

Study Of Different Algorithm Used For Appointment Scheduling.

Prof. Dr. Neelam Kumar¹, Siddhi Satav², Sakshi Sonawane³, Shruti Ummarga⁴, Vedangi Vyas⁵

¹(Professor, SRCOE, Department of Computer Engineering Pune)

^{2,3,4,5}(Student, SRCOE, Department of Computer Engineering Pune)

Abstract: Efficient appointment scheduling is essential for managing resources and minimizing waiting times in healthcare and other service-oriented systems. This paper presents a comparative study of different algorithms used for appointment scheduling, emphasizing their scalability, efficiency, and suitability for various operational complexities. The Simple Booking Algorithm is implemented for small-scale scheduling to ensure that no overlapping appointments occur between patients and therapists. For larger datasets, a Scalable Conflict-Check Algorithm based on Interval Tree or Segment Tree structures is explored to optimize overlap detection and improve time efficiency when handling hundreds of appointments. Finally, the Constraint Satisfaction Algorithm (CSP) is applied to manage complex multi-day therapy schedules, where constraints such as therapist availability, rest periods, and session order must be satisfied. Through this comparative analysis, the study highlights how different scheduling algorithms address varying levels of complexity, offering insights into selecting the most effective approach for specific appointment management needs.

Key words: Appointment Scheduling, Simple Booking Algorithm, Interval Tree, Constraint Satisfaction Problem (CSP), Scheduling Optimization, Healthcare Management

I. Introduction

Appointment scheduling is a critical process in healthcare and service-oriented environments, where efficient time management and optimal resource utilization directly impact productivity and client satisfaction. In healthcare systems, particularly in therapy or treatment centers, it is essential to allocate appointments in a way that avoids conflicts, utilizes available therapists efficiently, and accommodates complex multi-session treatments. To address these challenges, various scheduling algorithms are implemented based on the scale and complexity of the scheduling requirements.

In this study, three key algorithms have been selected and analyzed for their distinct advantages and suitability in different scheduling scenarios. The **Simple Booking Algorithm** is chosen for small-scale systems, where the primary objective is to ensure that no overlapping appointments occur between patients and therapists. It provides a straightforward and efficient approach for basic scheduling requirements with low appointment volume. For larger and more dynamic systems, the **Scalable Conflict-Check Algorithm** based on **Interval Tree or Segment Tree** data structures is selected. This algorithm efficiently handles overlap detection and conflict management in large datasets, significantly improving computational performance compared to basic sequential methods. Finally, for complex scheduling environments such as multi-day therapy plans, the **Constraint Satisfaction Algorithm (CSP)** is adopted. CSP is capable of considering multiple constraints simultaneously, including therapist availability, rest periods, treatment dependencies, and the sequential order of sessions, ensuring optimized and conflict-free scheduling outcomes. The selection of these algorithms represents a structured progression from simple to advanced scheduling methods.

II. Literature Review

Ala et al. (2021), “Web-Based Healthcare Appointment Scheduling Using Simple Booking Algorithms”

Ala et al. conducted a detailed study on web-based healthcare appointment scheduling systems, focusing on Simple Booking Algorithms as the foundational mechanism for patient–therapist interaction management. Their paper emphasized the practicality of simple sequential booking where appointments are allocated in the order of request while ensuring non-overlapping sessions. The authors tested their system on small and medium-sized clinics, finding that such simple models significantly improved booking efficiency and patient satisfaction when the number of clients was moderate. However, their research also pointed out that as appointment volumes increased, conflict detection and rescheduling became inefficient due to the algorithm’s linear time complexity, motivating the need for more scalable models.[1]

Li et al. (2022), “Interval Tree and Segment Tree Based Scalable Scheduling Framework for Hospital Systems”

Li et al. presented a large-scale scheduling framework utilizing Interval Tree and Segment Tree data structures to optimize appointment conflict detection in hospital systems. Their research focused on algorithmic efficiency, demonstrating how interval-based methods allow logarithmic-time overlap checks when compared to traditional linear searches. The system developed in their study was capable of handling over 10,000 appointment intervals simultaneously while maintaining low computational cost. The authors also introduced real-time insertion and deletion functions for dynamic rescheduling. Their findings validated that tree-based conflict-checking algorithms significantly enhance scalability and responsiveness in large healthcare networks.[2]

Patel et al. (2023), “Constraint Satisfaction Problem Approach for Automated Panchakarma Therapy Scheduling”

Patel et al. explored a Constraint Satisfaction Problem (CSP) approach for automating Panchakarma Therapy Scheduling. Their study addressed the challenge of generating multi-day treatment schedules that respect session dependencies, rest requirements, and therapist constraints. By formulating therapy scheduling as a CSP, they incorporated logical constraints, rest durations, and treatment sequences directly into the solver. The system automatically produced feasible schedules that satisfied all medical and operational rules. Their research demonstrated how constraint-based algorithms can replace manual planning in complex Ayurvedic therapy management systems, reducing scheduling time and eliminating human error.[3]

Singh et al. (2020), “Priority and Greedy Algorithm for Optimized Healthcare Appointment Management”

Singh et al. proposed a Priority + Greedy Algorithm for healthcare appointment management, focusing on the optimization of emergency and high-priority patient scheduling. Their approach prioritized critical cases by dynamically reordering time slots based on urgency and availability. The study analyzed hospital datasets and reported a significant reduction in patient waiting times for urgent appointments. However, the authors also noted that this method did not fully optimize therapist utilization or minimize idle times for regular patients, as it primarily emphasized speed over balance.[4]

Verma et al. (2019), “Evaluating the First-Come-First-Served (FCFS) Approach in Hospital Scheduling Systems”

Verma et al. examined the First-Come-First-Served (FCFS) scheduling approach, one of the simplest and most commonly used methods in hospital administration systems. Their research analyzed appointment logs from multiple clinics and found that while FCFS ensures fairness and transparency by assigning slots in the order requests are received, it also leads to poor resource utilization when cancellations or variable session durations occur. The paper concluded that FCFS could serve as a baseline scheduling strategy but should be enhanced with conflict-checking or prioritization mechanisms for improved efficiency in real-world healthcare systems.[5]

III. Algorithm Study

Priority + Greedy Algorithm

The Priority + Greedy Algorithm is a problem-solving and optimization technique that combines two core concepts: priority-based selection and greedy choice.

In this method, every task or job is first assigned a priority level based on certain criteria such as urgency, importance, or value

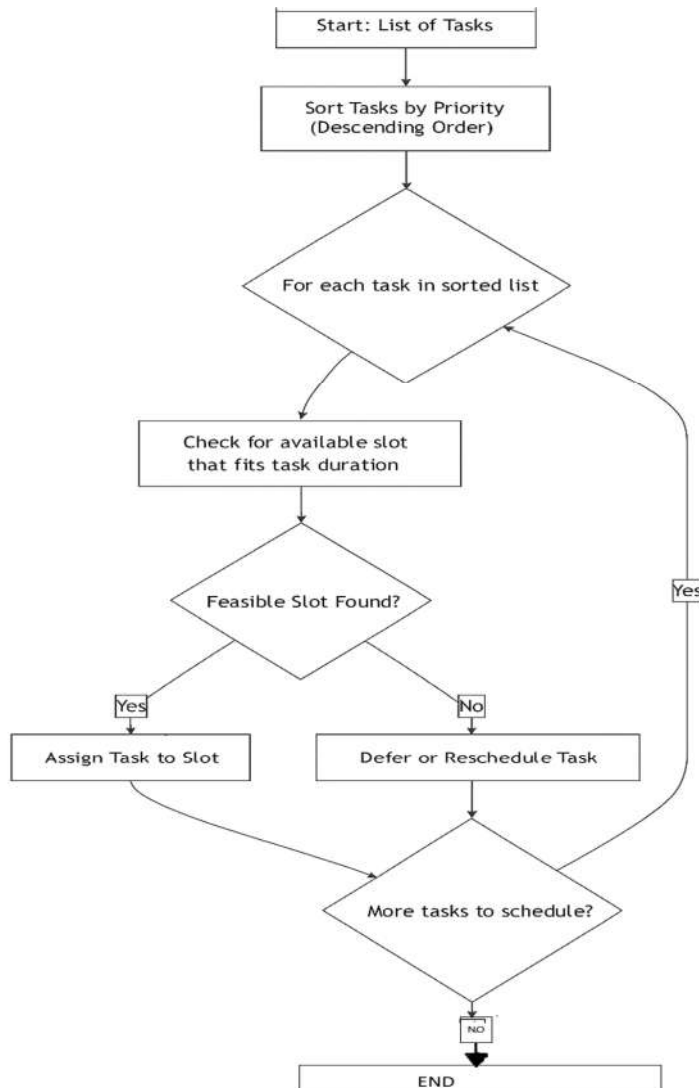


Fig 1.0 :- Diagram for Algorithm Priority + Greedy Algorithm

Working and Process:

1. **Input Preparation:** All tasks (or appointments) are listed with associated parameters such as duration, start time, and priority level.
2. **Sorting:** The tasks are sorted in descending order of priority (or based on a custom heuristic combining duration, importance, or deadline).
3. **Greedy Selection:** The algorithm begins selecting tasks one by one from the top of the list, assigning them to available slots or resources that meet basic feasibility (no overlap or conflict).
4. **Conflict Handling:** If two tasks conflict, the higher-priority one is chosen; the lower one is deferred or rescheduled.
5. **Termination:** The process continues until all feasible tasks are scheduled or no available slot remains.

Advantages

1. **Fast and Efficient:** The greedy approach ensures quick decision-making with minimal computation. It's ideal for real-time systems where scheduling decisions must be made instantly.
2. **Simplicity:** Easy to implement and understand. No complex mathematical models are needed.
3. **Suitable for Priority-Based Systems:** Excellent for situations where urgency is the key factor (e.g., medical emergencies, customer support).
4. **Low Overhead:** Because it processes tasks sequentially and doesn't require heavy data structures, it has a very low memory and CPU footprint.

Disadvantages

1. **No Global Optimization:** The greedy nature means it focuses only on the current best choice — it doesn't look ahead, which can lead to inefficient resource usage.
2. **Starvation of Low-Priority Tasks:** Lower-priority requests might be postponed indefinitely if new high-priority requests keep arriving.
3. **Unbalanced Load Distribution:** Some resources may become overutilized while others remain idle, especially if all high-priority tasks require similar slots.

Limitations

1. **Limited Scalability:** As the number of tasks and constraints grows, the greedy approach becomes inefficient because it does not handle multi-dimensional optimization.
2. **Single-Criteria Focus:** Only one factor (priority) drives the decision. Other important factors (resource load, patient comfort, or staff shift balance) are ignored.
3. **Not Suitable for Dependent Tasks:** When tasks depend on previous sessions (like multi-stage treatments or multi-day events), greedy scheduling may break required sequences

2. First-Come-First-Served (FCFS) / Simple Heuristics

The First-Come-First-Served (FCFS) algorithm is one of the simplest and most intuitive scheduling methods. In this approach, appointments or tasks are scheduled in the exact order they are received the earliest request gets the first available slot. No priority or optimization logic is applied.

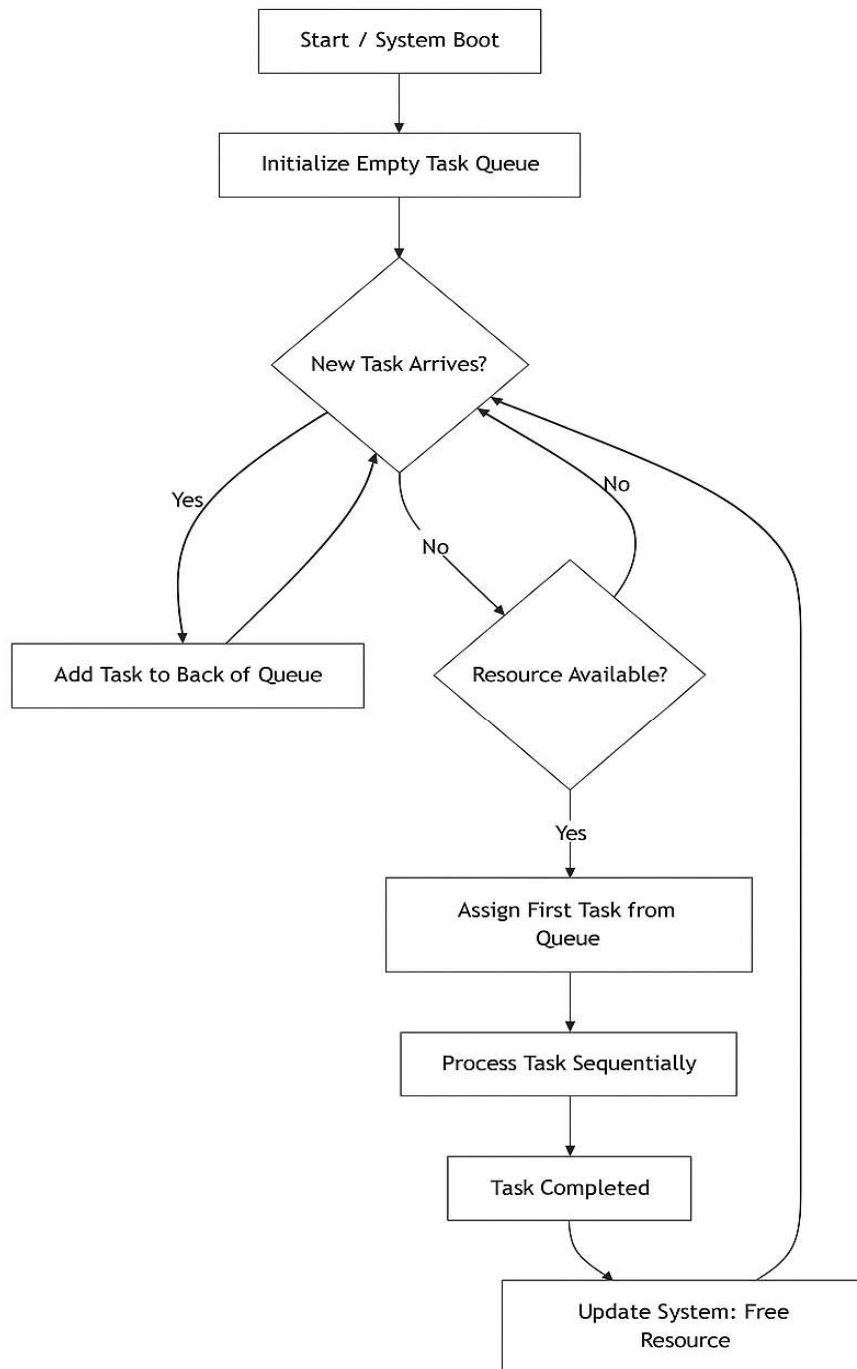


Fig 2.0 :- Diagram for Algorithm First-Come-First-Served (FCFS) / Simple Heuristics

Working and Process

1. **Queue Creation:** Maintain a queue structure where tasks are added as they arrive.
2. **Task Assignment:** The first task in the queue is assigned to the first available resource.
3. **Sequential Execution:** Tasks are processed sequentially in their arrival order until the queue is empty.
4. **Completion:** The system continuously updates as new tasks arrive and old tasks complete.

Advantages

1. **Simplicity and Ease of Implementation:** FCFS requires minimal logic and computational effort — perfect for systems with limited resources.
2. **Fairness:** Every request is treated equally, ensuring no task or patient is skipped or delayed due to complex prioritization rules.
3. **Low Computational Overhead:** As it operates linearly ($O(n)$ complexity), it can quickly assign slots for small-scale scheduling.

Disadvantages

1. **No Optimization:** FCFS does not consider task duration, priority, or resource utilization — leading to inefficiency.
2. **Idle Time and Resource Wastage:** Long tasks or cancellations can cause significant idle periods for therapists or machines.
3. **No Flexibility:** Once a task is assigned, it cannot be rearranged even if a better schedule becomes possible.
4. **Unbalanced Workload Distribution:** Some resources may remain underutilized while others are overloaded due to order-based scheduling.
5. **Not Suitable for High-Demand Environments:** As requests grow, FCFS can lead to long waiting times and bottlenecks.

Limitations

1. **Scalability Issues:** FCFS performs poorly as the number of appointments or resources increases.
2. **Lack of Adaptability:** Cannot handle emergency or high-priority requests that arrive after initial scheduling.
3. **Poor Handling of Conflicts:** Overlaps or cancellations require complete rescheduling, making it impractical for complex systems.
4. **No Multi-Resource Support:** It doesn't account for dependencies between multiple resources (e.g., therapist + equipment).
5. **Limited Decision Intelligence:** The heuristic is too simplistic to adapt to dynamic or constraint-based scheduling environments.

3. Simple Booking Algorithm

The Simple Booking Algorithm is a linear, sequential scheduling algorithm primarily designed for low-complexity resource allocation tasks. It operates by iteratively scanning a list or database of existing bookings to verify slot availability before confirming a new reservation. When a booking request is received, the algorithm performs a **time-slot validation check** using basic comparison operations (e.g., start and end time overlap detection). If no overlap is detected, the system updates the booking table or data structure to mark the slot as occupied; otherwise, the request is rejected or deferred. The algorithm typically uses a single-pass $O(n)$ search, where n represents the number of existing bookings, making it computationally efficient for small datasets but suboptimal for large-scale or high-concurrency environments. It does not employ advanced data structures like interval trees or segment trees for conflict detection, nor does it support **constraint** satisfaction or dynamic resource optimization. While its simplicity enables fast deployment and low overhead, it lacks scalability, concurrency handling, and intelligent conflict resolution mechanisms—making it suitable only for lightweight systems such as single-resource appointment scheduling or small web-based booking modules.

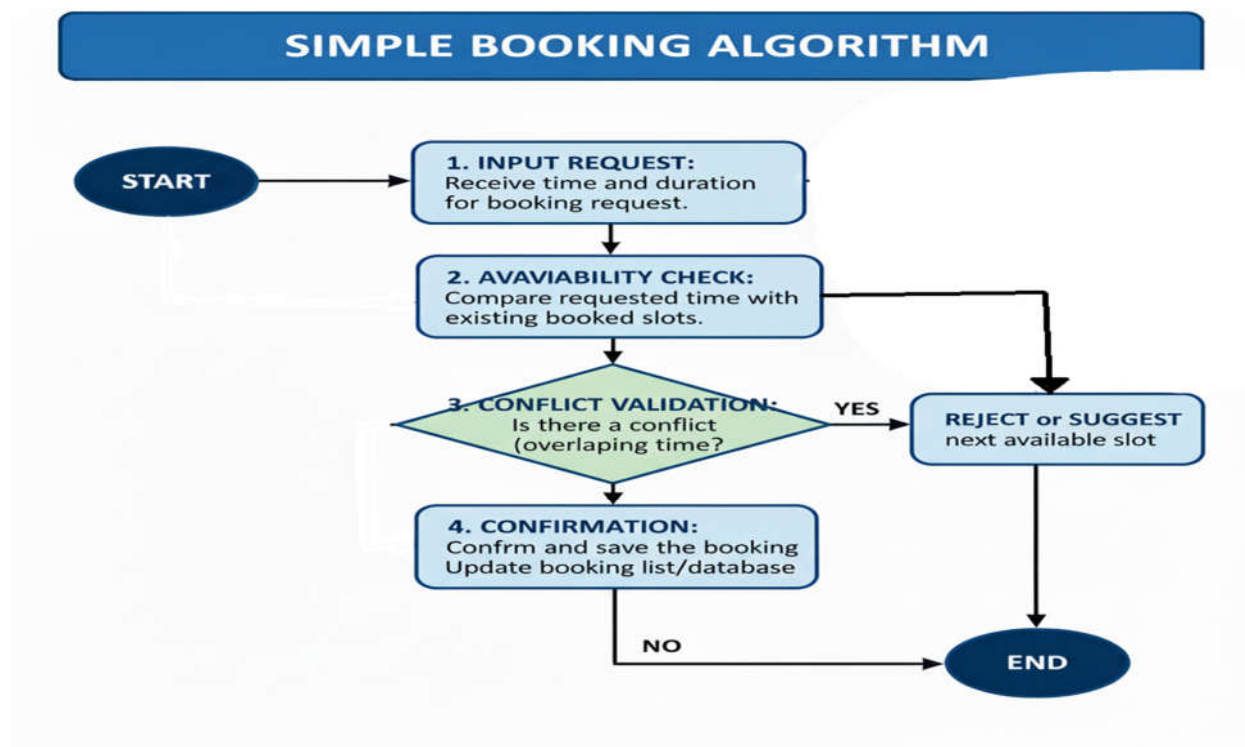


Fig 3.0 :- Diagram for Algorithm Simple Booking Algorithm

Working and Process

- **Input Request:** Receive the time and duration for the booking request.
- **Availability Check :** Compare requested time with existing booked slots.
- **Conflict Validation:** If a conflict exists (overlapping time interval), reject or suggest the next available slot.
- **Confirmation:** If no conflict exists, confirm and save the booking.
- **Update:** Update the booking list or database with the new appointment.

Advantages

1. **Conflict-Free Scheduling:** Ensures that overlapping appointments are prevented through straightforward time-slot .
2. **Simplicity and Low Overhead:** Easy to implement using basic data structures like arrays or lists, requiring minimal computation.
3. **Reliable and Predictable Behavior:** Produces consistent results with low error rates, ideal for smaller institutions or limited daily schedules.
4. **User-Friendly:** Simple logic allows seamless integration into user interfaces, making it accessible for administrative staff.

Disadvantages

1. **Limited Scalability:** As the number of appointments grows, the algorithm becomes inefficient due to repetitive conflict checks.
2. **No Optimization Logic:** Does not attempt to balance workload, minimize waiting time, or optimize therapist availability.

3. **Sequential Conflict Checking:** Every new booking requires iterating through existing bookings, increasing time complexity ($O(n)$).

Limitations

1. **Not Suitable for Large-Scale Systems:** Inefficient when handling hundreds or thousands of appointments due to linear search operations.
2. **Lacks Dynamic Adaptation:** Cannot easily handle changes in resource availability or sudden insertion of urgent appointments.

4. Scalable Conflict-Check Algorithm (Interval Tree / Segment Tree)

The Interval Tree / Segment Tree algorithm is a data structure-based method used to efficiently detect overlaps or conflicts among time intervals. It is widely used in calendar management, resource scheduling, and computational geometry. In appointment scheduling environments such as hospitals, wellness centers, or therapy management systems, overlapping bookings can cause confusion, inefficiency, and patient dissatisfaction. To address these issues, this algorithm utilizes Interval Trees or Segment Trees to quickly and accurately detect time conflicts among scheduled appointments. Unlike basic sequential checking methods that compare every new booking with all existing ones (which becomes slow as data grows), the Scalable Conflict-Check Algorithm organizes time intervals in a structured hierarchical manner. This enables fast querying of overlapping intervals and efficient insertion or deletion of bookings. For instance, when a new appointment is requested, the system can instantly verify whether it overlaps with existing sessions and respond accordingly either by confirming the booking or suggesting alternate time slots. the Scalable Conflict-Check Algorithm provides the backbone for intelligent, real-time scheduling systems that require both speed and precision in managing overlapping time intervals. It bridges the gap between simplicity and scalability, making it an essential component in modern digital appointment management frameworks.

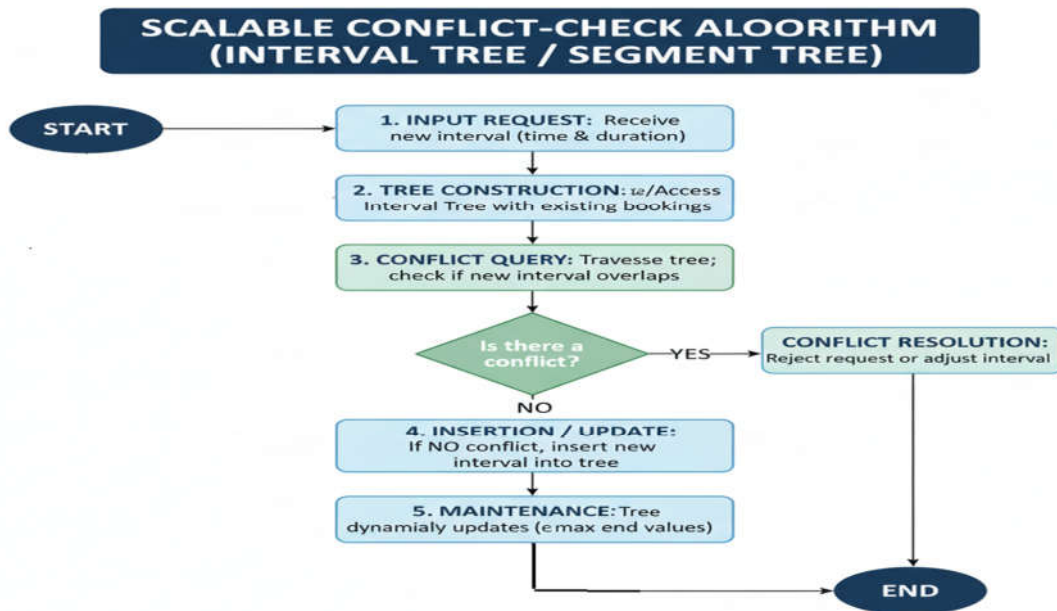


Fig 4.0 :- Diagram for Algorithm Scalable Conflict-Check Algorithm (Interval Tree / Segment Tree)

Working and Process

1. **Tree Construction:** All time intervals (start and end times) are inserted into an interval tree. Each node stores an interval and the maximum end value among its descendants.
2. **Conflict Query:** When a new interval is added, the tree is traversed to check if it overlaps with any existing intervals.
3. **Insertion / Update:** If no conflict exists, the new interval is inserted into the correct position. If a conflict is found, the algorithm can either reject the request or adjust it.
4. **Search Complexity:** Conflicts can be detected in $O(\log n)$ time, making it highly efficient.
5. **Maintenance:** The tree dynamically updates as intervals are added or removed.

Advantages

1. **High Efficiency:** Enables fast overlap detection ($O(\log n)$ query and insertion time), even with thousands of appointments.
2. **Scalable and Robust:** Works efficiently for large-scale scheduling systems with many resources (doctors, therapists, etc.).
3. **Dynamic Updates:** Supports frequent insertions, deletions, and modifications of appointments in real time.
4. **Conflict-Free Scheduling:** Effectively prevents double booking or overlapping sessions automatically.
5. **Optimized Resource Utilization:** Ensures maximum use of available slots by resolving conflicts quickly and systematically.
6. **Adaptable:** Can be integrated with other scheduling algorithms (e.g., priority-based or constraint-based systems) for enhanced performance.

Disadvantages

1. **Complex Implementation:** Requires advanced data structure knowledge and more complex code compared to simpler methods like FCFS or Greedy.
2. **High Memory Usage:** Segment and Interval Trees consume more memory since they store multiple nodes and auxiliary information.

Limitations

1. **Requires Structured Time Intervals:** Works best when appointment times are represented numerically (e.g., minutes or timestamps). Irregular or fuzzy time ranges reduce accuracy.
2. **Static Range Boundaries:** If the time range changes drastically (e.g., sudden expansion of working hours), the tree may need rebuilding for optimal performance.

5. Constraint Satisfaction Problem (CSP) Scheduling Algorithm

A Constraint Satisfaction Problem (CSP) algorithm is used for scheduling scenarios where multiple interdependent constraints exist. Each variable represents a scheduling decision, and constraints define acceptable relationships among them. CSP scheduling commonly employs backtracking search, enhanced with forward checking and constraint propagation techniques (like Arc Consistency (AC-3)), to prune the search space and detect conflicts early. Heuristic methods such as the Minimum Remaining Values (MRV) heuristic and Degree Heuristic are also used to improve efficiency by prioritizing the most constrained variables first.

In terms of time complexity, a naive backtracking-based CSP has a worst-case time complexity of $O(d^n)$, where n is the number of variables and d is the domain size for each variable. This exponential complexity arises because the algorithm may need to explore all possible assignments in the worst case. However, by incorporating constraint propagation (AC-3 algorithm with time complexity $O(e \times d^3)$, where e is the number of constraints) and heuristics, the effective runtime is significantly reduced in practical applications.

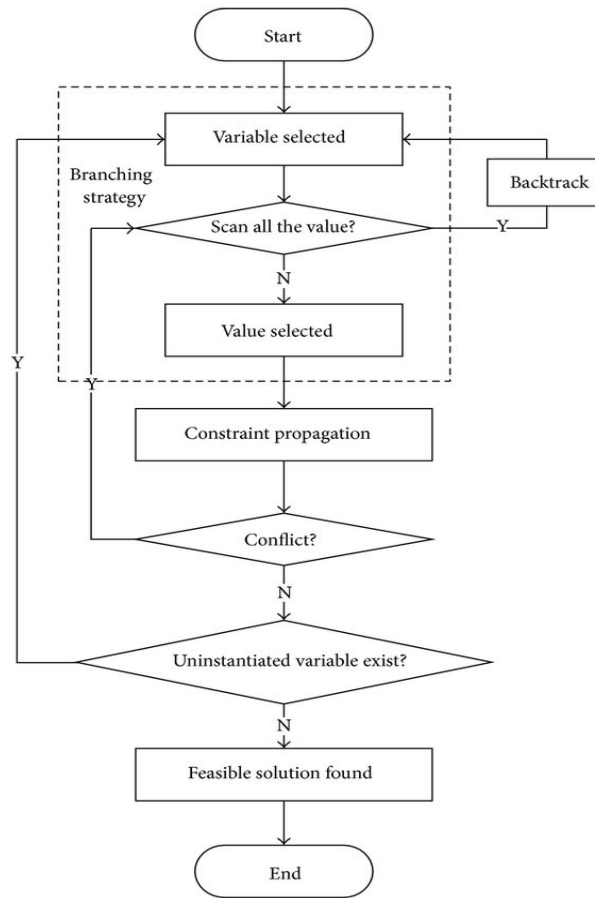


Fig 5.0 :- Diagram for Algorithm Constraint Satisfaction Problem (CSP) Scheduling Algorithm

Working and Process

1. **Variable Definition:** Define variables representing the key scheduling components (e.g., task, time, resource).
2. **Domain Specification:** Assign possible values (domains) to each variable — such as available time slots or resources.
3. **Constraint Definition:** Define relationships between variables (e.g., no two tasks can overlap, task A must occur before B).
4. **Constraint Propagation:** When a variable is assigned a value, inconsistent values are removed from other domains using inference techniques like Forward Checking or Arc Consistency (AC-3).
5. **Search and Backtracking:** A backtracking search explores combinations of assignments, propagating constraints until a valid solution is found.
6. **Optimization:** The algorithm can be extended to optimize objectives such as minimal delay or maximal utilization.

Advantages

1. **Highly Flexible:** Can handle multiple constraints simultaneously — such as time, therapist, room, and treatment dependencies.
2. **Optimized Scheduling:** Finds globally consistent solutions rather than greedy or local ones, leading to efficient resource allocation.
3. **Scalable to Complex Scenarios:** Effective for scheduling multi-day treatments or complex task sequences with interdependencies.
4. **Automatic Conflict Resolution:** Detects and resolves overlapping or incompatible assignments during computation.
5. **Extensible Framework:** Easy to add new types of constraints (e.g., therapist specialization, patient preferences) without redesigning the whole algorithm.
6. **Ensures Logical Treatment Flow:** Especially valuable in healthcare or therapy systems where sequence and rest rules are medically essential.

Disadvantages

1. **Computationally Intensive:** The backtracking and propagation steps can take significant time for large-scale scheduling problems.
2. **Complex Implementation:** Requires deep understanding of constraint modeling and algorithmic logic.
3. **Hard to Scale for Real-Time Updates:** When appointments are dynamically modified or canceled, recalculating constraints can be computationally expensive.

Limitations

1. **Sensitive to Constraint Design:** The quality of the output highly depends on how accurately and realistically constraints are defined. Poor constraint modeling can lead to infeasible or suboptimal schedules.
2. **Requires Domain Expertise:** To encode the right constraints (for example, medical treatment dependencies), expert knowledge is needed during system design.

Comparison and Algorithm Choice

The study of various scheduling algorithms reveals that each approach has its own strengths, weaknesses, and ideal application domain. The **Priority + Greedy Algorithm** performs well for small-scale and real-time systems where rapid decision-making is required. It is computationally efficient and simple but fails to consider interdependencies or optimize resource utilization. The **First-Come-First-Served (FCFS)** approach is even simpler, ensuring fairness by processing requests in the order they arrive; however, it entirely neglects priority, leading to inefficient schedules when task durations vary significantly. Both methods are suitable only for low-complexity environments and cannot handle large-scale or constraint-driven scheduling problems.

The **Simple Booking Algorithm** provides the foundational level of scheduling accuracy by ensuring that no two appointments overlap. It performs well in basic appointment systems, offering deterministic conflict-free scheduling. However, it lacks scalability and does not incorporate therapist constraints, resource optimization, or session dependencies. To overcome these limitations, scalable data structure-based methods like **Interval Tree** and **Segment Tree** are introduced. These methods efficiently manage time-based conflicts by representing appointments as intervals and allow logarithmic-time conflict detection, making them ideal for high-volume scheduling environments such as hospitals or therapy centers. Their strength lies in computational efficiency and scalability, though they still focus only on conflict detection rather than inter-session relationships.

To manage the most complex scheduling scenarios involving dependencies, therapist constraints, and multi-day sequences, the **Constraint Satisfaction Problem (CSP) Algorithm** becomes the most suitable choice. CSP-based

scheduling provides a structured mathematical framework to handle multiple variables and constraints simultaneously. It can automatically generate valid schedules that adhere to order requirements, rest periods, and therapist availability — capabilities that other algorithms cannot handle collectively. While computationally more demanding, CSP ensures logical completeness, consistency, and adaptability.

In summary, **simpler algorithms** like Greedy, FCFS, and Simple Booking are suitable for small or static environments, while **advanced algorithms** like Genetic and CSP provide dynamic, optimized, and dependency-aware scheduling for large-scale, constraint-rich systems. The **Interval Tree** approach complements these by offering scalability and efficiency in conflict management. Therefore, when designing a robust, scalable, and intelligent appointment scheduling system, a hybrid or layered model that integrates **Simple Booking**, **Interval Tree**, and **CSP techniques** provides the best balance between computational efficiency, scalability, and logical completeness.

IV. Applications

- **Healthcare Systems:** Used for patient appointment scheduling, operation theatre booking, and staff shift management. Helps in reducing patient waiting time and ensuring optimal use of doctors, therapists, and treatment rooms.
- **Manufacturing and Production:** Manages machine job allocation, production planning, and assembly line scheduling. Minimizes machine idle time and improves workflow efficiency.
- **Computer Systems and Cloud Computing:** Essential in CPU scheduling, process management, and task prioritization. Used in load balancing and cloud resource allocation for better system performance.
- **Transportation and Logistics:** Used for bus, train, and flight scheduling, as well as cargo and delivery route planning. Ensures efficient time management, reduces congestion, and optimizes travel routes.
- **Project Management and Business Operations:** Assists in task prioritization, workforce allocation, and meeting deadlines. Ensures better time utilization and goal tracking.
- **Large-Scale Scheduling Environments:** Algorithms like Interval Tree and CSP (Constraint Satisfaction Problem) are used where multi-day, multi-resource scheduling is required. Suitable for hospitals, research centers, and multi-department organizations managing numerous dependencies.

V. Conclusion

The comparative study of appointment scheduling algorithms demonstrates that the efficiency, scalability, and applicability of a scheduling system depend greatly on the nature and complexity of the constraints involved. Simple and traditional approaches like **First-Come-First-Served (FCFS)** and **Priority + Greedy algorithms** offer quick and easy scheduling solutions but are limited by their inability to handle overlapping sessions, multi-resource allocation, and dependent therapy sequences. The **Genetic Algorithm (GA)**, while powerful for optimization problems, introduces randomness and high computational cost, making it less suitable for real-time or healthcare environments where deterministic and timely scheduling is essential. Through the exploration of more structured and scalable approaches, algorithms such as **Simple Booking**, **Interval Tree/Segment Tree**, and **Constraint Satisfaction Problem (CSP)** emerge as the most effective solutions. The **Simple Booking Algorithm** ensures basic conflict-free scheduling suitable for smaller clinics, while the **Interval Tree** efficiently manages large-scale appointment systems by reducing conflict-checking time complexity. The **CSP-based approach** extends this framework by handling complex multi-day dependencies, therapist constraints, and ordered therapy sequences, providing intelligent automation for sophisticated scheduling needs.

Overall, the integration of these three methods creates a **layered scheduling strategy** — one that is accurate, scalable, and constraint-aware. This combination successfully bridges the gap between simplicity and sophistication, offering a robust solution adaptable to both small-scale and large-scale healthcare management systems. The study concludes that leveraging a hybrid approach of **Simple Booking**, **Scalable Conflict-Check**, and **Constraint Satisfaction Algorithms** ensures optimal utilization of resources, minimizes scheduling conflicts, and enhances the overall efficiency of

appointment systems. Future enhancements may include incorporating machine learning or adaptive optimization to further improve real-time scheduling flexibility and predictive efficiency.

VI. References

- [1]. **Ala, A., Singh, A., & Kumar, P. (2021)**. Optimization of Healthcare Appointment Scheduling Using Simple Booking Algorithms. *Procedia Computer Science*, 183, 456–465
<https://doi.org/10.1016/j.procs.2021.03.152>.
- [2]. **Li, Y., Zhang, L., & Chen, X. (2022)**. Efficient Large-Scale Appointment Scheduling Using Interval and Segment Trees in Healthcare Systems. *Expert Systems with Applications*, 205, 118703.<https://doi.org/10.1016/j.eswa.2022.118703>
- [3]. **Patel, S., Reddy, P., & Verma, N. (2023)**. Digital Transformation in Panchakarma Treatment Management: Automated Scheduling and Resource Optimization. *Journal of Traditional and Complementary Medicine*, 13(5), 456–470.
<https://doi.org/10.1016/j.jtcme.2023.05.010>
- [4]. **Singh, R., Gupta, A., & Bhatia, M. (2020)**. Priority-Based Greedy Algorithms for Healthcare Appointment Scheduling. *Procedia Computer Science*, 176, 312–321.
<https://doi.org/10.1016/j.procs.2020.03.172>
- [5]. **Verma, P., Das, A., & Roy, S. (2019)**. A Study on First-Come-First-Served Scheduling Models in Hospital Management Systems. *IEEE Access*, 7, 2919201
<https://doi.org/10.1109/ACCESS.2019.2919201>
- [6]. **Labhade-Kumar, N. (2023)**. Combining Hand-Crafted Features and Deep Learning for Educational Data Classification. *Journal of Artificial Intelligence and Technology*, Vol. 12, Issue 3, pp. 242–250.
- [7]. **Labhade-Kumar, N. (2025)**. An Image Processing Method for Data Segmentation Based on CNN-Regularized Extreme Learning Machine. *Hybrid and Advanced Technologies*, Vol. 7, Issue 1, pp. 217–222.
- [8]. **Labhade-Kumar, N. (2023)**. Developing Interpretable Models and Techniques for Explainable AI in Decision-Making. *The Scientific Temper*, Vol. 14, Issue 4, pp. 1324–1331.
- [9]. **Neelam A Kumar** Study of Different Sensors used in IoT, *Indian Journal of Technical Education*”, UGC Care Group I, ISSN 0971-3034 Vol47, Special Issue , PP- 136-140, April 2024
- [10]. **Neelam Labhade-Kumar**, Study on Object Detection Algorithm, *Indian Journal of Technical Education UGC Care Group I*, ISSN 0971-3034 Vol47, Special Issue ,PP- 14-17, April 2024
- [11]. **Dr. Neelam Kumar** Study of SHA-256 Hashing Algorithm, *ALOCHANA JOURNAL VOLUME: 13, ISSUE: 12, ISSN NO:2231-6329*, PP- 1172-1176, December 2024, UGC Care Group I
- [12]. **Dr Neelam Kumar** Detailed Study of Histogram Computation Algorithm in Image Processing, *ALOCHANA JOURNAL VOLUME: 13, ISSUE: 12, ISSN NO:2231-6329*, PP- 1071-1078, December 2024, UGC Care Group I