A Reinforcement Learning Approach for the Online Dynamic Home Health Care Scheduling Problem

Quy Ta-Dinh[1], Tu-San Pham[2], Minh Hoàng Hà[3*], Louis-Martin Rousseau[2]

[1]ORLab, Faculty of Computer Science, Phenikaa University, Hanoi, Vietnam.
[2]Department of Mathematics and Industrial Engineering, Polytechnique Montreal, Quebec, Canada.
[3]SLSCM, Faculty of Data Science and Artificial Intelligence, College of Technology, National Economics University, Hanoi, Vietnam.

*Corresponding author(s). E-mail(s): hoanghm@neu.edu.vn;

Over recent years, home health care has gained significant attention as an efficient solution to the increasing demand for healthcare services. Home health care scheduling is a challenging problem involving multiple complicated assignments and routing decisions subject to various constraints. The problem becomes even more challenging when considered on a rolling horizon with stochastic patient requests. This paper discusses the Online Dynamic Home Health Care Scheduling Problem (ODHHCSP), in which a home health care agency has to decide whether to accept or reject a patient request and determine the visit schedule and routes in case of acceptance. The objective of the problem is to maximize the number of patients served, given the limited resources. When the agency receives a patient's request, a decision must be made on the spot, which poses many challenges, such as stochastic future requests or a limited time budget for decision-making. In this paper, we model the problem as a Markov decision process and propose a reinforcement learning (RL) approach. The experimental results show that the proposed approach outperforms other algorithms in the literature in terms of solution quality. In addition, a constant runtime of less than 0.001 seconds for each decision makes the approach especially suitable for an online setting like our problem.

Highlights

- We introduce a Reinforcement Learning (RL) algorithm for scheduling home health care services in real-time. The algorithm optimizes patient acceptance decisions, maximizing patient care within resource constraints.
- Our RL approach surpasses greedy methods and delivers solution quality comparable to expensive simulation-based approaches, but with significantly faster execution times. This speed is critical for real-time decision-making. Unlike simulation-based methods, our approach maintains constant decision-making time regardless of problem size.
- We conduct an extensive analysis of our results, exploring the correlation between various factors and the decision to accept or refuse a patient, providing valuable insights into the decision-making process. Those analyses can help the healthcare provider to make proper policy to improve the quality of service in the future.

1 Introduction

Home Healthcare (HHC) is a service that involves patients receiving care from professional caregivers in the comfort of their own homes. This form of care offers numerous advantages to patients. Firstly, it allows them to remain in their familiar home environment, benefiting their mental well-being. Additionally, patients are less exposed to potential hazards commonly found in hospital settings. With an aging population putting immense pressure on the healthcare systems in many countries, HHC is an effective solution to lessen the burden on crowded hospitals and retirement facilities. Some countries, such as Sweden and Japan [1], are reforming their healthcare systems to integrate HHC services. The growing demands of HHC require a systematic approach to planning and scheduling. HHC scheduling can be viewed as a blend of assignment and vehicle routing problems. Given a set of patient requests, the problem consists of assigning patients to caregivers, deciding their visit schedules, and determining nurses' traveling routes. The complexity of this problem is compounded by a wide array of constraints that can vary from one HHC agency to another. These constraints typically encompass factors such as ensuring the continuity of care, considering the skill levels of available nurses, adhering to maximum working hours for nurses, and adhering to specific time windows for both nurses and patients, among others. In addition to this complexity, we face high uncertainty, including the stochastic arrival of patient requests. Given that there are often more requests than the available service capacity, HHC agencies frequently have to decline some patients.

In this paper, we study the Online Dynamic HHC Scheduling Problem (ODHHCSP), where an agency has to decide online whether to accept a patient request and how to schedule the patient in case of acceptance upon the patient's arrival. In the existing literature, most studies have primarily focused on the static version of the problem, where requests are scheduled in batches, and future information is not considered. Despite the advantage of batch scheduling, which provides more information for improved decision-making, most HHC agencies, in practice, adopt an online strategy, making decisions upon receiving each request. Various practical operational constraints drive this preference for online decision-making.

Within the literature, two approaches addressed the problem's online dynamic version. Bennett and Erera (2011) introduced a greedy heuristic that bases scheduling decisions on the cheapest insertion cost. This method remains myopic and does not consider information about future arrivals. Demirbilek et al. (2021) proposed a simulation-based approach that samples future scenarios to make decisions. However, this approach requires a long computation time, especially when the number of nurses is large. Consequently, it is unsuitable for real-time decision-making and lacks scalability when applied to larger scenarios.

We propose using a Reinforcement Learning (RL) approach for the ODHHCSP to fill this literature gap. RL learns to make decisions through a reward system. A RL agent interacts with an environment by taking action and then receiving a reward from the environment, which tells how good the action was. An RL agent can learn a policy to select a proper action based on the system state to maximize a cumulative reward through training.

As the training is done offline, an RL agent can make fast online decisions during execution, making it suitable for problems where the time budget for decision-making is limited.

We propose to model the problem as a Markov Decision Process (MDP) and use double-deep Q-Learning with experience replays as the RL algorithm. The experimental results show that our algorithm outperforms the existing methods in the literature. In addition, the approach is suitable for the problem's online setting as it can offer high-quality solutions in a short, constant run time (less than 0.001 seconds). We also provide analyses on the contribution of patients' characteristics and nurses' schedules to the decision outcome.

The rest of the paper is organized as follows. Section 2 presents the literature review on related problems. The problem description is provided in Section 3, and the solution methods are presented in Section 4. Section 5 reports and analyzes the experimental results, and finally, Section 6 closes the paper with conclusions and future works.

[1] https://www.emerald.com/insight/content/doi/10.1108/PAP-06-2020-0030/full/html

2 Literature review

The first attempt to address the home health care routing and scheduling problem (HHCRSP) can be dated back to 1997, when Begur et al. integrated a heuristic into a GIS system to develop a tool for solving the problem. The HHCRSP has been the subject of extensive research in numerous studies with different constraints and objective functions. We refer the readers to Fikar and Hirsch (2017) and Cissé et al. (2017) for comprehensive literature reviews on models and methods for this class of problem. More recently, Hecking et al. (2019) proposed a logic-based Benders decomposition approach for the HHCRSP problem to maximize the number of patients served over a given time horizon. The master problem was solved by Mixed Integer Linear Programming (MILP), while the sub-problems were tackled by Constraint Programming (CP). Grenouilleau et al. (2019) proposed a set partitioning formulation and a Large Neighborhood Search (LNS) framework to determine the assignment of home visits to a set of caregivers and the caregivers' routes over the given horizon to minimize penalties. They also took into account traffic delays using a time-dependent distance matrix. Grenouilleau et al. (2020) later extended the problem with a logic-based Bender Decomposition based on visit patterns and developed a metaheuristic based on LNS. They reduced the average computational time by 34% compared to Hecking et al. (2019).

While most studies focused on static problem settings where all patient requests are known a priori, few papers considered stochastic variants. Rodriguez et al. (2015) investigated a version with uncertainty in patient demands and solved a multi-objective problem using stochastic programming to find the best trade-offs between coverage on patient demand forecast and resource costs. Carello and Lanzarone (2014) considered a nurse-patient assignment problem to minimize nurses' overtime. The authors proposed a robust cardinality-constrained model and ran experiments on a 26-week rolling time horizon to investigate the impact of stochastic client demand. Nikzad et al. (2021) developed a matheuristic to solve multiple decision levels simultaneously, with uncertainty in terms of traveling and service time. Zhan et al. (2021) studied a home service routing and appointment scheduling problem with stochastic service time. The objective function is to minimize travel costs, the idling time of professional operators (nurses), and customers' waiting time. The problem was modeled as a MILP and solved using an L-shape method. A heuristic approach was also proposed to deal with large instances. However, the authors did not consider multiple visits and continuity of care.

Shi et al. (2019) proposed a robust optimization model for the home health care routing problem with consideration of uncertain travel and service times by reducing the problem to a deterministic version, which Gurobi, Simulated Annealing, Variable Neighborhood Search, and Tabu Search then solved. In Yang et al. (2018), a chance-constrained programming model was built with chance constraints on waiting time and overworking. The model is then solved by an Ant Colony Optimization meta-heuristic. Recently, Khorasanian et al. (2024) proposed a variant of HHCSRP that considers uncertainty in the number of required visits. In this problem, the number of visits for each patient is not predetermined. The necessity of the next visit is decided during the previous one: either one more visit is needed, or no further visits are required. The problem was conceptualized as a Markov Decision Process and solved using approximate linear programming techniques.

In this work, we focus on the dynamic setting of the HHCSP, where patient arrivals are revealed over time. In practice, many agencies do not have enough resources to accommodate all incoming patient requests. Therefore, they must decide whether to accept or reject a patient upon receiving a request. Very few studies considered the HHCSP in such a dynamic setting.

To the best of our knowledge, the first methods used to solve this problem were developed by Bennett and Erera (2011), in which the authors proposed two greedy insertion heuristics to maximize the number of patients accepted. The algorithms, however, were limited to instances with only a single nurse and were still myopic, i.e., they did not consider information on future requests.

Demirbilek et al. (2019) proposed a simulation-based approach for the same problem. Their algorithm, the Scenario-Based Approach (SBA), generated multiple future scenarios and made

decisions based on the simulation results. Demirbilek et al. (2021) extended the algorithm to tackle instances with multiple nurses. A weakness of their approach is that it requires multiple simulations to make a decision, which makes the algorithm unsuitable for online decisions and prevents it from scaling properly on larger instances. In this study, we propose a reinforcement learning approach to fill this methodological gap and solve more significant instances that arise often in practice. Unlike the SBA, our method trains the policy offline from simulations before we can use it online for decision-making. It provides adaptive actions for incoming requests with less computational effort, which is preferable in an online context.

Dynamic Vehicle Routing Problems

Since the literature on Dynamic Vehicle Routing Problem (DVRP) is vast, we suggest Soeffker et al. (2022) for the recent reviews. However, the number of works that applied RL to the DVRPs with stochastic customers is limited. Joe and Lau (2020) proposed a value-function approximation solution approach for the DVRP, in which they must update the planned routes after realizing a new customer. Their algorithm learns to estimate future rewards by Temporal Difference Learning and uses simulated annealing to optimize routes regarding immediate and predicted future rewards. Chen et al. (2022) used Deep Q-learning agents to decide customer admissions for trucks and drones in a same-usingy delivery context. The accepted customers are then assigned to vehicles by the cheapest heuristic. Basso et al. (2022) proposed a safe RL approach to solve the Dynamic Stochastic Electric VRP. The authors used a VFA-based algorithm to decide which next node to visit each time the vehicle leaves the customer's location. They considered two objective functions: minimizing energy consumption and the risk of battery depletion. Additionally, they proposed a second safety layer that uses energy cost predictions for the charging stations.

3 Problem definition

A HHC agency has a set of nurses $\mathcal{R}$ providing services for a set of patients $\mathcal{P}$. The studied problem is a variant of the classical HHCSP where patient requests arrive dynamically, and a decision must be made on whether to accept or reject a patient upon receiving a request. The objective is to maximize the number of patients served during a horizon.

Nurse

Each nurse $r \in \mathcal{R}$ has a working time window $[e_r, l_r]$, where e_r is the starting time of the working day when nurse r leaves home or office and l_r is the end of the working day, before which nurse r should return home or to office. All nurses work only during the weekdays, and no service is provided during the weekends. A nurse visits each patient at most once a day, provides services within a predefined duration, and then travels to the next one. Each nurse has a skill level, denoted as s_r. Some types of services require nurses at a certain skill level.

Patient

When a service request is received, a patient $p \in \mathcal{P}$ is registered to the system. The admission week of patient p is denoted as a_p. The requested care plan consists of an *episode of care* w_p (the number of consecutive weeks that the patient requires the service), a *visit frequency* f_p (the number of visits per week), a *visit duration* h_p, measured in minutes, and a minimum level skill of nurses s_p. The assigned nurse must have a skill level of s_p or higher. Each patient has a set of possible combinations of days to receive services, depending on their visit frequency. This combination is referred to as a *visit pattern*. No consecutive visits are allowed. The set of valid visit patterns for patient p is denoted as Ω_p. For example, if a patient p requires two visits per week, there are five valid visit patterns $\Omega_p = \{(Mon, Wed), (Mon, Thu), (Mon, Fri), (Tue, Thu), (Tue, Fri), (Wed, Fri)\}$. $\mathcal{T}_p$ is the set of possible visit times of a patient p in a day. In our problem settings, the visit time must be selected from a predefined set of equally-spaced appointment times of δ in minutes. For example, a typical patient has $\mathcal{T}_p = \{8:00, 8:15, ..., 16:00\}$ with $\delta = 15$. Some patients can restrict their visit times to fit their availability or preferences.

Other constraints

The agency must ensure the care continuity for accepted patients by assigning the same

nurse throughout their treatment. Furthermore, we should maintain consistent appointment times over treatment days, as well as ensure that patients receive their treatments on the same set of weekdays throughout their treatment period. The nurses' schedule must ensure they arrive at the customer locations before the appointment times. If they come before the visit time window, they must wait until the appointment to start the service. Figure 1 demonstrates a feasible weekly schedule for a single nurse with four patients.

Decision process

When a request is received, the agent must decide whether to accept or reject a patient. If the patient is accepted, several decisions need to be made: the days of the visit (visit pattern), the appointment time, and which nurse to assign to the patient. Thanks to the continuity constraints, we can define an insertion pattern $\lambda = \{\omega_\lambda, t_\lambda, r_\lambda\}$, where ω_λ is the visit pattern, t_λ is the appointment time, r_λ is the nurse's identification. Let p^-_{wd} and p^+_{wd} be the predecessor and the successor of p on the route of nurse r on day d of week w, respectively. The time taken by nurse r_λ to travel from p^-_{wd} to p, and from p to p^+_{wd} are denoted as $c(p^-_{wd}, p)$ and $c(p, p^+_{wd})$, respectively. An insertion pattern λ is considered feasible for a patient p if inserting p into the position defined by λ does not violate the traveling constraints (i.e., the nurse has sufficient time to travel from the preceding patient to p, finish the required service, and then travel to the succeeding one). Formally, an insertion pattern λ is valid for a patient p if it satisfies the following constraints:

$$e_{r_\lambda} + c(p^-_{wd}, p) \leq t_\lambda \leq l_{r_\lambda} - c(p, p^+_{wd}) \tag{1}$$

$$t_{p^-_{wd}} + h_{p^-_{wd}} + c(p^-_{wd}, p) \leq t_\lambda \quad \forall w \in \{a_p + 1, .., a_p + w_p\}, d \in \omega_\lambda \tag{2}$$

$$t_\lambda + h_p + c(p, p^+_{wd}) \leq t_{p^+_{wd}} \quad \forall w \in \{a_p + 1, .., a_p + w_p\}, d \in \omega_\lambda \tag{3}$$

$$s_{r_\lambda} \leq s_p \tag{4}$$

$$t_\lambda \in T_p \tag{5}$$

$$\omega_\lambda \in \Omega_p \tag{6}$$

Constraints (1) guarantee the working time window of nurse r_λ. The traveling constraints are enforced by inequalities (2) and (3). Constraint (4) ensures the requirement on the nurse's skill level. Finally, constraints (5) and (6) make sure the appointment time and visit pattern are valid for the given patient.

4 Methodology

In this section, we first present the cheapest insertion heuristics proposed in Bennett and Erera (2011) for solving the HHCSP with a single nurse, which also are the base components of our approach. The scenario-based approach (SBA) introduced in Demirbilek et al. (2019) is then re-described and used as the experiments' baseline algorithm. Finally, we provide the theoretical background of reinforcement learning (RL) and present our RL framework for solving the ODHHCSP.

4.1 Cheapest insertion heuristics and scenario-based approach

The cheapest insertion heuristics enumerate all possible insertion patterns and find the positions to insert the patient into the schedule to minimize the summation of increased costs. If there is at least one feasible spot for insertion, the patient is accepted and scheduled to the cheapest option. Otherwise, the request is rejected. The procedure is described in Algorithm 1. Given a request from patient p, nurse r is checked for sufficient skill level (Line 4). If the requirement is met, the algorithm iterates over feasible schedules of p, which include visit patterns and appointment times (Line 5 - Line 9). Then, time-related constraints for them are verified by checking the insertions on future routes (Line 11), and the total costs of these schedules are calculated (Line 12). Finally, the algorithm returns the position with the lowest cost (Line 25). The authors proposed two versions of the greedy heuristic. The distance-based heuristic (DH) minimizes the increase in a nurse's traveling distance. In contrast, the capacity-based heuristic (CH) maximizes the remaining capacity of a nurse (i.e., maximizing the nurse's number of remaining potential service slots).

The scenario-based approach (SBA) was proposed by Demirbilek et al. (2019) to solve the same problem. The SBA is based on simulating future scenarios to evaluate whether a request should be

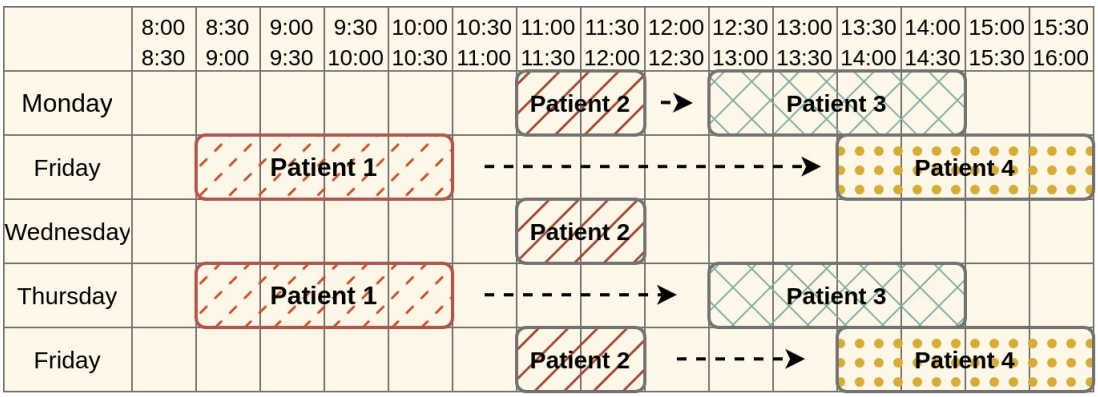


Fig. 1: An illustration on the schedule of a single nurse in a week.

Algorithm 1 Cheapest insertion heuristic (CIH)

Require: Patient p, nurse r

$c^* \leftarrow \infty$
$\lambda^* \leftarrow \{\emptyset, -1, -1\}$
$\Omega_p \leftarrow$ the list of valid visit patterns of patient p
if $s_p \leq s_r$ **then**
  for all visit pattern $\omega \in \Omega_p$ **do**
    for all visiting time $t \in \mathcal{T}_p$ **do**
      $c \leftarrow 0$
      for all week $w \in \{a_p + 1, .., a_p + w_p\}$ **do**
        for all day $d \in \omega$ **do**
          $\Gamma^r_{wd} \leftarrow$ planned route of r on day d, week w
          if inserting p into Γ^r_{wd} at time t does not violate travelling constraints of r **then**
            $c \leftarrow c + insertCost(\Gamma^r_{wd}, p, t)$
          else
            the insertion pattern $\{\omega, t, r\}$ is invalid
          end if
        end for
      end for
      if $\{\omega, t, r\}$ is valid and $c \leq c^*$ **then**
        $c^* \leftarrow c$
        $\lambda^* = \{\omega, t, r\}$
      end if
    end for
  end for
end if
return c^*, λ^*

accepted. A scenario consists of randomly generated patient requests for a week. Given a current request from patient p, the algorithm generates a set of future scenarios. Each scenario is solved using the cheapest insertion heuristic described in Algorithm 1. If patient p is accepted in at least one scenario, the patient is scheduled to the most recurrent accepted option. Otherwise, the patient is rejected. The approach is then extended for multiple nurses in Demirbilek et al. (2021).

4.2 Reinforcement learning

The framework of RL provides a mechanism for learning to take proper actions to gain maximum rewards over time. An RL agent learns by interacting with an environment, modeled as a Markov Decision Process (MDP). At every time step t, the agent observes an environment state S_t, selects and performs an action A_t, then collects a reward R_{t+1} and observes a new state S_{t+1} from the environment. The behavior of an

agent is defined by a policy (π) that maps states into actions. Based on past experiences, the goal of an agent is to learn an optimal policy that maximizes the discounted summation of future rewards $G_t = \sum_{k=t}^{\infty} \gamma^{k-t} R_{k+1}$, where γ is a discount factor. Numerous algorithms for RL are available in the literature. In value-based RL algorithms, given a policy π, the agent learns a state-action value function (Q-function) $Q^{\pi}(s,a) = \mathbb{E}[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a]$ that estimates the expected cumulative reward at state s if action a is taken. Q-learning proposed by Watkins (1989) is a value-based RL algorithm. It updates an estimated Q-function iteratively by sampling based on the Bellman equation:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_t + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$$

Once learned, the optimal policy $Q^*(s,a)$ can be derived by selecting the action that results in the best cumulative future reward $a_{\pi}(s) = \arg\max_a(Q_{\pi}(s,a))$. In Q-learning, it is important to control the trade-off between exploration (choosing an action randomly) and exploitation (selecting the best action based on the current estimated Q-function). For such purpose, we employ the classical ϵ-greedy heuristic, which chooses an action randomly with probability ϵ, and greedily with probability $1-\epsilon$, $0 \leq \epsilon \leq 1$.

In the initial stages of training, when data gathering and exploration are essential, ϵ is often set high, such as 1. As learning progresses, ϵ is gradually reduced to a lower bound, such as 0.1. This approach allows the agent to increasingly focus on exploiting the learned strategies while still allowing for a minimal level of exploration to avoid local optima.

A parameterized function approximator is often utilized instead of tabular methods to learn the Q function in applications with a large state or action space. Deep Q-network (DQN)(Mnih et al., 2013) uses a neural network as a nonlinear function approximator of the Q-function. The Q-function approximator is denoted as $Q(s,a,\theta)$, where θ is the parameters of the neural network. To estimate Q-values, the Q-network learns to minimize the mean square error (MSE) between the updated target $y_t^{DQN} = r_t + \gamma \max_{a'} Q(S_{t+1}, a')$ and the prediction $Q(S_t, A_t, \theta)$. A replay buffer stores the agent's experience to improve data efficiency by breaking correlations between consecutive samples. The Q-network is trained by sampling mini-batches uniformly from the replay buffer. It is well-known that DQN often overestimates the action-state values, which may prevent the agent from learning the optimal policy. This phenomenon can be caused by the fact that using the same network for selecting and evaluating actions is likely to choose the overestimated value, which can maximize the bias of the Q-network over time. To tackle this issue, Hasselt et al. (2016) proposed Double DQN (DDQN), which uses two separate networks for selecting and evaluating actions. Similarly to DQN, DDQN uses a neural network θ to select actions. An additional target network $\hat{\theta}$ estimates Q-values. The target is updated using the function:

$$y_t^{DDQN} = R_t + \gamma Q(S_{t+1}, \arg\max_{a'} Q(S_{t+1}, a', \theta), \hat{\theta})$$

$\hat{\theta}$ is updated periodically by copying the weights from θ.

4.3 Sequential decision process for the ODHHCSP

As mentioned above, we model the ODHHCSP as a MDP and solve it via RL approach. We now describe the main components of our MDP: the state space, the action space, the transition, and the reward function.

Decision point

In our problem setting, a time step t is triggered when the agency receives a patient's request. The corresponding patient is denoted as p_t.

State space

A state space consists of the necessary information for the agent to decide. The system state S_t at time step t consists of two main components: (1) information on the corresponding request and (2) information on the current schedules of nurses. The representation of the state space is presented in more detail in Section 4.4.

Action space and transition

At time step t, the agent faces $|\mathcal{R}| + 1$ potential actions: accepting and assigning the patient to a specific nurse $r \in \mathcal{R}$ or rejecting the request. If the patient is rejected, no further action is required. However, if a patient is accepted and assigned to a nurse, denoted as r_t, the algorithm proceeds to update the schedule of nurse r_t using the cheapest insertion heuristic (Algorithm 1). The assignment uses the distance-based insertion heuristic on Γ_{r_t}.

Following the processing of the t^{th} request, the system transitions to a *post-decision* state denoted as $\hat{S}_t = \{\emptyset, \hat{\Gamma}_t\}$.

Here, $\hat{\Gamma}_t$ represents the updated set of routes based on A_t. Furthermore, the model advances from $\hat{S}_t$ to the subsequent pre-decision state S_{t+1} when the system realizes the arrival of the $(t+1)^{\text{th}}$ request.

Reward function

When a patient is accepted and assigned to a valid nurse, a reward equal to 1 is returned. Conversely, the reward is set to 0 if the patient is rejected. To discourage the agent from choosing invalid actions, the environment penalizes the agent with a negative reward of $R_{invalid}$ whenever an invalid assignment is observed. Our implementation uses $R_{invalid} = -10$.

$$R_t = \begin{cases} 0, & \text{if } p_t \text{ is rejected.} \\ 1, & \text{if } p_t \text{ is assigned to a valid nurse.} \\ -10, & \text{if } p_t \text{ is assigned to an invalid nurse.} \end{cases}$$

4.4 State representation

To make proper scheduling decisions, the RL agent needs access to relevant information on the incoming request and the current nurses' schedules, referred to as an *observation*. We select a subset of valuable characteristics of the system's state as a compact feature vector. The feature vector consists of the following components:

Incoming patient: The first part of the observation is the information on the incoming request. For an incoming patient p, the episode of care w_p, visit frequency f_p, and visit duration h_p are required. In addition, we compute the cost of assigning the patient to each nurse using the cheapest insertion heuristic. The size of this feature is hence $(|\mathcal{R}| + 3)$.

Nurses' schedules: To capture the nurses' availability, we include the number of assigned visits, total idle time, and total traveling time in the next $\bar{w}$ weeks for each nurse, where $\bar{w}$ is the maximum episode of cares allowed in a problem setting. This component comprises $(3|\mathcal{R}|)$ numbers. An illustration of the idle and traveling time during a nurse's working day can be found in Figure **??**.

Additional information: The third component consists of other relevant information, including:

- A binary vector indicating which nurses are valid and available for the current request ($|\mathcal{R}|$ variables, one for each nurse).
- A binary variable implying whether the current state is post-decision.
- A binary variable indicating if there is at least one nurse available for the current request.
- A variable representing the time left in the week, measured in minutes from the decision point until the finish.
- For each nurse $r \in \mathcal{R}$, we calculate the minimum reduction in the number of consecutive appointments in the best scenario when p is inserted into r's planned routes. The number of available slots in the best scenario within an interval $[l, e]$ between two corresponding patients p_l and p_e is given by the equation from Bennett and Erera (2011):

$$nb_{best}(l, e) = \min(\lfloor \frac{e - l - \overline{w}_p - \delta}{\overline{w}_p + \delta} \rfloor, \lfloor \frac{e - l - \delta \lceil \frac{c(p_l^+, p_e^-)}{\delta} \rceil}{\overline{w}_p} - 1 \rfloor)$$

The reduction then can be calculated by searching for the cheapest insertion pattern. Then, we compute the total reduction over all days in the pattern. We need to use $|\mathcal{R}|$ variables, one corresponding to each nurse, to record this total reduction.

In our implementation, with the post-decision state, the value of the feature vectors related to the patient's information is set to zero. Each feature value is extracted from the state and is normalized to the [0, 1] range before being passed to the agent. The feature vector's total length is $6|\mathcal{R}| + 6$.

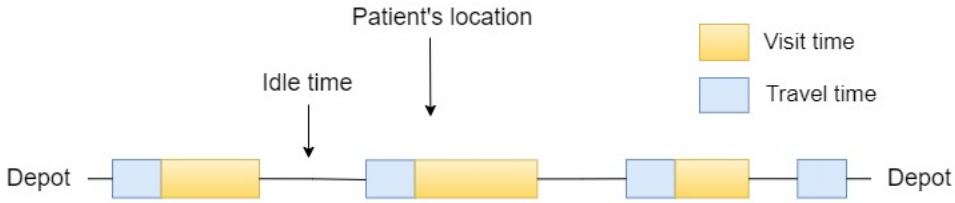


Fig. 2: The timeline of a nurse's working day.

4.5 Scheduling a patient

When a request of a patient is received, the algorithm makes a decision using the procedure outlined in Algorithm 2. Initially, the algorithm examines the current state of the schedule to determine if it can feasibly accommodate the patient's request, respecting all constraints (Line 1). If no slot is found, the request is marked as invalid. It is therefore denied [2]. In contrast, when a slot is available, a trained RL model determines whether to accept or reject the patient. To this end, the algorithm extracts a feature vector from the current state (Line 3). This vector is used in the RL model to estimate the action value (Line 4). As such, an action is derived (Line 5) to decide whether the request is accepted or rejected. If accepted, it also determines which nurse is assigned to the patient. When the request is approved, the patient is assigned to the corresponding nurse, and the schedule is updated using Algorithm 1.

[2]It is worth noting that during the training phase, we consider invalid requests as they allow the agent to observe environmental changes. However, during the deployment, it is not required to consider invalid requests.

Algorithm 2 Decision method

Require: Patient p_t
1: $S_t \leftarrow$ observe the current state
2: **if** There is an available slot for accepting p_t **then**
3: $ft \leftarrow$ Extract feature from S_t
4: $Q(S_t) \leftarrow$ Estimate value of ft from the neutral network
5: $A_t \leftarrow \arg\max_{a'}(Q(S_t, a'))$ ▷ Derive action
6: **if** A_t is accepted **then**
7: $r_t \leftarrow$ nurse's ID from A_t
8: Assign patient p_t to r_t and determine the schedule using Algorithm 1
9: **else**
10: Reject patient p_t
11: **end if**
12: **else**
13: Reject patient p_t
14: **end if**

5 Experimental results

In this section, we present the results of our computational experiment. First, we discuss the generation of test instances and the experimental setup in Section 5.1. Then, we compare the performance of the benchmarks with that of the RL-based algorithm in Section 5.2. An analysis of the algorithm is presented in Section 5.3. Experiments were conducted on an AMD Ryzen 7 3700X @3.600GHz CPU running Pop!OS 22.04. The algorithms were implemented in Python 3.10.

5.1 Data generation

All instance settings are adopted from Bennett and Erera (2011), which are generated to mimic real data. In the following, we describe the components of the test data in more detail.

Nurses

All nurses work from 08:00 to 16:30, five days a week, from Monday through Friday. A nurse can

have a skill level of 1, 2, or 3. The higher the level, the more tasks the nurse can perform.

Patients

All patients have the same set of possible appointment times with intervals of 15 minutes: $\mathcal{T} = \{08:00, 08:15, 08:30, ..., 16:30\}$. The episode of care for each patient lasts four weeks. A care plan can have a frequency of 1, 2, or 3 visits per week with probabilities of 0.05, 0.35, and 0.60, respectively. Each visit lasts 1 hour. Visits on consecutive days within a week are prohibited (e.g., a visit pattern of (Monday, Tuesday) is forbidden).

Locations

The patient locations are generated within a squared geographic region represented by a 60x60 grid (units in kilometers). The traveling time between locations is 3 minutes per one grid unit. The coordinates of the home depot for nurses are generated randomly. Three location distributions are considered: *Uniform* (U), *Clustered* (C), and *Uniform-Clustered* (UC).

In U instances, the patient locations are distributed uniformly over the entire geographic region. In C instances, they are sampled from three square sub-regions in the considered region. The sub-regions are defined by the coordinates of two opposite corners of the corresponding squares: [(8, 20), (18, 30)]; [(52, 64), (67, 79)]; and [(43, 27), (63, 47)]. In UC instances, 30% of the requests come from these sub-regions, while the remaining patients are drawn randomly from the entire region.

Patient arrivals

Inter-arrival times between patient requests are modeled with an exponential distribution. The rate of this distribution determines the average number of patient requests per time unit, and requests can arrive at any point during the 8-hour working day. The selected arrival rate of each instance setting is influenced by the number of nurses available: more nurses might correlate with a capacity for a higher request rate. We choose specified arrival rates for each setting related to the number of nurses to represent low, medium, and high-demand scenarios. Based on the number of nurses, a detailed list of these arrival rates can be found in Table 1. For the purpose of this paper, we generated patient requests in the simulations over ten weeks.

Parameters		Inter-arrival	Avg. patient
# Nurses	Demand	time (minutes)	per day
3	low	640	0.80
	medium	300	1.70
	high	120	4.25
6	low	320	1.59
	medium	150	3.40
	high	60	8.50
12	low	160	3.19
	high	75	6.80
15	low	128	3.98
	high	60	8.5

Table 1: Inter-arrival time and average number of patients for each instance setting

Generating initial schedules

A warm-up period is utilized to generate initial schedules. The process begins with an empty calendar, and the first three weeks of patient requests are scheduled using the distance-based cheapest insertion heuristic (Section 4.1).

5.2 Comparison between the RL approach and the baselines

We re-implemented the scenario-based approaches (SBA) proposed in Demirbilek et al. (2021) and both versions of the cheapest insertion heuristics in Bennett and Erera (2011): the distance-based (DH) and capacity-based (CH); and used them as baselines. The SBA simulated over 75 scenarios for each patient request to make decisions.

Our RL approach utilized a feed-forward neural network comprising two hidden layers. Each layer contained $60 \cdot |R|$ neurons. The ReLU activation function was used for function approximation. The agent's exploration was guided by the parameter ϵ, which decayed from 1 to 0.1 throughout the initial 20% of the time steps. We sampled a batch of 2048 transitions from the replay buffer at each iteration to train the Q-network. Every 100 time steps, we performed gradient descent to update the Q-network's parameters, using a step size of $\alpha = 0.01 \cdot 0.999^{10000 \cdot \#time\_steps/H}$, where H is the total number of time steps of the training phase. The target network receives updates every 5000 steps. We set the discount factor γ to 0.998.

We trained the models in $4 \cdot 10^7$ time steps based on preliminary experiments to ensure convergence. For every $2 \cdot 10^4$ time step, we evaluated the model using the same testing set of 3 instances. We plotted the training curves and, for comparison, included results from the testing set produced by the baseline methods. We trained an RL model for each instance set, which varied by the number of nurses, arrival rates, and patient location distribution. For each instance size (the number of nurses), we tested three demand levels (high, medium, and low) and three location distributions (uniform, clustered, and uniform-clustered). An example of the training curves for the instance settings with three nurses is shown in Figure 3 where the x-axis represents the training epoch and the y-axis indicates the total number of accepted patients on the validation set. As the training phase progresses, we notice that the solution quality significantly improves until reaching 10^7 trained time steps. After around $2 \cdot 10^7$ time steps, the validated value reaches stability.

To provide a detailed examination of the algorithms' performance, we conducted additional tests on the trained RL model and the four baseline methods (DH, CH, SBA + DH, and SBA + CH) over 30 instances. The results are reported in Table 2. For each instance, we computed the total number of accepted patients ($\#patients$) over the whole planning horizon. For better readability, we use the number of accepted patients provided by DH as the reference baseline. For the other approaches, we calculated the improvement (in percentage) on the total number of accepted patients compared to DH as follows:
$imp_A(\%) = 100 \times \frac{\#patients_A - \#patients_{DH}}{\#patients_{DH}}$, where $\#patients_A$ is the number of accepted patients provided by approach A.

The results in the table represent averages over the 30 instances. Given the considerably slower speed of the SBA-based methods to make decisions, we report their running times (in seconds, denoted as $time$). We do not provide in details the computation time of other approaches, including the greedy heuristics (DH, CH) and RL-based ones, because they have a negligible running time, less than 1 second per request.

The first observation from Table 2 is that CH generally underperforms compared to DH in both low and medium-demand scenarios. However, in high-demand scenarios, CH tends to exhibit slight improvements over DH. Consequently, SBA based on DH also outperforms the one based on CH. SBA+DH consistently improves upon the results from DH across all tested instances, with the most significant improvements noted in higher-demand scenarios. This can be attributed to low-demand environments, where the benefit of rejecting a client is minimal. Yet, the decision-making time per patient for SBA+DH is considerably long, especially in high-demand scenarios. SBA+CH takes up to 20 seconds to decide for a single patient in high-demand cases. In contrast, the RL approach consistently outperforms the other methods in both solution quality and computational speed. RL delivers the highest improvement across almost all instance settings, with one exception: the high-demand scenario with cluster distribution. In this particular setting, while the improvement percentages for RL and SBA+DH over DH are closely matched (2.49% vs. 2.62%), RL has a constant runntime and is significantly faster than SBA+DH.

To examine the algorithms' performance on the larger instances, we also tested them on instance settings with 6, 12, and 15 nurses. The results are reported in Tables 3, 4, and 5, respectively. As can be seen in the obtained results, in settings with 6 nurses, our RL approach achieves the best performance among all tested methods in low and medium-demand scenarios. However, in high-demand situations, the SBA+CH approach outperforms the RL approach regarding solution quality. Nevertheless, it is important to note that the SBA+CH approach requires significantly more time to make decisions. In high-demand UC instances, SBA+CH takes up to 113 seconds to make decision for a single patient. It is five times higher than that required in a similar setting with 3 nurses, highlighting that SBA-based algorithms are not scalable for larger scenarios. In contrast, while the improvement provided by RL is slightly lower than that of SBA+CH, the RL approach is significantly faster than SBA+CH, making it more suitable for larger instances.

In the case of settings with 12 nurses, we conducted tests using algorithms for both low and high-demand scenarios with a uniform patient distribution (U). The results, as presented in Table 4, show that the RL approach outperforms other methods in low-demand scenarios, exhibiting an improvement of 3.57% over DH.

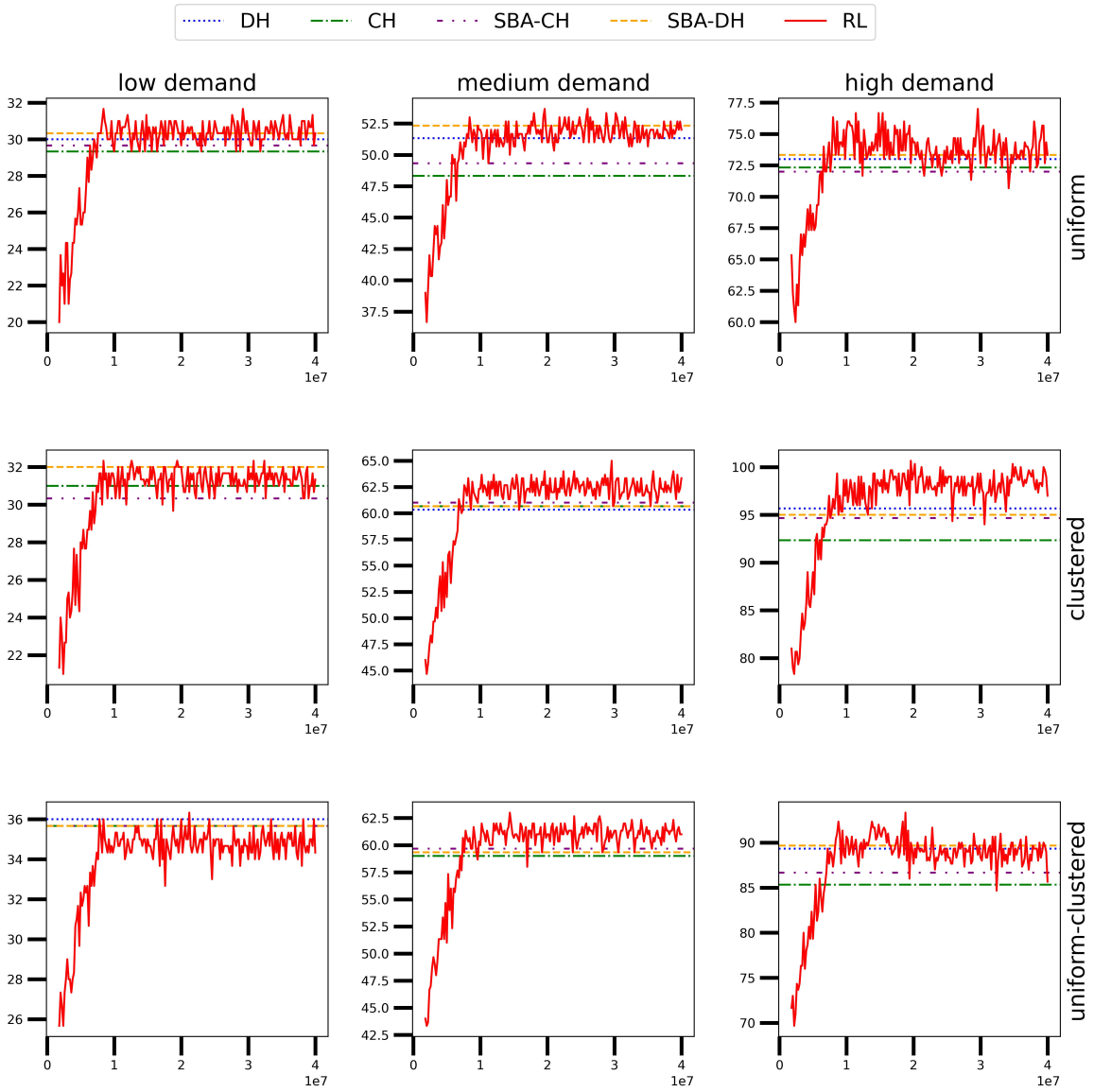


Fig. 3: Learning curves for instance settings with three nurses.

		DH	CH		SBA+DH			SBA+CH			RL	
		# patients	# patients	imp (%)	# patients	imp (%)	time	# patients	imp (%)	time	# patients	imp (%)
U	low	30.70	30.17	-1.57	31.00	0.95	2.59	30.40	-0.81	4.97	31.50	**2.68**
	medium	51.97	50.47	-2.71	52.57	1.25	4.10	50.63	-2.38	7.84	53.33	**2.73**
	high	74.90	75.37	0.82	76.77	2.56	10.51	76.40	2.19	17.85	77.30	**3.22**
C	low	36.23	36.60	1.27	36.63	1.21	2.81	36.70	1.50	5.83	37.07	**2.43**
	medium	61.13	61.17	0.18	61.60	0.72	4.95	61.57	0.86	10.39	62.43	**2.12**
	high	94.63	92.97	-1.61	97.10	**2.62**	7.28	94.03	-0.53	15.42	96.97	2.49
UC	low	34.33	34.13	-0.35	34.43	0.24	2.76	34.07	-0.57	5.77	35.07	**2.10**
	medium	59.40	57.73	-2.62	59.67	0.50	4.70	58.47	-1.41	9.43	60.20	**1.47**
	high	88.07	88.20	0.33	90.50	2.85	9.72	88.50	0.67	19.85	91.40	**3.92**

Table 2: The total number of accepted patients for instance settings with 3 nurses.

		DH	CH		SBA+DH			SBA+CH			RL	
		# patients	# patients	imp (%)	# patients	imp (%)	time	# patients	imp (%)	time	# patients	imp (%)
U	low	67.40	66.43	-1.23	67.97	0.94	8.93	66.70	-0.87	18.73	69.23	**2.79**
	medium	112.20	109.63	-2.24	113.13	0.85	14.22	110.70	-1.28	28.77	114.30	**1.92**
	high	168.43	169.73	0.86	173.13	**2.81**	60.72	172.03	2.21	109.42	170.97	1.52
C	low	75.53	75.63	0.19	75.90	0.47	10.00	75.67	0.23	25.14	77.87	**3.11**
	medium	131.50	130.63	-0.64	132.53	0.81	18.11	131.53	0.05	42.02	135.23	**2.88**
	high	199.13	199.17	0.08	206.90	**3.94**	68.09	201.30	1.16	145.40	203.87	2.41
UC	low	71.57	70.87	-0.81	71.83	0.37	9.78	71.37	-0.19	23.29	73.20	**2.27**
	medium	126.10	124.43	-1.26	128.13	1.64	16.17	125.13	-0.68	35.38	129.83	**3.02**
	high	185.67	187.47	1.12	192.70	**3.88**	55.42	190.10	2.56	113.55	189.60	2.20

Table 3: The total number of accepted patients for instance settings with 6 nurses.

		DH	CH		SBA+DH			SBA+CH			RL	
		# patients	# patients	imp (%)	# patients	imp (%)	time	# patients	imp (%)	time	# patients	imp (%)
U	low	136.60	135.80	-0.57	137.87	0.94	50.84	137.40	0.62	126.15	141.43	**3.57**
	high	240.13	237.87	-0.92	243.30	**1.31**	102.27	239.40	-0.27	199.74	242.43	0.97

Table 4: The total number of accepted patients for instance settings with 12 nurses.

In comparison, SBA+DH and SBA+CH show improvements of only 0.94% and 0.62%, respectively. In high-demand scenarios, the SBA+CH approach stands out as the top performer, achieving a notable improvement of 1.31% over DH. However, it requires on average, more than 100 seconds to make decision for a single request.

As can be seen in Table 5 representing the result of 15-nurse instances, SBA+CH and SBA+DH achieve the best performance on low- and high-demand instances, respectively. The reinforcement learning agent outperforms DH and CH. In these cases, the running time of the SBA-based approaches rapidly increases, taking more than 600 seconds to solve the problem in the high-demand setting, which is three times longer than the time taken to solve instances with 12 nurses. This shows the disadvantage of the reference approaches in terms of scalability.

In summary, the experimental results across four different instance sizes consistently demonstrate that the RL approach provides high-quality solutions with rapid running times, making it well-suited for real-time decision-making in problems like the one under consideration.

5.3 The impact of patients' characteristics on decision outcomes

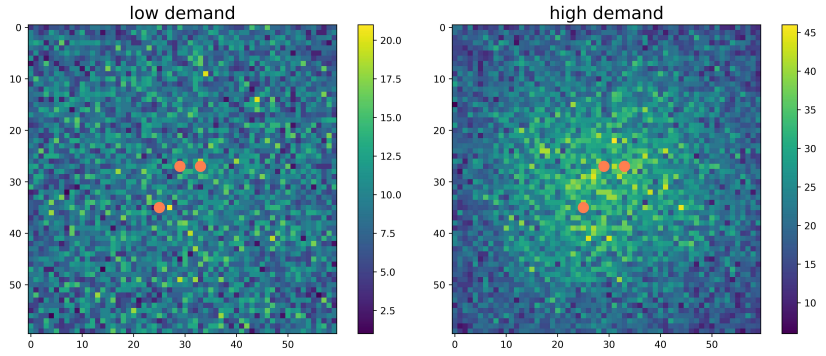
In this section, we analyze the impact of patients' characteristics on the RL agent's decision. Our initial analysis focuses on the correlation between patient locations and the acceptance rate. As illustrated in Figure 4, we present heat maps depicting a total number of accepted patients based on their respective locations across multiple simulations for instance settings with 3 nurses. The locations of nurses' home depots are highlighted with orange points for reference.

We begin by examining instances with a uniform patient distribution. In the low-demand setting, the acceptance rate is relatively consistent across the geographic region. In contrast, the high-demand scenarios result in a more concentrated acceptance rate, with the highest acceptance rate observed in the vicinity of the depot. This can be explained by the fact that in the low-demand setting, nurses have more availability, and most requests can be accepted. Conversely, in a high-demand setting, the agency has to reject some requests, and those requests closer to the depot are more likely to be accepted due to shorter travel times for nurses.

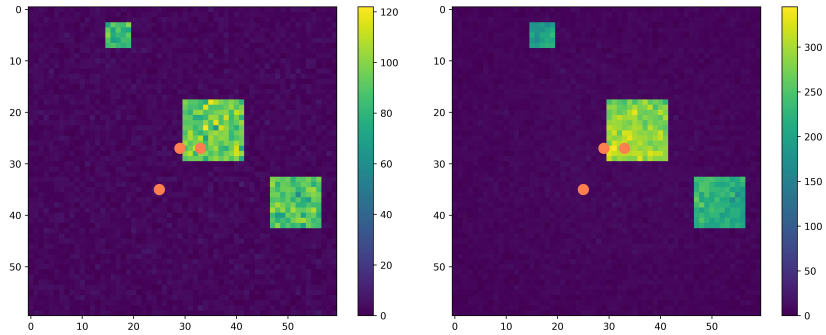
In simulations featuring a uniform clustered distribution setting, it is evident that patients within the clusters have a higher acceptance rate

		DH	CH		SBA+DH			SBA+CH			RL	
		# patients	# patients	imp (%)	# patients	imp (%)	time	# patients	imp (%)	time	# patients	imp (%)
U	low	156.73	157.67	0.60	158.07	0.85	59.20	159.13	**1.53**	158.56	158.93	1.40
	high	248.83	248.90	0.03	254.50	**2.28**	286.77	252.80	1.60	634.94	249.03	0.08

Table 5: The total number of accepted patients for instance settings with 15 nurses.



(a) Uniform distribution



(b) Uniform - Clustered distribution

Fig. 4: Heat maps of patient acceptance by locations over 1000 simulations for 3-nurse, low-demand (left), and high-demand (right) instances.

.

than those outside the clusters. This can be attributed to patients located in densely populated neighborhoods benefiting from shorter travel times for nurses providing services to their neighbors, resulting in a higher acceptance rate. Furthermore, the cluster closest to the depot has a higher acceptance rate than the more distant clusters. Similar to the uniform distribution, the acceptance rate is more evenly distributed in low-demand scenarios compared to high-demand scenarios, which exhibit a higher acceptance rate in the vicinity of the depot. These analyses highlight that the relative distance between nurses' locations and patients' locations plays a crucial role in determining the acceptance rate. This suggests that healthcare agencies should consider hiring nurses in more distant locations to increase the chances of patients in remote areas being served.

We then analyze the relationships between the total service time, the cheapest insertion cost, and the acceptance rate. The calculation of the total service time for patient p is the product of their episode of care w_p, visit frequency f_p, and visit duration h_p. A scatter plot representing the correlations between the total service time required,

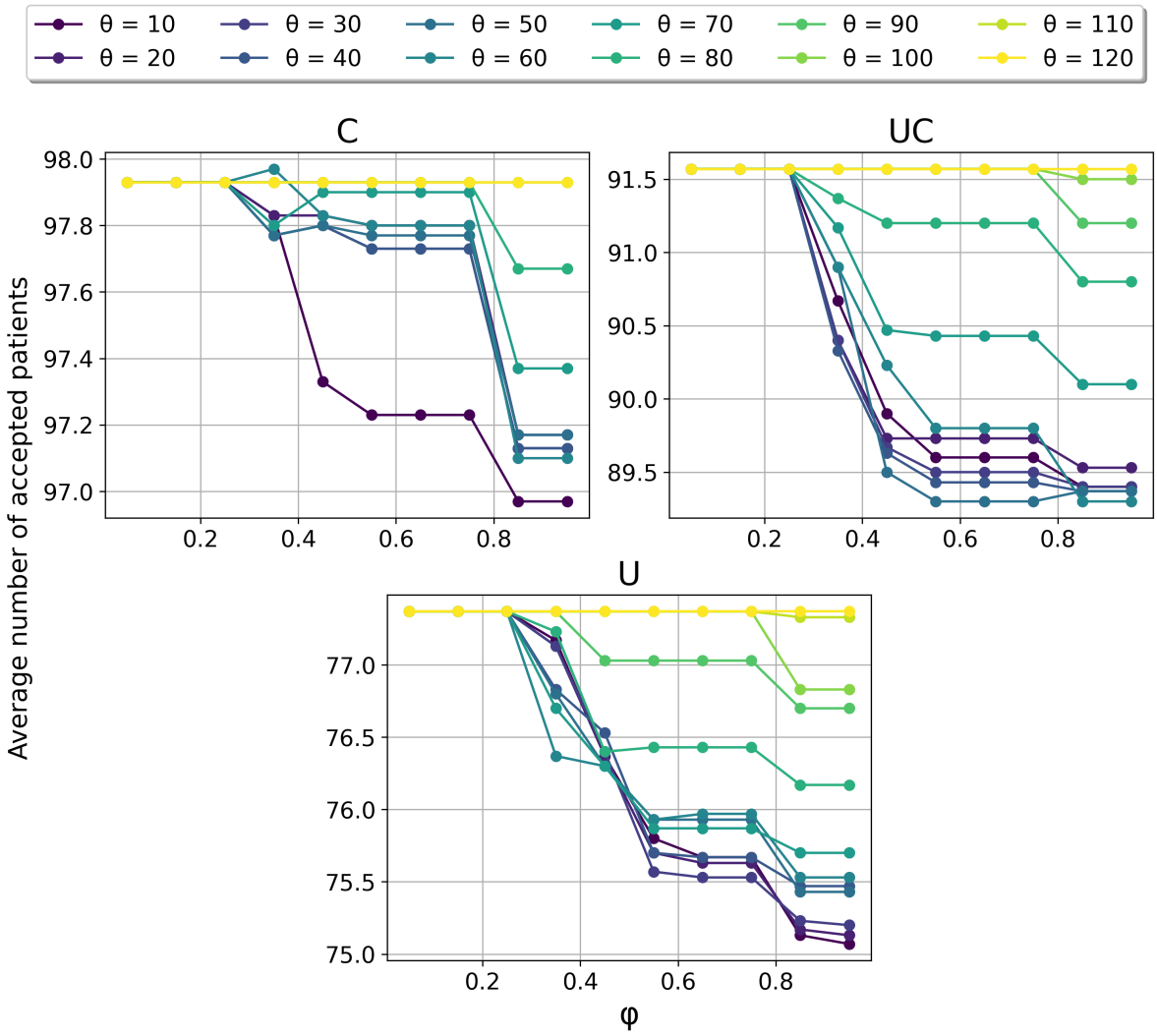


Fig. 5: The impact of varying acceptance probabilities (ϕ) and distance thresholds (θ) on the total number of accepted patients in a high-demand scenario involving three nurses.

the cheapest insertion cost, and the acceptance rate on the 6-nurse, high-demand, and clustered instances can be found in Figure 6. The plot makes it evident that patients with longer service time requests face a higher likelihood of being rejected. This correlation is intuitive, as patients with longer service time requirements consume more resources. The required service times can be categorized into three groups: 4 hours, 8 hours, and 12 hours. Most patients requesting 4 hours of service are accepted. For patients requesting 8 hours of service, the deciding factor becomes the cheapest insertion cost, with patients having a lower cheapest insertion cost being more likely to be accepted. For patients asking for 12 hours of service, the majority are rejected, except when their cheapest insertion cost is exceptionally low.

5.4 Addressing fairness for distant patients: an alternative policy

As highlighted in the previous analysis, our standard policy tends to reject patients located far from nurse depots to optimize decision-making efficiency. We now introduce an alternative policy designed to enhance the fairness and ensure

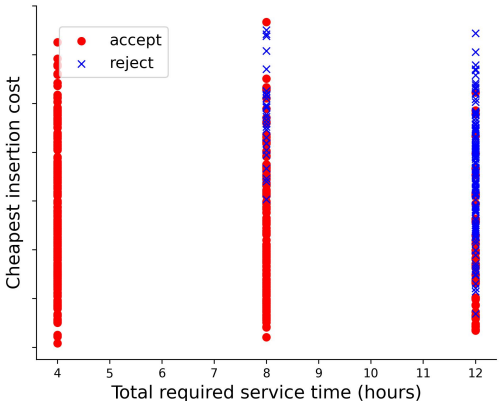


Fig. 6: The correlations between the total service time, the cheapest insertion cost, and the decision outcome on 6-nurse, high-demand, and clustered instances.

broader spatial coverage. This policy modifies our existing framework to allow for the acceptance of distant patients based on certain probabilities. We define two additional parameters: θ, the minimum distance from the patient to the nearest nurse depot, identifying a patient as distant; and ϕ, the probability of accepting a distant patient.

The policy is defined by adding a decision layer to our standard policy. When a feasible patient arrives, at first the standard policy on current state is run. If the decision is to reject the arriving patient, we check the distance from the patient to the nearest nurse depot. If the distance is greater than or equal to θ, we classify the patient as distant and assign them a probability ϕ of being accepted and scheduled using the cheapest insertion heuristic.

We conducted simulations with various settings of θ and ϕ in a high-demand scenario involving three nurses. The impact of the probability ϕ on the average number of accepted patients over 30 simulations for each setting is illustrated in Figure 5. As can be seen from the figure, the acceptance rate of patients begins to decline at a ϕ value of 0.25 and continues to decrease significantly as ϕ increases. For $\theta \geq 70$, significant changes in the policy only occur at higher values of ϕ. This is because the distant patients are rarely feasible for assignment and thus are few in number. Conversely, lowering θ results in greater degradation in policy performance, affected by the increase in the number of patients considered distant. These patterns are consistent across all tested scenarios: Uniform, Clustered, and Uniform-Clustered.

The results demonstrate a clear trade-off between accommodating distant patients and the total capacity of a home healthcare center, constrained by limited resources. These findings provide valuable empirical insights, helping policymakers adjust their strategies to achieve greater equity in patient care.

6 Conclusions

In this paper, we investigated the Online Dynamic Home Health Care Scheduling Problem (ODHHCSP), where healthcare agencies face real-time decisions regarding patient acceptance upon receiving requests. These decisions involve complex scheduling and routing tasks, all with the overarching goal of maximizing the number of patients served. We formulated this challenging problem as a Markov Decision Process (MDP) and applied a reinforcement learning (RL) approach. Our experimental findings demonstrate that the RL approach significantly improves the number of patients served compared to existing approaches in literature while maintaining a notably shorter computational runtime. This efficiency makes our approach well-suited for real-time decision-making scenarios. We conducted thorough analyses to explore the relationships between patients' characteristics and decision outcomes. These analyses provide valuable recommendations for healthcare agencies to enhance service quality. For future research, we aim to diversify our objectives beyond maximizing the patient acceptance rate.

For instance, we intend to minimize nurses' average travel time per visit and balance workloads among nurses. Finally, we recognize the importance of considering stochastic factors, such as variable travel times and nurses' availability, in problem formulation to enhance the realism of our model.

References

R. Basso, B. Kulcsár, I. Sanchez-Diaz, X. Qu (2022) Dynamic stochastic electric vehicle routing with safe reinforcement learning. Transportation Research Part E: Logistics and Transportation Review 157:102496

S.V. Begur, D.M. Miller, J.R. Weaver (1997) An integrated spatial dss for scheduling and routing home-health-care nurses. Interfaces 27(4):35–48

A.R. Bennett, A.L. Erera (2011) Dynamic periodic fixed appointment scheduling for home health. IIE Transactions on Healthcare Systems Engineering 1(1):6–19

G. Carello, E. Lanzarone (2014) A cardinality-constrained robust model for the assignment problem in home care services. European Journal of Operational Research 236(2):748–762

X. Chen, M.W. Ulmer, B.W. Thomas (2022) Deep q-learning for same-day delivery with vehicles and drones. European Journal of Operational Research 298(3):939–952

M. Cissé, S. Yalçındağ, Y. Kergosien, E. Şahin, C. Lenté, A. Matta (2017) Or problems related to home health care: A review of relevant routing and scheduling problems. Operations Research for Health Care 13:1–22

M. Demirbilek, J. Branke, A. Strauss (2019) Dynamically accepting and scheduling patients for home healthcare. Health Care Management Science 22(1):140–155

M. Demirbilek, J. Branke, A.K. Strauss (2021) Home healthcare routing and scheduling of multiple nurses in a dynamic environment. Flexible Services and Manufacturing Journal 33(1):253–280

C. Fikar, P. Hirsch (2017) Home health care routing and scheduling: A review. Computers & Operations Research 77:86–95

F. Grenouilleau, A. Legrain, N. Lahrichi, L.M. Rousseau (2019) A set partitioning heuristic for the home health care routing and scheduling problem. European Journal of Operational Research 275(1):295–303

F. Grenouilleau, N. Lahrichi, L.M. Rousseau (2020) New decomposition methods for home care scheduling with predefined visits. Computers & Operations Research 115:104855

H.v. Hasselt, A. Guez, D. Silver (2016) Deep reinforcement learning with double q-learning. In: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence. AAAI Press, AAAI'16, p 2094–2100

A. Heching, J.N. Hooker, R. Kimura (2019) A logic-based benders approach to home healthcare delivery. Transportation Science 53(2):510–522

W. Joe, H.C. Lau (2020) Deep reinforcement learning approach to solve dynamic vehicle routing problem with stochastic customers. In: Proceedings of the International Conference on Automated Planning and Scheduling, pp 394–402

D. Khorasanian, J. Patrick, A. Sauré (2024) Dynamic home care routing and scheduling with uncertain number of visits per referral. Transportation Science

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M.A. Riedmiller (2013) Playing atari with deep reinforcement learning. ArXiv abs/1312.5602

E. Nikzad, M. Bashiri, B. Abbasi (2021) A matheuristic algorithm for stochastic home health care planning. European Journal of Operational Research 288(3):753–774

C. Rodriguez, T. Garaix, X. Xie, V. Augusto (2015) Staff dimensioning in homecare services with uncertain demands. International Journal of Production Research 53(24):7396–7410

Y. Shi, T. Boudouh, O. Grunder (2019) A robust optimization for a home health care routing and scheduling problem with consideration of uncertain travel and service times. Transportation Research Part E: Logistics and Transportation Review 128:52–95

N. Soeffker, M.W. Ulmer, D.C. Mattfeld (2022) Stochastic dynamic vehicle routing in the light of prescriptive analytics: A review. European Journal of Operational Research 298(3):801–820

C.J.C.H. Watkins (1989) Learning from delayed rewards. PhD thesis, King's College, Oxford

X. Yang, T. Zhang, N. Mao, Q. Chen, L. Bai, W. Chen (2018) The stochastic home health care routing problem in dense communities with multi-appointment. In: 2018 15th International Conference on Service Systems and Service Management (ICSSSM), IEEE, pp 1–6

Y. Zhan, Z. Wang, G. Wan (2021) Home service routing and appointment scheduling with stochastic service times. European Journal of Operational Research 288(1):98–110