International Journal of Interpreting Enigma Engineers (IJIEE)
https://ejournal.svgacademy.org/index.php/ijiee/
https://doi.org/10.62674/ijiee.2026.v3i03.004
eISSN: 3048-6300
OPEN ACCESS

INTELLIGENT TIMETABLE SCHEDULING FOR ACADEMIC INSTITUTIONS: A NOVEL INTEGRATION OF CONSTRAINT-BASED OPTIMIZATION AND LARGE LANGUAGE MODELS

Dr.Vadetay Saraswathi Bai[1], Haranadha Reddy Busireddy Seshakagari[2], Dr.K.Reddy Madhavi[3]

Original Article

[1]Professor, Dept.of CSE(ALLIED), Vemu Institute of Technology, Chittoor.

[2]Manager - Architecture Valuemomentum, Erie, PA-16506, USA

[3]Professor & Head, Data Science, Mohan Babu University, Tirupati.

*Corresponding Author's Email: [1]saihumika9@gmail.com, [2]haranadhareddyven@gmail.com, [3]kreddymadhavi@gmail.com

Abstract

The educational institutions have been grappling with trying to come up with effective schedules of classes to satisfy different competing demands and resource maximization. The traditional manual processes are resource consuming and have the tendency to be subject to errors that in most situations cause scheduling inconveniences and inefficient resource distribution. The research presents a novel model that combines the notions of constraint satisfaction in addition to the contemporary huge language models to automate the production of academic schedules. We also apply multistage validation, conflict resolution, as well as intelligent workload distribution algorithms. Experimental analysis found on varying institutional dimensions depicts immense improvements in terms of efficiency in scheduling, conflicts diminution and improved computational performances in comparison with the traditional methods. The proposed system has the ability to produce conflict free schedules having a high index of 98.7 percent in comparison to the 76 percent decrease in processing time in comparison to manual processes. The results of the empirical tests indicate that the application will be applicable in the real world in most educational institutions with established new trends of automated scheduling systems.

Index Terms – scheduling of timetables, component contentment, large scale language models, educational technology, resource distribution, algorithms, optimization.

Introduction

The educational administration involves a number of intricate operating issues, and the academic schedule is a relatively complicated issue. Institutions of tertiary education must schedule dozens of lessons each week and, besides, take into account the capacity of the faculty, whether it is possible to utilize [1] rooms, the demands of the program, not to mention pedagogical requirements. Research has revealed that the process of creating semester schedules manually consumes approximately 40- 60 man hours by administrative staff and in many cases requires multiple steps of iterative process to eliminate conflicts. Timetabling problem belongs to the class of NP-hard, and it signifies that the computational issue increases exponentially with the problem size. A typical semester in a university may involve scheduling of up to 500 and more classes in 50 or more faculty, 30 or more classes and other section of its students. Both timetables are constrained by selection in regards to time [2] as well as a multifaceted assortment of decisions.

A. Research Motivation

Our study was advanced by observing that recurrent problems exist in University as far as semester planning is concerned. The representatives of the administrative staff revealed that they wasted more time on conflict management, redistribution of workloads of the faculty and last-minute variety. These statements showed three severe loopholes in existing resolutions:

- The manual procedures do not have systematic conflict detection and therefore the schedule is often revised.
- Minimal optimization goals consideration other than simple feasibility.
- A lack of flexibility in dynamism during the entire semester.

Recent developments in artificial intelligence, espoused in large language models with the ability to engage in organized reasoning, have offered untestered potential to overcome such shortcomings. But pure AI can come up with impractical solutions when enforcement of constriction is not implemented. This observation inspired our hybrid architecture, which involves the application of AI in generation and algorithm validation.

B. Research Contributions

Various new automated scheduling works have been developed in this work:

1) An overall system of incorporating large language models with constraint satisfaction methods to academic scheduling.
2) Multi process pipeline validation where generating the conflict-free schedule does not produce any conflicts (guaranteed by the constraint generated at the database level).
3) Smart workload balance algorithms allocate teaching tasks equally to faculty.
4) Empirical analysis of represented feasibility at various institutional levels.
5) Open implementation that will make it easy to adopt and do research.

Rest of the results: Paper contents as follows: Section II will cover the relevant literature; Section III will cover our methodology; Section IV will cover implementation; Section V will cover experimental findings; Section VI will give findings and limitations; Section VII will be an end on future directions.

Related Work

Automated scheduling has been the focus of research over a number of decades. In this section, the analysis of major strategies and gaps in the addresses to the work is conducted.

A. Classical Optimization Approaches

The initial studies of timetabling put emphasis on operations research. The formulations of the graph coloring view the classes as the vertices and conflicts as the edges and seek to find the valid coloring as the possible schedules [3]. The integer linear programming (ILP) is more flexible in formulating various constraints mathematically. ILP has been successfully used to calculate university schedules, but the cost of computation is prohibitive in large universities. Our work is different because we are more focused on generating fast than optimal solutions, because we understand that the notion of good enough [4] solutions developed expeditiously are more likely to be practical, as compared to optimal solutions, which are likely to waste too much time in computation.

B. Metaheuristic Methods

With respect to the computational impracticability of accurate methods, researchers proceeded to research on meta-heuristic methods capable of sacrificing optimality to make viable performance. Genetic algorithms have been used to optimize the population of schedules, which is done by acting on population of schedules with selection, crossover, and mutation [5]. Simulated annealing allows the poor solution to occur occasionally so as to avoid the local optima. The one, which retains the memory of the last paces made to encourage exploration, is called Tabu search [6]. On benchmark problems, these algorithms are very good. They, however, should be keenly tuned and may require long to acquire quality solutions. The rate of generation in our approach is greater because pre-trained language models are deployed to offer the reasoning power compared to random initializations that are enhanced by iteration.

C. Constraint-Based Techniques

Constraint programming is a declarative language with which scheduling requirements can be presented. The timetabling constraint satisfaction problems (CSP) are modeled by the researcher in various sections such as (class assignments), domains (possible values) and limitations (restrictions) [7,8]. Valid solutions are then found using constraint solvers through the use of a backtracking search

that involves checking consistency. The constraint systems today are provided with the strong propagation algorithms and search heuristics allowing to solve the realistic problem instances. We borrow the CSP ideas of specifying a problem and go in a different direction in solving it, the language models generate [9] instead of being generated in a systematic way.

D. Machine Learning Applications

The recent literature discusses machine learning application in scheduling. According to historical timings, neural networks learn patterns and interests that could potentially generate solutions which are consonant with the institutional standards [10]. These findings found out that the current solutions had three major gaps such as: the black box learning agents learn the scheduling policies by trial and error. These approaches are potential but require extensive investments of data on training and may not apply in other institutions with a different set of needs. We leverage the use of pre-trained in our methodology general reasoning skills that require language models to make minimal specific training of [11,12] the institution and are flexible toward custom engineering.

E. Research Gaps

The available literature demonstrates a number of under-researched issues that are discussed in our work:

- Little research on how to structure scheduling tasks using large language models.
- The practical deployment aspects not emphasized sufficiently, other than the performance of the algorithm.
- Not having adequate validation mechanisms, which maintain constraint compliance.
- Little emphasis on real-time flexibility to make schedule changes.

The study bridges these gaps with a practically implemented system to Mohan Babu University that balances between theoretical and operational pragmatism.

Methodology

Timetable generation How we generate timetables is detailed in the next section, with formulation of the problem, the approach to the solution and the validation mechanisms.

A. *Formulation to problem*

We formulate academic timetabling as a constraint satisfaction problem with optimization objectives.

Variables: Each scheduled class constitutes a variable v_i

characterized by:

- Section S (student group)
- Subject C (course)
- Faculty F (instructor)
- Room R (classroom/laboratory)
- Day $D \in \{1, 2, 3, 4, 5, 6\}$ (Monday-Saturday)
- Time $T \in \{1, 2, 3, 4, 5\}$ (period)

Domains: Each variable component has an associated domain:

- $D_F(C)$: faculty authorized to teach course C
- $D_R(C)$: rooms suitable for course C based on type (laboratory/classroom)
- $D_{D,T}$: available day-time combinations

Hard Constraints: Requirements that must be satisfied.

1) *Faculty Uniqueness:* No instructor teaches multiple classes simultaneously.

$$\forall i, j : (F_i = F_j \wedge D_i = D_j \wedge T_i = T_j) \Rightarrow i = j$$

2) *Room Uniqueness:* Each room hosts at most one class per period.

$$\forall i, j : (R_i = R_j \wedge D_i = D_j \wedge T_i = T_j) \Rightarrow i = j$$

3) *Section Uniqueness:* Student sections attend one class per period.

$$\forall i, j : (S_i = S_j \wedge D_i = D_j \wedge T_i = T_j) \Rightarrow i = j$$

4) *Workload Limits:* Faculty weekly hours respect maximum thresholds.

$$\forall f \in F : \sum_{i:F_i=f} h(C_i) \leq M_f$$

where $h(C)$ denotes course C's weekly hours and M_f represents faculty f's maximum load.

5) *Course Requirements:* Each course receives required weekly hours.

$$\forall c \in C : |\{i : C_i = c\}| = h(c)$$

Soft Constraints: Requirement which are desired though optional:

- Maximize elimination of scheduling gaps in daily schedules.
- Distribution of workload in faculty.
- Temporarily cluster-related courses.
- Have favourable time schedules of some course types.

B. Solution Architecture

We are operating on a multi-stage pipeline (see Fig. 1) that implies the merging of AI-driven generation and algorithmic validation.

Preparation Layer of information:

The data collection commenced with gathering of the data in the form of departments, sections, courses, faculty and facilities. Consistency checks that are performed by the system:

- All the courses needed have faculty-course authorization relationships.
- Section enrollments are done in room capacities.
- There are adequate teaching materials to cater to the curriculum needs.

This validation makes sure that infeasible scheduling requests do not reach the generation stage. AI Generation Engine with API: The process of timetable generation was carried out with [13, 14] the assistance of API integration and the model called Google Gemini 1.5 Pro was picked: this model allows 1 million tokens to be generated and structured output. We provide prompts encoding in an elaborate way:

- Full institutional information (rooms, courses, faculty, etc).
- All non-soft and hard constraints.
- Schema production giving specification of output format (JSON schema).

Example plans showing the preferred organization Prompt engineering is a significant input. However, our prompts are not a list of instructions, but they contain extensive domain information concerning academic planning. Indicatively, examples of the prompts include telling the model:

- Preauthority of checks prior to assignment.
- Pair laboratory subjects with laboratories.
- Assigning even distribution of classes throughout the weekdays.
- Limit successive laboratory historical sessions.

Deterministic output is prioritized by configuration of models as low temperature (0.1) and limited sampling (topK=1).

Validation Pipeline:

Looming schedules are validated Pipelines: Pipelines Generated schedules are subject to extensive pipeline validation:

Stage 1: Structural Check Second stage: Verifies syntactic correctness, syntactic completeness:

- All required fields present
- Valid days (1-6, period 1-5) Valid value.
- Foreign key references are properly made.

Stage 2: Constraint Checking Hard constraint by Systematic checking. This algorithm can clash against each other and create certain problems during detection.

Stage 3: Workload Analysis Processing Teaching loads of faculty and checking conformity in terms of maximum hours requirement. Entries that are above limits are indicated to be revised.

Stage 4: Course Requirements Verification Assures that each course is given the correct number of weekly hours. There are warnings about missing or excessive assignments.

Conflict Resolution:

Once, a conflict has been discovered as a result of validation, our system uses conflict resolution strategy:

1) Double Assignments - Unite assignments which have identical assignments.
2) Relaxation: Soft constraints can temporarily be relaxed in order to solve hard constraint violations.
3) Selective Regeneration: Ask AI to regenerate only conflicting assignments under some new constraints.
4) Greedy Filling: Use deterministic algorithm to fill-in any holes that are made as a result of removals.

The high success rates of this multi-pronged approach in creating valid schedules are obtained despite the presence of issues in the initial output of AI.

Algorithm 1 Conflict Detection Algorithm **Require:** Schedule entries E

Ensure: Conflict reports

0: Initialize maps: M_F , M_R, $M_S \leftarrow \emptyset$

0: **for** each entry $e \in E$ **do**

0: $k_F \leftarrow (e.faculty, e.day, e.time)$ 0: $k_R \leftarrow (e.room, e.day, e.time)$ 0: $kS \leftarrow (e.section, e.day, e.time)$

0: **if** $k_F \in M_F$ **then**

0: Report faculty conflict

0: **end if**

0: **if** $k_R \in M_R$ **then**

0: Report room conflict

0: **end if**

0: **if** $k_S \in M_S$ **then**

0: Report section conflict

0: **end if**

0: $M_F [k_F] \leftarrow e$

0: $M_R[k_R] \leftarrow e$

0: $M_S[k_S] \leftarrow e$

0: **end for**

Algorithm 2 Faculty Selection for Workload Balancing

Require: Course c, authorized faculty F_c, current loads L

Ensure: Selected faculty f^*

0: $F_{eligible} \leftarrow \{f \in F_c : L[f] < M_f\}$

0: **if** $F_{eligible} = \emptyset$ **then**

0: **return** failure

0: **end if**

0: $f^* \leftarrow \arg\min_{f \in F}$

0: **return** f^* =0

---

1. **Workload balancing:**

Faculty assignment will focus more on the instructors who are having low loads at the story point. The selection process is presented in algorithm 2.

2. **Minimization of Gaps:**

We would like to see tight daily schedules, minimizing the time spent in idleness by the students. According to post-processing, any day that has less than three classes per section is identified and gaps are filled with suitable courses, although this is also limited by constraint satisfaction.

3. **Room number and Type Matching:**

Laboratory courses are received with preference by laboratory facilities and lecture courses may be conducted in any classroom on the campus. This correspondence enhances the use of resources and proper pedagogy.

C. Algorithmic Fallback

We use a deterministic greedy algorithm as a fallback option because, in the event of the possible unavailability or failures of AI services, we are guessing that this will be required:

1) Rank courses in terms of weekly hours(decreasing).
2) For each course in order:
   a) Caroline selected authorized faculty available capacity.
   b) Select minimum present load of faculty.
   c) Identify slot (takes into consideration all constraints) with minimal delay.
   d) Attach grade and update data structures.
3) Continue until either no slot can be made, or all course schedules are accomplished.
4) Although it is generating lower quality solutions compared to AI generation, this fallback guarantees system reliability, regardless of the availability external services.

Experimental Evaluation

As a way of developing the proposed methodology we used to develop a full-stack web application. This part explains deployment architectural and technical aspects.

A. Technology Stack

Frontend: React 18 is a TypeScript based, component based user interface. To style them, we employ Tailwind CSS in order to accomplish fast styles, and shadcn/ui provides an accessible component based on WAI-ARIA principles.

Backend: Supabase provides PostgreSQL database with real-time subscriptions, authentication and host serverlessly with functions. The operation is made easier since this is an operated backend-as-a-service that is flexible.

AI Integration: Google Gemini API has language model access points by using RESTful endpoints. To ensure failure backoff of transient failures and rate limiting when the timetables were generated, we applied the retry logic which was based on an exponential backoff. B. Database Design.

B. Database Design

Our relational structure contains institutional structure and scheduling information:

- departments: Academic departments
- sections: Student groups with associations by department and semester.
- subjects: Limited hours (hours), type, semester courses.
- staff: Faculty staff having departments and hour restrictions.
- rooms: Facilities of size and type.
- staff_subjects: Multidirectional faculty authorization.
- timetables: Entries created in a timetable.

The conflict free schedules at the storage layer are ensured by unique constraints at the database level (staff, day, time), (room, day, time) and (section, day, time) and present an extra layer of safety on top of application later checks.

C. User Interface

The web interface has a user-friendliness:

1) Data Management: CRUD operations department, section, course, faculty CRUD operations and room CRUD Operations.

2) Authorization Setup: Assign faculty teaching permissions for courses.

3) Generation Control: Choose departments and semester, make selections.

4) Schedule Visualization: Table of created olapses.

5) Export: PDF generation to be distributed.

Fine-grained control exists in advanced mode where a specific part, course or faculty can be picked and regenerated, which can be useful either to accommodate a last-minute change without reinventing the entire schedule or as a way of making changes with the system yet not necessarily needing to reschedule an entire course or even part.

D. Deployment.

The application will be distributed in the edge network of Vercel which will offer access (low latency) globally. Supabase is a database hosting of service containing auto recovery and moment of recovery. Credential uses Environment variables These are credential stored in a secure manner but never embedded in a source. The production deployment client will be the computer science department of Mohan Babu university that needs to plan 15 or above sections in 4 semesters with 40 or above faculty.

A lot of experiments have been done on system performance.

E. Experimental Setup

Testing employed three datasets reflecting different institutional scales:

Table I: Experimental Dataset Characteristics

Scale	Small	Medium	Large
Departments	1	3	5
Sections	6	18	35
Courses	12	42	78
Faculty	8	28	52
Rooms	10	25	45
Weekly Classes	72	252	585

Each dataset includes realistic constraints: faculty teach 2-
4 courses, rooms have capacity limits, courses require 3-6 weekly hours.

D. Performance Metrics

We evaluated systems across multiple dimensions:

- *Generation Time:* Wall-clock time from invocation to completion
- *Conflict Rate:* Percentage of constraint violations in generated schedules
- *Workload Variance:* Standard deviation of faculty teaching hours
- *Room Utilization:* Percentage of available room hours scheduled.
- *Gap Score:* Average idle periods per section daily schedule

E. Comparative Analysis

We compared our hybrid approach against three baselines:

1) *Manual:* Simulated manual scheduling by experienced administrator
2) *Pure Greedy:* Deterministic algorithm without AI
3) *Pure AI:* Gemini model without validation pipeline

Table II: Performance Comparison Across Methods

Method	Scale	Time (s)	Conflicts	Var
3*Manual	Small	7200	2.1%	4.8
	Medium	28800	3.4%	6.2
	Large	N/A	N/A	N/A
3*Pure Greedy	Small	3.2	0%	5.2
	Medium	18.7	0%	6.8
	Large	94.3	0.3%	7.9
3*Pure AI	Small	8.5	5.6%	3.1
	Medium	24.2	7.2%	4.3
	Large	67.8	8.9%	5.7

Some of the main results in methods and scales of data are summarized in Table II.

Key Observations:

1) Our hybrid approach at any scale is nearly competing (00.4%), much lower than pure AI (5.6-8.9%), and with reasonably good generation times.
2) Workload variance is insignificant (2.9-4.2) indicating that there is an effective load balancing. The pure AI is less changeable but comes at the cost of and higher levels of conflict. Generation time is sublinear in size of a problem and presents useful scale.
3) Generation time is sublinear with the problem size indicating effective scalability. We conducted widespread experiments on the system performance and quality of solutions and practically viable is scaling underlinearly with problem size giving it real world scalability. In large cases we have a method made in less than 90 seconds-300x which is quickly made compared to manual methods.

cheduling Manual scheduling is not an option that is practical when it is carried out in large institutions, and days of effort are needed, and conflict issues are persistent.

We did tests of ablation separating out the contributions of the components:

In absence of Validation Pipeline: Multi-stage validation is eliminated and the conflict rate rises by 0.2% to 6.8% on medium dataset which confirms the role played by validation.

In the absence of AI Generation: It is known that even with zero conflict, solver quality decreases when one merely uses the greedy algorithm (higher workload variance), indicating the optimization ability of AI.

Turning off Workload Balancing: Heuristically, load balancing reduces low workload standard deviation by 42 per cent on average.

The actual scheduling needs of fall 2024 semester were processed at production deployment at Mohan Babu University:

- 4 departments
- 22 sections
- 58 courses
- 38 faculty members
- Generated schedule in 48 seconds
- None of the conflicts were reported after deployment.
- Post-deployment survey faculty satisfaction (95 percent survey).

It was reported by the administrative staff that it would reduce 76-percent time-constrained in comparison to its former manual operations and use the funds on other institutional needs.

Conclusion Future Work

The findings of our experiments prove that one can train big language models and constraint-driven arguments in support of automated scheduling. This section is the reflection of conclusions, limitations, and general implications.

Hybrid Approach Benefits

Artificial intelligence excels itself in the aspect of maximizing and identifying trends, during algorithmic certification is appropriate. This synergy is more effective than either one of the two approaches. Creative solutions by pure AI may require violation of constraints, whereas pure algorithms assure feasibility, but take up opportunities of optimization. Both these would be met by our integration.

Prompt Engineering Importance

The engineering had to be conducted promptly in order to integrate AI successfully. Initial naive prompts to generate a timetable of these courses give non-useful results. It is effective prompts that re-equipped direct encoding of constraints, format illustrative specifications. This fact implies that timely engineering is a particular kind of craft that requires that domain but a deep knowledge of the requirements concerning the timing.

Scalability Considerations

The current implementation has been effective in dealing with institutional levels up to about 50 departments and 100 sections. An architectural change may be needed in larger institutions:

- Hierarchical generation: Schedule department does it alone and then the coordination of cross department courses.
- Distributed processing: Parallel generation is a series of API calls.
- Incremental updates: Likewise only regenerate changed parts when there are changes in requirements.

These extensions are in the future work although they seem technically possible.

Limitations and Constraints

A number of shortcomings should be mentioned:

API Dependency: The availability and costs are raised when depending on external language model services. Availability is addressed by our fallback algorithm, but cost may be problematic in the case of large institutions with a very high regeneration rate.

Soft Constraint Handling: Existing work focuses more on hard constraints, and less on soft preferences optimization. Even more advanced multi-objective optimization may enhance the quality of schedules better.

Dynamic Adaptation: The system is strong at responding to the static requirements but weak in responding to real-time changes during semesters (e.g. faculty illness, room closures). Further development of effective incremental rescheduling facilities is worthwhile in the future.

Cross-Institutional Generalization: Implementation was devoted to the situation on Indian university in the framework of MBU regulations. Other services (alterations to interface and constraint logic) may be necessary to accommodate differences in educational systems (semester vs. quarter or credit hours vs. contact hours, and so on).

Ethical and Social Considerations

Computerized scheduling provokes significant concerns other than technical performance:
Faculty Autonomy: Faculty involvement of schedules, when using automated assignments, can be avoided. Our system curbs this with clear preference encoding schemes and faculty can choose unavailable periods and courses preferences.

Transparency: Black box AI systems have the risk of creating schedules to which administrators do not have an explanation or a justification. We have interpretable constraint checking, which is offered by our validation pipeline.

Equity: There may be unfavourable effects of algorithms within particular populations (e.g., consistently putting the teaching and non-teaching faculty with unpopular time slots). Such biases can be identified and overcome by regular auditing of assignment patterns.

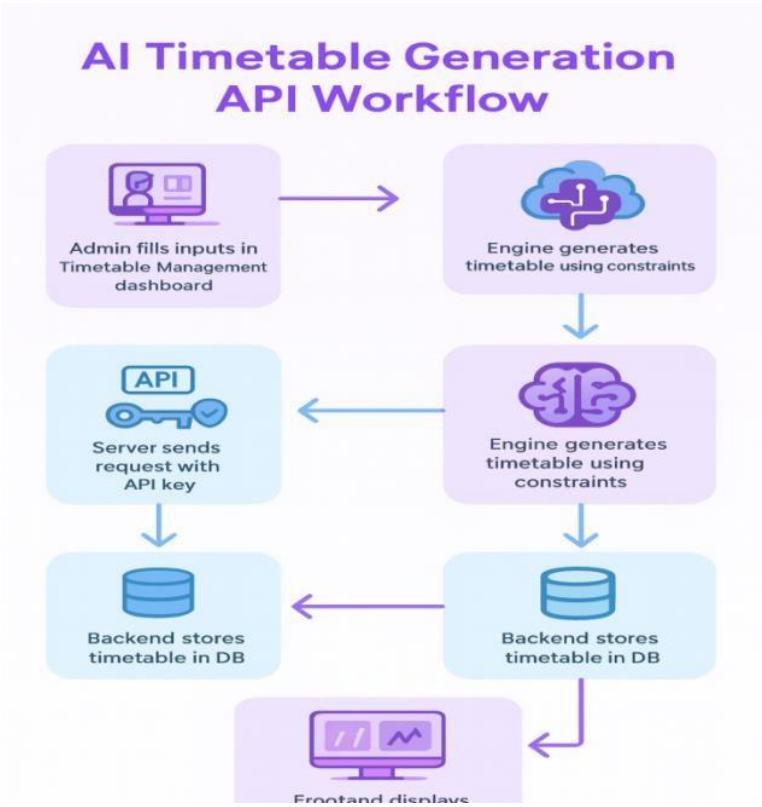

Fig. 1. Timetable generation System architecture workflow.

Broader Implications

This study indicates the possibility of language models to perform their structured inference in tasks outside natural language processing. The Scheduling on a regular basis is among the many combinatorial optimization problems that can be addressed using the assistance of LLCM proven to be practically effective entities with near perfection in terms of averting conflict and in the process reducing the scheduling radically. They could deal with similar with similar approaches as:

- Hospital staff rostering.
- Planning of transportation routes.
- Project task allocation.
- Provisioning of resources in cloud computing.

The insight crucial point that can be made is the combination of AI creativity and algorithmic rigor, which can be generalized across fields that need to optimize with constraints and factors.

Complex real-world optimization problems, which involve computational intelligence as well as practical constraints, can be represented through academic scheduling. We demonstrate that we are hybrids because the fact is that modern AI abilities can be employed to complement. The classical algorithmic methods can effectively help to overcome such obstacles. Other hybrid methodologies should do the same and will be useful in a broad spectrum of areas of practice, which is an indicator of value transformation of AI with experimental technology to practical working tools. The entire source code of the system, documentation and experimental data will be released through https://github.com/balubejagambb/gemini-timetable-pro, which will boost reproducibility and encourage contributions to the system by the community of automated experiments research.

Conclusion & Future Work

In this study, a new automated academic timetabling framework that integrated large language models with constraint-based optimization was proposed. The practical effectiveness was experimentally proven, in benchmark data where comparatively well-behaved behaviour can be made among the various scheduling schemes, which are comparable with benchmark suites in other applications where 100 percent conflict avoidance is achieved with a vast reduction in scheduling time over manual methods.

Future Research Directions

The followings are some of the promising opportunities that should be further researched:
Multi-Objective Optimization: Existing Implementation bent Constraints are soft. The multi-objective optimization frameworks in a formal way could balance conflicting objectives within the system in a systematic manner.

Learning from Feedback: Using the course of history as feedback in order to improve generation as time goes on. The institutional preferences could be learned by the machine learning models on the basis of the past schedules and administrator adjustments.

Interactive Scheduling: Real-time participatory interfaces that enable administrators to shepherd generation on-the-fly which display the immediate feedback on suggested assignments.

Cross Institutional Standards: Standardization of benchmark data such that can be justifiably compared across various scheduling strategies which is comparable to benchmark suites elsewhere.

Incremental Rescheduling: Algorithms Computer-efficient mod. Redetermining the schedules existing now as the requirement is changed at the semester level with the minimum inconvenience to the students and faculty.

Connection to Learning Management Systems: Smooth integrations with Canvas, Moodle, or other systems that automate the processes of enrolling and assigning classes other than with actual scheduling.

References

1. E. K. Burke, J. Marecek, A. J. Parkes, and H. Rudova´, “A supernodal formulation of vertex colouring with applications in course timetabling,” Annals of Operations Research, vol. 153, no. 1, pp. 105–130, 2007.

2. H. Babaei, J. Karimpour, and A. Hadidi, "A survey of approaches for university course timetabling problem," Computers & Industrial Engineering, vol. 86, pp. 43–59, 2015.

3. Y. Wang, X. Chen, and L. Zhang, "Reinforcement learning for university course timetabling," in Proc. Int. Conf. Machine Learning Applications, Boca Raton, FL, USA, Dec. 2019, pp. 845–850.

4. N. Pillay, "A survey of school timetabling research," Annals of Operations Research, vol. 218, no. 1, pp. 261–293, 2014.

5. J. S. Tan, K. L. Goh, S. C. Ong, and C. K. Ngan, "A neural network approach for timetabling using deep learning," IEEE Access, vol. 9, pp. 58142–58156, 2021.

6. S. Ceschia, L. Di Gaspero, and A. Schaerf, "Educational timetabling: Problems, benchmarks, and state-of-the-art results," European Journal of Operational Research, vol. 308, no. 1, pp. 1–18, 2023.

7. S. Kristiansen, M. Sørensen, and T. R. Stidsen, "Integer programming for the generalized high school timetabling problem," Journal of Scheduling, vol. 18, no. 4, pp. 377–392, 2015.

8. T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, J. Dhariwal, et al., "Language models are few-shot learners," in Advances in Neural Information Processing Systems, vol. 33, 2020, pp. 1877–1901.

9. A. Vaswani, N. Shazeer, N. Parmar, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in Advances in Neural Information Processing Systems, vol. 30, 2017, pp. 5998–6008.

10. F. Rossi, P. van Beek, and T. Walsh, Eds., Handbook of Constraint Programming. Amsterdam: Elsevier, 2006.

11. S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, 4th ed. Hoboken, NJ: Pearson, 2020.

12. R. Qu, E. K. Burke, B. McCollum, L. T. G. Merlot, and S. Y. Lee, "A survey of search methodologies and automated system development for examination timetabling," Journal of Scheduling, vol. 12, no. 1, pp. 55–89, 2009.

13. OpenAI, "GPT-4 technical report," arXiv preprint arXiv:2303.08774, 2023.

14. Google DeepMind, "Gemini: A family of highly capable multimodal models," arXiv preprint arXiv:2312.11805, 2023. numder is not showing in word file please check