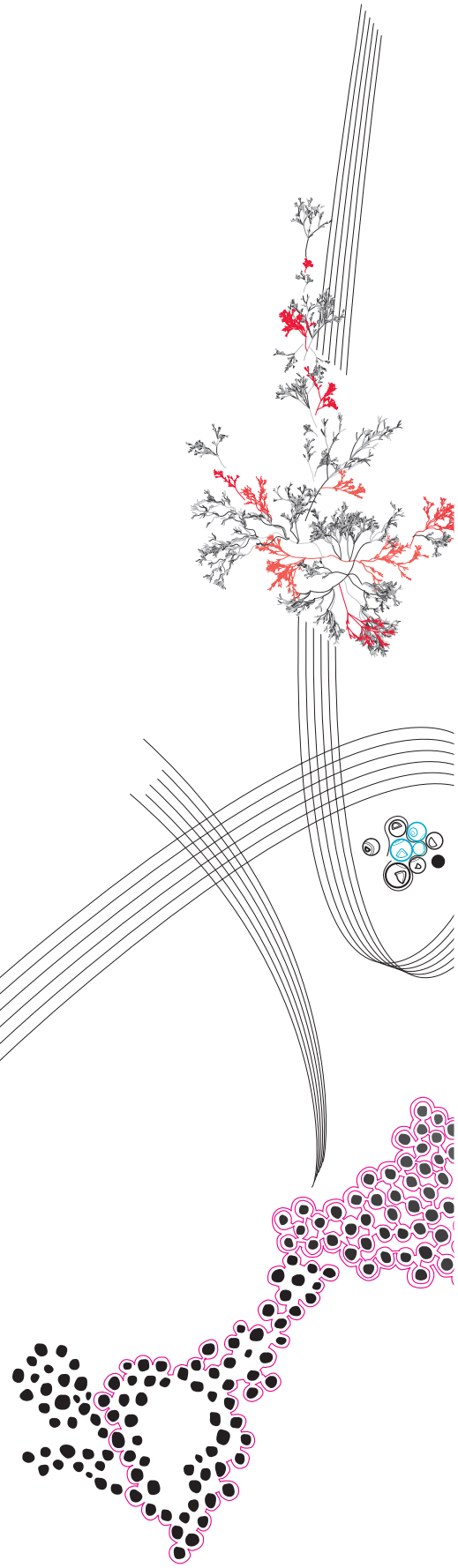

BSc Thesis Applied Mathematics

Assignment of Inspection Visits to Customers with Different Urgencies

Amalie Zeraïria

Supervisor: dr. J.C.W. van Ommeren

August, 2023

Department of Applied Mathematics
Faculty of Electrical Engineering,
Mathematics and Computer Science

UNIVERSITY OF TWENTE.

Preface

Before you lies the result of my final research for the Bachelor Applied Mathematics. During the past years I gathered many theoretical knowledge and practical skills, which I applied to this research. I really enjoyed my research and I am quite satisfied with my results.

I want to thank my supervisor dr. J.C.W. van Ommeren for his supervision and support throughout my research. Besides, I also want to thank dr.ir. G.F. Post for his help with implementing my ideas in the program of P.C.A. B.V.

Assignment of Inspection Visits to Customers with Different Urgencies

Amalie Zeraïria*

August, 2023

Abstract

We are dealing with a company employing damage experts. The company has contracts with several insurance companies, and a damage expert must visit the client when they submit a damage declaration. The clients have different urgencies (SLA) and when a client calls with a visit request, the planner of the company assigns the client to a certain day and time slot, not earlier than tomorrow.

When to schedule a client is based on the company's policy. In this paper an answer to the following research question will be formulated: "Can we find a(n optimal) policy that outperforms the already existing policies and considers the SLA, the load of the upcoming day(s), and the previous assignments of visits to days and their geographical spread?".

In order to answer this question, first the problem is mathematically formulated as a Markov Decision Process(MDP). Once formulated, an optimal policy is found using dynamical programming. This optimal policy has been compared with an already existing policy and performs better on some aspects. This is computationally really complex, since the number of possible states is a gigantic number. So an optimal policy is found for a small example.

To get more realistic results, the simulation program from the company P.C.A. B.V. is used to implement a new, self-invented policy based on a waiting list and to compare the results with already existing policies. This self-invented policy outperforms the other policies in a lot of perspectives. However whether this is the best for a company and its clients, is a question for them to answer.

*Email: a.zerairia@student.utwente.nl

Contents

1 Introduction

In the field of scheduling and routing problems, a lot of research has already been done, see [5][4]. However, the search for optimizing strategies seems to be a never-ending process. In this paper we will analyze and model a real-life business scheduling problem and try to come up with strategies that improve the situation from the point of view of both the client and the company.

We are dealing with a company employing damage experts. The company has contracts with several insurance companies, and a damage expert must visit the client when they submit a damage declaration. The contracts have different service level agreements (SLAs), which represent the number of days before the employee needs to visit the client. The clients' and employees' houses are geographically spread out over map. When a client calls with a request for getting a visit from a damage expert, the planner of the company assigns the client to a certain day and time slot. When to schedule a client depends on the policy of the company, which depends on what they value the most. Do they want to minimize the waiting times for their clients? Or do they prefer geographical clustering of the clients and meanwhile minimizing the travel costs?

With this research we aim to find an optimal policy that considers the SLA, the load of the upcoming day(s), and the previous assignments of visits to days and their geographical spread.

In order to find this strategy, we will try to formulate this scheduling problem as a Markov Decision Process (MDP) [7][8]. Once formulated, the MDP can be solved with dynamical programming using a policy iteration algorithm [1]. We will try to adapt this algorithm in such a way that our situation could be implemented.

Once we have found the optimal policy, we will run a simulation to retrieve a schedule for the upcoming days and compare these results with results from already existing policies, such as scheduling a client in the first possible time slot.

We value a policy according to four different variables; the percentage of clients that gets rejected, the percentage of clients that is visited too late, the average waiting time per client and the average travel costs per day.

Our research problem comes from a Dutch company P.C.A. B.V. [2], where they have made a model to simulate the scheduling problem using several policies [6]. We will try to come up with a better policy and implement this self-invented policy in their simulation.

So our research question is: "Can we find a(n optimal) policy that outperforms the already existing policies and considers the SLA, the load of the upcoming day(s), and the previous assignments of visits to days and their geographical spread?".

2 A Mathematical Approach: Markov Decision Process

2.1 Model

2.1.1 Markov Decision Process

In the next section we will formulate our situation as a Markov Decision Process(MDP). An MDP is a discrete-time stochastic control process. It is based on an environment and its decision-maker, in our case the latter is the company's planner. With an MPD one can model the decision making and can solve it by using dynamical programming. [8][7]

An MDP can be represented by a 5 tuple $< T, S, A(s), P(s'|s,a), R(s'|s,a) >$ where:

- T is the set of time instants when a decision is made. Each x minutes a new time instant takes place. The smaller x is chosen, the more accurate the MDP.
- S is the state space, consisting of all possible states that the system can be in.
- A(s) are all possible actions which can be taken while in state $s \in S$.
- $P(s'|s,a)$ is the probability that from state s we go to s' after taken action a, with $\sum_{s' \in S} P(s'|s,a) = 1$.
- $R(s'|s,a)$ contain all the rewards that the decision-maker obtains after taking action a and the state goes from s to s'.

The goal of the MDP is for the decision maker to find a policy to make decisions that maximizes the total reward. Given all $P(s'|s,a)$ and $R(s'|s,a)$ $\forall s, s' \in S, a \in A(s)$, the optimal policy π that maximizes the expected discounted reward can be found when solving the MDP with dynamic programming, for $\pi(s) \in A(s)$. A common way to solve this is by using policy iteration [3].

2.1.2 Problem Description

In order to fit into the format of an MDP, the situation has to be simplified and some assumptions have to be made.

Say, we have a rectangular area with one employee and several clients in it. If you divide the grid in D equal rectangular shaped districts, obtained by placing vertical lines in the area, the location of the client will be expressed in district number $d \in \{1, 2, ..., D\}$. The more districts we take, the more appropriate our model becomes.

Clients are calling with appointment requests and they arrive according to a Poisson(λ_k) distribution, with k representing the clients' SLA. When a client calls, the planner has to decide when to schedule this client according to their characteristics and the schedule of the upcoming days.

A client c_k arrives with two characteristics:

1. The district number d.
2. The SLA; the number of days within the client has to be visited according to their contract.

We will deal in this example with K different SLAs. An example of the priority set K is as following: $K = \{7, 5, 3, 1\}$ where $k \in K$ stands for the number of days that the client should be visited within. When exceeding this deadline, the appointment gets the label "late". Several clients could have the same contract and therefore the same SLA.

This process is a Markov Chain since the schedule at time $t = t_0 + 1$ is only influenced by the schedule at $t = t_0$ and the clients that is calling with an appointment request, the past is not important.

2.1.3 Assumptions

The following assumptions are made to simplify the model:

- There is only 1 employee in the company.
- The service times are constant.
- A time slot takes 60 minutes.
- An employee can visit at most 8 clients per day, 1 client per time slot.
- An employee arrives always in time at a client's house.
- Travel times fit perfectly in a time slot, in such a way that the employee can always visit 8 clients a day.
- At the end of the day, say at time $t = 480$ minutes, no client is calling and we shift the schedule one day ahead.
- It is not possible that two clients call at the same time. So at most one client has called in the time between current state s' and the previous state s.
- A calling client can be scheduled at earliest the day after they call.
- The maximum SLA is 10 days, so we can schedule at most 10 working days ahead, so our upcoming schedule is at most 10 days long.

2.1.4 Formulating the MDP

Time space Every x minutes the decision maker takes a decision and chooses an action based on the current state.

State Space In an MDP, the transition probabilities for going from one state to another should only depend on the current state. So all the information that we need that influences the transition probabilities, should be part of the state.

First, the state consists of a schedule looking like this: 8 time slots per day, 10 days long. A time slot is either empty or it contains the location of the client who is scheduled in that time slot. Other information about the client is not important to put in the schedule. Where to schedule a calling client into a schedule might also depend on the location of the full time slots, based on the policy. As mentioned before, we assume that the travel times fit always perfectly inside time slots. However, we do take into account different travel times between consecutive clients. The higher the travel times, the more expensive it gets for the company.

As mentioned before, we will only deal with one single employee, but in case you want to add more employees, you have to add parallel schedules.

Second, to determine to what state you go next, it is important to know whether a client has called since the last decision time and if so, who was calling.

Third, the current time is part of the state. To make sure that we are only looking at the schedule of the upcoming 10 working days, we need to update the time and move on to the next day when the current day has ended.

So the state of the MDP consists of three aspects:

1. An 8x10 matrix with its rows representing time slots and its columns representing the days. This represents the schedule of the upcoming days. An entry is either:
   - empty
     or
   - filled with the district number d of a client located in this district when this client is scheduled at that time slot.
2. The client calling now at time t. If no client calls, this will be 0.
3. The time in minutes from start of the day.

Actions Which actions can be taken, depends on the current state. When a client calls in our current state, you can take the following two actions:

- Schedule on (time slot, day) for example (6,2). Remember that day 1 is the first day where clients can be scheduled, that is "tomorrow".
- Reject the client, id est not assigning the client to a time slot.

When no client is calling, there are no actions that can be taken.

Transition Probabilities Starting in state s consisting of the following three aspects:

1. The schedule for the upcoming 10 days.
2. A client calling of type k and from district d, or no client at all.
3. The current time in minutes.

The following two actions could be taken:

- schedule at (time slot, day)

  or
- reject.

When the time t satisfies t **mod** $480 = 0$, the day has ended and we shift the schedule one day ahead so that the next day becomes our new current day.

No matter what the current state is and whatever the decision-maker has chosen to do, the next state will contain the updated schedule, the new time and the calling client. The former is already determined by the action that has been taken and the time is also determined. So for the transition probabilities, we will only look at the arrivals of clients.Therefore we can formulate the probability that it reaches state s' using the arrival rates of client.

Clients arrive according to a Poisson process. Clients of type $k = 1, 2, ..., K$ have arrival rate λ_k per x minutes. When there are only clients of type $k = 5, k = 3$ and $k = 10$ for example, every other k satisfies $\lambda_k = 0$.

Let us now look at the transition probabilities. After a decision is made, the next state is reached depending on if a client has called within the last x minutes, and if yes, from which type it was.

So, we have arrival rates λ_1, λ_2, ..., λ_K and define $\lambda = \sum_{k \in K} \lambda_k$.

The probability that no customer has called in the last x minutes is the probability that the time that it takes till the next call X is larger than x, that is:

$$P(\text{no one calls}) = P(X > x) = e^{-\lambda x}.$$

Then the probability that a client from type k has called, is the probability that there is a call times the probability that type k calls before the other types, that is:

$$P(\text{client of type k calls}) = (1 - e^{\lambda x}) \cdot \frac{\lambda_k}{\lambda},$$

which satisfies

$$\sum_k (1 - e^{\lambda x}) \cdot \frac{\lambda_k}{\lambda} + e^{-\lambda x} = 1.$$

Reward Function The reward function describes the reward or penalty that belongs to going from state s to s' when action a is taken. We will divide the reward in two parts: the instant reward and the travel costs.

Instant Reward If the action is to reject a client it leads to an instant penalty, id est a negative reward. If the action is to schedule a client on a certain day and time slot, this yields a penalty when the SLA is not satisfied, and a reward when scheduled within the SLA. An important note here is that we do not make a difference between being one day too late and being many days too late. Rejecting a client is punished more than scheduling a client too late. So the penalty does not increases as the patient is waiting longer. If no one calls, then no action taken, and this yields no reward. If it is the end of the day, the schedule shifts one day which also yields in no reward.

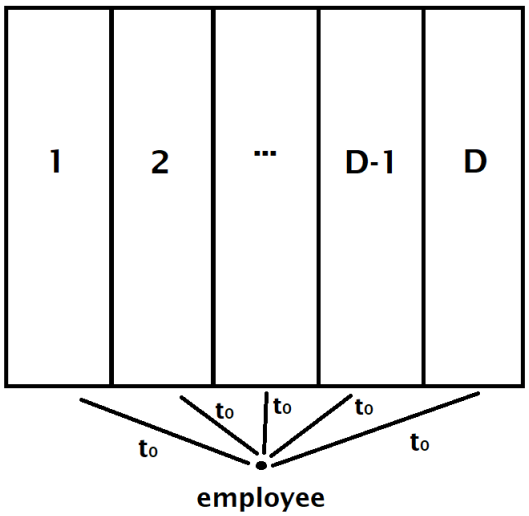

FIGURE 1: The geographical map, divided into districts.

Travel Costs At the end of the working day, the first day of the schedule is evaluated before it gets removed by the shift of the schedule. The evaluation happens based on the travel costs. In figure 1, one can find a schematic graph of the geographical map. We distinguish three costs, which are implemented in the functions as negative rewards.

1. The costs for travelling inside a district, t_{inside}.
2. The costs from travelling from one district to another, t_{outside}.
3. The costs from travelling from the employees' home to one district and vice versa, t_{start}.

Say, the employee lives outside the districts, at the same distance to each district. Staying at his home does not cost anything, but travelling to and from a district costs t_{start}. If there is an empty slot, the employee goes back to his house.

We charge travel costs for every ride that has been made on that day. We chose our districts in such a way that travelling from district 1 to 2 has costs that are equal to the costs of travelling from district 3 to 4. When travelling to consecutive districts, once $t_{outside}$ is charged. If the employee has to cross through multiple districts, then we multiply $t_{outside}$ with the difference in district number.

Example If for example the schedule of the day is like in table 1, then first from the employees house to district 1 costs t_{start}, then back to the employee's house is t_{start}, then to 4 is t_{start}, then to 2 is $(4-2) * t_{\text{outside}}$, then to 0 is t_{start}, then to 5 is t_{start}, then inside 5 is t_{inside}, then to 1 is $(5-1) * t_{\text{outside}}$, then to 0 is t_{start}. That yields a total cost of $T = t_{\text{start}} + t_{\text{start}} + t_{\text{start}} + 2 \cdot t_{\text{outside}} + t_{\text{start}} + t_{\text{start}} + t_{\text{inside}} + 4 \cdot t_{\text{outside}} + t_{\text{start}}$.

With this reward function we make sure that we prefer geographical clustering. So we prefer visiting the districts in the order 1-1-2-2 over 1-2-1-2. Since the employee goes back to his home in between shifts when a shift is empty, the model prefers 2-2-empty over

slot 1	district 1
slot 2	empty
slot 3	district 4
slot 4	district 2
slot 5	empty
slot 6	district 5
slot 7	district 5
slot 8	district 1

TABLE 1: Caption

2-empty-2.

Now that we have formulated an MDP, we can start with trying to solve it by using policy iteration.

2.2 Finding an Optimal Policy

We will try to find the optimal policy by performing a policy iteration algorithm on a small example. One can imagine that it takes a large amount of time to solve this algorithm, since the state space is extremely large.
Let us calculate first the size of the state space by using the following formula:

$$\text{size of the state space} = (\#\text{districts}+1)^{(\#\text{days}*\#\text{slotsperday})}*(1+\#\text{districts}*\#\text{priorities})*\#\text{timestepsperday}. \tag{1}$$

So if we are dealing with 10 days, 8 slots per day, 8 time steps per day, 4 districts, 4 priorities, and time steps of 60 minutes, so 8 per day; then the state space is $5.96 * 10^{57}$, by formula (1).

2.2.1 Policy Iteration

Policy iteration is an algorithm which starts with an initial policy, and evaluates it by calculating the value $V^\pi(s)$ of the policy π in state s, called policy evaluation, see [1][3]. The formula that describes the policy evaluation is:

$$V^\pi(s) = \sum_{s' \in S} (P_{\pi(s)}(s'|s)[r(s,a,s') + \gamma V^\pi(s')], \text{ where } V^\pi(s) = 0 \text{ for terminal states.} \tag{2}$$

Formula 2 describes the value of $\pi(s)$, the action chosen in state s according to policy π. The transition probability is $P_{\pi(s)}(s'|s)$ instead of $P_a(s'|s)$, since we are only dealing with the action that $\pi(s)$ provides. The corresponding reward is $r(s,a,s')$ and γ is the discount factor.

Then for each iteration it looks for a strictly improved policy, and if that does not exist, the current policy is already optimal.
In order to do that, the expected reward from state s when doing action a first and then following policy π is defined as in formula (3).

$$Q^\pi(s,a) = \sum_{s' \in S} P_a(s'|s)[r(s,a,s') + \gamma V^\pi(s')] \tag{3}$$

If there exists an a that satisfies $Q^\pi(s,a) > Q^\pi(s,\pi(s))$, then $\pi(s)$ is updated to a. If no further improvement is possible, the optimal policy is found.

2.2.2 Policy Iteration Algorithm

The work of [1] gives us a convenient framework to perform policy iteration on our MDP. We have implemented our MDP in this python algorithm to calculate the optimal policy. Different parameters are used so that one can adapt it to a specific situation and get the desired results.

Since a computer cannot solve a gigantic state space within a reasonable amount of time, we took the following parameters to run this policy iteration:

- 3 days to schedule ahead
- 3 time slots per day
- 2 districts
- 2 different SLAs: SLA = 1 and SLA = 2

Looking at the arrivals of clients, we assume that the clients are randomly drawn from the districts and the arrival rate per district is equal to 1.5 clients per day per district. So the total arrival rate of clients is 3 per day, the same as the number of time slots per day. A client has SLA = 1 with probability 0.4 and SLA = 2 with probability 0.6.

The rewards are defined as in section 2.1.4. The following fixed costs and rewards are used:

- a cost of $t_{\text{inside}} = 10$
- a cost of $t_{\text{outside}} = 60$
- a cost of $t_{\text{start}} = 15$
- a reward of 40 if the client is visited in time
- a cost of 50 if the client is visited too late
- a discount factor of 0.9

The outcome of this algorithm is the optimal policy for these parameters.

2.2.3 The Optimal Policy

The optimal policy calculated with policy iteration gives us for each state the best action to perform in order to maximize the rewards. Let us look at a few examples of states and their corresponding optimal action, to see whether the optimal policy is intuitively and does make sense.

TABLE 2: Example of a schedule of the upcoming days, with 1 and 2 being district numbers.

	Day 1	Day 2	Day 3
Slot 1	1		
Slot 2	2	1	
Slot 3	2	1	1

Example 1 Let the state be as follows:

- The schedule of the upcoming days as in table 2.
- A client from district 2 and SLA = 1 is calling.
- Time = 60 minutes.

In this example, the schedule is full for the first day, so the client is not visited in time. Since the negative reward of being too late is the same for 1 or 2 days late, we expect that the MDP policy schedules the client at day 3. Looking at the current time, there are still two hours left for today, so we expect on average 2 more clients to call. Maybe one of those two clients has priority 2 and could be visited in time when scheduled on day 2.
And indeed, the best action according to the policy is to schedule the (time slot, day) = $(2, 3)$.

Example 2 Let the state be as follows:

- The schedule of the upcoming days as in table 3.

TABLE 3: Example of a schedule of the upcoming days, with 1 and 2 being district numbers.

	Day 1	Day 2	Day 3
Slot 1	1		
Slot 2	2	1	
Slot 3	2	1	1

- A client from district 2 and SLA = 2 is calling.
- Time = 60 minutes.
  In this case, the best choice is to schedule at (time slot, day) = $(1, 2)$. This seems quite logical, since the only difference between this example and the previous one, is the SLA. Since SLA = 2 in this case, the client can be visited in time when scheduled on day 2. This is more important (according to our reward constants) than geographically clustering the clients, so it does not matter that the other visits on day 2 are in district 1.

Example 3 Let the state be as follows:

- The schedule of the upcoming days as in table 4.

TABLE 4: Example of a schedule of the upcoming days, with 1 and 2 being district numbers.

	Day 1	Day 2	Day 3
Slot 1			
Slot 2	2	1	
Slot 3	2	1	1

- A client from district 2 and SLA=1 is calling.
- Time = 120 minutes.

In this case the best choice is to schedule the client at (time slot, day) = $(1, 1)$. This makes sense since the SLA equals 1, so if scheduled on the first day, the client will still be visited in time and the reward will be positive.

Example 4 Let the state be as follows:

- The schedule of the upcoming days as in table 5.

TABLE 5: Example of a schedule of the upcoming days, with 1 and 2 being district numbers.

	day 1	day 2	day 3
1.			
2.	2	1	
3.	2	1	1

- A client from district 2 and SLA=2 is calling.
- Time = 120 minutes.

In this case the best choice is still to schedule the client at (time slot, day) = $(1, 1)$, which makes sense, since it would be on time if it is scheduled in day 1 or in day 2. Then the geographical advantage of placing it before a client from district 2 is decisive.

Example 5 Let the state be as follows:

- The schedule of the upcoming days as in table 6.
- A client from district 1 and SLA=2 is calling.
- Time = 0 minutes.

TABLE 6: Example of a schedule of the upcoming days, with 1 and 2 being district numbers.

	day 1	day 2	day 3
1.			
2.	1	1	
3.	2	1	1

The best choice is (time slot, day) = $(1, 2)$, which makes sense since it has SLA=2, and we prefer clustering, so we choose day 2. Also since we are only at the start of the day, the probability of a client calling today with SLA=1 is still reasonable large, so this spot will be reserved for that potential client.

Example 6 Let the state be as follows:

- The schedule of the upcoming days as in table 7.

TABLE 7: Example of a schedule of the upcoming days, with 1 and 2 being district numbers.

	day 1	day 2	day 3
1.			
2.	1	1	
3.	2	1	1

- A client from district 2 and SLA=2 is calling.
- Time = 120 minutes.

Now the best choice is (time slot, day) = $(1, 1)$. The only difference in state in comparison with example 5 is the time. The current time equals 120 minutes, and since we have a decision moment every hour, this is the last decision moment. Which means that if we do not schedule this client on day 1, this first slot will stay empty. So in this case we prefer filling the empty slots over geographical clustering. Additionally, there is a reasonable chance that the day after this, a client with SLA=2 calls, so we want to keep a free spot on day 2.

In conclusion, the optimal policy retrieved from the dynamical programming does make sense. We have studied some interesting examples and the decisions of the optimal policy are quite intuitively.

2.2.4 Simulation

Now that we have retrieved the optimal policy, we want to evaluate this policy. As mentioned earlier, we value a policy according to four different variables:

1. the percentage of clients that gets rejected
2. the percentage of clients that is visited too late
3. the average waiting time per client
4. the average travel costs per day

In order to conclude whether or not our retrieved optimal policy performs good, we are going to compare it with an already existing policy; the first possible slot policy. This policy always chooses to schedule the client in the first empty slot in the schedule of the upcoming days.

Expectations Since for the first possible slot policy it is not a goal to minimize the travel times and cluster clients geographically, we expect that the travel costs for this policy are quite high. However, since it chooses for every patient the first empty slot we expect that the waiting times for clients are quite low. According to the first possible slot policy, a client will be rejected if and only if the schedule of the upcoming 3 days is completely filled. The probability of this happening is really low, since we chose the arrival rate per day the same as the number of time slots per day.

We expect the MDP optimal policy to result in lower travel times than the first possible slot policy, since it tries to minimize the travel times. However, this will probably be at the expense of the waiting times, which we expect to be higher than for the first possible slot policy.

2.2.5 Results

A simulation has been run to see what the schedule looks like and to compare the two above mentioned policies with each other. Per day there are 3 decision times. This coincides with the time slots, of which there are also 3 per day. In order to make sure the system has stabilized, the simulation runs over 3000 days. The results for the first possible slot policy as well as the MDP optimal policy, can be found in table 8.

TABLE 8: The performance of the first possible slot policy and the MDP optimal policy

	Clients rejected [%]	Clients too late [%]	Average waiting time per client [days]	Average travel costs per day [units]
First possible slot policy	0.0000	0.0000	1.0000	152.4633
MDP optimal policy	0.0000	6.5504	1.6029	116.6433

As expected, the waiting time is higher for the MDP optimal policy, and the travel costs are higher for the first possible slot policy. The question is, to what extend is the MDP optimal policy really optimal? This is basically up to the company. What do they value the most?

In conclusion, we have found an optimal policy for our specific situation and parameters. If wanted, a company can adjust the parameters based on their wished and needs. In that way, a new optimal policy can be retrieved.

3 A Practical Approach

Since with the use of our computationally complex MDP, we could not simulate a realistic case, this chapter will cover a realistic simulation of the problem. Two already existing policies are being observed and evaluated. Finally, our main goal is to come up with our own policy which outperforms the two already existing policies.

3.1 Simulation

The company P.C.A B.V. [2] made a simulation program with the following assumptions and parameters:

- The clients' locations and SLAs are randomly generated, their location as coordinates in a grid, and their SLA is randomly drawn from a triangular distribution between 5 and 10 days.
- There are 25 employees who can visit clients.
- There are 8 time slots per day, so each employee can visit 8 clients per day.
- There is a total arrival rate of 200 clients per day.

The simulation runs 200 working days. When a client calls, they get either rejected or scheduled immediately. The appointment takes place no earlier than the day after the call. It aims to schedule every client according to their SLA, but allows a 50% extension.

3.2 Existing Policies

This simulation schedules clients based on a policy. There are already some policies implemented by the company and in this paper we will take a look at two of them.

3.2.1 First Possible Day Policy

When a client calls, the first day in the schedule with a free time slot is chosen. If there is more than one free time slot on that day, the slot which yields the least extra travel time will be chosen.

3.2.2 Minimum Travel Time Policy

Within the client's SLA, all the possible time slots are considered. Thereafter, the time slot with the least extra travel time will be chosen.

3.3 Self-Invented: Waiting List Policy

In this section our self-invented policy will be explained. In the former two strategies, the clients get rejected or scheduled immediately when they call. The possible appointment options are starting from tomorrow onward.

Instead of immediately assigning the client to a certain time slot, in this policy the client will be put on a waiting list, sorted on priority, their SLA. Only the next

day will be scheduled, with clients with the highest priority who add the least extra travel time. So we loop over the time slots on that day, and consider all the clients with the highest priority. Thereafter we schedule the client that adds the least extra travel time. When every slot on that day is filled, we move on to the next day. Using a waiting list, information of 'future' clients is provided which makes the geographical clustering easier and better. The clients will get their appointment information one day in advance, since we only schedule the next day. We call this policy the Waiting List Policy.

3.4 Results

3.4.1 First Possible Day Policy

The results and performance evaluation of the First Possible Day Policy can be found in table 9 and 10.

TABLE 9: The performance of the First Possible Day Policy from the company's perspective.

Idle duration	2.86%
Travel duration	33.52 %
Task duration	63.62%

TABLE 10: The performance of the First Possible Day Policy from the clients' perspective.

Clients rejected	14.95%
Clients too late	69.97 %
Clients in time	15.08 %

3.4.2 Minimum Travel Time Policy

The results and performance evaluation of the Minimum Travel Time Policy can be found in table 11 and 12.

TABLE 11: The performance of the Minimum Travel Time Policy from the company's perspective.

Idle duration	2.94%
Travel duration	31.78 %
Task duration	65.28%

TABLE 12: The performance of the Minimum Travel Time Policy from the clients' perspective.

Clients rejected	12.78%
Clients too late	69.28 %
Clients in time	17.94 %

3.4.3 Self-Invented: Waiting List Policy

The results and performance evaluation of the self-invented Waiting List Policy can be found in table 13 and 14.

TABLE 13: The performance of the Waiting List Policy from the company's perspective.

Idle duration	3.30%
Travel duration	25.10 %
Task duration	71.60%

TABLE 14: The performance of the Waiting List Policy from the clients' perspective.

Clients rejected	1.55%
Clients too late	63.78 %
Clients in time	34.67 %

3.4.4 Comparison

The Waiting List Policy outperforms the other two policies in many aspects. The travel duration is decreased significantly and the system is way more efficient. The percentage of rejected clients is improved drastically and way more clients are visited in time. However, there is a downside for the clients, they do not know their appointment immediately after they called the company with a request. The clients get their appointment details only one day prior to the appointment itself.

4 Conclusion

Now that the research has been done, we can answer the question: "Can we find a(n optimal) scheduling policy that outperforms the already existing policies and considers the SLA, the load of the upcoming day(s), the schedule and the geographical location of the clients?". The answer is yes, we have found an optimal policy by formulating the problem as a Markov Decision Process and solving a dynamical program using policy iteration. Whether it outperforms other policies is up to the company and what they value the most. It performs good in the sense that it prioritizes geographical scheduling. However, the waiting times for clients are not as low as possible.

Since we could not test and simulate this MDP-based optimal policy for a decent realistic situation, we used the algorithm of the company P.C.A. B.V. to compare their policies with a self-invented one. The results of this self-invented Waiting List Policy are outstanding. It definitely outperforms the other policies in many aspects.

5 Discussion

Markov Decision Process One could conclude from the results that the first possible day policy is better for the client than the MDP based policy, since less clients are visited too late and the waiting times are lower. However, for the optimal policy we see something particular, the travel costs are way less than for the first possible slot policy. Whether the clients find it disturbing to be visited a little later, is a question which we cannot answer, that information could be gathered via a client survey. For now, we value geographical clustering really high, so our optimal policy does what we want it to do.

In my opinion it is not a bad thing to let the client wait half a day longer so that the travel times and travel costs significantly decline. However, this is really depending on the severeness of the damage. There are definitely situations where we want to visit the client as soon as possible, independent of the extra travel time they cause.

A reason for the average waiting time being higher for the MDP Policy than for the First Possible Slot Policy, could be the fact that the reward for being too late does not distinguish being one day too late versus being ten days too late. It would be better to have a reward function where it is way worse to be visited ten days too late then one day too late.

Self-invented Waiting List Policy The waiting list policy outperforms the already existing policy in many ways. The main downside of this policy is that the client does not immediately knows their appointment date and time when calling, but they know it only one day in advance. This could be no problem for some kind of companies, it does really depend on what kind of service the company provides. For medical services this could be really worse, however for package delivering it would be fine to get a text or an email the evening before the visit. Actually, this is already used in a lot of services.

I would say that this policy could be implemented in specific real-life companies, and our research provides a clear overview of the advantages and downsides of this policy.

6 Recommendations

Since this research was spread over only one quartile of an academic year, there are still a lot of interesting issues left to research. In this section a few of these topics will be listed for future research purposes.

1. A new MDP could be formulated with the waiting list policy implemented.
2. In the MDP, the reward function could be adapted such that the costs get higher as the client is visited more days too late.
3. It could be interesting to make it worse to visit a client with SLA=1 too late than a client with SLA=5. In that way the severeness of the damage is taken into account.
4. To solve the problem that some clients with severe damage should be scheduled immediately instead of being put on a waiting list, a new policy could be formulated where in most cases the client will be put on a waiting list, but some clients with high priority will be scheduled immediately when they call. For example the rule to schedule clients with SLA=1 immediately, but put the rest on the waiting list, could be implemented. These results would be very interesting.

References

[1] Policy iteration x2014; Introduction to Reinforcement Learning — gibberblot.github.io. `https://gibberblot.github.io/rl-notes/single-agent/policy-iteration.html`. [Accessed 11-Jul-2023].

[2] Service and maintenance planning - PCA — pca.nl. `https://pca.nl/en/functionalities/service-and-maintenance-planning/`. [Accessed 11-Jul-2023].

[3] blackburn. Reinforcement Learning : Solving MDPs using Dynamic Programming (Part 3) — towardsdatascience.com. `https://towardsdatascience.com/reinforcement-learning-solving-mdps-using-dynamic-programming-part-3-b53d32341540`. [Accessed 11-Jul-2023].

[4] Jalel Euchi, Salah Zidi, and Lamri Laouamer. A new distributed optimization approach for home healthcare routing and scheduling problem. *Decision Science Letters*, 10(3):217–230, 2021.

[5] Engin Pekel. Solving technician routing and scheduling problem using improved particle swarm optimization. *Soft Computing*, 24(24):19007–19015, September 2020.

[6] Gerhard F. Post and Stefan Mijsters. On the call intake process in service planning. In Patrick De Causmaecker, Ender Özcan, and Greet Vanden Berghe, editors, *Proceedings of the 13th International Conference on the Practice and Theory of Automated Timetabling - PATAT 2022*, volume III, pages 265–278, 2022. 13th International Conference on the Practice and Theory of Automated Timetabling, PATAT 2022, PATAT 2022 ; Conference date: 30-08-2022 Through 02-09-2022.

[7] Sheldon M Ross. *Introduction to probability models*. Introduction to Probability Models. Academic Press, San Diego, CA, 9 edition, December 2006.

[8] Tao Zhang, Shufang Xie, and Oliver Rose. Real-time job shop scheduling based on simulation and markov decision processes. In *2017 Winter Simulation Conference (WSC)*. IEEE, December 2017.