Scalable and Ethical Medical Scheduling: An ASP-Based Framework with Patient-Centered Experimental Validation

Alina Vozna[1,2], Andrea Monaldini[1,2], Stefania Costantini[1,3], Dawid Pado[1] and Valentina Pitoni[1]

[1]*Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila,Italy*
[2]*University of Pisa, Largo B. Pontecorvo, Pisa, 56127, Italy*
[3]*Gruppo Nazionale per il Calcolo Scientifico - INdAM, Roma, Italy*

Abstract

This paper presents a modular, patient-centered scheduling framework for medical appointments based on Answer Set Programming (ASP). The system integrates Blueprint Personas—structured patient models reflecting clinical, cognitive, and social factors, to enable personalized and ethically informed decisions. We validate the approach through simulations with up to 2000 patients across scenarios of increasing complexity. Results show that the system efficiently generates optimal, constraint-compliant schedules in real time, with linear scalability. This confirms the effectiveness of ASP as a transparent and adaptable tool for healthcare scheduling.

Keywords

Blueprint Personas, Answer set programming, Healthcare AI, Real-Time Decision Making

1. Introduction

Efficient scheduling of medical appointments is vital to optimizing healthcare delivery and improving patient outcomes. Traditional systems, often based on manual or rule-based mechanisms, struggle to balance competing constraints such as patient urgency, clinician availability, and accessibility, resulting in inefficiencies such as extended waiting times, no-shows, and underused resources [1, 2].

This paper builds on a previously accepted work in which we introduced a patient-centered scheduling framework grounded in Answer Set Programming (ASP) [3, 4, 5, 6], a declarative paradigm suitable for modeling complex decision-making with multiple constraints. ASP enables real-time adaptability, constraint satisfaction, and transparent reasoning, making it particularly suited for dynamic healthcare scenarios. A distinguishing feature of our approach is the integration of Blueprint Personas [7], structured patient models that capture individual preferences, accessibility needs, and clinical priorities. These personas enable personalized, ethically aligned scheduling decisions that account for social, cognitive, and economic factors. The proposed solution is designed to complement centralized healthcare infrastructures such as Italy's *Centro Unico di Prenotazione* (CUP) [8], which currently lack support for nuanced and patient-centered constraints. In contrast, our ASP-based model supports complex prioritization, optimizes resource allocation, and reduces administrative overhead.

In this paper, we focus on the experimental validation of the framework, providing a detailed empirical evaluation of its performance and adaptability in realistic healthcare scheduling scenarios. Our work confirms that a modular and fully declarative scheduling system, centered on individual patient needs, can effectively manage complex constraints in real time.

Joint Proceedings of the Workshops and Doctoral Consortium of the 41st International Conference on Logic Programming (ICLP-WS-DC 2025), September 12, 2O25, Rende, Italy

Corresponding author.
alina.vozna@student.univaq.it (A. Vozna); andrea.monaldini@student.univaq.it (A. Monaldini); stefania.costantini@univaq.it (S. Costantini); valentina.pitoni@univaq.it (V. Pitoni)
0009-0009-0179-6948 (A. Vozna); 0009-0004-2518-2055 (A. Monaldini); 0000-0002-5686-6124 (S. Costantini); 0000-0002-4245-4073 (V. Pitoni)

2. Related Work

The scheduling of medical appointments has been extensively investigated using queueing theory, simulation, and optimization methods to address uncertainty in service delivery and resource limitations [1]. Recent research extends these approaches through decision-support systems for outpatient scheduling [9].

Optimization techniques, including Mixed-Integer Linear Programming (MILP), metaheuristics, and Stochastic Programming, have been employed to improve access and resource usage [10]. Algorithms such as Fuzzy Ant Lion Optimization (FALO) and Discrete Event Simulation (DES) integrated with behavioral models have enhanced scheduling fairness and adaptability to cancellations [11, 12]. Reinforcement learning and deep learning methods have also shown promise in predicting patient behavior and minimizing waiting times [13, 14].

Answer Set Programming (ASP) has emerged as an effective approach in complex healthcare scheduling. It has been applied to operating room [15] and nurse scheduling [16], and to clinical pathway planning using Logic-Based Benders Decomposition (LBBD) for chronic patients with comorbidities [17]. Kanias [18] further demonstrated the integration of ASP with databases for fairness-aware rescheduling at scale.

Despite these advances, practical issues such as dynamic constraints, real-time updates, and individual patient needs remain insufficiently addressed [19]. Decentralized models (e.g., DCOPs) have introduced negotiation-based agent scheduling [20], but still lack holistic patient modeling.

Our approach leverages the declarative expressiveness of ASP to ensure strict constraint satisfaction, explainable reasoning, and adaptability. Unlike data-driven or heuristic-based techniques, our method integrates medical, behavioral, and accessibility constraints through patient-centered Blueprint Personas. This bridges the gap between theoretical optimization and real-world scheduling needs in centralized healthcare systems.

3. Problem Description and Formalization

The scheduling of medical appointments is a complex task which requires coordination among clinical urgency, patient preferences, provider availability, and resource limitations. We model the scheduling process as a multi-objective constrained optimization problem and solve it using Answer Set Programming (ASP), which enables the declarative specification of constraints and preferences.

The problem involves three primary entities: patients, doctors, and clinics. Patients have varying urgency levels and may express preferences regarding time slots, clinics, and environmental conditions. Doctors operate under workload limits and are affiliated with specific clinics. Clinics offer appointment slots and are subject to resource and budget constraints.

The goal is to generate appointment assignments that satisfy all hard constraints, such as compatibility, availability, and accessibility, while optimizing soft objectives like minimizing patient waiting time, balancing doctor workload, and satisfying patient preferences.

3.1. User Modeling and Personas

Effective appointment scheduling extends beyond clinical coordination; it must also address individual patient needs. To this end, we adopt **Blueprint Personas**, structured, evidence-based patient archetypes developed in European digital health programs [21]. These personas integrate clinical, social, behavioral, and cognitive dimensions, providing an abstraction layer that enhances patient-centered decision-making.

Patient Personas. Patient profiles incorporate medical conditions, environmental constraints, and preferences about clinics, time slots, sensory conditions, and specific doctors. Although digital literacy is not directly used in scheduling logic, it supports system-level customization such as interface selection or telemedicine eligibility. We define three levels of patient complexity **Generally healthy individuals** prioritize convenience but have no critical constraints; **Patients with chronic conditions** require recurrent appointments, cost-sensitive scheduling, and consistent follow ups; **Patients with complex needs** may need sensory-friendly environments, accessible infrastructure, and caregiver support. Travel distance is also encoded, with telemedicine and home care set to zero. The following ASP snippet illustrates a sample patient profile (p), expressing clinic (c) and sensory preferences, doctor requirements, and appointment slots:

```
patient(p1, "Mario", "Rossi", "L'Aquila").
disabled(p1).
preference(p1, c3).
sensory_preference(p1, "noise").
doctor_preference(p1, "GP", "chronic_diseases", 10).
appointment_preference(p1, c3, 1850, 2000).
distance(p1, c3, 15).
```

Listing 1: Patient Profile with Preferences and Constraints

Clinician Personas. Clinicians are described by their specialty, operational context, and system engagement, enabling the scheduler to anticipate real-world constraints and improve care coordination. Visits are categorized by type, chronicity, cost, and whether in-person attendance is required. Multi-session treatments specify minimum and maximum intervals:

```
visit_type(v1, "Cardiology", "Heart Attack", 0, 0, 0).
visit_cost(v1, 1000).
required_sessions(v1, 2).
session_interval(v1, 14, 28).
```

Listing 2: Visit Type Definition and Constraints

Clinic availability and patient requests are represented with timestamps and individual urgency levels, enabling fine-grained scheduling aligned with medical and personal needs:

```
patient_interval(p1, v1, 21, 28).
need(p1, v1, 3).
availability(c1, v1, 1727308800).
```

Listing 3: Patient Needs Preferences and Clinic Availability

3.2. Rules of Inference

In the proposed ASP-based model, inference rules are employed to derive utility scores from patient preferences. These rules are not enforced as hard constraints but instead support the optimization process by assigning weighted values to specific scheduling features.

Preference-Based Indicators. Patient and doctor preferences are translated into utility indicators. For example, a preferred clinic or a doctor meeting a patient's specialization and experience requirements contributes positively to the score:

```
clinic_preference_effect(Patient, Clinic, 1) :- preference(Patient, Clinic).
clinic_preference_effect(Patient, Clinic, 0) :- not preference(Patient, Clinic).

doctor_preference_effect(Patient, Doctor, 1) :-
    patient(Patient, _, _, _),
    doctor(Doctor, _, _, _, _, Type),
    doctor_experience(Doctor, Specialization, YearsExperience),
    doctor_preference(Patient, Type, Specialization, RequiredYears),
    YearsExperience >= RequiredYears.
% Additional rules cover non-matching or underqualified doctors
```

Listing 4: Clinic and Doctor Preference Effects

Preference indicators such as doctor_preference_effect are introduced as auxiliary predicates to modularize the representation of soft constraints. Instead of defining multiple weak constraints, we aggregate these indicators in a single optimization statement (Listing 7), enabling fine-grained weight control and a unified cost model. This design choice improves readability, maintainability, and explainability of the optimization process.

Time and Sensory Alignment. Preferences for appointment time windows and sensory conditions (e.g., noise sensitivity) are also integrated. A patient receives a utility bonus if the appointment time aligns with their preferred interval:

```
appointment_preference_effect(Patient, Time, Clinic, 1) :-
    availability(Clinic, _, _, Time),
    appointment_preference(Patient, _, Start, End),
    X = (((Time \ 86400) * 3600) * 100) + (((Time \ 3600) / 60) / 3) * 5,
    X <= End, X >= Start.
% Additional rules penalize non-aligned time slots
```

Listing 5: Effect of Time Preferences

Environmental mismatches (e.g., noisy clinic during a patient's sensitive period) are penalized using the following rule:

```
sensory_penalty(Patient, Clinic, Time, Level) :-
    sensory_preference(Patient, Type),
    environmental_condition(Clinic, Type, Level, Start, End),
    Time >= Start, Time <= End.
sensory_penalty(Patient, Clinic, Time, 0) :- not sensory_preference(Patient, _).
```

Listing 6: Sensory Penalty Calculation

Optimization Statement. These indicators are aggregated in the ASP solver's optimization statement. It seeks to minimize travel and waiting time while maximizing preference satisfaction. The following expression encodes the trade-offs:

```
#minimize {
    (Distance * 10000) + WaitTime + (Penalty * 1000) -
    (ClinicPreference * 10000) -
    (DoctorPreference * 1000) -
    (AppointmentPreference * 1000) :
        distance(Patient, Clinic, Distance),
        appointment(Patient, Clinic, Doctor, _, Time),
        current_time(CurrentTime),
        WaitTime = Time - CurrentTime,
        sensory_penalty(Patient, Clinic, Time, Penalty),
        clinic_preference_effect(Patient, Clinic, ClinicPreference),
        doctor_preference_effect(Patient, Doctor, DoctorPreference),
        appointment_preference_effect(Patient, Clinic, Time, AppointmentPreference)
}.
```

Listing 7: Optimization Function Using Weighted Utilities

This optimization balances utility-based soft preferences with operational constraints, enabling the generation of personalized, explainable, and ethically grounded schedules.

The weights used in the optimization function (Listing 7) were empirically chosen based on clinical priorities and expected user impact. Waiting time is penalized most heavily, reflecting the importance of timely access to care. Travel distance and sensory discomfort carry medium weights, as they significantly affect user satisfaction, particularly for vulnerable patients. Preference satisfaction (clinic, doctor, and time) is weighted moderately to encourage alignment with user needs, without overriding urgency or feasibility constraints. These values were fine-tuned through iterative testing to balance solver performance and realistic scheduling behavior. The generation of feasible appointment combinations is handled via standard ASP choice rules. These rules non-deterministically produce all admissible assignments from available slots, constrained by clinical needs and visit requirements. For instance, appointments are generated with cardinality bounds based on required sessions, enabling the solver to explore multiple valid configurations before applying constraints and optimization.

4. System Architecture

The scheduling system is designed as a layered architecture built upon a microservice-based infrastructure. Its modular structure ensures scalability, maintainability, and seamless integration with external healthcare platforms.

The **User Layer** consists of front-end interfaces and IoT components that allow patients or clinicians to trigger appointment requests. These interfaces may include web applications or automated health monitoring systems capable of initiating bookings based on specific thresholds.

At the core, the **Business Logic Layer** implements the decision-making pipeline through two lightweight Flask-based services. The first service handles REST API requests, performs user authentication (e.g., via JSON Web Tokens), and translates user input into ASP facts. The second service aggregates these facts, builds a complete ASP program every 60 seconds, and solves it using the Clingo solver. The value of 60 seconds is chosen as a compromise between latency perceived by the user and computational load: this time window allows you to accumulate asynchronous requests and process them in batch with Clingo. This allows you to optimize the efficiency of the solver, avoiding too frequent launches.

The **Data Management Layer** relies on a MySQL database that stores persistent information, such as clinic schedules, patient preferences, budget constraints, and environmental parameters. All interactions adhere to ACID principles to guarantee consistency.

The system offers asynchronous RESTful APIs to support functionalities like submitting appointment requests, retrieving booking status, and querying clinic availability. By decoupling the solver from the interface logic and data storage, the architecture supports independent scaling and updates of individual components.

5. Experimental Evaluation

To demonstrate the capabilities of our ASP-based scheduling framework, we present three realistic scenarios reflecting common healthcare challenges. All experiments were executed using Clingo 5.7.0.[1] The scenarios highlight how the system handles multiple constraints, such as urgency, accessibility, and preferences, while producing optimal and constraint-compliant solutions.

The experiments were run on a machine with an Intel Core i7-9750HF CPU @ 2.60GHz, 16GB RAM, and Windows 11, using Clingo version 5.7.2. The scheduling system processes appointment requests in batches, triggered every 60 seconds, simulating asynchronous request handling in clinical settings.

Three experimental scenarios were defined to evaluate the behavior of the system under different levels of complexity:

- **Scenario A**: All patients request the same type of visit (stress test) in a limited set of clinics. This simulates emergency-like settings with concentrated demand.
- **Scenario B**: Two types of visits are available, and patients are evenly split between the two. This scenario increases diversity while retaining overlap.
- **Scenario C**: Each patient requests a different type of visit, simulating the most complex real-world configuration with individualized care pathways.

[1] https://github.com/potassco/clingo/releases/tag/v5.7.0

The synthetic datasets used in these experiments included up to 2000 patients, each associated with unique urgency levels and preferences. The simulated environment also included 50 doctors with varying specializations and years of experience, along with 28 clinics featuring different accessibility configurations and environmental conditions. A total of 100 types of visits and more than 1,000 appointment slots were generated, ensuring coverage of a wide range of temporal and structural scheduling constraints. All data included attributes for preference handling (clinic, doctor, time range, sensory conditions), urgency, distance, and accessibility.

5.1. Scalability Results

To evaluate the scalability of our ASP-based scheduling model, we implemented a dedicated Python script that dynamically generates synthetic test instances based on a user-defined number of patients. This generator produces valid ASP facts on-the-fly, including patient profiles, clinics, doctors, distances, visit types, appointment availability, and temporal constraints. All data is automatically structured to conform to the input requirements of the Clingo solver.

For each test run:

- The number of clinics is computed as $\lceil \text{patients}/100 \rceil + 1$
- The number of doctors is set to $\lceil \text{patients}/10 \rceil + 1$
- Each patient is assigned a unique visit type, simulating a fully individualized scheduling scenario (Scenario C)
- The number of availability entries grows with

Appointment slots are generated as unique Unix timestamps, randomly distributed between 30 and 60 days from the current time. This ensures both realism and uniqueness in slot assignments, preventing overlaps and enforcing valid temporal boundaries. Each test instance is grounded and solved by Clingo in a batch, with the total resolution time measured using Python's built-in timing functions. This method enables fully reproducible and parameterizable scalability experiments. Moreover, the script structure allows quick adaptation to alternative scheduling scenarios (e.g., high contention or overbooking cases) by modifying the `num_of_visit` or distribution logic. The programmatic control of scaling factors such as `patients × clinics × slots` provides insights into how grounding complexity affects solver performance and justifies the observed runtime behavior. Specifically, we demonstrate that linear scaling is achieved in low-contention, diversified scenarios, while denser configurations lead to a faster increase in grounded rules and computation time.

Figure 1 shows the resolution time (in seconds) as the number of patients increases from 100 to 2000, across the three scenarios.

We can observe: **Scenario A** led to the highest computational burden, when more than 200 patients request the same slot range, the solver exceeds the 60-second batch timeout; **Scenario B** showed better performance due to partial diversification; **Scenario C**, with completely individualized visit types, scaled linearly, requiring only 14 seconds to assign 2000 patients. This demonstrates that the system handles complex, heterogeneous scheduling tasks more efficiently than highly constrained, homogeneous ones. It is thus well-suited for real-world outpatient settings with varied visit demands.

5.2. Optimization Behavior

The objective function defined in Listing 7 consistently guided the solver toward preference-aligned, resource-efficient solutions. Across all experiments: Patients with high urgency were prioritized and scheduled earlier; Sensory preferences and travel distance were balanced against clinic and doctor preferences; Patients were never assigned to slots that violated accessibility, budget, or clinical compatibility.

The optimization logic, driven by declarative preference scoring, ensured ethically grounded and explainable decision-making even under load.

The ASP-based system demonstrates robustness, flexibility, and suitability for real-world healthcare scheduling tasks. The results validate the system's applicability in healthcare settings, where rapid

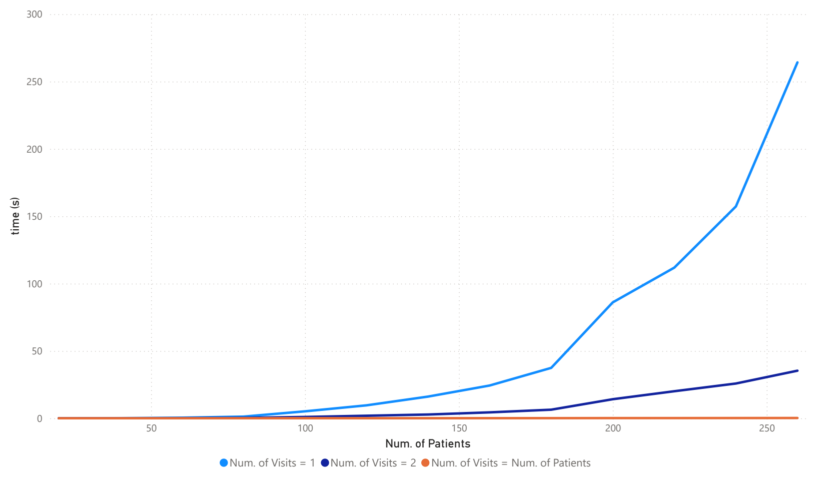

Figure 1: Solver resolution time for different patient loads across three scenarios. Scenario C exhibits the best scalability.

adjustments to cancellations, emergencies, or fluctuating resource availability are essential. Across all test cases, the ASP solver produced optimal schedules in under 0.04 seconds, confirming the feasibility of real-time decision-making even with complex constraints. The number of grounded rules and atoms remained manageable (below 10,000), indicating that the model scales well for small to medium-sized clinics. Preliminary tests with increased patient loads (up to 50 patients) showed that the computation time grows linearly, suggesting good scalability potential with incremental solving techniques.

6. Conclusion and Future Work

This paper presented an ASP-based framework for medical appointment scheduling that balances institutional constraints with patient-centered preferences. The system incorporates rich user modeling via Blueprint Personas and supports multi-session visits, budget constraints, and diverse scheduling needs.

Scalability tests with up to 2000 patients show that the model can compute optimal schedules in under 15 seconds. Inference rules and optimization logic enable explainable decisions that incorporate urgency, preferences, and fairness.

To strengthen the framework and address key limitations, future work will focus on: (1) the integration of real-world clinical datasets and user feedback to enhance external validity; (2) a comparative evaluation against established baselines to better understand computational and qualitative trade-offs; (3) the development and adoption of fairness metrics, equity indicators, and possibly qualitative user studies, in order to substantiate ethical claims with measurable evidence; (4) extending the system to support dynamic rescheduling in response to unforeseen events (e.g., resource failures, clinician unavailability), thereby improving resilience in realistic clinical settings; and (5) minimizing idle time between appointments via soft constraints, extending budget reasoning to the patient side, and integrating digital literacy into the scheduling logic.

Declaration on Generative AI

During the preparation of this work, the author(s) used X-GPT-4 in order to: Grammar and spelling check. After using these tool(s)/service(s), the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] D. Gupta, B. Denton, Appointment scheduling in health care: Challenges and opportunities, IIE transactions 40 (2008) 800–819.

[2] M. H. Alrefaei, A. Diabat, Modelling and optimization of outpatient appointment scheduling, RAIRO-Operations Research-Recherche Opérationnelle 49 (2015) 435–450.

[3] C. Baral, Knowledge representation, reasoning and declarative problem solving, Cambridge university press, 2003.

[4] V. Lifschitz, B. Porter, F. Van Harmelen, Handbook of knowledge representation, Elsevier, 2008.

[5] S. Costantini, A. Formisano, Answer set programming with resources, Journal of Logic and Computation 20 (2010) 533–571.

[6] V. Lifschitz, Answer set programming, volume 3, Springer Cham, 2019.

[7] A. Monaldini, A. Vozna, S. Costantini, Blueprint personas in digital health transformation (2024).

[8] Ministero del Lavoro, della Salute e delle Politiche Sociali, Sistema CUP: Linee guida nazionali, https://www.salute.gov.it/imgs/C_17_pubblicazioni_1577_allegato.pdf, 2009. Accessed: 2025-04-02.

[9] T. Niu, B. Lei, L. Guo, S. Fang, Q. Li, B. Gao, L. Yang, K. Gao, A review of optimization studies for system appointment scheduling, Axioms 13 (2023) 16.

[10] Y.-H. Kuo, H. Balasubramanian, Y. Chen, Medical appointment overbooking and optimal scheduling: tradeoffs between schedule efficiency and accessibility to service, Flexible Services and Manufacturing Journal 32 (2020) 72–101.

[11] A. Ala, V. Simic, D. Pamucar, E. B. Tirkolaee, Appointment scheduling problem under fairness policy in healthcare services: fuzzy ant lion optimizer, Expert systems with applications 207 (2022) 117949.

[12] X. Fan, J. Tang, C. Yan, H. Guo, Z. Cao, Outpatient appointment scheduling problem considering patient selection behavior: data modeling and simulation optimization, Journal of Combinatorial Optimization 42 (2021) 677–699.

[13] A. Ala, F. Chen, Appointment scheduling problem in complexity systems of the healthcare services: A comprehensive review, Journal of Healthcare Engineering 2022 (2022) 5819813.

[14] H. Qiu, D. Wang, Y. Wang, Y. Yin, Mri appointment scheduling with uncertain examination time, Journal of Combinatorial Optimization 37 (2019) 62–82.

[15] C. Dodaro, G. Galatà, M. Gebser, M. Maratea, C. Marte, M. Mochi, M. Scanu, Operating room scheduling via answer set programming: Improved encoding and test on real data, Journal of Logic and Computation 34 (2024) 1556–1579.

[16] M. Alviano, C. Dodaro, M. Maratea, Nurse (re) scheduling via answer set programming, Intelligenza Artificiale 12 (2019) 109–124.

[17] P. Cappanera, M. Gavanelli, M. Nonato, M. Roma, Logic-based benders decomposition in answer set programming for chronic outpatients scheduling, Theory and Practice of Logic Programming 23 (2023) 848–864.

[18] S. Kanias, An ai approach to large-scale medical appointment (re) scheduling using asp., in: ENIGMA@ KR, 2023, pp. 35–42.

[19] A. Kuiper, J. de Mast, M. Mandjes, The problem of appointment scheduling in outpatient clinics: a multiple case study of clinical practice, Omega 98 (2021) 102122.

[20] M. Hannebauer, S. Müller, Distributed constraint optimization for medical appointment scheduling, in: Proceedings of the fifth international conference on Autonomous agents, 2001, pp. 139–140.

[21] C. E., European innovation partnership on active and healthy ageing, https://ec.europa.eu/eip/ageing/home_en.html, 2024. Accessed: 2024-09-23.