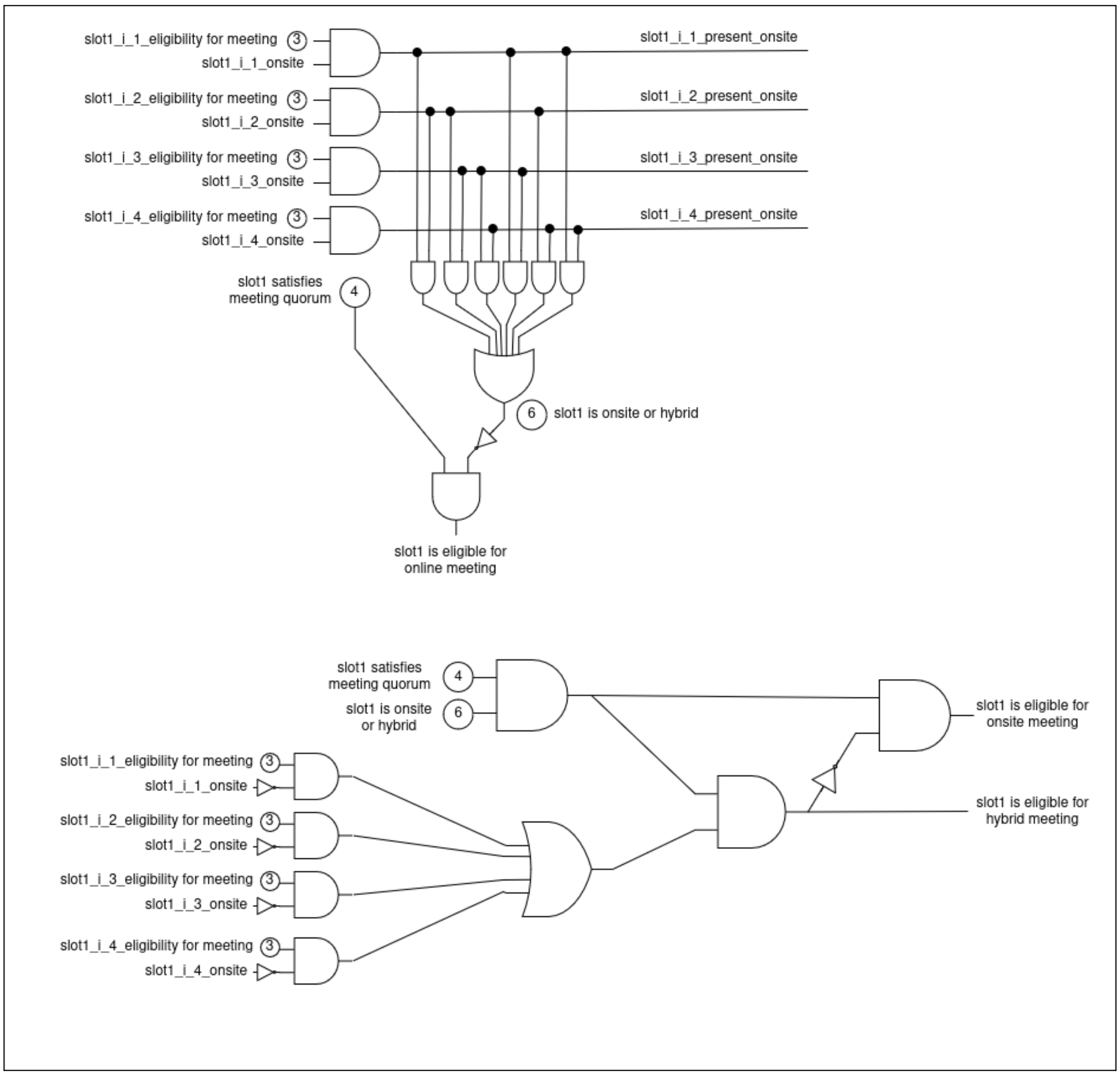

slot1_i_1_eligibility for meeting
slot1_i_1_onsite
slot1_i_2_eligibility for meeting
slot1_i_2_onsite
slot1_i_3_eligibility for meeting
slot1_i_3_onsite
slot1_i_4_eligibility for meeting
slot1_i_4_onsite
slot1_i_1_present_onsite
slot1_i_2_present_onsite
slot1_i_3_present_onsite
slot1_i_4_present_onsite
slot1 satisfies meeting quorum
slot1 is onsite or hybrid
slot1 is eligible for online meeting
slot1 is eligible for onsite meeting
slot1 is eligible for hybrid meeting

Figure 3.13: Circuit diagram for selecting the meeting mode

3.3.6 Simplified version of algorithm on meeting quorum satisfiability (figure 3.7)

Currently in *algorithm* 3 (figure 3.7), combinations of participants are created based on the size of meeting quorum defined as an input. Since that algorithm has an exponential time complexity, while looking for a heuristic to solve it, it was noticed that same functionality can be achieved by following algorithm.

In following algorithm, there is an "If" condition, which is not mapped to boolean circuit. But, when implementing this algorithm using a programming language, it is possible to use syntax of "If" structure of the relevant programming language.

Algorithm 6 Identifying meeting quorum satisfiability of time slots (Simplified version of **algorithm 3**)

```
1: Input 1: Eligibility of each individual for meeting, in each time slot
2: Note: Input 1 is obtained from output of previous algorithm
3: Input 2: Numerical meeting quorum value
4: Input 3: Availability of each individual in each time slot (at any location)
5: Output: Existence of quorum satisfying time slot/s, for the meeting
6:
7: union of time slots = $availability(i\_1) \cup \cdots \cup availability(i\_n)$
8: availability of a meeting quorum satisfying time slot = false
9: quorum satisfiability of slots = initiate an empty dictionary structure
10:
11: for each slot_$x$ in (union of time slots) do
12:     True_count = calculate the number of eligible participants in slot_$x$ using Input 1
13:     if True_count $\geq$ meeting_quorum then
14:         meeting quorum satisfiability of slot_x = True
15:     else
16:         meeting quorum satisfiability of slot_x = False
17:     end if
18:     Store (meeting quorum satisfiability of slot_$x$) in (quorum satisfiability of slots)
        dictionary
19:
20:     (availability of a meeting quorum satisfying time slot) = (availability of a meeting
        quorum satisfying time slot) OR (meeting quorum satisfiability of slot_$x$)
21: end for
22: Return (quorum satisfiability of slots) dictionary, (availability of a meeting quorum
    satisfying time slot) value
```

Figure 3.14: Identifying meeting quorum satisfiability of time slots (Simplified version of **algorithm 3** depicted by figure 3.7)

When calculating the time complexity of above algorithm, it can be observed that there is a "for" loop, which iterates through the number of time slots. In addition, within that "for" loop, there is a mathematical addition, which calculates the number of eligible participants within each time slot. Since number of time slots can be any positive integer, and since number of eligible participants in each time slot can be any positive integer, time complexity is calculated as $0(n \times n)$, which gets simplified into $O(n^2)$. Because, all other operations within the "for" loop are operations of constant time complexity, and exist at parallel level with mathematical addition. Therefore, those operations can be neglected in time complexity calculation. Accordingly, maximum possible time complexity of above algorithm is $O(n^2)$.

3.4 Analysis on complexity of the problem

3.4.1 Analysis based on time complexity

When above all algorithm segments are considered, it is observed that they do not have to be executed in parallel to each other. But, since there is an order of proceeding in operations, five algorithms are supposed to be executed from *algorithm* 1 to *algorithm* 5, sequentially. Therefore, when calculating the maximum possible time complexity of the overall problem, which has been divided into sub-problems, time complexities of those sub-problems should be compared. Following table depicts the maximum possible time complexity of each algorithm, described above.

Algorithm	**Maximum time complexity**
Algorithm 1 (fig. 3.3)	$O(n^2)$
Algorithm 2 (fig. 3.5)	$O(n^3)$
Algorithm 3 (fig. 3.7)	$O(\sqrt{n^3} \times 2^n)$
Simplified version of algorithm 3 (fig. 3.14)	$O(n^2)$
Algorithm 4 (fig. 3.9)	$O(n^2)$
Algorithm 5 (fig. 3.12)	$O(n^3)$

Table 3.3: Maximum time complexities of algorithm segments in boolean circuit mapping

Above time complexities are calculated for generating the boolean circuit dynamically, for a particular scenario, based on inputs provided. When above time complexities are considered, it can be observed that *algorithm* 1, *algorithm* 2, *simplifeid version of algorithm* 3, *algorithm* 4 and *algorithm* 5 have polynomial time complexities. Because, they have constant values such as 2, 3, as the power of base n. But in original version of *algorithm* 3 (figure 3.7), which is implemented using logic gates entirely, maximum possible time complexity is approximately $O(\sqrt{n^3} \times 2^n)$. This is a non-polynomial time complexity. Because, its dominant component of 2^n increases exponentially, when value of n increases, with constant base 2. Accordingly, it can be observed that, when this algorithm containing 5 segments is implemented using *simplified version of algorithm* 3, maximum possible time complexity is $O(n^3)$, but not

$O(\sqrt{n^3} \times 2^n)$.

In above each algorithm segment excluding *simplified version of algorithm* 3, after generating the boolean circuit, it takes a polynomial time duration to simplify, for achieving the output [20]. But, it becomes negligible in comparison to above time durations spent for generating the boolean circuit. In addition, *simplified version of algorithm* 3 directly provides its output within time complexity of $O(n^2)$. It means that, our problem is solvable within polynomial time.

Therefore, it can be concluded that, our problem belongs to P problem category, which takes polynomial time to solve. Our algorithm including *simplified version of algorithm* 3 segment is recommended for solving the problem within polynomial time, since *algorithm* 3 has an exponential time complexity.

3.4.2 Analysis on NP-completeness of problem

When related work associated with meeting scheduling problems are considered, most of them are introduced as NP-complete problems ([2]). NP-complete problems are problems belonging to both, NP and NP-hard categories ([21]). NP problems are problems that cannot be solved within a polynomial time duration, like explained earlier.

On the other hand, NP-hard problems cannot be solved during a polynomial time, as well as usually involve an optimization of variables. In addition, NP-hard problems cannot be verified within polynomial time. Graph edge coloring problem is a famous NP-hard problem. Graph edge coloring problem is, finding the number of colors required for coloring all edges of a graph, such that no two adjacent edges get the same color. Bofill et al have mapped the Business-to-Business meeting scheduling problem to edge coloring problem, for showing that it is NP-hard [2]. In our research, attempts to map our problem to an NP-hard problem like that failed.

Satisfiability problems such as knapsack problem, circuit satisfiability problem, MaxSAT problem and subset sum problem are considered as famous NP-complete problems. Therefore, it is possible to prove that any problem is NP-complete, by mapping it to one of those already known NP-complete problems [21]. Our research problem was mapped to a boolean circuit as explained earlier, and then was simplified by involving a conditional block as an intermediate step. Because, pure circuit implementation spends an exponential time to solve the problem. Even that exponential time version was not mapped to any NP-complete problem. When pure circuit mapping itself is considered, obviously our requirement is to find whether privacy preserved meeting is possible or no, when inputs are known. But, our requirement is not to optimize the inputs of circuit for conducting a privacy preserved meeting as the output. **Therefore, it is concluded that our problem**

does not belong to NP-complete category.

But, our problem has the possibility to be an NP-complete or NP-hard problem, if optimization related sub-problems are concerned. As examples, identifying the privacy-preserved meeting with highest number of participants for discussing a set of documents, scheduling a series of privacy-preserved meetings, distributing a set of documents into several meetings such that they can be discussed without violating the privacy can be cited. Such problems cannot be solved in polynomial time. If they are analyzed deeply in a future research, sub-problems which spend exponential time even for verification (NP-hard problems) maybe identified.

As depicted earlier in figure 2.2, NP, NP-hard and NP-complete regions represent the non-polynomial time solvable problem categories explained earlier. P region represents the polynomial time solvable problems, which includes our research problem as well.

Chapter 4

Implementation

4.1 Implementation of boolean circuit

4.1.1 Choice of technology

As explained earlier, our research problem was mapped to a boolean circuit. And, that circuit was elaborated as five sequential algorithms, representing consecutive sub-sections of the entire boolean circuit. Since it is required to show that those algorithms are convertible to a practical application in real world, a proof of concept was developed in our research [22]. In addition, simplified version of circuit including a conditional block as an intermediate step also was implemented separately [23].

After analyzing potentially eligible programming languages and technologies, Python was selected as the programming language for implementing the proofs of concept representing pure circuit version [22] and simplified version [23] both, explained earlier. Because, in our analysis, it was noticed that Python has libraries containing built-in functions, executing the behavior of logic gates. In algorithms explained above, AND, OR and NOT logic gates are utilized. Therefore, when Python is used to implement above algorithms, it is not required to simulate the behavior of logic gates by us, since there are built-in Python functions. When developing the proofs of concept, we used hard coded inputs. When executing the program for a new scenario, those inputs can be updated easily by editing the Python code.

In addition, it is required to create mathematical combinations in *algorithm* 3 (figure 3.7), as explained earlier. In Python, there are libraries containing functions that support to create such combinations. Because, creating combinations is a problem of exponential time complexity. After compiling the Python code with a compiler, execution can be done in a console environment, such as Linux terminal or the console of a suitable Integrated Development Environment (IDE).

4.1.2 Details on technology used for circuit implementation

Following table depicts the the version numbers and other specific details, regarding the technologies used to implement the programs, as proofs of concept.

Technology	**Name and version**
Programming language	Python 3.7.0
Compiler	GCC 7.2.0
Library for logic gates	Sympy 1.2
Library for mathematical combinations	Itertools bundled with Python 3.7.0
Output console	Terminal of Ubuntu 22.04.4 LTS

Table 4.1: Technologies used to implement the proofs of concept for algorithms

4.1.3 Overview of the Python programs

Above algorithms are designed to accept the **individuals** present in **groups** in access control lists of documents, including the agenda, as input. Therefore, in these both Python programs [22] and [23], when defining the access control lists of documents, they should be defined in terms of singleton sets (i.e. individual participants), instead of aggregate groups containing multiple individuals. But, *public* group present in access control list of any document should not be resolved, since it is required to decide the eligible location for conducting the meeting. Following example depicts how access control lists should be defined in the Python program, for documents including the meeting agenda.

```
doc1 = {"public"}
doc2 = {"i_2", "i_3", "i_4", "i_5", "i_6"}
agenda = {"public", "i_1", "i_2", "i_3", "i_4", "i_5"}
```

Figure 4.1: Sample code snippet for defining access control lists of documents

In addition, availability of individuals in different time slots, at different locations is stored in program using a dictionary. Sample section from this dictionary is depicted in the code snippet below.

```
availability_dictionary = [
    {"person": "i_1", "slot": "slot1", "location": "onsite"},
    {"person": "i_1", "slot": "slot2", "location": "remote_private"},
    {"person": "i_2", "slot": "slot1", "location": "remote_public"}
]
```

Figure 4.2: Sample code snippet for availability dictionary

Further, meeting quorum also should be defined as an input variable of integer type, in the program. Following example depicts how to define it, to mention the minimum number of participants required for conducting the meeting.

```
meeting_quorum = 3
```

Figure 4.3: Sample code snippet for defining meeting quorum

When above mentioned inputs are defined by user, all required inputs are present in both Python programs [22] and [23]. In each Python program, there are five functions corresponding to five algorithm segments as depicted in table below.

Pure circuit version [22]	**Simplified version** [23]
Algorithm 1 (fig. 3.3)	Algorithm 1 (fig. 3.3)
Algorithm 2 (fig. 3.5)	Algorithm 2 (fig. 3.5)
Algorithm 3 (fig. 3.7)	Simplified version of algorithm 3 (fig. 3.14)
Algorithm 4 (fig. 3.9)	Algorithm 4 (fig. 3.9)
Algorithm 5 (fig. 3.12)	Algorithm 5 (fig. 3.12)

Table 4.2: Algorithm segments used for two versions of proofs of concept

Each function has parameterized inputs. Some of those inputs are from inputs defined by user as explained above. In both programs, these five functions are executed in sequential order, such that some outputs from initial functions are provided to later functions, as parameterized inputs. Following table depicts the name of Python function in code, corresponding to each algorithm explained earlier. In addition, output expected from execution of each function is also mentioned in the table.

#	Function name in code	Output of the function
3.3	`doc_analysis_validation`	Validity of each individual for meeting, by document analysis
3.5	`time_slot_and_participant_analysis`	Validity of each individual, in each time slot, by analyzing locations with expected privacy level for documents
3.7	`meeting_quorum_analysis`	Meeting quorum satisfiability of each time slot, by validated participants
3.14	`meeting_quorum_analysis`	Meeting quorum satisfiability of each time slot, by validated participants
3.9	`find_earliest_eligible_slot`	Earliest privacy preserving and quorum satisfying time slot for the meeting
3.12	`select_meeting_mode`	Meeting mode for each privacy preserving as well as quorum satisfying time slot identified

Table 4.3: Python functions corresponding to algorithm segments present in proofs of concept (# column represents the figure number of algorithm segment, while algorithm segment '6' (figure 3.14) represents the simplified version of algorithm segment '3' (figure 3.7))

As it is described in problem analysis section, *algorithm* 3 contains an operation to create mathematical combinations. Since that operation has exponential time complexity of $O(\sqrt{n^3} \times 2^n)$, it has been implemented using *combinations* function of *itertools* library in Python. Because, there is no specific standard method to create combinations within polynomial time. Accordingly, it can be stated that, *itertools* library was used by us, as an existing heuristic approach to create mathematical combinations.

When each program out of 2 programs is compiled and executed, following outputs are displayed in the console, as boolean outputs. Please note that values may change depending on the input. But, format of output will be as explained below.

```
Participant validity by doc analysis:
{'i_2': True, 'i_4': True, 'i_6': False, 'i_1': False,
'i_5': True, 'i_3': True}
```

Figure 4.4: Output depicting participants validated by document analysis

In above output, *true* or *false* states whether each individual is validated by analyzing access control lists of documents, including the agenda document as well. If an individual has no access to all the documents discussed in the meeting, particular individual will be marked as $false$.

```
Time Slot: slot1
    slot1_i_2_eligibility: True
    slot1_i_4_eligibility: True
    slot1_i_6_eligibility: False
    slot1_i_1_eligibility: False
    slot1_i_5_eligibility: False
    slot1_i_3_eligibility: True
```

Figure 4.5: Output depicting eligibility of each participant in a time slot, based on analysis of location with required level of privacy for documents

Above section of output displays the eligibility of each individual to attend the meeting, in each time slot. To decide the eligibility of an individual in a certain time slot, algorithm analyzes the location of that individual, with the expected level of privacy for documents in the meeting.

```
Slot: slot1 - Quorum Satisfiability: True
Slot: slot2 - Quorum Satisfiability: True
Slot: slot3 - Quorum Satisfiability: True
Slot: slot4 - Quorum Satisfiability: False
```

Figure 4.6: Output depicting meeting quorum satisfiability of each time slot

Above section of output states whether each time slot satisfies the meeting quorum, by having enough number of participants at privacy preserving locations, according to the expected level of privacy for documents in the meeting.

```
Slot 1 eligibility as earliest slot: True
Slot 2 eligibility as earliest slot: False
Slot 3 eligibility as earliest slot: False
Slot 4 eligibility as earliest slot: False
```

Figure 4.7: Output depicting earliest privacy preserving time slot for the meeting

In above section of output, that depicts the earliest quorum satisfying time slot for meeting, only one slot is supposed to have *true* value. Because, once after identifying a quorum satisfying time slot among all the time slots arranged in sequential order, algorithm

4 (figure 3.9) marks all other time slots after the identified time slot as, $false$. If no quorum satisfying time slot is present, all time slots will be marked as $false$, since it is impossible to identify an earliest eligible time slot.

```
Slot1:
    Eligibility online: False
    Eligibility hybrid: False
    Eligibility onsite: True
```

Figure 4.8: Output depicting meeting mode for each privacy preserving time slot identified

Above section of output depicts the eligible meeting mode for conducting the meeting, in each time slot. Algorithm 5 (figure 3.12) is designed such that more than one meeting mode cannot be activated (cannot be *true*) for a particular time slot. If a time slot is not suitable to conduct a privacy preserving meeting with equal or more number of participants than the meeting quorum value, all three meeting modes will get $false$ for that time slot, to display that meeting should not be conducted in particular time slot.

In overall, above overview describes the inputs, outputs and behavior of the Python programs, implemented as proofs of concept, associated with algorithms explained earlier. In our research, those algorithms are converted to the Python code such that maximum possible time complexities identified do not get exceeded.

4.1.4 Application of the Python program

Python programs developed by us as proofs of concept are available in GitHub ([22] and [23]). There are comments in the programs, for easily understanding its content. They can be cloned and executed by anyone in an environment supporting Python 3.7.0 version. Input sections can be edited before each execution, to observe the variation of result. Variation of results in different scenarios is explained with more details in the section below at 5.1.

4.1.5 Extension of the Python program

In addition to simple implementations explained above as the proofs of concept, extended versions of both Python programs were developed by us. Regarding them, [24] is the extended version of pure boolean circuit implementation [22], while [25] is the extended version of simplified version [23]. Each extended version has the capability to include aggregate participant groups, rather than only singleton sets, in access control lists of documents. Because, these extended versions contain the functionality of resolving aggregate participant groups, into their constituent participants. In addition, if a specific numerical meeting quorum is not defined as input, extended versions consider that all participants (the individuals defined as singleton sets and individuals after resolving aggregate participant groups except the "public" group) invited by agenda are required for the meeting. These extended versions are available in GitHub ([24] and [25]).

```
participant_groups = {
    "group_a": {"i_1", "i_2", "i_3"},
    "group_b": {"i_2", "i_3", "i_4"},
    "group_c": {"i_1", "i_5", "i_6", "i_7", "i_8"}
}
```

Figure 4.9: Definition of aggregate participant groups

For example, when aggregate participant groups are defined as above, and $access(agenda)$ is defined as below, then extended versions resolve $access(agenda)$ into its individual participants.

```
agenda = {"public", "group_a", "i_4", "i_5"}
```

Figure 4.10: Definition of $access(agenda)$

Extended versions resolve above access control list into following format. And this group resolving functionality is valid not only for agenda document, but also for any other document.

```
agenda = {"public", "i_1", "i_2", "i_3", "i_4", "i_5"}
```

Figure 4.11: Format of $access(agenda)$, after resolving aggregate participant groups

In this scenario, if any numerical meeting quorum is not defined as input, extended versions consider that $meeting\_quorum = 5$, since there are five individuals in $access(agenda)$, after resolving the groups.

4.2 System Design

4.2.1 Computational Complexity and Scope of Implementation

We identified earlier that our meeting scheduling problem belongs to the class of **P** problems. This means it can be solved efficiently using deterministic algorithms within polynomial time. Our system implementation is therefore entirely based on this problem classification and the hypotheses we have formulated, ensuring that the solution remains within the scope of tractable, real-time computation.

Our implementation, as explained in the following sections, is fully grounded in this theoretical foundation and tailored to address the practical aspects of a decision support system for meeting scheduling.

4.2.2 Technology Selection and Integration Feasibility Study

Before implementing the system, we conducted a comprehensive feasibility study to identify the most suitable tools and platforms for integrating our meeting scheduling decision support system. The goal was to ensure seamless interoperability with widely used scheduling ecosystems while maintaining robust access control, data privacy, and participant context-awareness.

Why Google Workspace?

After evaluating multiple scheduling solutions and cloud platforms, we identified **Google Workspace** as the most appropriate ecosystem for our system. The decision was influenced by the following key requirements:

- **Document Storage and Access Control**: Since our system deals with meeting-related documents and agendas, along with their associated access permissions, a secure and flexible document storage platform was essential. **Google Drive** emerged as the most viable option due to its ability to:
  - Support document sharing with both *individual users* and *Google Groups.*
  - Offer detailed sharing settings, including view-only or editing permissions.
  - Integrate with Google Groups, allowing group-based permissioning that simplifies access control for larger teams.

  Thus, Google Drive and Google Groups were selected for managing document storage and access resolution.

- **Calendar-based Participant Context and Availability**: A core requirement of our system is to retrieve each participant's availability and working location in hourly time slots. **Google Calendar** proved to be the most effective tool for this purpose. However, different versions of Google Calendar offer varying levels of functionality:
  - **Google One** and **personal Google Calendars** support basic sharing but lack contextual metadata such as location labels and availability types.
  - **Google Workspace Calendar**, on the other hand, provides advanced features including:
    - Location tagging (e.g., `Work`, `Home`, `Remote Public`) using custom or predefined labels.

  * Calendar statuses such as `Available`, `Out of Office`, and `Focus Time`, which are crucial for realistic and privacy-aware scheduling.

Therefore, Google Workspace Calendar was identified as the most suitable choice for our implementation, due to its superior capabilities in capturing and managing participant context.

However, this option introduced a significant constraint: **access to these advanced features requires an enterprise-level Google Workspace account with admin privileges**, which involves monetary costs and organizational setup that exceeded our current project scope.

Practical Implementation Decision

Due to the financial limitations of accessing Google Workspace, we proceeded with implementation using **personal Google Calendar accounts**. Although these lack some advanced context features (e.g., location tagging and status labels), they still support shared calendars and basic availability tracking, which allowed us to demonstrate the core functionality of our decision support system effectively.

Conclusion of Feasibility Study

In conclusion, the integration of Google Drive, Google Groups, and Google Calendar—adapted for our environment—forms the backbone of our system architecture. Each platform was selected to fulfill a specific functional requirement:

- Google Drive: Document storage and permission management.
- Google Groups: Simplified group-based sharing and access resolution.
- Google Calendar (personal): Hourly availability and scheduling feasibility with locations.

While Google Workspace remains the ideal long-term platform for deployment, our implementation with personal Google services serves as a valid and practical proof-of-concept for the proposed scheduling model.

4.3 System Implementation

This section describes the implementation of our Decision Support System (DSS), which integrates Google Workspace services—namely Google Drive, Google Groups, and Google Calendar—to support informed decision-making in the context of meeting organization and access control. Our system assists users in evaluating key parameters such as participant availability, document confidentiality, and access permissions before proposing feasible meeting arrangements.

The system's objective is to aid decision-makers in coordinating meetings that align with both organizational policies and contextual constraints, such as privacy requirements and collaborative roles. By extracting and analyzing structured data from Google Workspace, the system provides a comprehensive overview of the resources and participants involved, empowering users to make transparent, privacy-aware, and logistically feasible scheduling decisions.

Integration with Google Drive

Google Drive serves as the primary repository for all meeting-related documents, including the meeting agenda and any supporting materials discussed during the meeting. Each document within the shared folder is associated with metadata that specifies its confidentiality level—either public or private. In addition, the access permissions for each file are governed by Google Drive's native sharing settings, which define access at both the individual and group levels. These groups are managed via Google Groups, enabling role-based or team-based access control. The agenda file itself also includes a confidentiality label that reflects the overall privacy classification of the meeting.

Utilization of Google Groups

Google Groups is employed to manage collective access permissions efficiently. Instead of manually specifying access for each participant, users can assign documents to predefined groups (e.g., engineering team, project managers, external reviewers), allowing streamlined control over who can view or edit specific documents. This also enhances scalability and consistency in access control policies.

Integration with Google Calendar

Google Calendar is used to track employee availability and working locations. The calendar data is organized into one-hour time slots, where each slot contains availability status and work location details for individual employees. This structured calendar data enables the scheduling algorithm to make informed decisions based on when participants are free and whether they are working remotely or onsite.

4.3.1 Scheduling Workflow

The scheduling process begins when a user provides a link to a Google Drive folder. This folder contains the meeting agenda and associated documents. The system parses each file to extract access permissions and confidentiality labels, enabling it to determine who should be included in the meeting. The agenda's confidentiality label dictates the sensitivity of the meeting as a whole. Combined with calendar data and group membership information, the system automatically identifies appropriate time slots and manages secure access to all relevant materials.

4.3.2 Participant Determination Logic

To determine valid meeting participants, the system follows a structured four-step logic:

1. **Document Access Resolution**
    - Each document in the provided Google Drive folder may be shared with individual users or with Google Groups.
    - For documents shared with groups, the system queries its internal database to resolve each group into its individual members.
    - This ensures that all access permissions are evaluated at the individual user level.
2. **Intersection/Union Computation Based on Confidentiality**
    - If *all documents* are labeled as **public**, the system takes the **union** of all users who have access to any document.
    - If *any document* is labeled as **private**, the system computes the **intersection** of users who have access to *all* documents.
    - This step ensures appropriate access control based on document sensitivity.
3. **Agenda-Based Filtering**
    - The resulting participant set from the previous step is further **intersected** with users who have access to the agenda file.
    - This ensures that all final participants are authorized to access both the meeting documents and the agenda.
4. **Minimum Participation Check**
    - The final participant list must contain **at least two users**.
    - If fewer than two eligible participants are found, the meeting is considered invalid and is not scheduled.

4.3.3 Quorum Identification

In many organizational contexts, meetings require a minimum number of participants (a quorum) to proceed with decision-making. Our system provides flexible support for quorum identification as follows:

1. **Optional Quorum Requirement**
    - The user can specify whether a quorum is required for the meeting.

   - If quorum is not required, the system proceeds with the full list of eligible participants determined in the previous step.

2. **User-Defined Quorum Value**
   - If quorum is required, the user inputs the minimum number of participants needed.
   - This value is used as a threshold for evaluating available time slots.

3. **Validation Against Available Participants**
   - During availability computation, the system checks each potential time slot to ensure that the number of available participants meets or exceeds the quorum.
   - Time slots that do not satisfy the quorum requirement are excluded from final recommendations.

4.3.4 Time Slot Identification

After identifying valid participants and setting quorum (if required), the system determines feasible meeting time slots. This process considers participant availability, working location, meeting type (public or private), and a user-defined date range. The steps involved are as follows:

1. **Date Range Selection**
   - The user begins by specifying a feasible date range for the meeting.
   - This range defines the window within which the system searches for valid time slots.

2. **Calendar Data Retrieval**
   - The system fetches scheduled events for each selected participant within the specified date range by querying their Google Calendars.
   - Each event represents a one-hour time slot and includes the participant's availability for that time period, along with a location attribute.

3. **Location Categorization**
   - The system maintains each participant's home and work locations in its database.
   - Each 1-hour time slot in the date range is analyzed and classified into one of the following location categories:
     - **Onsite:** Participant is at their recorded work location.
     - **Remote Private:** Participant is at their home location.
     - **Remote Public:** Participant is at any other location.

4. **Meeting Type Consideration**
    - For *public* meetings, participants in any location (*onsite*, *remote private*, or *remote public*) are eligible.
    - For *private* meetings, only participants in *onsite* or *remote private* locations are considered, to uphold confidentiality.
5. **Quorum Validation (if required)**
    - If quorum is specified, the system checks each time slot to ensure that the number of available participants meets or exceeds the quorum.
    - Time slots that do not satisfy this condition are discarded.
6. **Final Time Slot Compilation**
    - After applying all constraints — date range, permissions, availability, meeting type, quorum, and location — the system compiles a list of valid time slots.
    - These are presented to the user as scheduling recommendations.

4.3.5 Privacy-Preserving Slot Selection via Average Security Score

To promote secure and privacy-conscious meeting environments, the system evaluates each valid time slot using an *Average Security Score* metric. This score quantifies the privacy level of a meeting based on the working locations of its participants.

Location Scoring Each participant's working location is mapped to a predefined numeric score reflecting its privacy level:

- **Work (Most Secure)**: 1
- **Home**: 2
- **Remote Public (Least Secure)**: 3

Computation Method For each time slot where the meeting quorum is satisfied:

1. Filter the list of participants involved in that time slot.
2. Map each participant's location label to its numeric score.
3. Compute the average of these scores using the formula:

$$\text{Average Security Score} = \frac{\sum_{i=1}^{n} s_i}{n}$$

where s_i is the score for the i^{th} participant and n is the number of participants in the slot.

Example Consider a time slot with three participants:

Participant	**Location**
A	Work (1)
B	Home (2)
C	Remote Public (3)

The Average Security Score is computed as:

$$\frac{1+2+3}{3} = 2.0$$

Interpretation

- A lower score indicates a more secure meeting environment.
- Among feasible time slots, the one with the lowest Average Security Score is recommended to preserve participant privacy.

Chapter 5

Results of Execution

5.1 Results of execution of Python programs

When considering the results of Python programs, both programs including the pure boolean circuit [22] and simplified version of boolean circuit [23] generate the same results, when inputs are same. Because, they both fulfill the same functionality, following different algorithms. Therefore, in this section of analyzing results, hereafter both versions are generally referred as **"program"**.

As explained in code overview (section 4.1.3), program contains three inputs including the access control lists of documents, availability details of individuals in time slots (availability dictionary) and the meeting quorum. When these inputs are edited and program is executed, it can be observed that output changes. Possible outputs of the program can be summarized as below.

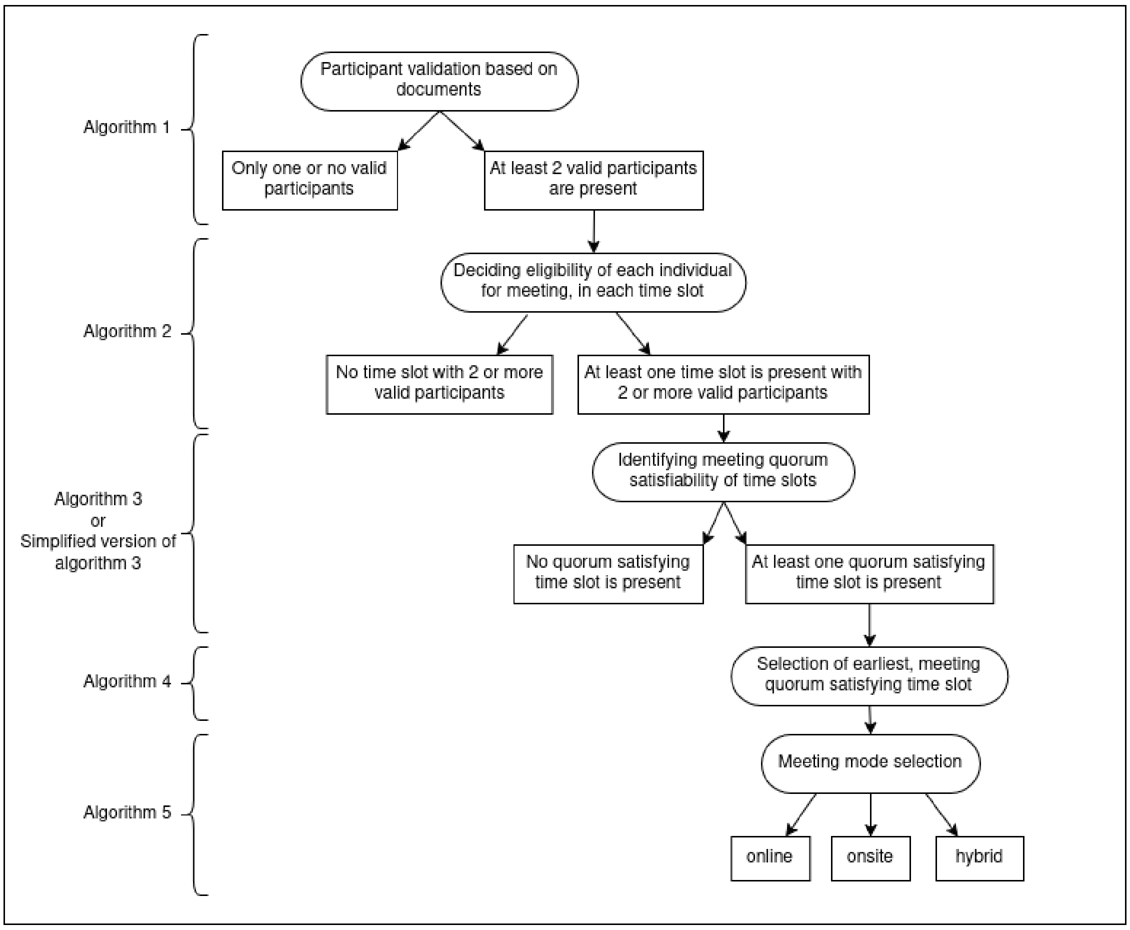

Algorithm 1
Participant validation based on documents
Only one or no valid participants
At least 2 valid participants are present
Algorithm 2
Deciding eligibility of each individual for meeting, in each time slot
No time slot with 2 or more valid participants
At least one time slot is present with 2 or more valid participants
Algorithm 3 or Simplified version of algorithm 3
Identifying meeting quorum satisfiability of time slots
No quorum satisfying time slot is present
At least one quorum satisfying time slot is present
Algorithm 4
Selection of earliest, meeting quorum satisfying time slot
Algorithm 5
Meeting mode selection
online
onsite
hybrid

Figure 5.1: Possible results of program execution

In addition, though each negative output shows the end of process in above diagram for simplicity, when considering the boolean circuit, processing does not stop abruptly in that manner. Instead of stopping, after a reaching a certain negative output, all remaining outputs also become negative ($false$), in boolean circuit. Three negative outputs identified in the program execution are listed below.

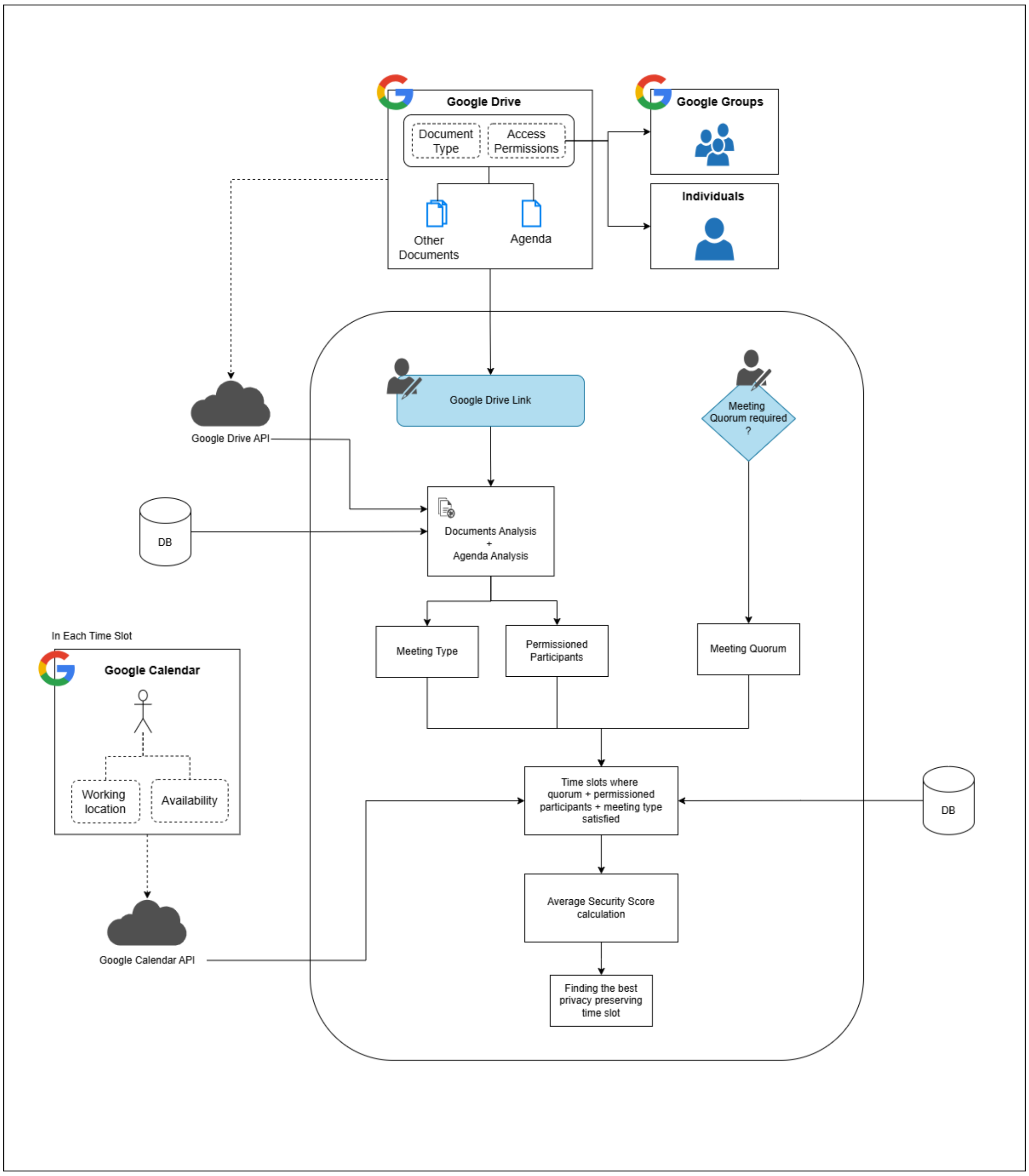
- *Algorithm* 1: Only one or no valid participants
- *Algorithm* 2: No time slot with 2 or more valid participants
- *Algorithm* 3: No quorum satisfying time slot is present

Accordingly, for reaching a positive output which means that meeting can be conducted, none of above three negative outputs should be encountered in program execution. In other words, if program approaches at least one negative output among above three, then it is impossible to conduct the particular meeting as a privacy preserving meeting.

In our research, we have considered variation of output with changes in input, to check whether they validate our hypothesis. After breaking down possible outputs of algorithm segments as depicted above, sections of output supporting to validate the hypothesis were analyzed. Here, it is needed to note that, some sections of output are not required to validate the hypothesis. Such sections were introduced by us, as additional contributions, to enhance the quality of research outcome.

5.2 Results of execution of public platform based implementation

This section outlines the functioning and practical results of the proposed system after deployment. The system acts as a decision support tool for privacy-aware meeting scheduling by integrating multiple Google Workspace services. The complete workflow of the application is illustrated in the architecture diagram (see figure 5.2).



Google Drive
Document Type
Access Permissions
Other Documents
Agenda
Google Groups
Individuals
Google Drive API
DB
Google Drive Link
Meeting Quorum required ?
Documents Analysis + Agenda Analysis
Meeting Type
Permissioned Participants
Meeting Quorum
In Each Time Slot
Google Calendar
Working location
Availability
Google Calendar API
Time slots where quorum + permissioned participants + meeting type satisfied
DB
Average Security Score calculation
Finding the best privacy preserving time slot

Figure 5.2: System Architecture Overview

5.2.1 Execution Flow Overview

The system's execution begins when a user submits a Google Drive folder link containing meeting-related documents and an agenda file. The system then performs the following key steps:

1. **Access Resolution and Participant Identification**: It parses document permissions (individual and group-based), resolves Google Groups into users, and applies confidentiality rules (union or intersection logic based on document privacy) to determine the eligible participant set.

2. **Quorum Validation (Optional)**: If the user specifies a required quorum, the system filters out slots that do not meet the minimum participant count.

3. **Date Range Filtering**: Users select a desired date range, which restricts the scheduling search to relevant time windows.
4. **Availability and Location Retrieval**: The system queries Google Calendar for each eligible participant to obtain their hourly availability and working location within the selected range.
5. **Time Slot Matching**: Based on the meeting type (public or private), the system evaluates each slot for whether all (or any) participants are simultaneously available, satisfying quorum and location constraints.
6. **Privacy-Aware Scoring**: For every feasible slot, an *Average Security Score* is computed based on participant location types (Work, Home, or Remote Public), aiding in selection of the most privacy-preserving option.
7. **Final Output**: The system outputs a ranked list of valid time slots, highlighting the slot with the lowest average security score as the most suitable choice. Metadata such as participant emails, locations, and score breakdowns are included for transparency.

5.2.2 Outcome and Usability

The application efficiently streamlines the decision-making process around scheduling secure and inclusive meetings. It offloads the cognitive burden of manually verifying access, balancing participation, ensuring confidentiality, and aligning availability—making it highly practical in collaborative, privacy-sensitive work environments.

The successful execution demonstrates how contextual integration of Drive, Calendar, and Groups can power robust decision support for modern scheduling scenarios.

Chapter 6

Evaluation

6.1 Foundation of evaluation

In our research, as our contribution, initially entities in privacy-preserving meeting context were defined formally, using standard notations. After that, using those formal definitions as the foundation, research problem was mapped to a boolean circuit for analyzing it easily. Specially for analyzing the complexity of the problem, boolean circuit was depicted in algorithmic representation. Time complexity of the problem was calculated by considering the quantity of input variables and loops present in that algorithmic representation.

Since it was not possible to solve the problem within polynomial time using pure boolean circuit, a simplified version of the algorithm was developed. In this version, only one algorithm segment is different, out of five algorithm segments present in pure boolean circuit.

After that, we implemented algorithmic representation of pure boolean circuit and its simplified version both, using Python. And then outputs of the Python programs were manually checked, by setting the inputs. Those results were cross-validated with the outputs of algorithms, to ensure that implemented Python programs align with the algorithms. Then possible results of the execution of Python programs regarding selection of eligible participants, eligible time slot and eligible meeting mode, were discussed. And, we highlighted when a negative output showing that a privacy-preserved meeting is not possible will be delivered. These results were discussed, considering the segments of the algorithms, for aligning with the algorithmic level analysis of the problem.

Later, an analytical investigation (section 6.2) was conducted on results obtained by execution of Python programs. This investigation is for showing how segments of algorithm contribute to validate the hypothesis built in the beginning of research. Further a sample data set (section 6.3) was introduced to observe the variation of results in Python program execution. Since we have published our Python program implementation and this sample dataset in GitHub, anyone can test to check the validity.

Using existing public platforms such as Google-API, a simple system was implemented to prove that privacy-preserved meeting organization is practically viable. Therefore, it can be confirmed that, this is not limited to a theoretical concept. But, this is a practical approach, which can be improved further in future, to organize privacy-preserved meetings in practical context.

Accordingly, it is observed that there is a flow of logically related steps in this research, which started from basic set of definitions, and was developed up to a practical application that can be tested by anyone. Therefore, it can be determined that, there is no mismatch in intermediate steps, followed in this research.

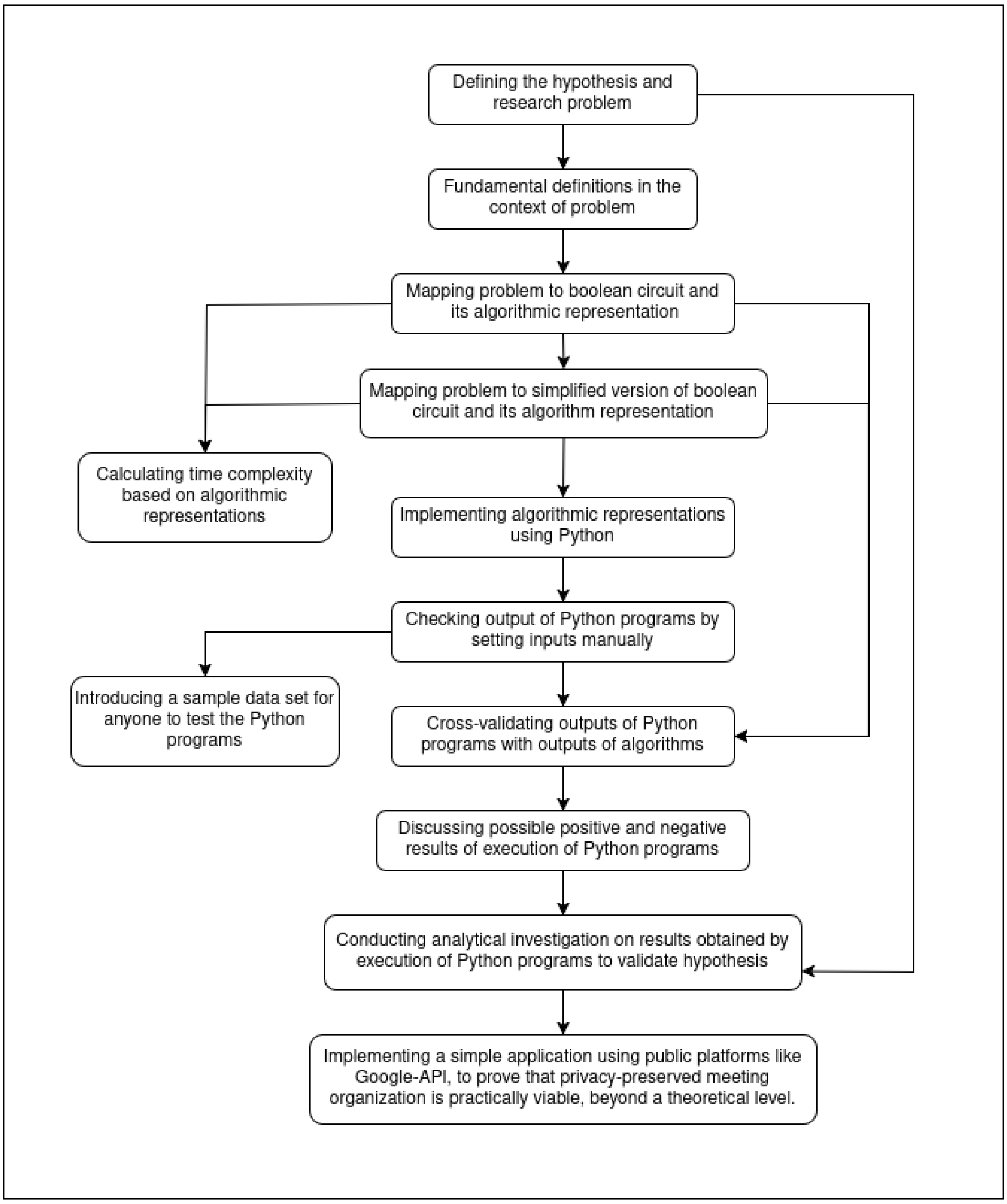

Defining the hypothesis and research problem
Fundamental definitions in the context of problem
Mapping problem to boolean circuit and its algorithmic representation
Mapping problem to simplified version of boolean circuit and its algorithm representation
Calculating time complexity based on algorithmic representations
Implementing algorithmic representations using Python
Checking output of Python programs by setting inputs manually
Introducing a sample data set for anyone to test the Python programs
Cross-validating outputs of Python programs with outputs of algorithms
Discussing possible positive and negative results of execution of Python programs
Conducting analytical investigation on results obtained by execution of Python programs to validate hypothesis
Implementing a simple application using public platforms like Google-API, to prove that privacy-preserved meeting organization is practically viable, beyond a theoretical level.

Figure 6.1: Flow of logically related steps in our contribution of research

6.2 Analytical investigation on the results of algorithm

When *algorithm* 1 (figure 3.3) is considered, its output shows whether each individual is allowed to discuss all documents included in the meeting. In this case, we consider having at least two valid participants as the positive output, since at least two participants are required for conducting a meeting. This decision is made by analyzing the access control lists of documents associated with the meeting, including the meeting agenda. Following flow chart depicts the process within *algorithm* 1 (figure 3.3), to generate the expected output.

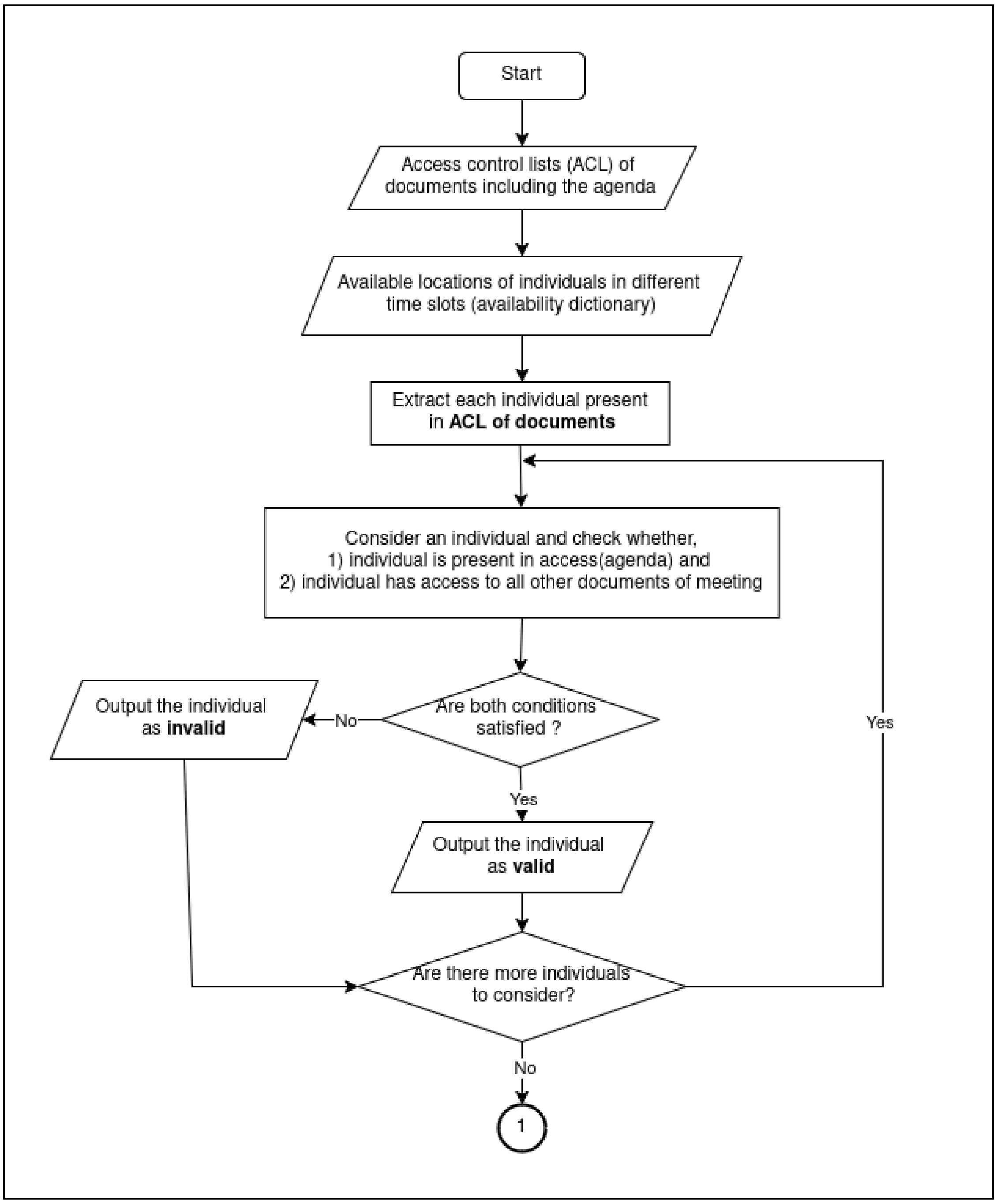

Start
Access control lists (ACL) of documents including the agenda
Available locations of individuals in different time slots (availability dictionary)
Extract each individual present in **ACL of documents**
Consider an individual and check whether, 1) individual is present in access(agenda) and 2) individual has access to all other documents of meeting
Are both conditions satisfied ?
No
Output the individual as **invalid**
Yes
Output the individual as **valid**
Are there more individuals to consider?
Yes
No
1

Figure 6.2: Participant validation by document analysis in *algorithm* 1

Output of *algorithm* 2 (figure 3.5) shows whether each individual is allowed to discuss all documents of the meeting, in each time slot. As discussed in problem analysis section, to make this decision, program checks whether at least one private document is present in meeting. If at least one private document is present, an individual present at a public location in a certain time slot is not allowed to attend this meeting, in that time slot. However, whether a document is private or public (by presence of "*public*" group), is stated in the access control list of the document. Following flow chart depicts the process within *algorithm* 2 (figure 3.5), to generate the expected output.

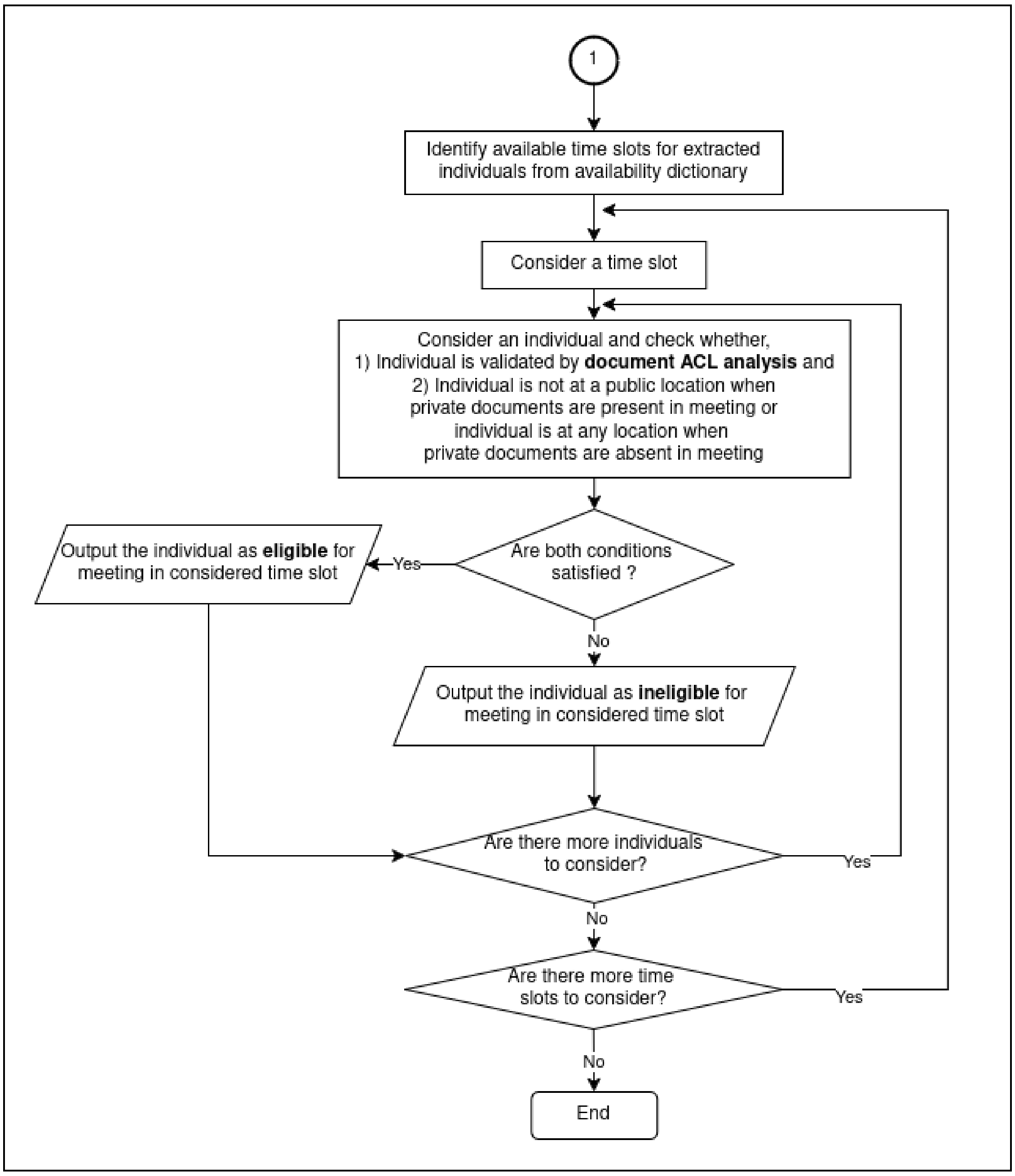

1
Identify available time slots for extracted individuals from availability dictionary
Consider a time slot
Consider an individual and check whether,
1) Individual is validated by **document ACL analysis** and
2) Individual is not at a public location when private documents are present in meeting or individual is at any location when private documents are absent in meeting
Are both conditions satisfied ?
Yes
Output the individual as **eligible** for meeting in considered time slot
No
Output the individual as **ineligible** for meeting in considered time slot
Are there more individuals to consider?
Yes
No
Are there more time slots to consider?
Yes
No
End

Figure 6.3: Deciding eligibility of each individual in *algorithm* 2

By observing these two segments of flow chart connected by a connector, it can be understood that access control lists of documents play a vital role as an essential input, to decide the eligible meeting participants. It means that, **"Meeting participant selection is primarily dominated by the access control lists of the documents presented including meeting agenda"** expression is validated by analyzing the process within *algorithm* 1 and *algorithm* 2 collectively, as in above flow chart segments.

After identifying the eligibility of each individual to attend the meeting in each time slot, next result expected from the program is whether each time slot satisfies the meeting quorum. Though meeting quorum is not relevant to privacy of meeting data, it is an important feature in meeting domain, to determine whether minimum required number of participants is present in meeting. In our research, *algorithm* 3 (figure 3.7) and *simplified version of algorithm* 3 (figure 3.14) enhance the quality of research outcome, by checking whether each time slot satisfies the meeting quorum, with enough eligible meeting participants, for a privacy-preserved meeting. In other words, our requirement is to satisfy the meeting quorum, without violating the privacy of meeting data. Accordingly, identification of quorum satisfying time slots as well, depends on access control lists of documents of the meeting. Meeting quorum value should be be defined by user as an input, in program.

After identifying quorum satisfying time slots, it is required to identify the earliest one among them. If there are more than one quorum satisfying time slots, *algorithm* 4 (figure 3.9) selects the earliest one among them. If only one time slot satisfies the meeting quorum, it becomes the earliest one. However, since *algorithm* 4 (figure 3.9) depends on *algorithm* 3 depicted by figure 3.7 (or *simplified version of algorithm* 3 depicted by figure 3.14), it can be stated that, outcome of *algorithm* 4 (figure 3.9) indirectly depends on access control lists of documents. Result of *algorithm* 3 depicted by figure 3.7 (or *simplified version of algorithm* 3 depicted by figure 3.14) and *alogorithm* 4 (figure 3.9) do not support to validate the hypothesis, since these two algorithms do not focus on privacy of meeting data.

After quorum satisfying time slots are identified by *algorithm* 3, *algorithm* 5 decides the privacy-preserving meeting mode for each of such time slots, by analyzing the locations of eligible participants. Every quorum satisfying time slot belongs to one meeting mode out of three modes—onsite, online and hybrid. Essential input required to generate the meeting mode of a time slot as output, is the available location of every eligible participant, in the particular time slot. The logic for selecting the meeting mode is discussed in “meeting mode selection” (3.3.5) section of problem analysis.

Accordingly, when logic used for meeting mode selection is analyzed, it can be observed that locations of participants decide the meeting mode of a particular time slot. Therefore, **“the choice of meeting mode depends on the participants’ locations.”** phrase of hypothesis is validated by *algorithm* 5 (figure 3.12). In ensemble, it can be concluded that, hypothesis is validated by our algorithm.

6.3 Relationship between input and the result

To observe the way how relationship between input and output supports the hypothesis as explained above, a sample data set has been designed. That sample data set is available in GitHub [26]. The data set has been designed such that availability dictionary is same for all scenarios. Only access control lists of documents and meeting quorum are changed, for observing the variation of output. There are comments in the data set, for applying data in either of two programs (pure boolean circuit [22] or simplified version [23]) conveniently, by a user. However, since our data set is not an exhaustively large data set, anyone can extend this data set to include more scenarios, and execute the program to observe how result changes from scenario to scenario. To validate the analytical investigation of result presented by us above, it is required to observe whether results align with the hypothesis, in each execution.

Chapter 7

Conclusion

In modern organizations, scheduling meetings is not just about finding a common time slot. It is about ensuring that the right people are in the right place, with the right access, and under the right constraints. Traditional scheduling systems fall short when privacy of meetings comes into play. This research began with the central problem: How can we create a privacy-preserved meeting scheduling model that respects document access permissions, adapts to participant locations, and selects the appropriate mode (virtual, physical, or hybrid), all while preserving the privacy of the meeting? We proposed a novel framework that integrates access control logic (via document ACLs) and participant location constraints to determine both participant eligibility and suitable meeting modes (physical, virtual, or hybrid). Our work aimed to provide a practical, efficient, and privacy-aware solution that moves beyond traditional time-slot matching.

Our literature review uncovered a fragmented research landscape. While several constraint-based models—such as MIP, CP, and SAT-based methods as used in [2] have been successfully used for structured scheduling tasks, these approaches often lack real-time adaptability and do not inherently preserve privacy.

Adaptive models like Genetic Algorithms (GAs) and Theory of Constraints (TOC) as used in [13] offer flexibility but suffer from high computational costs and lack mechanisms to safeguard sensitive participant data. On the other hand, privacy-centric distributed models such as DCOP and DVCSP as used in [11] enhance confidentiality and user privacy but struggle with scalability and optimality, particularly in high-density participant environments. Our literature review highlighted how early research using computational geometry achieved notable efficiency but did not account for privacy. Techniques like Cube-and-Conquer as used in [17] and Soft Conflict Pseudo Boolean (SPB) constraints from SAT and MaxSAT as used in [2] research have pushed the boundaries in constraint satisfaction and adaptability. However, these still do not fully resolve the inherent conflict between speed and meeting privacy. We can clearly see that a consistent limitation emerges here. No existing framework adequately balances scheduling efficiency, flexibility, and privacy.

Recognizing this research gap, our solution introduced a hybrid model that integrates access control-based participant filtering, location-aware scheduling, and agenda sensitivity-based meeting mode resolution. Technically, we began by modeling the core logic of the problem using Boolean circuits to assess complexity and isolate exponential components. We then transitioned it to a polynomial-time algorithm, which replaced the non-scalable parts while preserving logical integrity. Our final complexity analysis placed the overall problem in class P, enabling practical, real-world use.

To validate the feasibility of our approach, we developed a working prototype in Python, integrated with Google Calendar APIs to simulate real-world deployment. This prototype successfully demonstrated our model's ability to automate participant selection, enforce access-based eligibility, and determine appropriate meeting modes while ensuring privacy boundaries of the meeting.

While our contributions mark a step forward, a critical analysis reveals areas for growth:

- We assumed document ACLs are well-structured and that user location data is available and reliable. However, in real-world organizational systems, these inputs are often inconsistent, outdated, or incomplete. This assumption limits the generalizability of our current solution.

- Our prototype was evaluated in a controlled environment with a limited number of participants and documents. Its performance in large-scale enterprise environments—where constraints are highly dynamic—remains untested. Further benchmarking is needed to assess its robustness and responsiveness under load.

- While our focus was on constraint satisfaction and feasibility, we did not address multi-objective optimization such as minimizing schedule conflicts or maximizing participation diversity. Future iterations could include optimization metrics and support for soft constraints.

- The user-facing component of our prototype remains basic. A practical scheduling tool must be not only functionally correct but also user-friendly, offering intuitive interfaces, visual explanations of scheduling logic, and flexible override mechanisms.

Our work directly addresses the gap highlighted in the literature: the absence of integrated, privacy-preserving, and computationally feasible scheduling models. Unlike prior approaches that optimize either efficiency or privacy, but not both, we provide a path for balanced integration.

In conclusion, our research provides the groundwork for a new class of smart, privacy-conscious scheduling systems. We have shown that it is possible to reconcile access control, location-awareness, and agenda sensitivity within a scalable framework. Yet, realizing the full vision requires extending our model to be dynamic, adaptive, and human-centered. At the end of this research, we believe we have taken an important first step toward building practical, intelligent, and privacy-aware scheduling systems. We are convinced that the future of scheduling lies not in isolated improvements to optimization techniques or user interfaces, but in integrated, context-aware solutions that reflect how people actually collaborate securely in a flexible way, and often across physical and virtual boundaries. We also recognize that our contribution is part of a much broader evolution in scheduling research. The field is shifting from static rule-based systems toward adaptive, personalized, and policy-driven automation. Our work contributes to this trajectory by showing that such a shift is not only possible, but computationally achievable. Going forward, we are excited about extending this work in several directions such as scaling it across organizations, integrating predictive models, and improving user interaction to realize the full vision of smart, privacy-conscious scheduling as a service.

Chapter 8

Limitations and Future Work

As privacy-preserved meeting organization is a relatively less-explored research area, this work required a thorough investigation into existing meeting organization techniques and related literature. Prior to formulating our approach, we conducted an in-depth study to ensure that no existing solutions effectively addressed the privacy aspects in meeting scheduling. This foundational understanding not only helped shape the direction of our research but also highlighted the novelty and relevance of our contribution. There are certain limitations in our current work, and several promising directions remain open for future exploration.

One notable limitation of our research lies in the practical implementation aspect, specifically in the integration of Google Workspace services. Although we intended to incorporate Google Workspace in a more advanced and comprehensive manner to enhance the usability and scalability of the application, we were constrained by the associated costs of accessing and utilizing the full range of Google APIs and enterprise-level services. As a result, the current implementation is limited to a simpler version that demonstrates the core functionalities without leveraging the complete potential of the Google Workspace ecosystem. Future work may address this limitation by exploring funding opportunities or alternative platforms with similar capabilities.

Another limitation in our research lies in the simplification of participant location categorization. For the sake of implementation feasibility and conceptual clarity, all participant locations were broadly classified into three main categories: onsite, remote private, and remote public. While this categorization was sufficient for our proposed model, it does not fully capture the complexity of real-world scenarios, where accurately identifying and verifying participant locations can be a challenging task due to factors such as mobility, dynamic availability, and privacy concerns. Mostly a real-time location verification method may be required for practical deployment in diverse and dynamic environments.

An additional limitation of this research is the absence of a real-world survey involving meeting participants from various organizations to better understand their perceptions of privacy in the context of meeting data. While such a survey could have provided valuable insights into practical privacy concerns and user expectations, conducting it proved to be challenging. The primary reason was the general reluctance of individuals to disclose sensitive details about their meetings, which raised concerns about participant engagement and the reliability of any results obtained. Consequently, the our research relies more on theoretical and technical assumptions regarding privacy, rather than empirical evidence drawn from organizational settings.

As a study situated within the information systems domain, this research was designed with a strong emphasis on demonstrating the practical applicability of its results. To this end, we were motivated to develop a working application using publicly available platforms, which serves as a proof of concept for the proposed privacy-preserved meeting organization framework. However, the development and integration effort required for this implementation placed significant demands on the research timeline. As a result, we were unable to allocate sufficient time for a formal verification of the problem mapping process. Formal verification is essential to ensure the correctness and consistency of the theoretical model and its mapping to computational constructs. Therefore, a comprehensive formal verification remains an open task and is recommended as future work to validate and potentially refine the current problem formulation.

In this research, we have shown that the proposed problem is solvable in polynomial time, using a Boolean circuit-based approach to model the constraints and decision logic. While effective, this method may not be the most optimized in terms of computational efficiency. There is potential for exploring alternative polynomial-time approaches such as graph-based models and integer programming that may yield more efficient or scalable solutions. Therefore, identifying and evaluating such alternative methods remains a promising direction for future work.

In this research, we focused on organizing a meeting based on a given set of documents, assuming that all documents need to be discussed within a single session. However, in real-world scenarios, it may not always be feasible to cover all materials in one meeting due to privacy contraint emphasized by us. In addition, there can be many other meeting related constraints such as time constraints, participant availability, and document complexity. In such cases, it becomes necessary to intelligently divide the documents across multiple meetings, while still preserving privacy of meeting data included in documents. This introduces an additional layer of complexity, transforming the problem into an optimization challenge. Solving this would require a detailed analysis of document access permissions and participant roles. A heuristic or metaheuristic-based approach could be effective for addressing this challenge, especially in large-scale or dynamic settings. Therefore, this extension represents a meaningful and practical future research direction.

Another promising direction for future work is to extend the privacy-preserved meeting organization problem by incorporating additional real-world constraints that were not addressed in the current study. For instance, meeting precedence—where certain meetings must occur before others due to dependency in topics or decision-making processes—is a common scheduling requirement in organizational settings. Additionally, the number of available onsite physical locations can significantly impact scheduling feasibility, especially when accommodating hybrid or in-person meetings with space limitations. These types of constraints introduce new layers of complexity, requiring more advanced constraint satisfaction or optimization techniques. Relevant work in the literature, such as that by Bofill et al. [2], has explored constraint optimization problems involving similar real-world conditions, but without a focus on privacy preservation. Integrating such constraints into the privacy-preserved framework could enhance the practicality and applicability of the solution, making it more adaptable to diverse organizational needs.

In our research, the concept of meeting quorum is introduced as an additional constraint to ensure that a minimum required number of participants are available to attend the meeting within a given time slot. This basic quorum check serves as a foundational requirement for ensuring that the meeting has sufficient representation to be meaningful and productive. However, this constraint can be further refined by considering not only the overall minimum number of participants but also the inclusion of essential participants whose presence is critical for the success or purpose of the meeting. In some cases, certain participants may hold key roles, such as decision-makers or subject matter experts, whose absence could render the meeting ineffective, regardless of the overall quorum. By incorporating such “must-have” participants into the quorum check, we can ensure that meetings are not only numerically sufficient but also functionally viable. This improvement would lead to a more robust and realistic model for meeting scheduling and participant allocation, making it another promising avenue for future research.

As part of our research, we developed a simple application using Google APIs to demonstrate the practical applicability of our privacy-preserved meeting organization solution. This prototype serves as a proof of concept, validating the feasibility of the approach while ensuring privacy protection. However, this initial version is basic and focused on core functionalities. Future work could involve developing a more advanced application with enhanced features, such as refined privacy controls, integration with additional platforms, improved user interfaces, and better scalability to support larger organizations and more complex meeting scenarios.

Therefore, it can be said that our research represents a pioneering initiative in the relatively under-explored domain of privacy-preserved meeting organization. By addressing the intersection of meeting organization and privacy concerns, our work introduces new perspectives and methodologies in a field that has received limited attention in the literature. This research not only lays the groundwork for future advancements in secure and efficient meeting management but also opens up new avenues for integrating privacy preservation with organizational processes, which could have significant implications for industries where confidentiality and data security are paramount.

References

[1] J. Jo, Y. Chae, H. Jang, and J. Kong, "Federated-access management system and videoconferencing applications: Results from a pilot service during covid-19 pandemic," *Electronics*, vol. 10, p. 2239, 09 2021.

[2] M. Bofill, C. Coll, M. Garcia, J. Giraldez-Cru, G. Pesant, J. Suy, and M. Villaret, "Constraint solving approaches to the business-to-business meeting scheduling problem," *Journal of Artificial Intelligence Research*, vol. 74, pp. 263–301, 2022. Accessed 29-01-2025, `https://dl.acm.org/doi/pdf/10.1613/jair.1.12670`.

[3] G. Golkarnarenji and U. Ali, "A study of it personnel awareness on video conferencing security," tech. rep., Department of Computer Science, Electrical and Space Engineering, Luleå University of Technology, 2012. Accessed: 2025-03-06, `https://www.academia.edu/3056220/Unified_Communications_Security_A_study_of_IT_personnel_awareness_on_video_conferencing_security_recommendations`.

[4] A. Culafi, "Microsoft ai researchers mistakenly expose 38 tb of data," *TechTarget*, September 2023. Accessed: 2025-03-02, `https://www.techtarget.com/searchsecurity/news/366552399/Microsoft-AI-researchers-mistakenly-expose-38-TB-of-data`.

[5] M. Ozer, Y. Kose, G. Kucukkaya, and E. Varlioglu, "The shifting landscape of cybersecurity: The impact of remote work and covid-19 on data breach trends," *IEEE Computer Science and Computer Engineering (CSCE)*, 2024. Accessed: 2025-03-06, `https://arxiv.org/html/2402.06650v2`.

[6] M. Bispham, S. Creese, W. Dutton, P. Esteve-Gonzalez, and M. Goldsmith, "Cybersecurity in working from home: An exploratory study," *SSRN Electronic Journal*, 2021. Accessed: 2025-03-06, `https://www.researchgate.net/publication/353661008_Cybersecurity_in_Working_from_Home_An_Exploratory_Study`.

[7] R. Jennings, "Zoom fails grow: 530,000 passwords leaked, details for sale by hacker," 2021. Accessed: 2025-03-06, `https://techbeacon.com/security/zoom-fails-grow-530000-passwords-leaked-details-sale-hacker`.

[8] S. Salloum, T. Gaber, S. Vadera, and K. Shaalan, "Phishing email detection using natural language processing techniques: A literature survey," *Procedia*

Computer Science, vol. 189, pp. 19–28, 2021. Accessed: 2025-03-06, https://www.sciencedirect.com/science/article/pii/S1877050921011741.

[9] Condeco, "How to decide who should attend your meeting?," 2024. Accessed: 2025-03-06, https://www.cnbc.com/2021/08/01/zoom-reaches-85-million-settlement-over-user-privacy-and-hacker-zoombombing.html.

[10] T. Tsuruta and T. Shintani, "Scheduling meetings using distributed valued constraint satisfaction algorithm." No publication year available.

[11] F. Enembreck and J.-P. A. Barthès, "Distributed constraint optimization with mulbs: A case study on collaborative meeting scheduling," *Journal of Network and Computer Applications*, vol. 35, no. 1, pp. 164–175, 2012. Accessed: 2025-03-06, https://www.sciencedirect.com/science/article/pii/S1084804511000543.

[12] F. Berger, R. Klein, D. Nussbaum, J.-R. Sack, and J. Yi, "A meeting scheduling problem respecting time and space," *Geoinformatica*, vol. 13, no. 4, pp. 453–481, 2009. Accessed: 2025-03-06, https://doi.org/10.1007/s10707-008-0053-4.

[13] V. V. Peteghem and M. Vanhoucke, "A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problem," *European Journal of Operational Research*, vol. 201, no. 2, pp. 409–418, 2010. Accessed: 2025-03-06, https://www.sciencedirect.com/science/article/pii/S037722170900191X.

[14] J. Zheng, Z. Chen, C.-M. Li, and K. He, "Rethinking the soft conflict pseudo boolean constraint on maxsat local search solvers," *arXiv preprint*, 2024. Accessed: 2025-03-06, https://arxiv.org/abs/2401.10589.

[15] E. M. Goldratt, "What is this thing called theory of constraints and how should it be implemented?," 1984. Accessed: 2025-03-06, https://www.academia.edu/7095271/Theory_of_Constraints_Eliyahu_M_Goldratt.

[16] D. Golmohammadi, "A study of scheduling under the theory of constraints," *International Journal of Production Economics*, vol. 165, pp. 38–50, 2015. Accessed: 2025-03-06, https://www.sciencedirect.com/science/article/pii/S0925527315000833.

[17] M. J. H. Heule, O. Kullmann, S. Wieringa, and A. Biere, "Cube and conquer: Guiding cdcl sat solvers by lookaheads," in *Hardware and Software: Verification and Testing* (K. Eder, J. Lourenço, and O. Shehory, eds.), vol. 7261 of *Lecture Notes in Computer Science*, pp. 50–65, Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

[18] A. Popescu, S. P. Erdeniz, A. Felfernig, M. Uta, M. Atas, V. M. Le, K. Pilsl, M. Enzelsberger, and T. Tran, "An overview of machine learning techniques in constraint solving," *Journal of Intelligent Information Systems*, vol. 58, no. 1, pp. 91–118, 2022.

[19] C. P. Pfleeger, "Chapter 3: Security encryption systems," in *Security in Computing, Second Edition* (C. P. Pfleeger, ed.), ch. 3, pp. 69–76, Upper Saddle River, NJ: Prentice Hall, 2 ed., 1996.

[20] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Fourth Edition.* MIT Press, 2022. Accessed: 2025-04-04, `https://books.google.lk/books/about/Introduction_to_Algorithms_fourth_editio.html?id=HOJyzgEACAAJ&redir_esc=y`.

[21] W. L. Hosch, "P versus NP problem — Complexity Theory & Algorithmic Solutions — Britannica — britannica.com," 2024. Accessed 29-01-2025, , `https://www.britannica.com/science/P-versus-NP-problem`.

[22] T. Akmeemana, "sympy_solver.py," 2025. Accessed: 2025-02-07, `https://github.com/Thejana-A/IS-4101-git/blob/master/python_code/BOOLEAN_SOLVERS/sympy_solver/final_version/sympy_solver.py`.

[23] T. Akmeemana, "sympy_solver_simplified.py," 2025. Accessed: 2025-02-07, `https://github.com/Thejana-A/IS-4101-git/blob/master/python_code/BOOLEAN_SOLVERS/sympy_solver/final_version/sympy_solver_simplified.py`.

[24] T. Akmeemana, "sympy_solver_with_group_resolver.py," 2025. Accessed: 2025-03-08, `https://github.com/Thejana-A/IS-4101-git/blob/master/python_code/BOOLEAN_SOLVERS/sympy_solver/final_version/sympy_solver_with_group_resolver.py`.

[25] T. Akmeemana, "sympy_solver_simplified_with_group_resolver.py," 2025. Accessed: 2025-03-08, `https://github.com/Thejana-A/IS-4101-git/blob/master/python_code/BOOLEAN_SOLVERS/sympy_solver/final_version/sympy_solver_simplified_with_group_resolver.py`.

[26] T. Akmeemana, "Circuit result analysis data set," 2025. Accessed: 2025-02-07, `https://github.com/Thejana-A/IS-4101-git/blob/master/latex/CircuitResultAnalysis_DataSet`.

[27] U. of Wollongong, "Permutations & combinations hsc revision," 2025. Accessed: 2025-02-26, `https://documents.uow.edu.au/content/groups/public/@web/@eis/@maas/documents/mm/uow168693.pdf`.

[28] U. of Newcastle, “Permutations and combinations,” 2021. Accessed: 2025-02-26, https://www.newcastle.edu.au/__data/assets/pdf_file/0004/819139/Permutations-and-Combinations.pdf.

[29] W. L. Hosch, “Stirling’s formula,” 2025. Encyclopedia Britannica, Accessed: 2025-03-05, https://www.britannica.com/science/Stirlings-formula.

Appendices

A Maximum value of $n\mathrm{C}r$

The following proof shows that $n\mathrm{C}r$ is maximum when $r = \frac{n}{2}$ or $r \approx \frac{n}{2}$.

$$\binom{n}{r} = \frac{n!}{(n-r)!r!}$$

$$\binom{n}{r+1} = \frac{n!}{(r+1)!(n-r-1)!}$$

$$\frac{\binom{n}{r}}{\binom{n}{r+1}} = \frac{\frac{n!}{(n-r)!r!}}{\frac{n!}{(r+1)!(n-r-1)!}}$$

$$\frac{\binom{n}{r}}{\binom{n}{r+1}} = \frac{\frac{n!}{(n-r-1)!(n-r)r!}}{\frac{n!}{r!(r+1)(n-r-1)!}}$$

$$\frac{\binom{n}{r}}{\binom{n}{r+1}} = \frac{\frac{n!}{(n-r-1)!(n-r)r!}}{\frac{n!}{r!(r+1)(n-r-1)!}} = \frac{r+1}{n-r}$$

Therefore, when $\frac{\binom{n}{r}}{\binom{n}{r+1}} < 1$,

$$\frac{r+1}{n-r} < 1$$

$$r+1 < n-r$$

$$r < \frac{n-1}{2}$$

When $\frac{\binom{n}{r}}{\binom{n}{r+1}} = 1$,

$$\frac{r+1}{n-r} = 1$$

$$r+1 = n-r$$

$$r = \frac{n-1}{2}$$

When $\frac{\binom{n}{r}}{\binom{n}{r+1}} > 1$,

$$\frac{r+1}{n-r} > 1$$

$$r+1 > n-r$$

$$r > \frac{n-1}{2}$$

According to definition of $n\mathrm{C}r$ in mathematics, n is a positive integer ($n \in \mathbb{Z}^+$), r is a non-negative integer ($r \in (\mathbb{Z}^+ \cup \{0\})$), and $n \geq r$ ([27], [28]).
$\therefore$ For $r = \frac{n-1}{2}$ to be satisfied, n must be an odd integer.

When n is an odd integer, the variation of the magnitude of $n\mathrm{C}r$ can be graphically depicted as below, based on the above calculations.

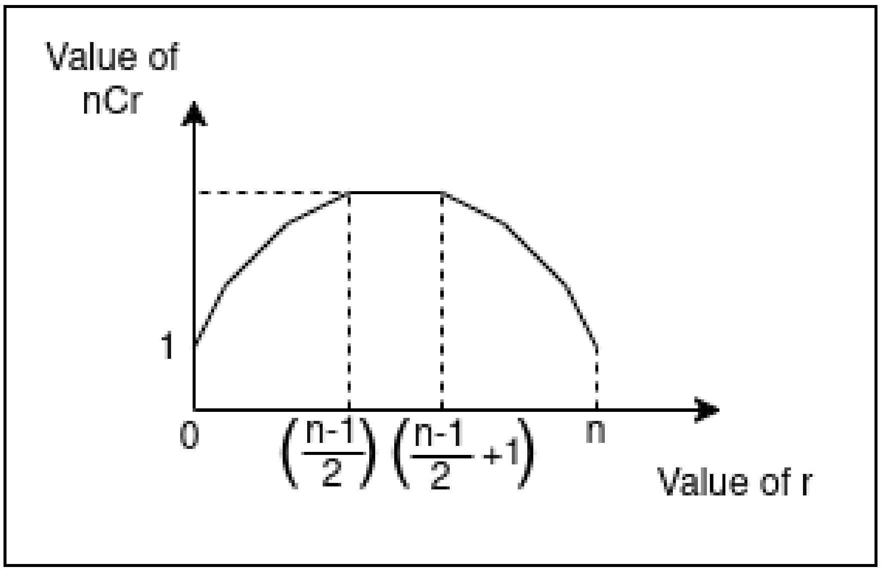


Figure 1: Variation of nCr for different r values, when n is an odd integer

By analyzing that graph, it can be proven that nCr is maximum when $r \approx \frac{n}{2}$. Because, $\frac{n}{2}$ is not an integer, when n is odd. ($\frac{n}{2} \approx \frac{n-1}{2}$ and $\frac{n}{2} \approx \left(\frac{n-1}{2}\right) + 1$)

When considering values from 0 to n for r, in nCr, the following symmetric pattern is observed.

$$nC0 = \binom{n}{0} = \frac{n!}{(n-0)!0!}$$
$$\binom{n}{n-0} = \frac{n!}{(n-0)!(n-(n-0))!} = \frac{n!}{(n-0)!0!}$$
$$\therefore \binom{n}{0} = \binom{n}{n-0}$$

$$nC1 = \binom{n}{1} = \frac{n!}{(n-1)!1!}$$
$$\binom{n}{n-1} = \frac{n!}{(n-1)!(n-(n-1))!} = \frac{n!}{(n-1)!1!}$$
$$\therefore \binom{n}{1} = \binom{n}{n-1}$$

$$nC2 = \binom{n}{2} = \frac{n!}{(n-2)!2!}$$

$$\binom{n}{n-2} = \frac{n!}{(n-2)!(n-(n-2))!} = \frac{n!}{(n-2)!2!}$$
$$\therefore \binom{n}{2} = \binom{n}{n-2}$$

It means that, $\binom{n}{r} = \binom{n}{n-r}$. But, in that pattern, when n is an even number, and $r = \frac{n}{2}$ is reached, following observation is noticed.

$$r = \frac{n}{2}$$
$$n - r = n - \frac{n}{2} = \frac{n}{2}$$
$$\therefore \binom{n}{\frac{n}{2}} = \binom{n}{n-\frac{n}{2}}$$

It means that, value of $\binom{n}{r}$ when $r = \frac{n}{2}$ is available only for $r = \frac{n}{2}$. Because, $r = n - r$.
Based on previous inequalities, following relationships are present for even n values.
When $\frac{\binom{n}{r}}{\binom{n}{r+1}} < 1$,

$$r < \frac{n-1}{2}$$

Since $\frac{n-1}{2} < \frac{n}{2}$,

$$r < \frac{n-1}{2} < \frac{n}{2}$$

When $\frac{\binom{n}{r}}{\binom{n}{r+1}} > 1$,

$$r > \frac{n-1}{2}$$

When n is an even integer, the variation of the magnitude of $n\mathrm{C}r$ can be graphically depicted as below, based on the above factors.

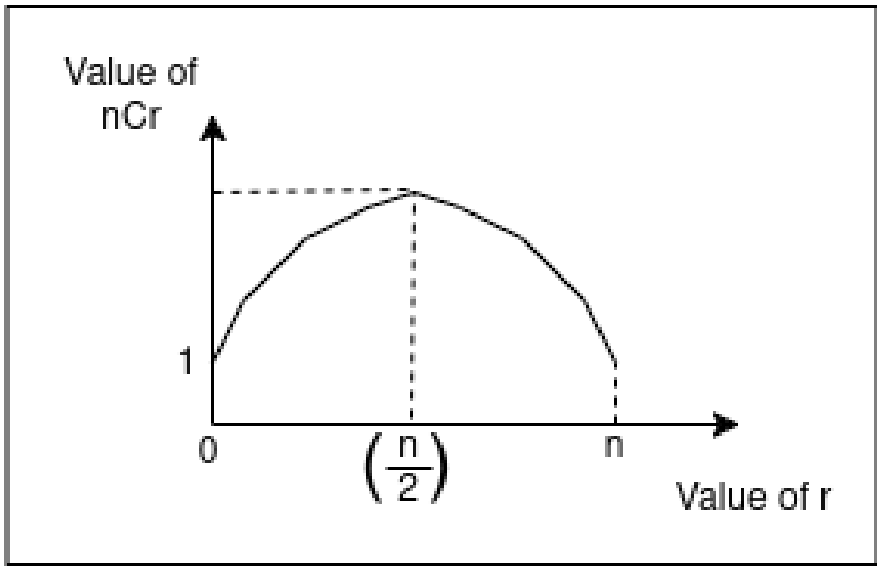


Figure 2: Variation of nCr for different r values, when n is an even integer

In ensemble, it is concluded that, $\binom{n}{r}$ is maximum when $r = \frac{n}{2}$ or $r \approx \frac{n}{2}$.

B Time complexity of $\binom{n}{\frac{n}{2}}$ for large n values

$$n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \quad \text{(By Stirling's approximation, [29])} \quad \Rightarrow \quad 1$$

$$\therefore \quad \left(\frac{n}{2}\right)! \approx \sqrt{\frac{2\pi n}{2}}\left(\frac{n}{2e}\right)^{n/2} \quad \Rightarrow \quad 2$$

$$\binom{n}{\frac{n}{2}} = \frac{n!}{(\frac{n}{2})!(n-\frac{n}{2})!} = \frac{n!}{(\frac{n}{2})!(\frac{n}{2})!} \text{ (By nCr combination)}$$

$$\frac{n!}{(\frac{n}{2})!(\frac{n}{2})!} \approx \frac{\sqrt{2\pi n}\left(\frac{n}{e}\right)^n}{\sqrt{\frac{2\pi n}{2}}\left(\frac{n}{2e}\right)^{\frac{n}{2}} \cdot \sqrt{\frac{2\pi n}{2}}\left(\frac{n}{2e}\right)^{\frac{n}{2}}} \quad \text{(By 1 and 2)}$$

$$\frac{n!}{(\frac{n}{2})!(\frac{n}{2})!} \approx \frac{\sqrt{2\pi n}\left(\frac{n}{e}\right)^n}{\pi n\left(\frac{n}{2e}\right)^n}$$

$$\frac{n!}{(\frac{n}{2})!(\frac{n}{2})!} \approx \sqrt{2\pi n} \cdot \frac{n^n}{e^n} \cdot \frac{2^n e^n}{\pi n n^n}$$

$$\therefore \quad \frac{n!}{(n/2)!(n/2)!} \approx \sqrt{\frac{2}{\pi n}} \cdot 2^n$$

$$\text{It means that, } \binom{n}{\frac{n}{2}} \approx \sqrt{\frac{2}{\pi n}} \cdot 2^n$$

$$\therefore O(\binom{n}{\frac{n}{2}}) \approx O(\frac{2^n}{\sqrt{n}})$$

C Time complexity of $\binom{n}{\frac{n-1}{2}}$ for large n values

In $\binom{n}{\frac{n-1}{2}}$, when $2m+1$ is substituted for n, and m is substituted for $\frac{n-1}{2}$,

$$\binom{n}{\frac{n-1}{2}} = \binom{2m+1}{m}$$

$$\binom{2m+1}{m} = \frac{(2m+1)!}{(m+1)!m!} \text{ (By nCr combination)} \Rightarrow \quad 1$$

$$\binom{2m+1}{m+1} = \frac{(2m+1)!}{(m+1)!m!} \text{ (By nCr combination)} \Rightarrow \quad 2$$

$$\frac{(2m+1)!}{(m+1)!m!} = \frac{(2m+1)! \times (2m+2)}{(m+1)!m! \times (2m+2)}$$

$$\frac{(2m+1)! \times (2m+2)}{(m+1)!m! \times (2m+2)} = \frac{(2m+2)!}{(m+1)!m! \times 2(m+1)}$$

$$\frac{(2m+2)!}{(m+1)!m! \times 2(m+1)} = \frac{(2m+2)!}{2 \times (m+1)!(m+1)!}$$

$$\frac{(2m+2)!}{2 \times (m+1)!(m+1)!} = \frac{1}{2} \times \frac{(2m+2)!}{(m+1)!(m+1)!}$$

$$\frac{1}{2} \times \frac{(2m+2)!}{(m+1)!(m+1)!} = \frac{1}{2} \times \binom{2m+2}{m+1} = \frac{1}{2} \times \binom{2(m+1)}{m+1} \Rightarrow \quad 3$$

By considering 2 and 3,

$$\binom{2m+1}{m+1} = \frac{1}{2} \times \binom{2(m+1)}{m+1}$$

$$\therefore O(\binom{2m+1}{m+1}) = O(\binom{2(m+1)}{m+1}) \Rightarrow \quad 4$$

By 1 and 2,

$$\binom{2m+1}{m+1} = \frac{(2m+1)!}{(m+1)!m!} = \binom{2m+1}{m}$$

$$\therefore O(\binom{2m+1}{m+1}) = O(\binom{2m+1}{m}) \Rightarrow \quad 5$$

Accordingly by 4 and 5,

$$O(\binom{2m+1}{m}) = O(\binom{2(m+1)}{m+1})$$

By above proof B,

$$O(\binom{2(m+1)}{m+1}) \approx O(\frac{2^{2(m+1)}}{\sqrt{2(m+1)}})$$

When 2(m+1) is considered as an integer n,

$$O(\frac{2^{2(m+1)}}{\sqrt{2(m+1)}}) \approx O(\frac{2^n}{\sqrt{n}})$$

$$\therefore O(\binom{n}{\frac{n-1}{2}}) \approx O(\frac{2^n}{\sqrt{n}})$$