Online Multi-appointment Scheduling for Outpatient Examinations with Genetic Programming

Huan Zhang [a], Yang Wang [a,*], Jin-Kao Hao [b], Lihui Zeng [c,*], Li Luo [d,*]

[a] *School of Management, Northwestern Polytechnical University, 710072 Xi'an, China*

[b] *LERIA, Université d'Angers, 2 Boulevard Lavoisier, 49045 Angers, France*

[c] *West China Hospital, Sichuan University, 610041 Chengdu, China*

[d] *Business School, Sichuan University, 610065 Chengdu, China*

Computers & Operations Research 194: 107548, 2026.

Abstract

Centralized appointment platforms in hospitals allow patients to submit their appointment requests online and receive real-time confirmations of available date and time slots, which simplifies the appointment process and reduces the administrative burden. However, existing multi-appointment scheduling systems in hospitals typically rely on simple dispatching rules, which are myopic and insufficient to balance two critical patient needs: minimizing the total number of hospital visits and reducing the number of waiting days. To address this issue, we formulate a novel online multi-appointment scheduling problem (OMASP) that explicitly targets both objectives. Due to the complexity of designing effective dispatching rules manually, we propose a genetic programming (GP) approach to automatically evolve high-performing dispatching rules. The proposed GP framework integrates a multi-fidelity simulation evaluator and a feature selection mechanism to enhance both performance and interpretability. Using real data from a large hospital, we design simulation scenarios that reflect both the number of patient examinations and the daily arrival rates of appointment requests. Computational results show that our proposed GP method achieves an objective improvement of 13.88% compared to baseline methods. The performance gains are particularly pronounced in scenarios with a high average number of examinations per patient and elevated daily request volumes. By analyzing the behavior of the evolved rules through explicit simplification and a new visualization method, we provide valuable managerial implications that can inform and enhance hospital appointment scheduling practices.

Keywords: Multi-appointment scheduling; Hyper-heuristic; Dispatching rule; Genetic programming

1 Introduction

In recent years, high-ranking hospitals in China have accelerated the adoption of centralized appointment platforms for medical examinations. These platforms allow patients to submit requests through mobile apps and web portals and can instantly schedule multiple examinations for the same patient, reducing delays in appointment confirmation. By eliminating the need for in-person visits and long waits, centralized appointment platforms help simplify the appointment process and thus improve the patient experience.

However, the algorithms used by these platforms often rely on simplistic rules, such as "earliest-day" and "single-day" strategies. The "earliest-day" rule schedules appointments at the earliest available dates, aiming to shorten waiting days but often resulting in fragmented appointments across multiple days. Conversely, the "single-day" rule consolidates all appointments on the latest common available date, minimizing the number of visits but potentially extending the overall waiting days. These rules limit patients' ability to balance two conflicting objectives: reducing the number of hospital visits and minimizing waiting days. Furthermore, these dispatching rules are inherently myopic—they prioritize immediate scheduling for individual patients without considering subsequent requests. As a result, fragmented early schedules can compromise the flexibility and efficiency of future appointments. An effective scheduling algorithm must therefore strike a balance between immediate needs and system-wide efficiency to improve the overall performance of hospital scheduling operations.

This challenge gives rise to the online multi-appointment scheduling problem (OMASP)—a dynamic and uncertain variant of the appointment scheduling problem that requires real-time decisions from a centralized perspective (Marynissen and Demeulemeester, 2019). Due to the stochastic nature of patient arrivals and varying examination requirements, OMASP involves significant uncertainty. In such contexts, dispatching rules provide a promising approach, enabling real-time scheduling decisions that adapt to system status and patient demands. This study aims to develop novel, high-performing dispatching rules and embed them within a heuristic framework to address real-time scheduling in large-scale, practical healthcare settings.

However, manually designing dispatching rules is labor-intensive, heavily reliant on domain expertise, and often requires extensive trial and error

* Corresponding authors.
Email addresses: sparkle.wy@gmail.com (Yang Wang), arronzeng@163.com (Lihui Zeng), luolicc@suc.edu.cn (Li Luo).

(Guo et al., 2022). To overcome these limitations, we adopt a hyper-heuristic approach that automates rule design (Burke et al., 2013, 2009). Specifically, we employ genetic programming (GP), a widely used evolutionary computation technique that has shown strong performance in online scheduling problems (Braune et al., 2022; Burke et al., 2007; Fan et al., 2021). As originally introduced by Koza (1992), GP evolves programs by representing them as hierarchical combinations of functions and terminals. Through iterative processes driven by selection, crossover, and mutation, GP explores a vast search space and continuously improves candidate solutions based on fitness. In our study, this framework is tailored to evolve interpretable and high-performing dispatching rules for OMASP, offering a scalable and effective solution for real-time centralized appointment scheduling.

Motivated by real-world challenges, this study develops a GP algorithm that evolves dispatching rules outperforming those commonly used heuristics, ultimately offering a practical decision-making tool for centralized appointment platforms. The main contributions of this paper are as follows:

(1) We formally define OMASP based on practices observed at our partner hospital, aiming to jointly minimize patients' waiting days and the number of hospital visits. We also propose a heuristic algorithm specifically designed for OMASP, which integrates dispatching rules to enable real-time decision-making for problems of practical scale.

(2) We introduce a new GP-based approach to automatically generate dispatching rules. As far as we know, this is among the first applications of GP to multi-appointment scheduling. Our GP is enhanced with a multi-fidelity simulation evaluator (Nguyen et al., 2019) that filters out poor-performing rules during evolution and an offline feature selection procedure (Mei et al., 2016) that eliminates redundant terminals to improve interpretability.

(3) We develop a discrete event simulation model using real data from a partner hospital in China for evaluation. Extensive experiments are conducted across varying numbers of patient examinations and daily arrival rates. Results confirm the effectiveness of the GP algorithm in solving OMASP, achieving an average performance improvement of over 13% compared to baseline methods.

(4) We extract concise and interpretable rules with strong performance through explicit simplification of the final GP outputs. To gain further insights, we propose a new visualization method to analyze rule behavior and derive managerial implications. Our findings reveal that daily patient arrival rates influence optimal appointment scheduling strategies. When

capacity permits, hospitals should prioritize patients requiring multiple examinations to reduce total visits. In contrast, during peak demand periods, minimizing patient queues should be the primary focus.

The remainder of the paper is organized as follows. Section 2 reviews related work and provides background on outpatient appointment scheduling. Section 3 introduces and formulates OMASP. Section 4 presents the proposed GP approach. Section 5 reports experimental results, while Section 6 presents further analysis. Finally, Section 7 concludes the paper and outlines future research directions.

2 Related work

This study focuses on the design of dispatching rules for outpatient appointments using genetic programming. Accordingly, this section reviews the relevant literature from two perspectives: outpatient appointment scheduling and genetic programming.

2.1 Outpatient appointment scheduling

The outpatient appointment scheduling problem has attracted continuous interest since it was introduced (Bailey, 1952). The primary focus is on developing an efficient appointment system to improve outpatient operations and reduce waiting times. According to Ahmadi-Javid et al. (2017), the decision-making process of an outpatient appointment system can be classified into three levels: strategic, tactical and operational. Among these, most researchers focus on the operational level, investigating how to determine the appointment time slot, date and equipment for individual patients. The relevant methods fall into two categories: (1) the rule-based approach (RBA) provides a simple way to handle uncertain situations, such as patient no-shows or unpunctuality (Cayirli and Yang, 2014; Cayirli et al., 2012); (2) the optimization-based approach (OBA) aims to establishing mathematical models and designing optimization algorithms to obtain high-quality solutions (Conforti et al., 2008, 2010).

In recent years, the multi-appointment scheduling problem (MASP) has gained increasing attention (Leeftink et al., 2020; Marynissen and Demeulemeester, 2019). In MASP, patients must secure appointments for a specific set of hospital resources. Based on the response time to appointment requests, MASP scheduling strategies can be classified into two types: (1) the online strategy (Hsieh, 2017; Kortbeek et al., 2017) requires immediate

response and decision, with the scheduling of patient appointments done sequentially based on the order of their arrival time; (2) the offline strategy (Chien et al., 2008; Saadani et al., 2014) waits and collects a certain number of appointment requests before applying algorithms to select patients from this list. Online systems are more common in practice, while offline methods receive more attention in the literature as an offline system is easier to model (Ahmadi-Javid et al., 2017). Existing research on MASP typically emphasizes patient satisfaction by optimizing appointment schedules to minimize access time, waiting time and lateness of patients. However, very few studies consider the number of visits as we do.

Compared to the literature on offline MASP, there are fewer studies focusing on online MASP (Marynissen and Demeulemeester, 2019). Azadeh et al. (2015) studied a scheduling problem in a pathology laboratory, formulating it as a mixed-integer programming model of a semi-online hybrid-shop scheduling problem and developing a genetic algorithm to solve it. Braaksma et al. (2014) presented an integer linear programming formulation for planning treatments in rehabilitation outpatient clinics, ensuring continuity of care while optimizing seven performance indicators. Luscombe and Kozan (2016) proposed a dynamic scheduling framework to provide real-time support for the emergency department, integrating parallel machine and flexible job shop theories and designing a tabu search heuristic algorithm to solve the problem. Pérez et al. (2013) studied appointment scheduling in nuclear medicine with strict time window constraints and derived a stochastic online scheduling algorithm to optimize patient and resource scheduling. Castro and Petrovic (2012) introduced a real-world radiotherapy scheduling problem, modeling it as a multi-objective optimization problem and applying mathematical programming and heuristics to generate scheduling solutions. In summary, the dominant approach in the literature on online MASP is to adopt the OBA, which focuses on deriving optimal solutions to maximize efficiency and meet operational constraints. However, our partner hospital requires the appointment system to provide real-time responses, which imposes strict limitations on computation time for these methods. On the other hand, the application of RBA is relatively rare, yet these approaches offer potential advantages in terms of simplicity and adaptability. Thus, this study explored the automated design of dispatching rules for OMASP using a GP approach to address this gap.

2.2 Genetic programming

GP has been successfully applied to online bin packing problems (Burke et al., 2007), boolean satisfiability problems (Fukunaga, 2008), travelling salesman problems (Keller and Poli, 2007), timetabling and scheduling (Bader-El-Den

et al., 2009) and emergency medical dispatch problems (MacLachlan et al., 2023). In particular, GP has been widely applied to production scheduling, as it is especially well-suited for encoding scheduling heuristics (Jakobović et al., 2007). The evolution of dispatching rules is the most common application of GP in this domain (Burke et al., 2013). It has been proven that GP can combine and reorganize functions and terminals to generate dispatching rules that outperform those created by humans, which encouraged us to apply GP in OMASP.

The traditional GP paradigm proposed by Koza (1992) demonstrates a powerful generality. However, it faces two major challenges: (1) the high computational cost of fitness evaluation; (2) the vast search space resulting from the combinations of terminals and functions. One stream in the literature discusses how to improve the efficiency of fitness evaluation with surrogate models. Hildebrandt and Branke (2015) proposed a phenotypic characterization based surrogate to preselect top individuals from an intermediate population. Nguyen et al. (2016) further proposed a simplified model-based surrogate, which is simpler to implement and requires no additional training compared to the former. Recent literature has designed multi-fidelity evaluators based on surrogate models, achieving better performance (Guo et al., 2025; Nguyen et al., 2019; Zhang et al., 2021a). These studies employ different evaluation methods throughout the GP run, depending on fidelity requirements. Another stream in the literature studies how to obtain a promising terminal subset with feature selection techniques. Mei et al. (2016) proposed a method that determines the importance of each terminal based on a set of good rules for feature selection. Furthermore, they proposed a Niching-based search framework for extracting a diverse set of rules (Mei et al., 2017). A recent study has also explored feature selection techniques in the context of dynamic flexible job shop scheduling (Zhang et al., 2021b). In summary, the application of both techniques has improved the search efficiency of GP.

While most studies focus on improving the efficiency of GP, fully understanding the decision-making process of rules remains a challenge (Zhang et al., 2024). Explicit simplification is a post-processing step aimed at removing redundant components from GP individuals (Javed et al., 2022; Zhang and Wong, 2008). However, even after simplification, the resulting rules often remain challenging to interpret. To address this, Nguyen et al. (2016) employed parallel coordinate plots to visualize dispatching rules and provide better insights into their behaviors. Parallel coordinate plots effectively display relationships among multiple terminals. However, excessive lines, intersections, and overlaps in the visualization may potentially hide critical patterns, such as the relationship between each terminal and rule decisions (Heinrich and Weiskopf, 2013). To overcome

these limitations, this study proposes a new visualization method that employs a color-coded representation to reveal the relationship.

3 Problem description and formulation

OMASP can be defined as follows. Consider an outpatient system consisting of n medical departments, indexed by $K = \{1, \ldots, n\}$, where each department k contains m_k homogeneous examination equipment, represented by $M_k = \{1, \ldots, m_k\}$. The system operates over an infinite planning horizon D, with time indexed by $d \in Z^+$. On any given day, each piece of equipment can be divided into multiple time slots for scheduling appointments. Let $T = \{1, \ldots, t\}$ denote the set of available time slots, where larger values of t correspond to later times of the day. A "resource"refers to the basic unit for scheduling patient appointments. The set I represents the collection of all resources in the outpatient system, where each resource $i \in I$ is defined by a tuple $i_{k,m,d,t} = (k_i, m_i, d_i, t_i)$. The components k_i, m_i, d_i, and t_i represent the department, equipment, date, and time slot associated with i, respectively. Let c_i represent the capacity of resource i, which will be consumed by the workload of examinations.

Let P denote the set of patients, whose appointment requests arrive dynamically at the centralized appointment platform over time. For each patient p, let s_p represent the submission date of requests, and E_p denote the set of examinations required by patient p. Each examination $e \in E_p$ must be assigned exactly one resource, which corresponds to a specific department, equipment, date, and time slot. We define $\bar{k}_e$ as the department associated with examination e. Because the actual examination durations are unknown, the hospital can only provide estimates based on previous experience. Let w_e represent the workload of examination e. The workload is measured as a multiple of the minimum work unit varying by departments. Such units often range from 5 to 20 minutes in practice. The total workload of examinations assigned to any resource cannot exceed its capacity. OMASP requires assigning resources to examinations for every patient $p \in P$, with the goal of maximizing overall patient satisfaction, which is measured based on two factors, the waiting days and the number of hospital visits.

To provide a clear description of the problem, we present a MILP model to formulate the offline version of MASP. We first introduce two binary variables to represent the assignment of appointments:

$$x_{p,e,i} = \begin{cases} 1, \text{if examination } e \text{ of patient } p \text{ is assigned to resource } i; \\ 0, \text{otherwise} \end{cases} \tag{1}$$

$$y_{p,d} = \begin{cases} 1, \text{if patient } p \text{ needs to visit hospital on date } d; \\ 0, \text{otherwise} \end{cases} \tag{2}$$

Such assignment decisions must satisfy the following constraints:

- Each examination must be assigned to exactly one resource, ensuring it is scheduled only once, as shown in Constraint (3). Let $I_{p,e}$ denote the set of feasible resources for examination e of patient p, defined as $I_{p,e} = \{i \in I \mid d_i > s_p,\, k_i = \bar{k}_e\}$. The resource in this set must belong to the same department as required by the examination and the date of them must be strictly later than the patient's submission date.

$$\sum_{i \in I_{p,e}} x_{p,e,i} = 1, \quad \forall p \in P, \forall e \in E_p. \tag{3}$$

- Let $I_{d,t}$ denote the set of resources available on date d and time slot t. For any two examinations in E_p, they cannot be assigned to the resource with the same date and time slot, as shown in Constraint (4).

$$\sum_{e \in E_p} \sum_{i \in I_{d,t}} x_{p,e,i} \leq 1 \quad \forall p \in P, \forall d \in Z^+, \forall t \in T \tag{4}$$

- The total workload of all examinations assigned to resource i must not exceed its capacity c_i, as shown in Constraint (5).

$$\sum_{p \in P} \sum_{e \in E_p} x_{p,e,i} w_e \leq c_i \quad \forall i \in I \tag{5}$$

- Let I_d denote the set of resources assigned on date d. Each patient needs to visit the hospital at most once per day, regardless of how many examinations are scheduled on that day. To capture this requirement, we introduce a binary variable $y_{p,d}$, as shown in Constraint (6).

$$\sum_{e \in E_p} \sum_{i \in I_d} x_{p,e,i} \leq |E_p|\, y_{p,d}, \quad \forall p \in P, \forall d \in Z^+ \tag{6}$$

The primary objective of offline MASP is to improve patient experience. In this model, the objective function $f(x, y)$ aggregates two components: $f_1(x)$ representing the total waiting time in days and $f_2(y)$ representing the total

number of hospital visits. The objective function is defined as:

$$f(x, y) = \frac{\alpha_1 f_1(x) + \alpha_2 f_2(y)}{|P|} \tag{7}$$

where $|P|$ denotes the total number of patients, α_1 and α_2 are weight coefficients reflecting the relative importance of waiting time and visit frequency. The components $f_1(x)$ and $f_2(y)$ are given by:

$$f_1(x) = \sum_{p \in P} \sum_{e \in E_p} \sum_{i \in I} x_{p,e,i}(d_i - s_p) \tag{8}$$

$$f_2(y) = \sum_{p \in P} \sum_{d \in Z^+} y_{p,d} \tag{9}$$

In our experiments, we set $\alpha_1 = 1$ and $\alpha_2 = 2.5$, following guidance from domain experts to reflect the relative importance of minimizing the number of visits over waiting time. The objective is to identify an appointment schedule that minimizes the function $f(x, y)$, subject to the constraints defined in Equations (3) through (6).Thus, we can formulate offline MASP as follows:

$$\begin{aligned} \min \quad & f(x, y) \\ \text{s.t.} \quad & (3) - (6) \end{aligned}$$

However, in practice, patient requests are not known in advance. Instead, they arrive dynamically at the centralized appointment platform and must be scheduled immediately upon arrival. OMASP adopts the same objective function and constraints as its offline version, aiming to optimize the overall patient experience without the precise information about future arrivals and their examination demands. Stochastic programming are not suitable for this setting, as they often involve solving multiple scenarios, which is computationally prohibitive for large-scale hospital scheduling problems that require responses within seconds. Therefore, we focus on heuristic methods that can deliver high-quality solutions in such limited computational time.

4 Online solution method

In this section, we first describe an online heuristic algorithm designed for centralized appointment platforms, which relies on dispatching rules. We then introduce the developed GP algorithm, which generates high-performing dispatching rules for the heuristic algorithm.

4.1 Heuristic algorithm

Centralized appointment platforms allow patients to conveniently submit their appointment requests through mobile apps or web portals, and must provide real-time appointment plan to their needs. A dispatching rule is a function that computes a priority value for a resource from its attributes and environmental information (Haupt, 1989). Dispatching rules can ideally rank and select available resource throughout the scheduling period for each appointment, and be embedded in a heuristic algorithm framework to generate an appointment plan in milliseconds.

Each time a request arrives, the heuristic algorithm outlined in Algorithm 1 is executed. Since this problem considers resources over an infinite planning horizon, the algorithm narrows the date range of resources to reduce the computational cost of scanning them. The algorithm first determines the earliest available date d_e for each examination $e \in E_p$, defined as the earliest feasible starting date on any resource that can accommodate the examination without exceeding its capacity. It then calculates the minimum value $d_{\min}$ and the maximum value $d_{\max}$ of d_e over all $e \in E_p$ for the date range of the candidate resource in I^{cnd} (lines 4-9). This is because resource before $d_{\min}$ are considered unavailable, while resource after $d_{\max}$ are effectively replaced by the resource on $d_{\max}$. Dispatching rule r is then applied to calculate priority values for each resource in I^{cnd} (lines 10-12). In cases where priority values are equal, ties are broken based on dates and time slots of resource, with preference given to earlier dates and time slots. For each examination $e \in E_p$, a resource is chosen that maximizes the total priority, while also ensuring that all problem constraints are satisfied (lines 13-17). Specifically, the algorithm first enumerates all examination sequences for the patient. For each sequence, it assigns the available resource with the highest priority to each examination in turn, generating a candidate appointment plan. This enumeration step for patient p has a time complexity of $O(|E_p|!)$. In our real-world dataset, the maximum number of examinations per patient is four, which results in at most 24 sequences per patient, making the computational cost negligible. Finally, all candidate plans are compared, and the one with the highest total priority is selected. This algorithm can implement various appointment strategies by applying different dispatching rules.

4.2 Overall GP framework

GP evolves individuals that represent mathematical expressions of dispatching rules. In our framework, we adopt the widely established

Algorithm 1 Heuristic algorithm based on dispatching rules

Input: Dispatching rule r, patient p and set of resource I
Output: Appointment plan for patient p
Let E_p denote the set of examinations of p
for $e \in E_p$ **do**
    Determine the earliest available date d_e for examination e
end for
$d_{\min} \leftarrow \min\{d_e \mid e \in E_p\}$
$d_{\max} \leftarrow \max\{d_e \mid e \in E_p\}$
$I^{cnd} \leftarrow \{i \in I \mid d_{\min} \leq d_i \leq d_{\max}\}$
for $e \in E_p$ **do**
    Apply rule r to calculate priority values of each resource $i \in I^{cnd}$
end for
Enumerate all possible sequences of examinations in E_p
for each sequence **do**
    Assign to e the available resource $i \in I^{cnd}$ with the highest priority
end for
Select the candidate appointment plan with the maximum total priority while satisfying all problem constraints
return Appointment plan for patient p

tree-based representation for individuals, as introduced by Koza (1992). In such a tree structure, the tree nodes represent functions, which are a series of arithmetic and logical operators, while the leaf nodes represent terminals, which are variables and parameters from the problem domain. Crucially, rather than applying GP in a generic fashion, we customize both the terminal and function sets to align with the specific characteristics and constraints of OMASP. This ensures that the evolved rules are semantically meaningful within the problem context.

Table 1 presents the terminal and function sets used to build dispatching rules. We initially selected 15 terminals from the problem domain and random integer values to support rule generation. These terminals primarily represent attributes related to patients, examinations, and departments. Specifically, WL is an attribute of the examination; TWL and NE are related to patient attributes; NM corresponds to departmental information; D, T, C, and RC are associated with resource characteristics. We also introduce three key parameters from our heuristic algorithm—EAD, $EEAD$, and $LEAD$—which define the range of candidate resources. Furthermore, we integrates factors commonly considered by human decision-makers during examination scheduling, such as AWL, RCE, NWQ, and WWQ, which account for workload balancing, resource efficiency, and queue management. All terminals are organized based on the entity they describe, as shown in Table 1. In addition, we selected seven commonly used functions: $+, -, \times, /, \max, \min$, and the conditional logic operator "if-then-else," which

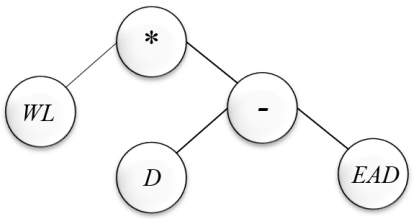


Fig. 1. Example of an expression tree

returns different values based on a specified condition. A protected division function is employed to avoid undefined results, returning 1 when the divisor is zero. These functions are widely used in the GP literature (Hildebrandt et al., 2010; MacLachlan et al., 2023; Nguyen et al., 2016). Figure 1 illustrates a simple example of an expression tree, which includes 2 functions: $+$ and $\times$, and 3 terminals: WL, D, and EAD. Consequently, the expression represented by this tree structure is $WL \times (D - EAD)$.

Table 1
List of terminals and functions

Category	Notation	Description
Examinations	WL	Workload of examination
	EAD	Earliest available date of examination, i.e., d_e in Algorithm 1
Patient	TWL	Total workloads of patient
	NE	Number of examination of patient
	EEAD	Earliest EAD of patient, i.e., $d_{\min}$ in Algorithm 1
	LEAD	Latest EAD of patient, i.e., $d_{\max}$ in Algorithm 1
Department	NM	Number of equipment in department
	AWL	Average workload of examinations in department
Resource	D	Date of resource
	T	Time slot of resource
	C	Capacity of resource
	RC	Remaining capacity of resource
Equipment	RCE	Remaining capacity of equipment on the same day
	NWQ	Number of patients waiting in the queue for equipment
	WWQ	Total workloads of patients waiting in the queue for equipment
Other	x	Random integer between 0 and 9
Arithmetic	$+$	Add
	$-$	Subtract
	$\times$	Multiply
	$\div$	Protected div
Mathematical	max	Maximize
	min	Minimize
Logical	if-then-else	Conditional logic operator

The pseudocode of our GP framework is shown in Algorithm 2. It first initializes the population of rules Pop and the global best rule $r*$ (lines 3-4). For each generation, the entire population is evaluated (lines 7-9). Based on the fitness obtained, the best rule of the current generation r_{gen} is further evaluated to decide whether it should replace the global best rule $r*$ (lines 10-15). Genetic operators are applied to generate an intermediate population Pop_{imd}, which is significantly larger than Pop (line 16). Next, each individual in Pop_{imd} and Pop is estimated to select the most promising rules to the next generation (lines 16-20). GP evolves its population iteratively and returns the best rule once the maximum number of generations is reached. (lines 21-23). In each step, three evaluation methods are employed: lazy evaluations, full evaluations, and surrogate evaluations. The specific method used depends on the accuracy requirements, as detailed in Section 4.2.2.

In the following section, we will introduce the initialization method, the genetic operators, the fitness evaluator, and provide a detailed description of the offline feature selection.

Algorithm 2 Genetic programming

1: **Input**: Maximum generations and population size
2: **Output**: Global best rule $r*$
3: $Pop = \{r_1, r_2, \ldots\} \leftarrow Initialization()$
4: Set $r* \leftarrow null$ and its fitness $fit_{r*} \leftarrow -\infty$
5: $gen \leftarrow 0$
6: **while** $gen < maxGen$ **do**
7: **for** $r \in Pop$ **do**
8: $fit_r \leftarrow LazyEvaluate(r)$
9: **end for**
10: $r_{\text{gen}} \leftarrow$ find the rule with the highest fitness in Pop
11: $fit_{r_{\text{gen}}} \leftarrow FullEvaluate(r_{\text{gen}})$
12: **if** $fit_{r_{\text{gen}}} > fit_{r*}$ **then**
13: $r* \leftarrow r_{\text{gen}}$
14: $fit_{r*} \leftarrow fit_{r_{\text{gen}}}$
15: **end if**
16: $Pop_{\text{imd}} \leftarrow GeneticOperator(Pop)$
17: **for** $r \in Pop_{\text{imd}}$ **do**
18: $fit_r \leftarrow SurrogateEvaluate(r)$
19: **end for**
20: Replace Pop with the top part of rules in Pop_{imd} and Pop
21: $gen \leftarrow gen + 1$
22: **end while**
23: **return** $r*$

4.2.1 Initialization and genetic operators

In the initialization phase, we used the ramped-half-and-half method (Koza, 1992) to randomly create expression trees for dispatching rules. By combining two generative methods: grow and full, with each generating 50% of the individuals in the population, diversity in tree shape and size is ensured. The grow method creates trees that are variably shaped by adding functions and terminals randomly to its root. The full method creates trees ensuring a specified depth by randomly choosing functions instead of terminals as long as the maximum depth is not reached. The depth parameter of the initial population ranged from 2 to 6, following Koza's classic configuration.

For the given representation, we rely on the standard operators: the subtree crossover and the subtree mutation (Koza, 1992). The subtree crossover creates new individuals by randomly exchanging subtrees between two selected parents, while the subtree mutation is performed by replacing a randomly selected subtree of an individual with a newly randomly generated subtree. Both operators selected parents with a tournament selection. Each time an individual is selected, there is a 90% probability of using crossover and a 10% probability of using mutation. The tournament selection size follows Koza's standard value of 7.

4.2.2 Fitness evaluator

We evaluate the performance of dispatching rules through an event-driven hospital simulator, which modeled the real-world operation of a centralized appointment platform. Each event represents the arrival of a patient appointment request. Algorithm 3 illustrates the individual evaluation process using discrete event simulation. Dispatching rule r is evaluated over a set of replications Ω. For each replication $\omega \in \Omega$, initialization is first performed to set up the examination resources. Then the simulation time is advanced event by event until the maximum simulation time is reached (lines 3-7). During each step, the request of a randomly generated patient p is processed with the aforementioned heuristic algorithm based on rule r, and the set of resource I is updated accordingly (lines 8-10). Its final output is the average objective value f_r across all replications, providing a measure of the performance of r.

It is well known that genetic operators in GP have a high likelihood of generating poorly performing dispatching rules. Evaluating such rules, especially when the evaluation process is computationally expensive, is wasteful (Nguyen et al., 2019). A large number of simulation replications are required to obtain reliable fitness values. Thus, we adopted a multi-fidelity

Algorithm 3 Individual evaluation with discrete event simulation

Input: Dispatching rule r and a set of replications Ω
Output: objective value f_r
for *each replication* $\omega \in \Omega$ **do**
  $I \leftarrow InitializeSimulation()$
  $time \leftarrow 0$
  while $time < maxTime$ **do**
    Advance *time* to the next event
    $p \leftarrow RandomPatient()$
    Process appointment request for p by using Algorithm 1
    Consume the capacity of resources in I
  end while
  Calculate objective value f_ω in current replication using Equation 7
end for
$f_r \leftarrow \sum_{\omega \in \Omega} f_\omega / |\Omega|$
return f_r

simulation evaluator introduced by Nguyen et al. (2019), which provides three methods to estimate the fitness of dispatching rules:

(1) Lazy evaluations are applied to evaluate the fitness of the entire population during each generation, as outlined in Algorithm 2. This strategy uses only one replication within each generation and a new replication in every generation to evaluate the fitness of individuals, which has been shown to be effective in improving the performance of evolved rules (Hildebrandt et al., 2010).

(2) Full evaluations are applied to identify the most potential rules by evaluating the best rule in the current generation. This strategy evaluates the true fitness of dispatching rules by measuring their average performance over 100 simulation replications.

(3) Surrogate evaluations are applied to preselect individuals from the intermediate population, enhancing the evolutionary process (Hildebrandt and Branke, 2015). In our approach, we adopted a simplified model-based surrogate proposed by Nguyen et al. (2016). Specifically, the number of equipment in each department and the daily arrival rates of patient requests in the simulator are both reduced to 20% of their original values, while keeping the simulation duration unchanged. The goal is to reduce simulation costs while maintaining an acceptable level of accuracy.

In summary, using three evaluation methods, we enhanced the effectiveness of GP, ensuring a balanced trade-off between computational cost and accuracy. Moreover, updating the global best rule involves comparing fitness between two rules from populations in different generations. However, since GP training uses a different set of simulation replications in each generation to improve the

generalization of the evolved rules, this comparison may not be accurate. To address this, we have designed a specialized fitness function, where the fitness of an individual is defined as the gap between the rule being evaluated and the benchmark rule. We have chosen the "earliest-day" rule as the benchmark rule. Recall that f_r denote the objective value of rule r. Let f denote the objective value of the benchmark rule under the same set of simulation replications. The fitness function is as follows:

$$fit_r = (f - f_r)/f \times 100 \tag{10}$$

4.2.3 Feature selection

Feature selection is a machine learning technique, aiming to select a small but complementary subset of features (Xue et al., 2015). In the context of GP, features refer mainly to the terminals which form the dispatching rules. It has been proven that including irrelevant features may affect the quality of rules evolved by GP (Hildebrandt et al., 2010). In this paper, feature selection serves as a preprocessing phase for GP. Its input is a set of 30 high-performing rules, each generated from an independent run of GP with an unselected terminal set. From this dataset, we assess the importance of each terminal using a contribution-based approach introduced by Mei et al. (2017), which is detailed in Algorithm 4.

This approach assumes that more important terminals tend to make greater contribution, and this contribution can be estimated by its impact on a set of high-performing rules. Since the OMASP studied in this paper is a minimization problem, rules with lower objective values should be given higher weights (lines 3-7). For a given terminal t, its contribution to a rule r is measured by replacing all occurrences of t with constant value 1 in the expression of r, and observing the resulting change in the objective function (lines 9-11). Full evaluations were used to ensure the reliability of this estimation. The final output is obtained by summing all weighted contributions values (lines 14-15). If the contribution value of a terminal is below a small threshold, it will be considered an irrelevant feature and removed from the terminal set. The remaining selected terminals will be used as parameters in subsequent GP runs.

5 Experiment Results

This section presents the application of our GP-based approach to various simulation scenarios generated using real-world data, referred to as surrogate-assisted genetic programming with feature selection (SGP-FS).

Algorithm 4 Terminal contribution calculation

Input: Rule set R and terminal t
Output: Contribution value c
for $r \in R$ **do**
    $w_r \leftarrow 1/(1 + f_r)$
end for
$w_{\min} = min\{w_r | r \in R\}$
$w_{\max} = max\{w_r | r \in R\}$
for $r \in R$ **do**
    $r' \leftarrow$ Replace terminal t in r with 1
    $f_{r'} \leftarrow FullEvaluate(r')$
    $c_r \leftarrow (f_{r'} - f_r)/f_r$
    $w'_r \leftarrow (w_r - w_{\min})/(w_{\max} - w_{\min})$
end for
$c \leftarrow \sum_{r \in R} w'_r c_r$
return c

5.1 Simulation scenarios

Using real data from a partner hospital, we developed a discrete event simulation model that replicates the actual operational environment. The configurations for departments, equipment, time slots, resource capacities, and examination workloads are all based on the hospital's actual settings. To simulate patient appointment requests, we randomly generate each request upon each patient's arrival. This involves first determining the number of examinations needed and then assigning each examination to a specific department. To support this, we collected examination data from January to September 2023. Figure 2 presents the workloads of six departments, highlighting CT and US as having the highest volumes. It also shows the distribution of the number of patient examinations per patient, revealing that most patients require only one examination, while those needing multiple examinations are comparatively rare. The simulation code and data has been made publicly available at our GitHub [1]. The key simulation parameters are summarized as follows:

- 6 medical departments, including CT (computed tomography), DR (digital radiography), MR (magnetic resonance imaging), US (ultrasound), ES (endoscopy), and UC (ultrasound cardiogram).
- 5 to 20 examination equipment in each department.
- 5 time slots per day, each representing a two-hour period.
- 2 to 6 workload units per time slot, varying by department and differing between weekdays and weekends.

[1] `https://github.com/ZhangHuan-NWPU/OMASP`

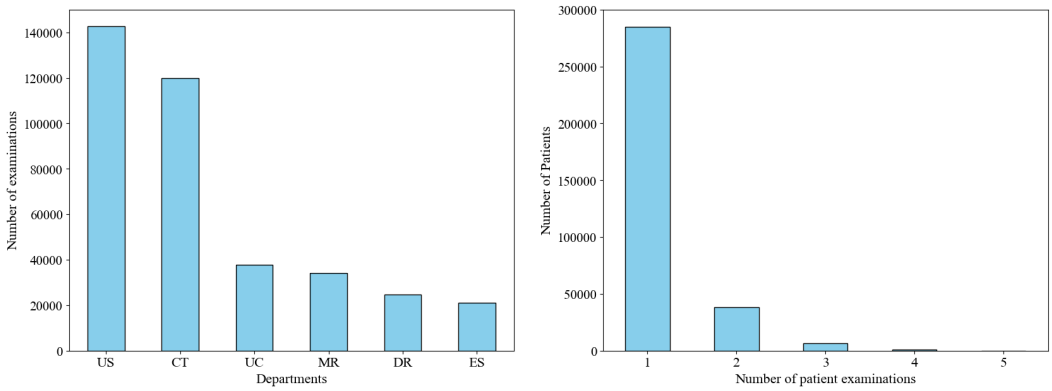


Fig. 2. Subfigure 1 shows the number of examinations for each department, while Subfigure 2 shows the number of patients for different examination numbers.

- Patient request arrivals follow a Poisson distribution.
- 1 to 4 examinations per patient.
- 1 to 4 units of workload per examination.

In each replication, the simulation process lasts for 15 days, with the first 5 days serving as a warm-up period. Each time a patient appointment request arrives, the simulator will call Algorithm1 to assign their examinations, and record the corresponding waiting days and the number of visits. Patients can submit appointment requests to the online system at any time. Whenever the simulation clock advances by 24 hours, the resource on the first day become unavailable. The outpatient scenarios can be further classified by varying two key factors:

- Number of patient examinations, defined at two levels. In the low-level scenario, most patients require only a single examination, with relatively few needing multiple examinations. The distribution is as follows: 80% for one examination, 15% for two, 5% for three, and 0% for four. In such cases, patients mainly care about reducing waiting days, as their appointments can typically be completed in one visit. As a result, their needs are easier to satisfy compared to patients with more complex requirements. However, since the objective of OMASP is to optimize the average patient experience, low-level scenarios may obscure the challenges faced by patients with multiple examination needs—who often require greater medical attention and value coordinated scheduling. To more comprehensively evaluate algorithm performance, especially in handling complex appointment demands, we define a high-level scenario by increasing the proportions of patients requiring two, three, or four examinations by 10% each. The resulting distribution is: 50% for one examination, 25% for two, 15% for three, and 10% for four. This setup better reflects the operational pressures of multi-examination scheduling and tests the robustness of the proposed approach.

- Daily arrival rate of patient requests, modeled using a Poisson distribution, with three levels corresponding to 80%, 100%, and 120% of the total examination resource capacity. This parameter controls the system load and enables us to examine algorithm performance under underloaded, balanced, and overloaded conditions.

By combining these two factors, we created 6 distinct outpatient scenarios and defined their names based on both characteristics. For example, <Low, 80% > designate a scenario with a low level of the number of patient examinations and a daily arrival rate of patient requests corresponding to 80% of total workloads of examination resources. For the low level of the number of patient examinations, the corresponding daily arrival rates are 354.26, 442.87, and 531.44 for 80%, 100%, and 120%, respectively. For the high level, the corresponding daily arrival rates are 234.40, 293.00, and 351.60, respectively.

5.2 Parameter settings

Table 2 shows the parameters for GP. We determined two key parameters, population size and maximum generations through tuning (see Section 6.1), while the other parameters were set according to the classic configurations in the literature (Hildebrandt and Branke, 2015; Koza, 1992; Mei et al., 2017). We tested several combinations of population size and maximum generations to evaluate their performance and selected the current parameter values. The GP method was implemented with Evolutionary Computation in Java (Luke, 2017). All experimental tests were run on a system with an Intel Core™ i7-10700 CPU @ 2.90GHz, and 16GB of RAM.

Table 2
Parameter settings

Parameter	Value	Section
Population size	512	Section 4.2
Maximum generations	50	Section 4.2
Crossover rate	90%	Section 4.2.1
Mutation rate	10%	Section 4.2.1
Selection method	tournament selection (size 7)	Section 4.2.1
Initialization method	ramped-half-and-half (minimum depth 2, maximum depth 6)	Section 4.2.1
Maximum depth	8	Section 4.2.1
Size of intermediate population	2560	Section 4.2
Replication times in full evaluations	100	Section 4.2.2

5.3 Results of feature selection

The results of feature selection are listed in Table 3. Column 1 reports the name of terminals. Columns 2-7 display the contribution values of each terminal across different scenarios, computed using the method described in Section 4.2.3. Column 8 shows their average contribution values. These results indicate that the importance of terminals is not fixed but depends on scenarios. Therefore, it is necessary to select a unique terminal set for each scenario in GP training. To determine the optimal value for the selection threshold, we ranked them in ascending order of their contributions, and the threshold was determined at the position where the slope changes most significantly. Figure 3 illustrates the variation in contributions across different scenarios, a threshold value of 0.08 is deemed appropriate for low-level scenarios and 0.1 for high-level scenarios. Terminals with contribution values above the threshold are selected. The contribution values of the selected terminals are highlighted in bold in Table 3.

Table 3
The results of feature selection across different scenarios.

Terminals	Scenarios						*Avg*
	Low-80%	Low-100%	Low-120%	High-80%	High-100%	High-120%	
WL	**0.08997**	**0.08500**	0.03092	0.04121	0.02650	**0.15690**	0.07175
EAD	0.00535	0.00174	0.00746	0.00881	**0.95837**	**0.27501**	0.20946
TWL	0.00105	**0.15364**	0.01926	**0.21933**	**0.19655**	**0.42763**	0.16958
NE	0.03606	**0.26179**	0.01790	0.02801	0.04594	0.00490	0.06577
EEAD	0.00997	0.05633	0.01658	**0.17393**	0.00153	0.03110	0.04824
LEAD	0.04660	0.02461	0.04118	**0.47396**	**0.22229**	**0.57925**	0.23132
NM	**0.11091**	0.00165	**0.12198**	0.00088	0.07254	0.09097	0.06649
AWL	0.02449	0.00000	0.04989	0.04878	0.00076	0.03362	0.02626
D	**0.20276**	**0.46008**	**0.16466**	0.00000	**0.53247**	**0.55852**	0.31975
T	**0.20340**	**0.32969**	**0.21800**	**2.69603**	**1.62138**	**1.92203**	1.16509
C	**0.12401**	0.06171	0.03648	**0.27918**	**0.25302**	0.00477	0.12653
RC	0.03627	0.00312	0.02122	0.06064	**0.27505**	**0.52155**	0.15297
RCE	0.02979	0.01962	0.00197	0.06583	**0.47026**	**0.35791**	0.15756
NWQ	**0.10351**	**0.08988**	0.01713	**0.18784**	0.01379	**0.31120**	0.12056
WWQ	**0.11894**	**0.20911**	**0.14844**	**2.14526**	**0.84312**	**1.03768**	0.75042

5.4 Computational results

For our experiments, we implemented two existing appointment scheduling rules commonly used in hospitals, denoted as AR_1 and AR_2, as well as a newly designed dispatching rule AR_3 that explicitly balances both objectives. All these rules can be simply represented as dispatching rules and

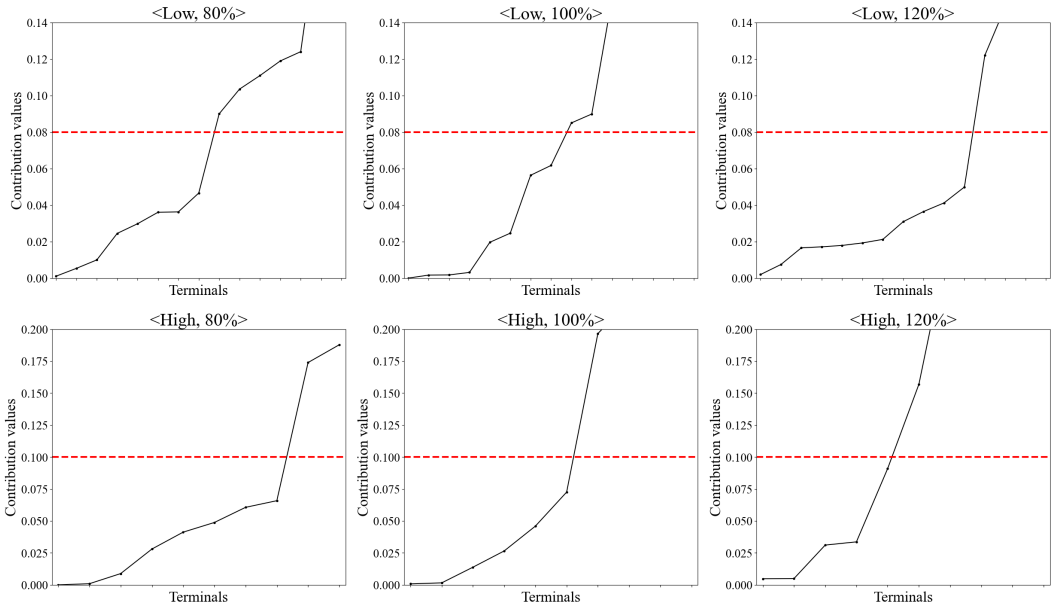


Fig. 3. The variation trend of contributions across different scenarios.

embedded in the proposed heuristic algorithm. In addition, we implement a primal–dual algorithm as a benchmark, as it is a widely adopted approach for online resource scheduling and patient scheduling problems (Buchbinder and Naor, 2009; Keyvanshokooh et al., 2021).

- AR_1 is implemented by assigning the highest priority to resource with the earliest date, which corresponds to the "earliest-day" rule. Note that AR_1 is also used as the benchmark rule for fitness evaluation in GP (see Section 4.2.2). Its expression is as follows:

$$priority = -D \tag{11}$$

- AR_2 is implemented by assigning the highest priority to resource with the latest date, which corresponds to the "single-day" rule. The dates of considered resources are limited to the range of $d_{\min}$ and $d_{\max}$, as shown in Algorithm 1. Therefore, all of a patient's appointments are consolidated into the same day. Its expression is as follows:

$$priority = D \tag{12}$$

- Considering that existing rules AR_1 and AR_2 are overly simplistic, we propose a new dispatching rule AR_3, motivated by insights from empirical observations and the literature (Burke et al., 2009). The proposed rule is designed to balance two objectives, minimizing patient waiting days and the number of hospital visits in a myopic and greedy manner. AR_3 consists of two parts. The first component, $EAD - D$, aims to minimize the patient's waiting days, similar to AR_1. The second

component is if $(LEAD > D)$ then δ else 0. Recall that d_e denotes the earliest available date for each examination e of a patient, and $d_{\max}$ is the maximum value of d_e among all examinations. So at least one examination for the patient can only be scheduled on $d_{\max}$, i.e. $LEAD$. If any examination is scheduled before $LEAD$, the patient may need to make one additional hospital visit at most. Based on this idea, AR_3 assigns a penalty δ if the date of resource is earlier than the latest available date $d_{\max}$ to reduce the number of hospital visits. We set $\delta = 2$ based on preliminary parameter tuning (see Section 6.1). Its expression is as follows:

$$priority = (EAD - D) - (\text{if } (LEAD > D) \text{ then } \delta \text{ else } 0) \tag{13}$$

- The primal–dual algorithm is derived from the dual formulation of the linear relaxation of the offline MASP. Upon the arrival of each patient, examinations are considered sequentially in decreasing order of their earliest available dates, and are assigned a resource determined by minimizing the reduced cost, which consists of the waiting date, a penalty for additional visits, and dynamic shadow prices reflecting resource congestion. After each decision, the shadow prices of the selected resource and the penalty for additional visits are updated accordingly.

In each scenario, SGP-FS was independently run 30 times. Since GP-based automated rule design requires relatively little human effort and time, it is potentially feasible and cost-effective to generate a specific dispatching rule for each outpatient scenario (Burke et al., 2013). The best dispatching rules evolved by SGP-FS were compared with three manually designed rules and the primal–dual algorithm, using a set of 1,000 independent simulation replications for all tests. In addition, Wilcoxon signed-rank tests were conducted to statistically compare SGP-FS with the other methods. Table 4 shows the computational results of the manually designed rules and the best rules evolved by SGP-FS. Column 1 gives the name of 6 scenarios, while the subsequent columns report the objective values obtained by AR_1, AR_2, AR_3, primal–dual algorithm and SGP-FS. The best objective values for each scenario are highlighted in bold. Here, + indicates that a method performs significantly better than SGP-FS, − indicates significantly worse performance, and = indicates no statistically significant difference. Columns 5 and 6 show the improvement of the evolved rules compared to the manually designed rules. Furthermore, we present the average values of the computational results in the last row.

The computational results lead to the following observations. First, the evolved rules outperform the manually designed rules and the primal–dual algorithm across all scenarios, achieving an average performance

improvement of at least 13.88%. In high-level scenarios, the improvement achieved by the evolved rules is significantly greater than that in low-level scenarios, indicating that evolved rules exhibit greater adaptability when handling patient appointment requests containing multiple examinations. Second, the primal–dual algorithm remains inferior to AR_3. This is mainly because it schedules examinations using candidate sets defined strictly by feasibility constraints and lacks the targeted pruning of candidate resources based on earliest and latest available dates. Moreover, It schedules each examination individually, whereas the proposed heuristic selects the appointment plan that maximizes the overall priority value across all examinations of a patient. In summary, GP shows the capability to evolve dispatching rules that exceed manually designed rules, especially pronounced in complex and high-demand environments.

Table 4
Computational results for the manually designed rules, the primal–dual algorithm, and the evolved rules across different scenarios.

Scenarios	AR_1	AR_2	AR_3	PD	SGP-FS	Imp. over AR_1	Imp. over AR_2	Imp. over AR_3	Imp. over PD
<Low, 80%>	4.9826 (-)	4.9878 (-)	4.9253 (-)	4.9378 (-)	**4.6802**	6.07%	6.17%	4.98%	5.22%
<Low, 100%>	6.5279 (-)	6.9059 (-)	6.7865 (-)	6.4607 (-)	**6.3016**	3.47%	8.75%	7.14%	2.46%
<Low, 120%>	8.7483 (-)	9.0811 (-)	8.6800 (-)	8.5830 (-)	**8.4542**	3.36%	6.90%	2.60%	1.50%
<High, 80%>	6.7479 (-)	6.6505 (-)	5.8008 (-)	6.7202 (-)	**4.6536**	31.04%	30.03%	19.78%	30.75%
<High, 100%>	10.4459 (-)	10.8380 (-)	9.3212 (-)	10.0859 (-)	**6.6992**	35.87%	38.19%	28.13%	33.58%
<High, 120%>	13.8657 (-)	15.2168 (-)	12.8373 (-)	13.5723 (-)	**10.1823**	26.56%	33.08%	20.68%	24.98%
Avg	8.5531	8.9467	8.0585	8.3933	**6.8285**	17.73%	20.52%	13.88%	16.41%

Table 5 reports the average waiting days and the number of hospital visits achieved by the manually designed rules, the primal–dual algorithm, and the evolved rules across different scenarios. Among all methods, SGP-FS consistently yields the shortest waiting days, whereas AR_2 consistently yields the smallest number of visits. The results indicate that the performance improvement of SGP-FS over the other methods is primarily driven by its substantial reduction in patient waiting days, while still maintaining a reasonable number of hospital visits.

6 Experimental analysis

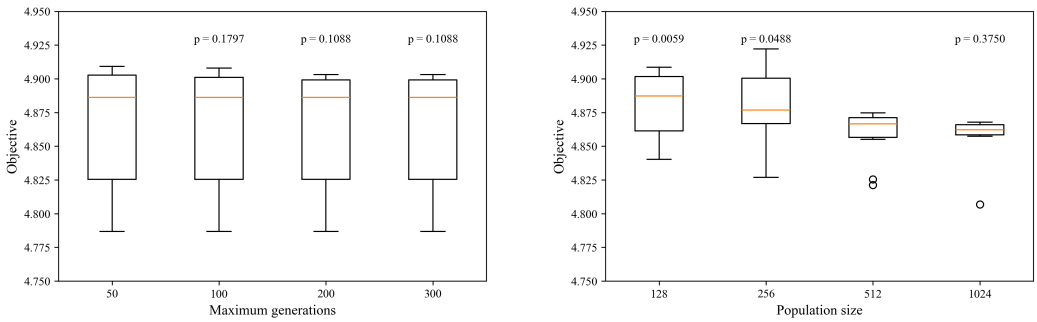
6.1 Sensitivity Analysis

First of all, we conduct a sensitivity analysis of key parameters in SGP-FS, including the maximum generations and the population size. We test each parameter while fixing other parameters to their default values. For every

Table 5
Waiting days and the number of visits for the manually designed rules, the primal–dual algorithm, and the evolved rules across different scenarios.

	Waiting Days					Number of Visits				
Scenarios	AR_1	AR_2	AR_3	PD	SGP-FS	AR_1	AR_2	AR_3	PD	SGP-FS
<Low-80%>	2.1331	2.4856	2.4150	2.2192	**1.9597**	1.1398	**1.0009**	1.0041	1.0875	1.0882
<Low-100%>	3.6096	4.4037	4.2734	3.6492	**3.5894**	1.1673	**1.0009**	1.0052	1.1246	1.0849
<Low-120%>	5.7188	6.5790	6.0921	5.7348	**5.5856**	1.2118	**1.0008**	1.0352	1.1393	1.1474
<High-80%>	3.1110	4.1170	3.1793	4.0882	**2.0577**	1.4548	**1.0134**	1.0486	1.0528	1.0384
<High-100%>	6.3738	8.2921	6.6418	6.7845	**3.7621**	1.6288	**1.0184**	1.0718	1.3206	1.1748
<High-120%>	9.7313	12.6651	9.7593	10.0134	**7.1265**	1.6537	**1.0207**	1.2312	1.4236	1.2223
Average	5.1129	6.4237	5.3935	5.4149	**4.0135**	1.3760	**1.0092**	1.0660	1.1914	1.1260

parameter setting and each scenario, 10 independent runs are performed. Wilcoxon tests are further applied to compare each setting with the default configuration. The results for the first scenario are illustrated in Figure 4. The results show that SGP-FS converges within 50 generations, and further increasing the number of generations does not yield statistically significant improvements. Increasing the population size leads to noticeable performance gains. However, once the population size exceeds 512, the marginal improvement becomes negligible.



(a) Maximum generations (b) Population size

Fig. 4. Boxplots of parameter settings for (a) the maximum generations and (b) the population size. The p-values above each boxplot report the significance of differences relative to the default configuration.

In this study, the objective function is defined with fixed weights. To investigate the impact of weight settings on the results, we fixed $\alpha_1 = 1$ and varied $\alpha_2 = 0, 1, 2.5, 5, 10$, and 20 to explore the relative importance on the waiting days and the number of visits. We ran SGP-FS 10 times to evolve dispatching rules under different weight settings. Figure 5 illustrates the average waiting days and the number of visits of evolved rules under different weight settings in the first scenario, with each point representing a specific weight settings and its corresponding α_2 value indicated. It can be

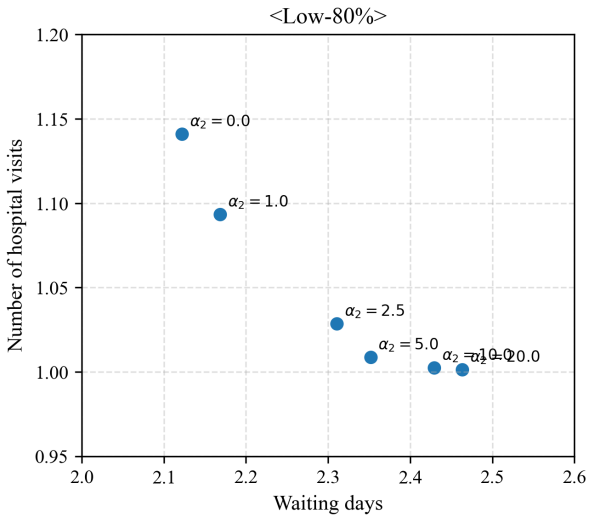


Fig. 5. Scatter plot of weight settings in the first scenario

observed that as α_2 increases, the average waiting days gradually increases, while the number of visits decreases, confirming that the two objectives are inherently conflicting and require explicit trade-offs. SGP-FS converges to different Pareto-optimal solutions under different weight settings, indicating that it is able to capture the conflict between objectives.

Finally, for the dispatching rule AR_3, the sensitivity analysis of the parameter δ is reported in Table 6. The best-performing values are highlighted in bold. The results indicate that the best performance is achieved when $\delta = 2$.

Table 6
Sensitivity analysis of parameter δ in dispatching rule AR_3.

Scenarios	$\delta = 0.5$	$\delta = 1$	$\delta = 2$	$\delta = 2.5$
Low-80%	4.9877	4.9068	**4.9253**	**4.9253**
Low-100%	6.5234	**6.5044**	6.7865	6.5254
Low-120%	8.7435	8.7302	8.6800	**8.6480**
High-80%	6.8204	6.0233	**5.8008**	6.8204
High-100%	10.4593	9.8592	**9.3212**	10.4592
High-120%	13.8647	13.5437	**12.8373**	13.8645
Avg	8.5665	8.2613	**8.0585**	8.5405

6.2 Component Analysis

In this section, we conducted a component analysis of the GP variants to investigate the effectiveness of of surrogate evaluations and feature selection. The analysis focused on two key metrics: test performance and program length, where smaller values are considered indicative of better outcomes. We carried

out Wilcoxon signed rank tests, with a significance level of 0.05, for statistical significance tests.

Two other GP variants were tested and compared in our experiments: (1) standard GP. Compared to the proposed GP, this variant also uses lazy evaluations and full evaluations in the evolutionary process; however, the key difference is that it does not adopt a surrogate model for preselection. (2) surrogate-assisted GP (SGP). As described in Algorithm 2, this variant integrates a simplified model-based surrogate to enhance the search mechanism, but does not include feature selection. Each GP variant will be independently run 30 times. The evolved rules were applied to 1,000 independent simulation replications, evaluating their performance and program length.

Table 7 presents the average test performance of the dispatching rules evolved by standard GP, SGP and SGP-FS across different scenarios, including objective values (with standard deviations in parentheses) and the statistical significance of comparisons with the other two methods. The best values are highlighted in bold. Here, + indicates the method is significantly better than others, − indicates the method is significantly worse than others, and = indicates no significant difference. The last two rows summarize the average performance and mean rank of each method across all scenarios. The mean rank is determined as follows: for each scenario, the three approaches are ranked based on their objective values, with the best method assigned a rank of 1, the second-best a rank of 2, and the worst a rank of 3. The mean rank represents the average ranking of each method across all scenarios based on its performance. Table 7 reveals that the rules evolved by SGP variants can dominate those generated by standard GP. While SGP outperforms other methods in certain scenarios, SGP-FS achieves better overall performance and ranking.

Table 8 displays the average program length of the dispatching rules evolved by standard GP, SGP and SGP-FS across different scenarios. The program length is defined as the number of nodes in the rule's tree-based representation. A smaller number of nodes reduces the efforts needed for a human user to simulate the model's behavior, enhancing its interpretability (Mei et al., 2022). In terms of program length, SGP-FS achieves the shortest overall length and the highest average ranking, demonstrating its ability to balance the performance and interpretability. Standard GP achieves the shortest program length in certain scenarios, whereas SGP generally produces longer rules, demonstrating weaker interpretability.

In summary, these results underscore the effectiveness of incorporating surrogate evaluations and feature selection to improve GP performance.

While surrogate evaluations may reduce the interpretability of dispatching rules, feature selection helps mitigate this limitation.

Table 7
Mean (std.) test performance of standard GP, SGP and SGP-FS.

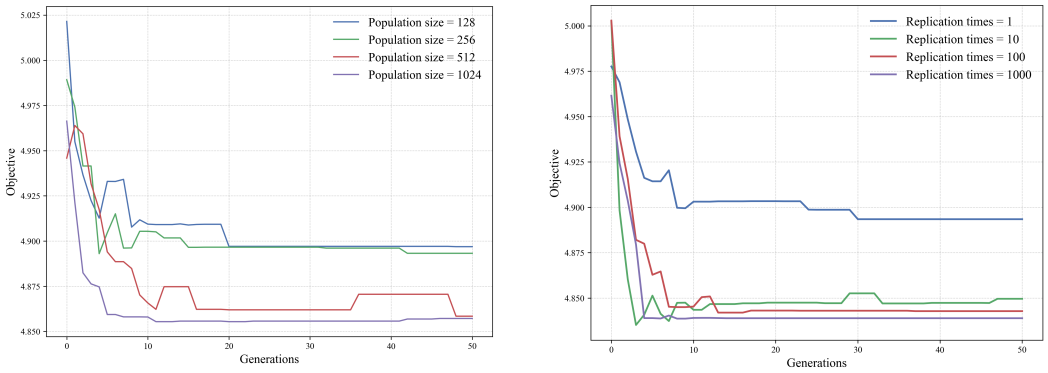
Scenarios	Standard GP	SGP	SGP-FS
<Low, 80%>	4.9295 (0.0585) (−−)	4.8727 (0.0749) (+ =)	**4.8428** (0.0761) (+ =)
<Low, 100%>	6.5453 (0.0766) (==)	**6.5255** (0.0865) (==)	6.5519 (0.0965) (==)
<Low, 120%>	8.7467 (0.0708) (==)	8.7232 (0.1252) (==)	**8.7000** (0.1024) (==)
<High, 80%>	5.5065 (0.2497) (= −)	5.3823 (0.2222) (= −)	**5.1511** (0.2884) (++)
<High, 100%>	8.5703 (0.6151) (− =)	**8.2381** (0.3824) (+ =)	8.2770 (0.4385) (==)
<High, 120%>	12.4937 (0.5257) (−−)	11.9714 (0.8035) (+ =)	**11.8007** (0.6570) (+ =)
Avg	7.7986 (0.2661)	7.6189 (0.2825)	7.5539 (0.2765)
Mean rank	2.83	1.67	1.5

Table 8
Mean (std.) program length of standard GP, SGP and SGP-FS.

Scenarios	Standard GP	SGP	SGP-FS
<Low, 80%>	23.8667(27.0845)(==)	26.7333(33.6800)(==)	**17.4667**(24.1057)(==)
<Low, 100%>	30.7667(33.7962)(==)	27.0000(34.0162)(==)	**19.2667**(19.4297)(==)
<Low, 120%>	**24.5333**(25.9957)(==)	37.0000(31.0794)(==)	31.6333(32.7935)(==)
<High, 80%>	22.9333(27.0235)(− =)	**13.4667**(25.6928)(++)	24.6667(19.1407)(= −)
<High, 100%>	15.6000(20.6291)(==)	28.8000(30.6430)(= −)	**11.3333**(15.9510)(= +)
<High, 120%>	**14.9333**(21.2764)(==)	28.0333(31.7354)(==)	19.1333(22.6179)(==)
Avg	22.1056(25.9676)	26.8389(31.1411)	20.5833(22.3398)
Mean rank	1.83	2.5	1.67

6.3 Convergence Analysis

In this section, we investigate the effects of the population size and the number of replications in full evaluation on the algorithm's behavior and convergence properties. We ran SGP-FS 10 times under different parameter settings and present the corresponding convergence curves in Fig 6. As shown in subfigure 6a, larger population sizes lead to higher-quality initial solutions and faster convergence, resulting in better final dispatching rules. As shown in subfigure 6b, increasing the number of replications in full evaluation improves the accuracy of fitness evaluation, thereby enhancing the stability and overall performance of the algorithm. However, this improvement incurs a substantial computational cost: increasing the number of replications from 100 to 1000 raises the average runtime of SGP-FS from 9.23 to 43.84 minutes. This suggests that further increasing the number of replications is inefficient.



(a) Population size (b) Replication times in full evaluation

Fig. 6. Convergence curves under different parameter settings: (a) population size and (b) replication times in full evaluation.

6.4 Runtime Analysis

This section analyzes the runtime of SGP-FS, as reported in Table 9. The runtime of SGP-FS is related to the daily arrival rate of patient requests, since higher arrival rates increase the duration of the simulation-based evaluation. Overall, since the dispatching rules produced by SGP-FS can be reused over a long period, the computational cost is acceptable.

Table 9
Average runtime of SGP-FS under different scenarios (in minutes).

Scenario	<Low, 80%>	<Low, 100%>	<Low, 120%>	<High, 80%>	<High, 100%>	<High, 120%>
Runtime	9.23	22.43	22.43	9.71	16.64	27.98

We also evaluates the online feasibility of the evolved rules by analyzing the computation time required to schedule one patient request. The computation times of different online scheduling methods are reported in Table 10. The results show that all methods incur very low computational overhead and fully satisfy the real-time requirements of the OMASP. Among them, the primal–dual algorithm exhibits the highest computational cost.

6.5 Robustness Analysis

This section investigates the robustness of SGP-FS by evaluating its performance under additional hospital configurations. Specifically, we construct a new set of settings in which the number of machines is doubled, while the number of examinations per patient and the daily arrival rate of patient requests remain unchanged. Using the same experimental setup and feature set as in the main computational study, we run SGP-FS under these

Table 10
Average computation time per patient for different online scheduling methods (in 10^{-5} seconds).

Scenarios	AR_1	AR_2	AR_3	PD	SGP-FS
<Low, 80%>	0.14	0.16	0.14	2.21	0.22
<Low, 100%>	0.08	0.08	0.09	1.81	0.10
<Low, 120%>	0.08	0.09	0.09	1.80	0.13
<High, 80%>	0.09	0.09	0.08	10.21	0.08
<High, 100%>	0.13	0.13	0.12	9.76	0.25
<High, 120%>	0.16	0.17	0.18	9.80	0.17
Avg	0.11	0.12	0.12	5.93	0.16

new settings and compare its performance with other benchmark methods, as reported in Table 11. The results show that SGP-FS remains effective across different hospital configurations. When the hospital setting changes, it is sufficient to re-evolve the dispatching rules.

Table 11
Computational results for the manually designed rules, the primal-dual algorithm, and the evolved rules in larger hospital settings.

Scenarios	AR_1	AR_2	AR_3	PD	SGP-FS	Imp. over AR_1	Imp. over AR_2	Imp. over AR_3	Imp. over PD
Low–80%	4.9644	4.9651	4.9074	4.9696	**4.6200**	6.94%	6.95%	5.86%	7.03%
Low–100%	6.5034	6.8897	6.4576	6.4300	**6.2355**	4.12%	9.50%	3.44%	3.02%
Low–120%	8.7171	9.0431	8.6711	8.5461	**8.2058**	5.87%	9.26%	5.37%	3.98%
High–80%	6.6801	6.5261	5.7617	6.7069	**4.5931**	31.24%	29.62%	20.28%	31.52%
High–100%	10.4402	10.7453	9.3743	10.0541	**6.8681**	34.22%	36.08%	26.73%	31.69%
High–120%	13.8561	15.1660	13.0710	13.5192	**9.8637**	28.81%	34.96%	24.54%	27.04%
Avg	8.5269	8.8892	8.0405	8.3710	**6.7310**	18.53%	21.06%	14.37%	17.38%

6.6 Behavior analysis of the evolved rules

In this section, we examine the behavior of the dispatching rules evolved by SGP-FS. These high-performance rules were selected from the GP output under three distinct high-level scenarios, where they demonstrated notable performance improvements. However, the GP-generated programs are typically complex and opaque, lacking clear structure or interpretability. To address this, we applied a two-step simplification process to extract interpretable rule representations: (1) Redundancy elimination. We simplified all redundant functions in the expressions. For example, expressions like $a + 0$ can be replaced with a, and the conditional statement "if-then-else" can be replaced with the result that always holds. (2) Structural pruning via subtree replacement. We systematically traversed each expression tree and attempted to replace subtrees with a constant value of 1. If the resulting tree

does not exhibit a statistically significant difference over 1,000 simulation evaluations, the change is accepted (Nguyen et al., 2016). Through this simplification procedure, we obtained three interpretable rules, denoted as R_1, R_2, and R_3, as shown in Table 12. The table also includes the expressions of manually designed rules, which were previously introduced in Section 5.4.

Table 12
Simplified expressions of the evolved rules.

Rules	Scenarios	Expressions
R_1	<High, 80%>	$max\{T, TWL\} - TWL$
R_2	<High, 100%>	$((D - EAD) * RCE - 1)/T$
R_3	<High, 120%>	$((((T + TWL) * RC)/WL) - 3 * LEAD)/D$
AR_1	-	$-D$
AR_2	-	D
AR_3	-	$(EAD - D) -$ (if $(LEAD - D)$ then 2 else 0)

As presented in the table, manually designed rules typically exhibit simple structures, making their behavior readily interpretable from their mathematical expressions. In contrast, evolved rules often incorporate a more nuanced combination of decision attributes. Although they may be less transparent, these rules capture the underlying trade-offs in the scheduling problem more effectively. To bridge this gap in interpretability, visualization serves as a powerful tool—not only to intuitively illustrate the contribution of each terminal but also to uncover latent patterns that are not immediately apparent from the symbolic expressions. This aids in deepening our understanding of how the evolved rules function.

To analyze the internal mechanics of these rules, we visualized the influence of each terminal on the computed priority values, thereby revealing how the rules discriminate among resources during decision-making. Specifically, we conducted extensive simulations to collect a large number of samples, where each sample corresponds to a resource at a decision point and records the values of all terminals along with the resulting priority value. For a given terminal, the samples were sorted in ascending order of the terminal's value and plotted along a standardized axis. A color map was then used to represent the associated priority values, with red indicating the highest priority and blue the lowest. If a terminal's color band shows a strong correlation with priority levels, it signals a meaningful influence of that terminal on the decision rule and warrants further attention.

For example, subfigures 7a and 7b in Figure 7 display the visualizations of AR_1 and AR_2, respectively. Below each color band, the corresponding terminal name is annotated, with the minimum and maximum observed values indicated at two ends of the color band. In subfigure 7a, the color bands of terminals EAD, $EEAD$, $LEAD$, and D transition from red to blue

as their values increase from bottom to top. This pattern reveals that AR_1 tends to assign higher priority to resources with earlier date values, underscoring the significant role these terminals play in its decision-making process. In contrast, subfigure 7b illustrates a similar emphasis on the same set of terminals in AR_2; however, the gradient direction is reversed. This indicates that AR_2 favors resources with later date values, demonstrating a distinct prioritization strategy compared to AR_1. For AR_3, the trade-off between minimizing waiting days and the number of visits is governed by the relative positions of terminals D, EAD, and $LEAD$. As a result, its decision-making behavior cannot be easily illustrated through visualization.

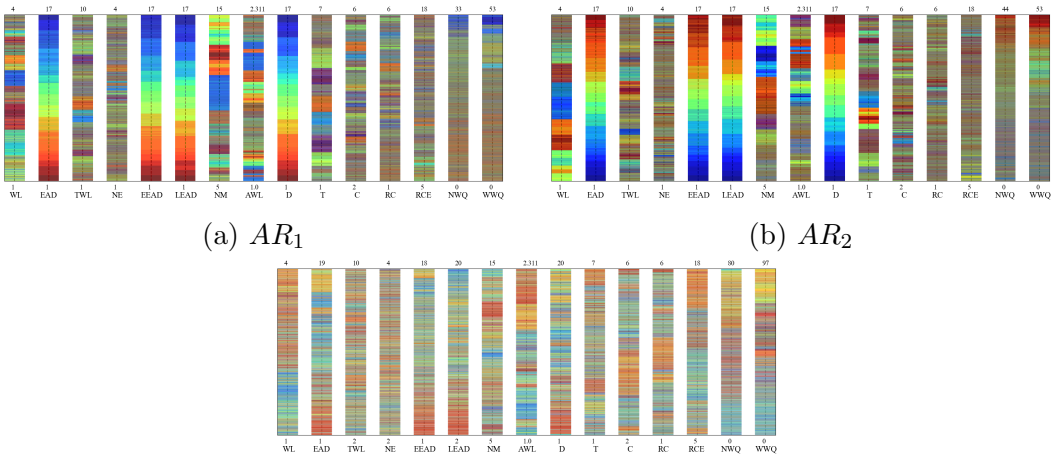
However, the evolved rules are inherently more complex in structure and behavior. The expression for R_1 highlights the significance of the terminals T and TWL. Specifically, when $T > TWL$, R_1 is effectively dominated by the value of T, thereby favoring the allocation of patient examinations to the latest available time slots, as illustrated in subfigure 8a. Conversely, when $TWL > T$, the rule evaluates to zero. In such cases, when multiple resources share the same priority value, tie-breaking is determined by the scheduling system based on earlier time slots. This design has practical implications: patients with larger TWL values—who often require multiple examinations—are assigned to earlier time slots, while those with smaller TWL values are scheduled in later time slots.

The behavior of R_2 can be characterized by two distinct cases, determined by the relationship between the terminals D and EAD, as inferred from its expression. When $D = EAD$, R_2 simplifies to $-1/T$, thereby favoring resources with larger values of T, which typically correspond to later time slots. In contrast, when $D > EAD$, R_2 remains positive due to the substantial contribution of the RCE term. Although RCE appears in the symbolic expression of R_2, the visual analysis in Figure 8 suggests that it has minimal influence on the rule's actual decision-making behavior. In this latter case, R_2 tends to assign higher priority to resources with smaller T values and large D values. This reflects a strategy that aims to consolidate multiple examinations for the same patient on a single day, thereby minimizing the total number of visits. Subfigures 8b and 8c illustrate the contrasting tendencies of R_2 with respect to T in these two cases. Importantly, patients requiring only one examination are consistently classified under case 1, as the candidate resources for such patients always have $D = EAD$, according to our heuristic algorithm 1. In contrast, patients with multiple required examinations are more frequently associated with case 2.

Finally, the visualization of $R3$ highlights the importance of the terminals EAD, $EEAD$, $LEAD$, D, NWQ and WWQ, while the influence of other terminals appears negligible. It can be observed that $R3$ behaves similarly to

AR_1, with the key difference being that the decisions of $R3$ are affected by NWQ and WWQ. Although these two terminals do not explicitly appear in the symbolic expression of $R3$, the visualization suggests that they are nonetheless implicitly correlated with its decision-making process. This indicates that $R3$ tends to assign higher priority to resources with lower NWQ and WWQ, reflecting a strategy that promotes queue balancing across examination equipment to improve overall system efficiency.

The behavioral analysis yields several important insights. First, centralized scheduling platforms should adopt adaptive appointment strategies based on daily patient arrival rates. When demand is within the capacity of outpatient examination resources, priority should be given to patients requiring multiple examinations by assigning them earlier dates and earlier time slots. As demand approaches capacity, the scheduling strategy must shift to prioritize examination consolidation. Specifically, multiple examinations for the same patient should be scheduled on the same day by using the latest available dates but earliest time slots. In contrast, patients with only one examination can be assigned to later time slots. When patient demand exceeds system capacity, the focus should shift to queue reduction and workload balancing across equipment. Effective measures include extending working hours or referring patients to affiliated facilities. These findings offer actionable guidance for centralized scheduling systems, contributing to better resource utilization and improved service quality.



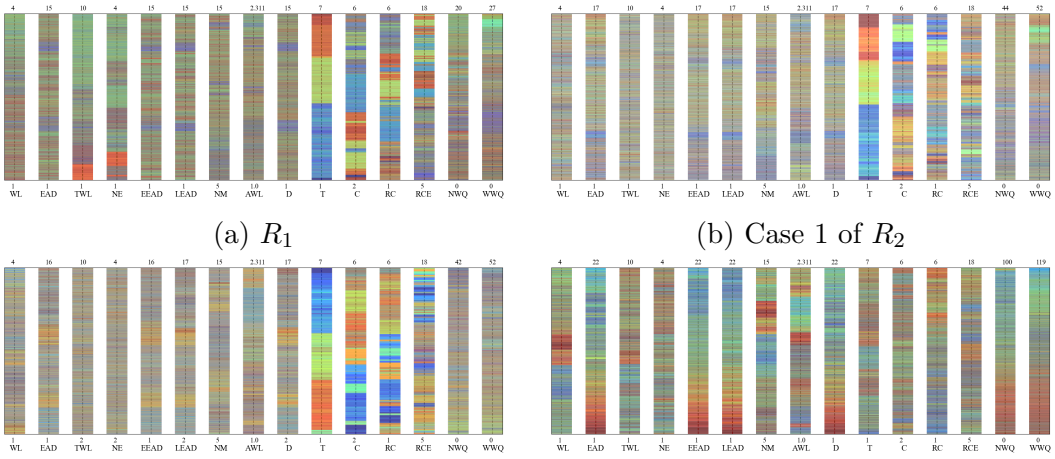
(a) AR_1 (b) AR_2

(c) AR_3

Fig. 7. Behavioral visualization of AR_1, AR_2 and AR_3.

7 Conclusion

With the advancement of information technology, many high-ranking hospitals in China have adopted centralized appointment platforms for



(a) R_1 (b) Case 1 of R_2

(c) Case 2 of R_2 (d) R_3

Fig. 8. Behavioral visualization of $R1$, $R2$ and $R3$.

medical examinations, allowing patients to submit requests online and receive appointment plans in real time. The effectiveness of such platforms hinges on the quality of the dispatching rules used to schedule appointments. However, manually designing these rules is challenging due to the complexity and dynamic nature of the scheduling environment. To address this, we proposed a genetic programming (GP) approach that automatically evolves effective dispatching rules. Our method integrates a multi-fidelity simulation-based evaluator and an offline feature selection mechanism to enhance learning efficiency and solution quality.

Using real data from a partner hospital, we developed a discrete event simulation model and constructed a range of scenarios that vary in both the number of examinations per patient and the daily volume of appointment requests. Computational results demonstrate that the GP-based approach outperforms commonly used appointment strategies, achieving an average performance improvement of at least 13.88%. The approach is particularly effective under high-load conditions characterized by large numbers of examinations and elevated patient arrival rates. To support managerial decision-making, we extracted interpretable dispatching rules from the evolved GP solutions through symbolic simplification and introduced a visualization technique to analyze rule behavior. Our analysis reveals that the optimal scheduling strategy is sensitive to the daily arrival rate: when capacity is sufficient, priority should be given to patients requiring multiple examinations; under high demand, minimizing queue lengths becomes paramount.

This study opens several avenues for future research, with two key directions identified for model enhancement. First, we intend to incorporate additional sources of real-world uncertainty, such as patient no-shows and the stochastic nature of examination durations. Second, we aim to enrich the

objective function by including hospital operational cost metrics, such as idle time and overtime, thereby facilitating a more comprehensive assessment of trade-offs between patient satisfaction and resource efficiency. Although this study focuses on the appointment scheduling problem, the underlying framework is generalizable and can be adapted to a broad range of resource allocation challenges commonly encountered in healthcare systems.

Acknowledgments

This work was partially supported by the National Natural Science Foundation of China (No. 72371200, 71971172, 72342014, 72371176).

References

Ahmadi-Javid, A., Jalali, Z., Klassen, K.J., 2017. Outpatient appointment systems in healthcare: A review of optimization studies. European Journal of Operational Research 258, 3–34.

Azadeh, A., Baghersad, M., Farahani, M.H., Zarrin, M., 2015. Semi-online patient scheduling in pathology laboratories. Artificial Intelligence in Medicine 64, 217–226.

Bader-El-Den, M., Poli, R., Fatima, S., 2009. Evolving timetabling heuristics using a grammar-based genetic programming hyper-heuristic framework. Memetic Computing 1, 205–219.

Bailey, N.T., 1952. A study of queues and appointment systems in hospital out-patient departments, with special reference to waiting-times. Journal of the Royal Statistical Society Series B: Statistical Methodology 14, 185–199.

Braaksma, A., Kortbeek, N., Post, G.F., Nollet, F., 2014. Integral multidisciplinary rehabilitation treatment planning. Operations Research for Health Care 3, 145–159.

Braune, R., Benda, F., Doerner, K.F., Hartl, R.F., 2022. A genetic programming learning approach to generate dispatching rules for flexible shop scheduling problems. International Journal of Production Economics 243, 108342.

Buchbinder, N., Naor, J., 2009. The design of competitive online algorithms via a primal—dual approach. Foundations and Trends in Theoretical Computer Science 3, 93–263.

Burke, E., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Qu, R., 2013. Hyper-heuristics: A survey of the state of the art. Journal of the Operational Research Society 64, 1695–1724.

Burke, E.K., Hyde, M.R., Kendall, G., Ochoa, G., Ozcan, E., Woodward, J.R., 2009. Exploring hyper-heuristic methodologies with genetic programming,

in: Computational intelligence: Collaboration, fusion and emergence. Springer, pp. 177–201.

Burke, E.K., Hyde, M.R., Kendall, G., Woodward, J.R., 2007. The scalability of evolved on line bin packing heuristics, in: 2007 IEEE Congress on Evolutionary Computation, pp. 2530–2537.

Castro, E., Petrovic, S., 2012. Combined mathematical programming and heuristics for a radiotherapy pre-treatment scheduling problem. Journal of Scheduling 15, 333–346.

Cayirli, T., Yang, K.K., 2014. A universal appointment rule with patient classification for service times, no-shows, and walk-ins. Service Science 6, 274–295.

Cayirli, T., Yang, K.K., Quek, S.A., 2012. A universal appointment rule in the presence of no-shows and walk-ins. Production and Operations Management 21, 682–697.

Chien, C.F., Tseng, F.P., Chen, C.H., 2008. An evolutionary approach to rehabilitation patient scheduling: A case study. European Journal of Operational Research 189, 1234–1253.

Conforti, D., Guerriero, F., Guido, R., 2008. Optimization models for radiotherapy patient scheduling. 4OR 6, 263–278.

Conforti, D., Guerriero, F., Guido, R., 2010. Non-block scheduling with priority for radiotherapy treatments. European Journal of Operational Research 201, 289–296.

Fan, H., Xiong, H., Goh, M., 2021. Genetic programming-based hyper-heuristic approach for solving dynamic job shop scheduling problem with extended technical precedence constraints. Computers and Operations Research 134, 105401.

Fukunaga, A.S., 2008. Automated discovery of local search heuristics for satisfiability testing. Evolutionary Computation 16, 31–61.

Guo, H., Liu, J., Zhuang, C., 2022. Automatic design for shop scheduling strategies based on hyper-heuristics: A systematic review. Advanced Engineering Informatics 54, 101756.

Guo, T., Mei, Y., Zhang, M., Sun, R., Zhu, Y., Du, W., 2025. Genetic programming with multifidelity surrogates for large-scale dynamic air traffic flow management. IEEE Transactions on Evolutionary Computation 29, 2671–2685.

Haupt, R., 1989. A survey of priority rule-based scheduling. OR Spectrum 11, 3–16.

Heinrich, J., Weiskopf, D., 2013. State of the art of parallel coordinates. Eurographics (State of the Art Reports) , 95–116.

Hildebrandt, T., Branke, J., 2015. On using surrogates with genetic programming. Evolutionary Computation 23, 343–367.

Hildebrandt, T., Heger, J., Scholz-Reiter, B., 2010. Towards improved dispatching rules for complex shop floor scenarios: a genetic programming approach, in: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation, pp. 257–264.

Hsieh, F.S., 2017. A hybrid and scalable multi-agent approach for patient scheduling based on petri net models. Applied Intelligence 47, 1068–1086.

Jakobović, D., Jelenković, L., Budin, L., 2007. Genetic programming heuristics for multiple machine scheduling, in: Genetic Programming: 10th European Conference, Springer. pp. 321–330.

Javed, N., Gobet, F., Lane, P., 2022. Simplification of genetic programs: a literature survey. Data Mining and Knowledge Discovery 36, 1279–1300.

Keller, R.E., Poli, R., 2007. Linear genetic programming of parsimonious metaheuristics, in: 2007 IEEE Congress on Evolutionary Computation, IEEE. pp. 4508–4515.

Keyvanshokooh, E., Shi, C., Van Oyen, M.P., 2021. Online advance scheduling with overtime: A primal-dual approach. Manufacturing & Service Operations Management 23, 246–266.

Kortbeek, N., van der Velde, M., Litvak, N., 2017. Organizing multidisciplinary care for children with neuromuscular diseases at the academic medical center, amsterdam. Health Systems 6, 209–225.

Koza, J., 1992. Genetic Programming: On the Programming of Computers by Means of Natural Selection. A Bradford book, MIT Press, Cambridge.

Leeftink, A., Bikker, I., Vliegen, I., Boucherie, R., 2020. Multi-disciplinary planning in health care: a review. Health Systems 9, 95–118.

Luke, S., 2017. ECJ then and now, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp. 1223–1230.

Luscombe, R., Kozan, E., 2016. Dynamic resource allocation to improve emergency department efficiency in real time. European Journal of Operational Research 255, 593–603.

MacLachlan, J., Mei, Y., Zhang, F., Zhang, M., Signal, J., 2023. Learning emergency medical dispatch policies via genetic programming, in: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1409–1417.

Marynissen, J., Demeulemeester, E., 2019. Literature review on multi-appointment scheduling problems in hospitals. European Journal of Operational Research 272, 407–419.

Mei, Y., Chen, Q., Lensen, A., Xue, B., Zhang, M., 2022. Explainable artificial intelligence by genetic programming: A survey. IEEE Transactions on Evolutionary Computation 27, 621–641.

Mei, Y., Nguyen, S., Xue, B., Zhang, M., 2017. An efficient feature selection algorithm for evolving job shop scheduling rules with genetic programming. IEEE Transactions on Emerging Topics in Computational Intelligence 1, 339–353.

Mei, Y., Zhang, M., Nyugen, S., 2016. Feature selection in evolving job shop dispatching rules with genetic programming, in: Proceedings of the Genetic and Evolutionary Computation Conference 2016, pp. 365–372.

Nguyen, S., Mei, Y., Xue, B., Zhang, M., 2019. A hybrid genetic programming algorithm for automated design of dispatching rules. Evolutionary Computation 27, 467–496.

Nguyen, S., Zhang, M., Tan, K.C., 2016. Surrogate-assisted genetic programming with simplified models for automated design of dispatching rules. IEEE Transactions on Cybernetics 47, 2951–2965.

Pérez, E., Ntaimo, L., Malavé, C.O., Bailey, C., McCormack, P., 2013. Stochastic online appointment scheduling of multi-step sequential procedures in nuclear medicine. Health Care Management Science 16, 281–299.

Saadani, N.E.H., Bahroun, Z., Bouras, A., 2014. A linear mathematical model for patients' activities scheduling on hospital resources, in: 2014 International Conference on Control, Decision and Information Technologies (CoDIT), IEEE. pp. 074–080.

Xue, B., Zhang, M., Browne, W.N., Yao, X., 2015. A survey on evolutionary computation approaches to feature selection. IEEE Transactions on Evolutionary Computation 20, 606–626.

Zhang, F., Mei, Y., Nguyen, S., Zhang, M., 2021a. Collaborative multifidelity-based surrogate models for genetic programming in dynamic flexible job shop scheduling. IEEE Transactions on Cybernetics 52, 8142–8156.

Zhang, F., Mei, Y., Nguyen, S., Zhang, M., 2021b. Evolving scheduling heuristics via genetic programming with feature selection in dynamic flexible job-shop scheduling. IEEE Transactions on Cybernetics 51, 1797–1811.

Zhang, F., Mei, Y., Nguyen, S., Zhang, M., 2024. Survey on genetic programming and machine learning techniques for heuristic design in job shop scheduling. IEEE Transactions on Evolutionary Computation 28, 147–167.

Zhang, M., Wong, P., 2008. Explicitly simplifying evolved genetic programs during evolution. International Journal of Computational Intelligence and Applications 7, 201–232.