Constraint-Driven Timetabling with Real-Time Updates: Design and Implementation

Appu H P
Dept. of Computer Science and Engg.
Malnad College of Engineering
Hassan, India
appuhp9844@gmail.com

Bhuvan M
Dept. of Computer Science and Engg.
Malnad College of Engineering
Hassan, India
bhuvanmuralidhara1@gmail.com

Bindu Prasad G S
Dept. of Computer Science and Engg.
Malnad College of Engineering
Hassan, India
binduprasad728@gmail.com

Aathish Shetty
Dept. of Computer Science and Engg.
Malnad College of Engineering
Hassan, India
aathishshetty65@gmail.com

B Uma
Dept. of Computer Science and Engg.
Malnad College of Engineering
Hassan, India
bu@mcehassan.ac.in

***Abstract*—The management of academic timetables presents a significant operational challenge for educational institutions worldwide. This paper presents an innovative scheduling system that transforms timetable generation from a time-consuming manual process to an automated, real-time solution. Our approach formulates the scheduling problem as a constraint satisfaction challenge, implementing Google's OR-Tools CP-SAT solver to generate conflict-free timetables that honor essential constraints such as faculty availability, classroom allocation, and curriculum requirements. The system's distinguishing feature is its integration of Firebase for continuous data synchronization, allowing immediate reflection of scheduling changes—including class cancellations, extra sessions, and teacher swapping—across all user platforms. Built on a modular C4 architecture, the system provides distinct interfaces for administrators, faculty, and students through Flutter-based applications that function consistently across device types. Empirical evaluation reveals dramatic efficiency improvements with schedule creation time substantially decreased compared to manual methods, while update propagation occurs almost instantaneously. By combining deterministic constraint resolution with cloud-based real-time capabilities, our implementation creates a responsive academic scheduling ecosystem that significantly enhances institutional operations while improving the experience for all educational stakeholders.**

***Index Terms*—Academic Timetabling, Constraint Programming, Real-Time Synchronization, Firebase, OR-Tools, Flutter, CP-SAT, Educational Technology.**

I. Introduction

Academic timetable creation represents one of the most challenging operational tasks in educational institutions. The inherent complexity arises from the need to coordinate multiple interdependent elements—courses, teachers, classrooms, and student sections—while adhering to various scheduling rules and preferences. Traditional timetabling approaches typically rely on manual processes that are not only labor-intensive but also prone to errors, conflicts, and inefficiencies.

A. The Scheduling Challenge

The timetabling problem is fundamentally a resource allocation challenge with multiple constraints. Educational institutions must schedule classes to optimize resource utilization while ensuring quality of education and stakeholder satisfaction. In practice, this means balancing competing objectives: minimizing idle time, maintaining pedagogical quality, and accommodating various preferences and restrictions.

The emergence of modern computational techniques has created new opportunities to address these challenges. In particular, constraint programming offers a powerful framework for expressing and solving complex scheduling problems, while cloud-based technologies enable dynamic updates and real-time communication that were previously impossible in traditional systems.

B. Computational Complexity

The academic timetabling problem belongs to the class of NP-hard problems, characterized by exponentially growing solution spaces as the problem size increases. For example, scheduling even a modest department with 10 teachers, 5 sections, 6 time slots per day, and 6 days per week creates a search space with potential combinations that exceed 10^{30} possibilities. This computational complexity makes exhaustive search methods impractical and necessitates specialized constraint satisfaction techniques.

C. Toward Dynamic Scheduling

Beyond the initial generation of feasible schedules, educational institutions require systems that can adapt to daily operational changes. Teacher absences, class swaps, rescheduling requests, and the need for extra sessions demand a timetabling system that can dynamically adjust while maintaining the integrity of the overall schedule. This paper presents a solution that addresses both the initial timetable generation challenge

and the ongoing need for real-time adaptability in day-to-day academic operations.

II. Problem Statement

Timetable management in educational institutions is complex and involves balancing faculty availability and curriculum constraints. Manual and static scheduling methods often cause conflicts, inefficiencies, and lack real-time adaptability. Existing systems fail to handle dynamic changes like absences or class swaps promptly. Therefore, an automated, real-time, conflict-free timetable generation system is essential to enhance flexibility, accuracy, and operational efficiency.

A. Key Challenges

The academic scheduling domain presents several interconnected challenges that require systematic resolution:

- **Faculty Scheduling Conflicts**: Ensuring instructors are not simultaneously assigned to multiple sections, which requires careful tracking of all teaching assignments across the timetable.
- **Infrastructure Constraints**: Optimizing the allocation of limited physical resources, particularly when classroom and laboratory availability creates bottlenecks in the scheduling process.
- **Curriculum Adherence**: Guaranteeing that each course receives its designated number of sessions per week according to program requirements and accreditation standards.
- **Specialized Session Requirements**: Accommodating practical laboratory work that requires uninterrupted blocks of consecutive periods while ensuring appropriate facility availability.
- **Schedule Efficiency**: Minimizing fragmentation in daily schedules by reducing isolated free periods that create inefficient use of time for both students and faculty.
- **Responsive Adaptation**: Developing mechanisms that allow for seamless handling of unexpected changes including faculty absences, class swaps, and impromptu session additions without compromising schedule integrity.

These multifaceted challenges necessitate a comprehensive solution that integrates sophisticated constraint programming for initial timetable creation with agile update capabilities for ongoing schedule management throughout the academic period.

III. Related Work

Academic timetabling has received significant attention within the operations research and artificial intelligence communities. Various approaches have been proposed to address its complexity.

El-Sakka [1] investigated the university course timetable problem at the Community College of the University of Sharjah, formulating it as a multidimensional constraint satisfaction problem. The proposed CLP/OPL model demonstrated efficiency in optimizing resource allocation while satisfying scheduling constraints, though it lacked mechanisms for real-time adjustments.

Ďuriš [2] approached the timetable problem through a tree-based search algorithm within the Constraint Satisfac- tion Problem (CSP) framework. His work included constraint validation tools that improved the theoretical understanding of timetable scheduling, but did not address implementation concerns for real-world dynamic scheduling environments.

Khodadadi [3] introduced the Dynamic Arithmetic Optimization Algorithm (DAOA), which demonstrated improved exploration and exploitation capabilities in optimization problems. While not directly applied to timetabling, this approach offers insights into handling complex dynamic behaviors relevant to academic scheduling.

Coviello [4] focused on train timetabling using the ATMO tool, which optimizes multiple conflicting objectives such as travel time and timetable stability. The application of Multi-Objective Ant Colony Optimization (MOACO) with Mixed Integer Linear Programming (MILP) to generate Pareto-optimal solutions provides valuable methodological insights, though in a different domain.

More recent work by Kumar et al. [5] explored machine learning approaches for automated timetable generation, while Maneesha et al. [6] investigated genetic algorithms for this purpose. Ambhore et al. [7] developed an automatic timetable generator focusing on user interface aspects, and Kavade et al. [8] integrated AI techniques in their smart timetable system.

While these approaches have advanced the field, most existing solutions focus primarily on the initial generation of timetables rather than providing mechanisms for real-time updates and synchronization. Our work extends the state of the art by integrating constraint-driven timetable generation with a real-time update framework, addressing both the initial scheduling problem and the operational challenges of dynamic schedule management.

IV. System Architecture

Our system employs a modular architecture based on the C4 model, which provides a hierarchical view of the soft- ware architecture from different levels of abstraction. This approach allows for clear separation of concerns, facilitates maintenance, and supports scalability.

A. C4 Model Overview

At the system context level, we identify three primary user types—administrators, teachers, and students—each with distinct interactions with the timetable system. Administrators provide input parameters, generate timetables, and manage changes. Teachers view personal schedules and can request modifications, while students access class schedules and receive notifications about changes.

Figure 1 illustrates the container-level architecture, which defines the high-level technological components of the system.

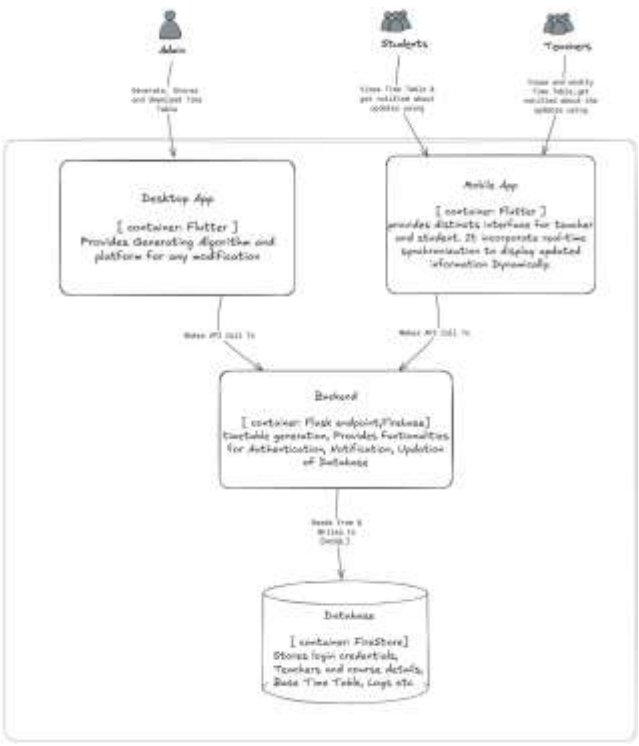

Fig. 1: Container-level architecture depicting the core system components and their interactions.

B. Core Components

The system comprises four main containers:

- **Admin Desktop Application**: Developed using Flutter, this container provides administrators with interfaces for managing academic parameters, triggering timetable generation, and handling schedule modifications.
- **Mobile Application**: Also built with Flutter, this container serves both teachers and students, offering personalized schedule views and real-time notifications.
- **Backend Engine**: Implemented using Python with OR-Tools integration, this container processes constraint-based timetable generation, validates changes, and manages data consistency.
- **Database**: Utilizing Firebase Firestore, this NoSQL database container stores all system data and enables real-time synchronization across devices.

C. Component-Level Design

At the component level, specific elements within each container handle specialized functions. The Admin Desktop Application includes modules for teacher management, course configuration, and timetable generation. The Mobile Application contains separate interfaces for teachers and students, along with notification components. The Backend Engine incorporates the CP-SAT solver, data validation services, and synchronization handlers.

D. Interaction Flow

The system follows a real-time interaction pattern where:

1) Administrators input parameters and trigger timetable generation through the Desktop Application.
2) The Backend Engine processes these inputs using the CP-SAT solver to generate a conflict-free timetable.
3) Generated timetables and subsequent changes are stored in the Firebase Firestore database.
4) Real-time database listeners in the Mobile Application immediately reflect changes to affected users.
5) Push notifications alert users to relevant schedule modifications.

This architecture ensures that all stakeholders have access to the most current schedule information at all times, while maintaining the integrity of the scheduling constraints.

V. Constraint Formulation

The timetable generation logic is formulated as a Constraint Satisfaction Problem (CSP) and solved using the CP-SAT solver from Google OR-Tools. The problem is defined using the following mathematical model:

A. Variables

- $S = \{s_1, \ldots, s_n\}$: Set of academic sections (e.g., A, B)
- $D = \{d_1, \ldots, d_6\}$: Days (Monday–Saturday)
- $T = \{t_1, \ldots, t_k\}$: Time slots (e.g., 8:30–9:30)
- $C = \{c_1, \ldots, c_m\}$: Courses, $L \subset C$: Lab courses
- $\mathcal{T}$: Set of teachers

B. Decision Variables

Teacher Assignment Variable:

$$X_{s,d,t} \in \mathbb{Z}_{\geq 0} \tag{1}$$

Represents the teacher index for section s, day d, time slot t. A value of 0 indicates no assignment.

Theory Course Assignment Variable:

$$T_{s,c,d,t} \in \{0,1\} \tag{2}$$

Binary indicator that equals 1 if theory course c is scheduled for section s on day d during time slot t.

Lab Course Assignment Variable:

$$L_{s,c,d,t_1,t_2} \in \{0,1\} \tag{3}$$

Binary indicator that equals 1 if lab course c is scheduled for section s on day d during consecutive time slots t_1 and t_2.

C. Domain

$$X_{s,d,t} \in \{0,1,\ldots,|\mathcal{T}_s|\} \text{ where } \mathcal{T}_s \subseteq \mathcal{T} \cup \{\text{None}\} \tag{4}$$

D. Constraints

1) **Theory Session Requirement**
Ensures that each theory course c meets its required number of weekly sessions $r_{s,c}$.

$$\sum_{d \in D} \sum_{t \in T} T_{s,c,d,t} = r_{s,c},$$

$$T_{s,c,d,t} = \delta(X_{s,d,t} = \text{teacher}(c))$$

2) **Lab Block Constraint**
Ensures that each lab course c is scheduled in valid contiguous slot pairs and assigned the required number of weekly sessions.

$$L_{s,c,d,t_1,t_2} = 1 \Rightarrow X_{s,d,t_1} = X_{s,d,t_2} = \text{teacher}(c),$$

$$\sum_{d \in D} \sum_{(t_1,t_2) \in \mathcal{P}} L_{s,c,d,t_1,t_2} = r_{s,c}$$

3) **Lab Exclusivity**
Ensures that a lab teacher assigned to a lab block does not take any other slots for that section on the same day.

$$L_{s,c,d,t_1,t_2} = 1 \Rightarrow X_{s,d,t'} \neq \text{teacher}(c) \quad \forall t' \in T \backslash \{t_1, t_2\}$$

4) **Lab Room Capacity**
Ensures that the number of concurrent labs does not exceed the number of available lab rooms R_{lab}.

$$\sum_{s \in S} \sum_{c \in L} L_{s,c,d,t_1,t_2} \leq R_{\text{lab}} \quad \forall d \in D, (t_1, t_2) \in \mathcal{P}$$

5) **Teacher Conflict**
Ensures that no teacher is assigned to multiple sections at the same time.

$$\sum_{s \in S} \delta(X_{s,d,t} = \tau) \leq 1 \quad \forall \tau \in \mathcal{T}, d \in D, t \in T$$

6) **Classroom Capacity**
Ensures that the number of theory classes does not exceed the number of available general classrooms R_{class}.

$$\sum_{s \in S} \delta(\text{theory}(X_{s,d,t})) \leq R_{\text{class}} \quad \forall d \in D, t \in T$$

7) **Saturday Cutoff**
Ensures that no classes are scheduled after 1:00 PM on Saturdays.

$$X_{s,\text{Sat},t} = 0 \quad \forall s \in S, t \in \{2{:}00\text{–}3{:}00, 3{:}00\text{–}4{:}00\}$$

8) **Slot Utilization**
Ensures that each section utilizes exactly the number of slots required by its assigned theory and lab courses.

$$\sum_{d \in D} \sum_{t \in T} \delta(X_{s,d,t} \neq 0) = \sum_{c \in C \backslash L} r_{s,c} + 2 \sum_{c \in L} r_{s,c}$$

9) **Compactness Objective**
Minimizes the number of idle slots that are surrounded by active class slots to improve schedule compactness.

$$\min \sum_{s \in S} \sum_{d \in D} \left(\sum_{t \in T} I_{s,d,t} \right)^2,$$

$$I_{s,d,t} = \delta(\text{idle}(t) \wedge \exists \text{ classes before/after } t)$$

A. Constraint Processing with OR-Tools

The implementation of these constraints in OR-Tools involves translating the mathematical formulations into programmatic constraints using the CP-SAT solver. The solver employs several advanced techniques:

- **Boolean Variable Creation**: For each possible assignment of teachers to slots and sections.
- **Constraint Translation**: Mathematical constraints are expressed using the solver's API functions such as AddBoolOr, AddBoolAnd, and AddLinearConstraint.
- **Conflict-Driven Clause Learning**: The solver identifies infeasible combinations of assignments and uses these insights to guide the search process efficiently.
- **Activity-Based Search**: The solver prioritizes variables that are involved in many constraints, accelerating the discovery of conflicts and valid solutions.
- **Objective Optimization**: After finding a feasible solution, the solver iteratively improves it to minimize the defined objective function.

This constraint programming approach ensures that generated timetables satisfy all specified requirements while optimizing for practical usability, specifically the compactness of the schedule to minimize idle time for both students and faculty.

VI. Implementation Details

A. Backend Implementation

The core of our system is implemented using Google's OR-Tools CP-SAT solver in Python. The backend engine processes input data about courses, teachers, sections, time slots, and constraints to generate feasible timetables.

The constraint implementation follows this process:

1) **Initialization**: Create a CP-SAT model and define integer variables for each section-day-timeslot combination.
2) **Constraint Addition**: Systematically add all hard constraints to the model.
3) **Objective Definition**: Set the objective function to minimize idle periods.
4) **Solve**: Invoke the solver with appropriate parameters.
5) **Solution Retrieval**: Extract and structure the solution for storage in Firebase.

B. Firebase Integration

Firebase Firestore serves as our real-time database, offering several advantages:

- **NoSQL Structure**: Flexible document-oriented storage that accommodates the complex relationships in academic timetables.
- **Real-time Synchronization**: Built-in capabilities for instant data propagation to all connected clients.
- **Authentication**: Secure access control for different user types.

The database schema organizes data into collections for users and timetables, with careful attention to optimizing read patterns for common queries.

C. Cross-Platform Applications

Both the admin desktop application and the mobile applications for teachers and students are built using Flutter, which offers:

- **Cross-Platform Compatibility**: Single codebase for deployment across Android, iOS, web, and desktop.
- **Reactive UI**: Widget-based architecture that efficiently reflects data changes.
- **Firebase SDK Integration**: Native support for Firestore listeners and Cloud Messaging.
- **Material Design**: Consistent and intuitive user interfaces.

D. Real-Time Update Mechanism

The real-time update mechanism follows this workflow:

1) A teacher initiates an action (e.g., class cancellation) through the mobile app.
2) The request is validated and processed by the backend.
3) The change is committed to Firestore with appropriate timestamps.
4) UI components in affected applications update automatically.
5) Firebase Cloud Messaging sends push notifications to relevant users.

This architecture ensures that changes propagate instantly across the system, maintaining data consistency and keeping all stakeholders informed.

VII. User Interface Design

The user interface design focuses on providing intuitive, role-specific experiences for administrators, teachers, and students.

A. Admin Desktop Interface

The administrator interface includes:

- **Dashboard**: Overview of system status and recent activities.
- **Configuration Panels**: Interfaces for managing teachers, courses, sections.
- **Timetable Viewer**: grid-based visualization of generated schedules.

Fig. 2: Admin Desktop Interface showing the timetable management view with grid-based visualization and control options.

B. Teacher Mobile Interface

The teacher interface provides:

- **Personal Schedule**: Day and week views of assigned classes.
- **Section View**: Timetables for taught sections.
- **Class Management**: Controls for cancellations, reschedules, and extra classes.
- **Notifications**: Alerts about schedule changes and approvals.
- **Profile**: Personal information and preferences.

C. Student Mobile Interface

The student interface features:

- **Section Timetable**: Complete schedule for the student's section.
- **Daily View**: Today's classes with timing details.
- **Notifications**: Alerts about cancellations, reschedules, and extra classes.
- **Profile**: Academic information and settings.

All interfaces implement real-time updates, automatically refreshing when the underlying data changes without requiring manual intervention from users.

(a) Student view with section timetable

(b) Teacher view with personalized timetable

Fig. 3: Mobile Application Interface implementations for different user roles.

(a) Class cancellation interface

(b) Extra class scheduling interface

Fig. 4: Teacher class management options for dynamic schedule modifications.

VIII. System Evaluation

We evaluated the system along multiple dimensions, including performance, usability, and operational impact.

A. Performance Metrics

- **Timetable Generation Time**: For the test case of three sections and two classrooms, the system generates a complete timetable in under 0.35 seconds, compared to several hours or days with manual methods.
- **Update Propagation Latency**: Changes to the timetable propagate to all connected devices in an average of 1.2 seconds, ensuring near-instantaneous communication of schedule modifications.
- **Constraint Satisfaction**: All generated timetables successfully satisfy 100% of the hard constraints, eliminating schedule conflicts entirely.

B. Usability Assessment

Feedback from stakeholders highlighted several qualitative improvements:

- Administrators reported significant reduction in time spent on timetable management and scheduling tasks.
- Teachers expressed appreciation for the immediate notifications about schedule changes and improved commu-
- Students noted higher satisfaction with academic organization and schedule transparency compared to the previous system.

C. Comparison with Traditional Methods

The system shows significant advantages over both manual methods and static software solutions:

- **Time Efficiency**: Reduces timetable creation from days to minutes.
- **Error Elimination**: Removes human errors through constraint validation.
- **Adaptability**: Enables real-time changes that would be impractical with static systems.
- **Communication**: Automates notification of changes to all stakeholders.
- **Resource Optimization**: Improves utilization of classrooms and faculty time.

These results demonstrate the effectiveness of combining constraint programming with real-time technologies for academic scheduling.

IX. Challenges and Limitations

Despite the system's successes, several challenges and limitations were encountered:

A. Technical Challenges

- **Connectivity Dependence**: The real-time update mechanism requires stable internet connectivity for optimal performance, though basic functionality is maintained during offline periods.

- **Computational Complexity**: The NP-hard nature of the timetabling problem means that as the problem size increases (more sections, teachers, or courses), generation time grows exponentially, potentially requiring optimization techniques for larger institutions.
- **Cross-Platform Consistency**: Maintaining UI consistency across desktop, and mobile platforms required careful design considerations and extensive testing.

B. Current Limitations

- **Constraint Flexibility**: The current implementation prioritizes hard constraints, with limited support for soft preferences like ”teacher prefers morning classes.”
- **Scaling Scope**: The system has been tested with three sections and two classrooms; scaling to larger departments will require architectural adjustments.
- **Conflict Resolution**: While the system prevents scheduling conflicts, it does not yet provide automated suggestions for resolving infeasible constraints.

These challenges inform our future development roadmap.

X. Conclusion and Future Work

This paper presented a constraint-driven timetabling system with real-time update capabilities, demonstrating how modern computational techniques and cloud technologies can transform academic scheduling from a static, error-prone process to a dynamic, responsive system. The integration of Google’s OR-Tools CP-SAT solver with Firebase’s real-time synchronization capabilities provides a robust solution to both the initial timetable generation challenge and the ongoing need for schedule adaptability.

The key contributions of this work include:

- A comprehensive constraint formulation that captures the essential requirements of academic scheduling.
- A modular system architecture that separates concerns and facilitates maintenance and scalability.
- An efficient real-time update mechanism that ensures all stakeholders have access to current schedule information.
- Cross-platform applications that provide intuitive interfaces for administrators, teachers, and students.

Future work will focus on several enhancements:

- **Scaling to Larger Institutions**: Extending the system to handle multiple departments and programs simultaneously.
- **Enhanced Academic Features**: Integrating course material management, attendance tracking, and performance analytics.
- **Advanced Optimization**: Implementing soft constraint preferences and multi-objective optimization to balance competing priorities.
- **Machine Learning Integration**: Incorporating predictive analytics for intelligent suggestions and automatic adjustment based on historical patterns.

These future directions will transform the system from a scheduling tool into a comprehensive academic management ecosystem, further enhancing the efficiency and effectiveness of educational institutions.

References

[1] T. El-Sakka, “University course timetable using constraint satisfaction and optimization,” *International Journal of Computing*, vol. 4, no. 3, 2015.
[2] V. Dˇ urisˇ, “Algorithmic verification of constraint satisfaction method on timetable problem,” *Mathematics and Statistics*, vol. 8, no. 6, pp. 728–739, 2020.
[3] N. Khodadadi, V. Snasel, and S. Mirjalili, “Dynamic arithmetic optimization algorithm for truss optimization under natural frequency constraints,” *IEEE Access*, vol. 10, pp. 16188–16208, 2022.
[4] N. Coviello, G. Medeossi, A. Nash, T. Nygreen, P. Pellegrini, and J. Rodriguez, “Automatic generation of timetable with the ATMO tool,” Technical report, 2021.
[5] P. Kumar, S. Sanakar, P. Kumar, S. M. Usman, and Vani, “Automated Timetable Generator Using Machine Learning,” *International Research Journal of Modernization in Engineering Technology and Science*, vol. 2, no. 8, Aug. 2020.
[6] G. Maneesha, T. Deepika, S. BhanuSri, N. R. Kumar, and P. S. Nagamani, “Automatic Time Table Generation Using Genetic Algorithm,” *Journal of Emerging Technologies and Innovative Research*, Jul. 2021.
[7] S. Ambhore, P. Walke, R. Ghundgrudkar, A. Alone, and A. Khedkar, “Automatic Timetable Generator,” *International Journal of Research in Engineering, Science and Management*, vol. 3, no. 3, Mar. 2020.
[8] R. Kavade, S. Qureshi, N. Veer, V. Ugale, and P. Agrawal, “Smart Time Table System Using AI and ML,” *International Journal of Creative Research Thoughts (IJCRT)*, vol. 11, no. 5, May 2023.