United International Journal for Research & Technology

Volume 07, Issue 07, 2026 | Open Access | ISSN: 2582-6832

Automated Class Scheduling System (ACSS) for ISUFST

Bryan C. Paderes[1] and Renante A. Diamante[2]

[1]Passi City College, Philippines

[2]Iloilo State University of Fisheries Science and Technology

Email: *paderesbryan08@gmail.com and rdiamantetip@gmail.com*

***Abstract*—** Manual class scheduling in higher education institutions is a complex and time-consuming administrative task that often results in conflicts, overlapping instructor assignments, room double-bookings, and inefficient allocation of classroom resources. Such inefficiencies can lead to operational delays, reduced teaching quality, and increased administrative workload. This study presents the development and implementation of an Automated Class Scheduling System (ACSS) specifically designed for ISUFST to address these challenges. The ACSS leverages a constraint-based scheduling algorithm to automatically assign instructors, classrooms, and time slots while ensuring adherence to institutional rules and minimizing conflicts. The system considers various constraints, including instructor availability, room capacity, appointment types (lectures and laboratories), course-year-section consistency, and batch allocations, to generate optimized timetables.

The system is implemented as a desktop application using Windows Presentation Foundation (WPF) and follows the Model–View–ViewModel (MVVM) architecture, with Microsoft SQL Server as the backend database for data management and persistence. Administrative functionalities include managing instructors and subjects, defining room capacities and configurations, setting availability schedules, and exporting generated schedules to image or PDF formats. Evaluation of the ACSS demonstrated its effectiveness in reducing manual scheduling time, preventing conflicts, and improving overall administrative efficiency. The automated generation of color-coded timetables by instructor, room, and section facilitates clear visualization and flexibility in scheduling adjustments.

By integrating intelligent constraint handling and user-configurable parameters, the ACSS provides a practical, reliable, and scalable solution suitable for the academic environment of ISUFST. Future enhancements may include the incorporation of student enrollment data and the adoption of advanced optimization techniques, such as genetic algorithms, to support larger institutions and further improve schedule efficiency.

***Keywords*—** automated class scheduling system, academic scheduling, timetabling optimization, constraint-based scheduling, resource allocation, scheduling algorithm.

I. INTRODUCTION

Class scheduling is a critical administrative task in higher education institutions, as it involves assigning instructors, classrooms, and time slots while avoiding conflicts. At ISUFST, scheduling is traditionally performed manually, a process that is time-consuming and prone to errors such as overlapping instructor assignments, room double-bookings, and inefficient utilization of classroom resources.

To address these challenges, this study proposes the design and implementation of an Automated Class Scheduling System (ACSS), which generates optimized schedules based on predefined constraints, including instructor availability, room capacity, and subject requirements. By automating the scheduling process, the system minimizes human error, ensures conflict-free timetables, and improves overall administrative efficiency at ISUFST.

II. RELATED WORK

Several universities have adopted automated scheduling systems to improve efficiency and reduce conflicts in academic timetabling. Common

approaches include genetic algorithms [1], which generate optimized schedules using evolutionary strategies; constraint satisfaction problems (CSPs) [2], which model scheduling as a set of constraints that must be satisfied; and heuristic-based methods [3], which apply rules to iteratively reduce conflicts and improve timetable quality. While these methods have proven effective, they often involve complex configuration or are primarily designed for large-scale institutions.

In contrast, the Automated Class Scheduling System (ACSS) presented in this study employs a constraint-based incremental allocation algorithm with batch-aware processing and multi-phase execution. This approach provides a practical and efficient solution tailored to the scale and scheduling requirements of ISUFST.

III. METHODOLOGY

The Automated Class Scheduling System (ACSS) is implemented as a Windows desktop application using Windows Presentation Foundation (WPF) and follows the Model–View–ViewModel (MVVM) architectural pattern. The system utilizes Microsoft SQL Server as its backend database.

The architecture is modular and consists of three primary components:

- **Presentation Layer (WPF with MVVM) –** Responsible for user interaction, data binding, and command handling.
- **Scheduling Engine Module –** Implemented as a separate project containing the core scheduling algorithm and constraint-processing logic.
- **Data Layer –** Handles database communication, data persistence, and relational integrity using SQL Server.

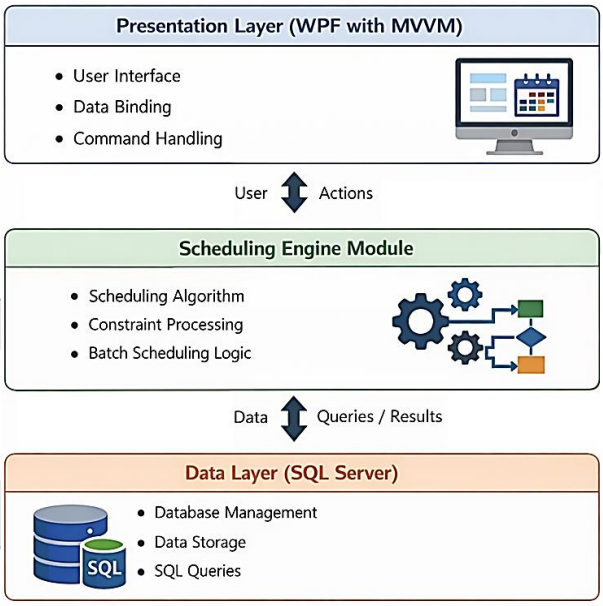

***Figure 1** shows the high-level system architecture.*

A. Scheduling Constraints

The system enforces the following constraints:

Constraint	Description
Instructor availability	An instructor cannot be scheduled in two classes at the same time. Availability is refined by existing appointments and lunch breaks.
Room assignment	A room can accommodate only one class per time slot. Room type (lecture/lab) must match subject type.

Room capacity	The classroom must have enough capacity for enrolled students. Students are divided into batches if capacity is exceeded.
Preferred time slots	Instructor preferred time slots are prioritized when available.
Section conflicts	Sections cannot be scheduled to overlap within the same semester, year level, and course.
Course-Year-Level-Section consistency	No two subjects with the same course, year level, and section can occupy overlapping time slots.
Semester and school year consistency	Schedules must match the selected semester and school year; cross-semester conflicts are prevented.
Batch allocation constraints	When student batches are used, each batch is scheduled in separate time slots without conflicts.
Appointment type constraint	Lecture and laboratory sessions must be assigned to appropriate room types and durations.

B. Scheduling Algorithm

The algorithm operates in the following steps:

- **Initialization:** All instructors, rooms, and subject loads are loaded, and availability maps are created for each day, incorporating working hours and lunch breaks.
- **Preprocessing:** Subject loads are split into smaller sessions based on appointment type (lecture or laboratory) and divided into batches if the number of students exceeds room capacity.
- **Incremental Allocation:** The algorithm iteratively assigns time slots and rooms for each batch of each subject. Before committing a schedule, it checks for conflicts including instructor availability, room assignment, section conflicts, course-year-section consistency, and semester/school-year consistency.
- **Finalization:** Successfully assigned appointments are stored in the database. Unsuccessful assignments are preserved for reassignment or future scheduling.
- **Reassignment:** Any previously unsuccessful schedules are retried in a separate pass, ensuring maximal schedule completion.

Pseudocode:

Input: List of Instructors, Rooms, Subject Loads, Semester ID, School Year ID
Output: Scheduled Appointments (Successful and Unsuccessful)

1. Initialize instructors, rooms, and subject loads.
2. Apply default working hours and lunch breaks.
3. Build an instructor availability map for each day.
4. Preprocess subject loads:
    - Split subjects based on appointment type (Lecture/Lab).
    - Divide students into batches if section size exceeds room capacity.
5. For each instructor:
   5.1 For each subject load (including split batches):
    - Identify suitable rooms based on appointment type.
6. For each room:
   6.1 For each available time slot within instructor and room availability:
7. Check for conflicts:
    - Instructor overlap
    - Room double-booking
    - Section conflicts (Course-Year-Level-Section)
    - Semester and school year consistency
8. If no conflicts:
    - Assign appointment in memory.
    - Update instructor and room availability.
    - Move to the next subject.
9. Otherwise:

- Mark appointment as unsuccessful.

10. Finalize scheduled appointments:
    - Save successful appointments to the database.
    - Remove subject loads if fully scheduled.
    - Retry unsuccessful appointments if applicable.
11. Return the list of successful and unsuccessful appointments.

IV. SYSTEM IMPLEMENTATION

The Automated Class Scheduling System (ACSS) was developed as a desktop application using Windows Presentation Foundation (WPF) and follows the Model–View–ViewModel (MVVM) architecture. The scheduling engine is implemented in C# as a modular component, while data persistence is handled using Microsoft SQL Server.

The system provides administrators with functionalities such as managing instructors and subject loads, defining room capacities and configurations, setting instructor availability schedules, automatically generating class timetables, and exporting schedules to image or PDF formats. An example of the system's schedule output is shown in Fig. 2.

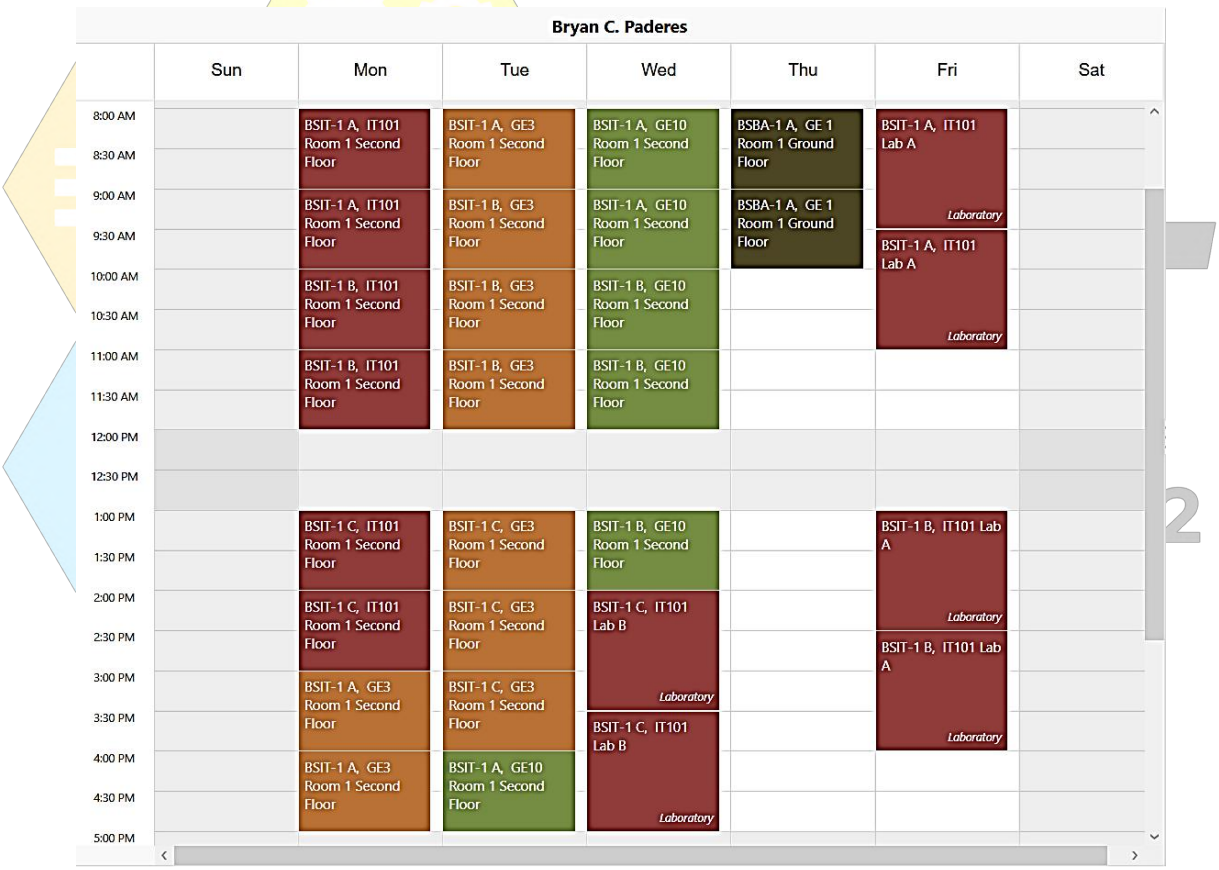

Figure 2. *Weekly Schedule of Instructor Bryan C. Paderes Showing Assigned Subjects, Time Slots, and Room Allocations*

The ACSS was evaluated based on three criteria: scheduling efficiency, conflict prevention, and administrative usability.

- **Time Efficiency:** The system significantly reduced schedule preparation time compared to manual scheduling. Automated generation eliminated repetitive conflict-checking and room allocation tasks.

- **Conflict Prevention:** The generated schedules satisfied all defined hard constraints, including instructor availability, room double-booking prevention, section overlap avoidance, and course–year–section consistency. No temporal conflicts were detected during system validation.
- **Administrative Usability:** The system streamlined instructor and room management processes, reducing manual workload during enrollment periods and improving overall scheduling reliability.

V. RESULTS AND DISCUSSION

The Automated Class Scheduling System (ACSS) successfully automated the generation of class schedules, detecting instructor and room conflicts and respecting instructor availability constraints.

The system produced color-coded timetables by instructor, room, and section, and allowed administrators to customize schedules using parameters such as room capacity, batch allocation, and lunch breaks.

Evaluation in a real academic setting demonstrated that the system generated conflict-free schedules efficiently and provided a user-friendly interface for administrative adjustments.

VI. CONCLUSION

The Automated Class Scheduling System (ACSS) provides an efficient solution to the scheduling challenges at ISUFST. By automating the scheduling process using a constraint-based algorithm, the system reduces scheduling conflicts, saves administrative time, and improves the allocation of classroom and instructor resources.

Future improvements may include integrating student enrollment data and adopting advanced optimization algorithms, such as genetic algorithms, to further enhance scheduling efficiency for larger institutions.

ACKNOWLEDGMENT

The author would like to thank colleagues at Passi City College and ISUFST for their guidance and feedback during the development of this system.

REFERENCES

[1] K. R. Apt, "The essence of constraint propagation," Theoretical Computer Science, vol. 221, no. 1–2, pp. 179–210, 1999. https://doi.org/10.1016/S0304-3975(99)00032-8

[2] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and J. R. Woodward, "A graph-based hyper-heuristic for educational timetabling problems," European Journal of Operational Research, vol. 176, no. 1, pp. 177–192, 2007. https://doi.org/10.1016/j.ejor.2005.08.012

[3] F. P. Diallo and C. Tudose, "Optimizing the scheduling of teaching activities in a faculty," Applied Sciences, vol. 14, no. 20, 9554, 2024. https://doi.org/10.3390/app14209554

[4] O. Alonge, W. Sakpere, and E. Adediran, "An automated timetable scheduler using NSGA-II for optimized scheduling in educational institutions," SSRN, 2024. https://doi.org/10.2139/ssrn.5015783

[5] A. N. Querubin and G. L. Contillo, "Optimizing class scheduling for the University of Northern Philippines using greedy algorithm," in Proc. 2024 9th Int. Conf. on Information Technology and Digital Applications (ICITDA), 2024, pp. 1–6. https://doi.org/10.1109/ICITDA64560.2024.10809509

[6] E. Rappos, E. Thiémard, S. Robert, and J. F. Hêche, "A mixed-integer programming approach for solving university course timetabling problems," Journal of Scheduling, vol. 25, no. 1, pp. 85–100, 2022. https://doi.org/10.1007/s10951-021-00715-5

[7] L. Paquete and T. Stützle, "Empirical analysis of tabu search for the lexicographic optimization of the examination timetabling problem," in Proc. 5th Int. Conf. on the Practice and Theory of Automated Timetabling, 2003, pp. 413–42