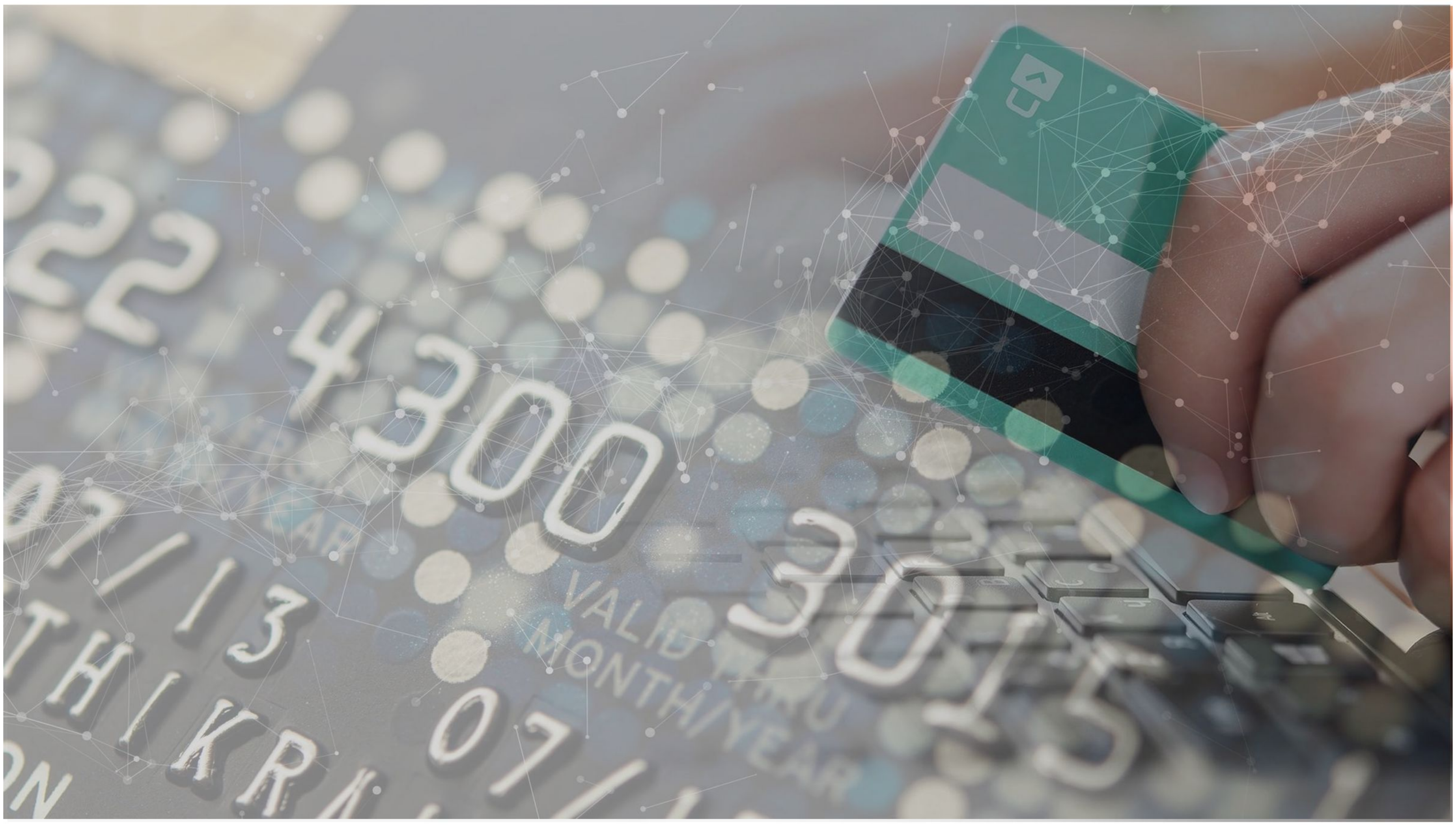
IMPROVING PAYMENT AUTHORIZATION RATES
BY INTELLIGENTLY ROUTING TRANSACTIONS



Author: Cristian Ciobotea

In collaboration with:

UNIVERSITY OF TWENTE.

adyen

Improving payment authorization rates

by intelligently routing transactions

Cristian Ciobotea

Master of Science
Computer Science
Information and Software Engineering
University of Twente, Enschede, the Netherlands

Student number	1583093	
E-mail	cioboteacristiann@gmail.com	
Project duration	1st of January - 31st of July 2016	
Supervisors		
	dr. ir. M.J. van Sinderen	University of Twente
	dr. N. Litvak	University of Twente
	C. Laumans	Adyen
Graduation committee		
	dr. ir. M.J. van Sinderen	
	dr. N. Litvak	
	dr. L. Ferreira Pires	

"Try to learn something about everything
and everything about something"
by Thomas Huxley (1825 - 1895, biologist)

Executive summary

Introduction

The payment industry has closely followed and benefited from the new technologies and e-commerce trends. Easy, fast and flexible payment methods have become a necessity. Shoppers expect to be able make payments on web, in-store, and on mobiles, no matter the currency, merchant or type of bank card. Merchants want to accept payments from anywhere in the world. However, payments are not always successful. In the payment flow there are many processing institutions, such as payment service providers (PSP), acquiring banks, credit card schemes, and card issuing banks. At a global scale, the interaction between all of these different systems is not perfect, resulting in refused payments.

What it is usually seen at Adyen, a technology company that processes payments, is that, on any given day, 10-20% of online card payments may get refused. Approximately half of refusals are for legitimate reasons because of insufficient funds, possible frauds, or expired cards, while the other half are caused by technical or inexplicable flaws in the payment flow. Adyen has tried to learn useful patterns in the failing payments and intervene where possible. However, this proved to be a very difficult task, given the diversity of payments and the dynamic character of the payment flow.

Research Goal and Research Approach

The main research goal is to **improve the authorization rate of genuine payments by designing an optimization algorithm for routing transactions in the payment flow, algorithm that supports unknown and unfixed authorization rates, such that payments are more efficiently processed.**

The research approach follows the methodology explained in [1]:

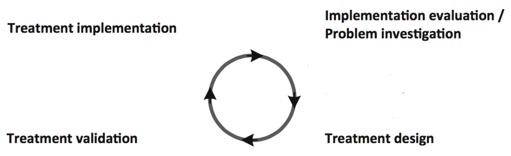


Figure: Engineering cycle [1]

In the problem investigation, we identify Adyen and its clients (the merchants) as the main stakeholders. The architecture framework is the payment flow of the transaction from the shopper to the merchant:

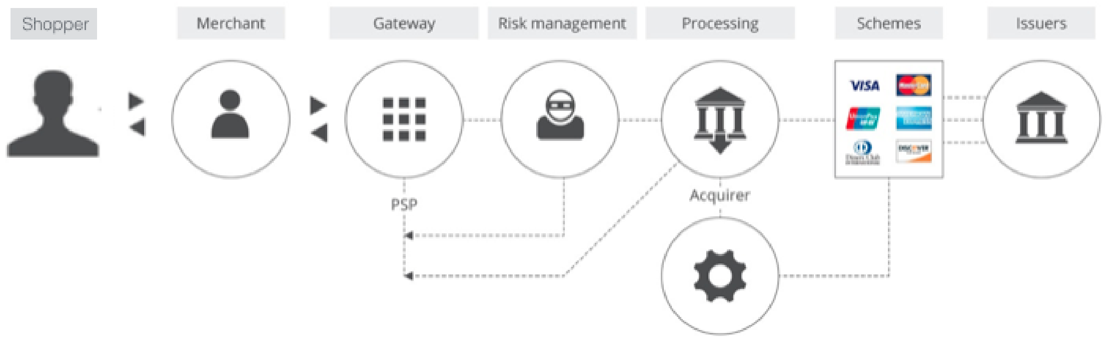


Figure: Architecture framework

In some markets (e.g.: in Spain, France or US), Adyen (as a gateway) can connect to two or more acquirers. It has been observed that the authorization rate of the transactions is influenced by the selection of one acquirer or the other to send transactions to. Choosing acquirers from the available ones and redirecting traffic to them is called **routing of payments**. However, the better acquirer, in terms of the highest authorization rate, is not evident and it depends on many factors. The most important factors are the country of the merchant and BIN (Bank Identification Number; usually, the first 6 digits of a bank card).

Current solution and proposed treatments

In the treatment design of the engineering cycle, we start by analyzing the requirements. The most important requirements are the overall authorization rate of transactions increase and the usability of the solution on the current platform.

Adyen has built the core infrastructure to enable routing of payments, but the answer to the question on **how** to route payments is still unknown. Currently, the routing rules are decided by the data analysts. They look at the past transactions, by country, merchant, BIN and try to find large differences in authorization rates (usually, larger than 1%) between acquirers. This method proved to be cumbersome and inefficient in terms of authorization rate improvements. After each setup, subsequent analyses were run. In many cases, they showed that the setup was not correct because the acquirer, that had been selected to route the most payments to, did not have the highest authorization rate as expected.

In this thesis, we analyze several possible treatments for payments routing and test them against payment data. These algorithms are part of the randomized experimental design (Play the Winner Rule) and reinforcement learning, the so-called multi-armed bandits (4 different versions or strategies: ϵ-Greedy and Pursuit strategy, Gradient bandit, Exp31, and Thompson sampling).

In the design of these algorithms, we have two phases: exploration (find the better acquirer) and exploitation (send traffic to the better acquirer). Each algorithm has different approaches on exploration and exploitation. Play the Winner Rule algorithm sends traffic to a single acquirer until a number of refused payments has been reached; when the refused transactions go over this threshold, then the algorithm switches the traffic to one of the alternatives and the count of failed transactions starts again. ϵ-Greedy always exploits, except for a fixed proportion of ϵ % transactions, which are randomly sent any of the available acquirers. Gradient bandit and Exp31 use more complex functions to compute the preference to an acquirer or the other. Thompson sampling uses a Beta distribution to calculate the probability of each acquirer of being the best one.

The main traits of these algorithms are that they do not make assumptions on the authorization rate of the acquirers and they never settle to an acquirer, but continuously try to identify change.

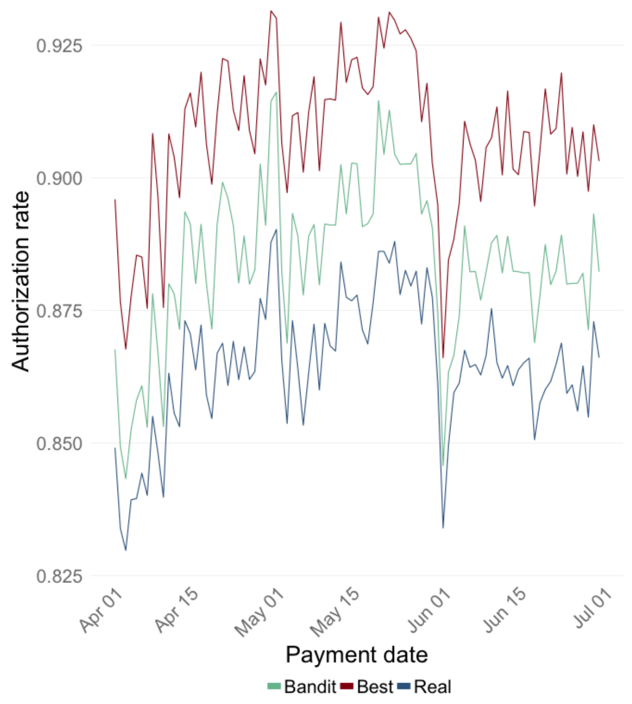
Some of these algorithms have parameters that can be set by the user (e.g.: thresholds, ϵ). In our case, we test the same algorithm with different combinations of parameters to see the impact of these parameters on the results.

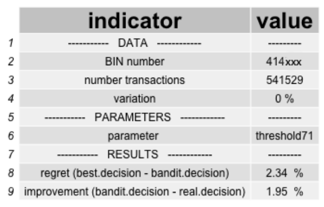
Results

In the treatment validation of the engineering cycle, we gathered real payment data for a merchant, in three markets: France, Spain, and US. In total, the sample data has 30.7 mil transactions, for approximately 17,000 different BINs. These BINs are grouped into 10 clusters per volume of transactions per day, variation of authorization rate and difference of authorization rate between the two acquirers. A subset of two BINs from each cluster has been extracted to run the algorithms.

We implemented the proposed algorithms offline. However, we tried to simulate the real world as much as possible. For example, we included the relevant fields when generating the payment requests and we added a time delay to the payment responses.

The below simulation shows the authorization rates per day after applying Play the Winner Rule algorithm for one of the US BIN. In this case, we can see that the use of the bandit algorithm is expected to bring 1.95% authorization rate improvement. However, there is still a regret of 2.34% authorization rate that could be further improved:





	indicator	value
1	----------- DATA -----------	---------
2	BIN number	414xxx
3	number transactions	541529
4	variation	0 %
5	----------- PARAMETERS -----------	---------
6	parameter	threshold71
7	----------- RESULTS -----------	---------
8	regret (best.decision - bandit.decision)	2.34 %
9	improvement (bandit.decision - real.decision)	1.95 %

Figure: Play the Winner Rule on a US BIN

In the plot below, we see the overall results of the selected algorithms. The one with the highest expected authorization rate improvement (of 2.84%) is the multi-armed bandit - Greedy and Pursuit bandit strategy:

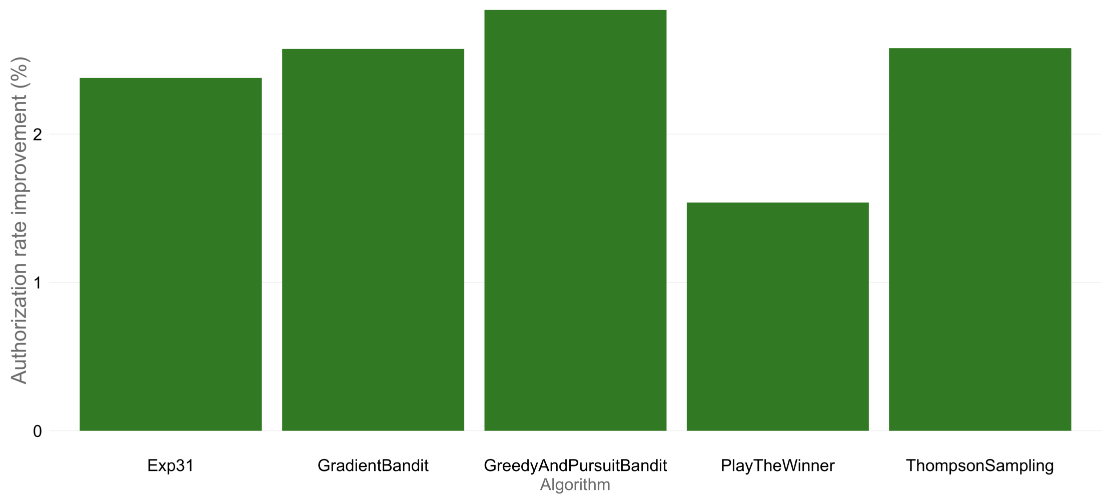


Figure: Authorization rate improvement weighted by the total number of transactions

However, there is a trade-off between the higher usability, but lower performance of the Play the Winner Rule and the lower usability, but higher performance of the Greedy and Pursuit bandit.

Conclusion

Multi-armed bandit is a neat solution for optimizing the authorization rates of payments and it can bring up to 2.84% more authorized payments. This result has been achieved by simulating payments requests and responses based on real data, and applying multi-armed bandit - Gradient and Pursuit bandit strategy.

The main research goal is achieved by utilizing any of the designed algorithms for routing payments to different acquirers. Implementing one of these algorithms will definitely reduce the effort of setting routing rules, compared to the current implementation.

Intelligent payment routing can be easily implemented and tested on the payment platform at Adyen and leverages the connections to multiple acquirers.

Future research

The multi-armed bandit has dozens of versions and strategies, but this thesis focuses only on small subset of them. To get a complete view on the performance of the multi-armed bandit, it would be useful to have an exhaustive testing of all strategies.

As already mentioned, some of the bandit strategies depend on parameters. Not only we can test different values for these parameters, but it would be useful to have another logical layer to learn the best parameters for each strategy.

Adyen can also modify the payment request itself (by tweaking various fields, such as merchant address data), procedure called payment flagging. There are many similarities between payment flagging and intelligent payment routing. Based on analogy, the same research and same algorithms could be used in payment flagging applications for the same purposes of increasing the authorization rate and make payment more efficient.

The failure rate due to unknown or technical reasons is much higher (up to 10% of all transactions). Thus, a lot more effort needs to be put in understanding the payment flow and the interactions between all financial institutions to cover all technical failures.

At the end of the day, payment processing is deterministic. Treating the other end as a black box is due to poor standardization of messages and adoption of those standards by all payment processing institutions. Until these standards will be perfectly implemented and adopted, using algorithms such as the multi-armed bandit has the potential of getting accepted thousands of payments world-wide, every day.

Preface

It is amazing how some things are taken for granted. Before diving into the payments industry, I realized that I had never really thought about how transactions work. At most, I had some concerns about possible frauds and extra costs. Now that I know much more about how payments happen, I find card payments and all kind of alternatives to be very exciting and I want to make use of them every time I have the chance. If prior to this research I had doubts that things can go wrong with my payments, I now know for a fact that things go wrong, in many cases. Contributing to this real world problem has been a major motivation for me during the current research.

I would like to thank Marten van Sinderen for his assistance throughout my entire second academic year at University of Twente and for emphasizing the importance of adding perspective to this research and to the results. Equally, I would like to thank Nelly Litvak for teaching me about the strictness of statistics and academic writing. Special thanks go to Chris Laumans for his everyday support, trust, time and for being open minded to new ideas. Thanks to Pieter Huibers and Bert Wolters for helping me solving all my implementation and integration issues. I am also grateful to Jelle, Rahul, Daniel, David, Traian, Ruben, and many others from Adyen that took their time to listen to my ideas and gave me useful feedback. Many thanks to Maikel for introducing me to Adyen's culture and for sharing with me his enthusiasm for doing great things for both the academic and business worlds.

Finally, thanks to my family and friends for always being beside me, even though I decided to study abroad and thus, we have had less time to spend together.

Enjoy reading!

Contents

List of Tables

List of Figures

1. Introduction

1.1. Problem identification and motivation

Payments are not new, they have been with us for a while. Electronic payments (such as debit cards, credit cards, electronic funds transfers, direct credits, direct debits, internet banking and e-commerce payment systems) were a disruptive technology at the time they were introduced. Also in the beginning, these systems were adopted by banks by simply building them on top of their infrastructure at that time. This way, with relatively little effort, banks managed to make these payments work.

Since then, technology world has witnessed another type of disruption, i.e. the expansion of the Internet, the dominance of the mobile world, and the IoT. These new trends majorly influenced the payments systems. Payments need to support all kind of innovative payment methods (e.g.: ApplePay, AliPay) at a whole new level in terms of volume. Last but not least the globalization of businesses and shoppers has intensified, which increased the diversity of systems involved in processing of payments. Nowadays, it is not uncommon for a person with a card issued by a bank in the Netherlands order and pay products from a merchant in Mexico.

Making payments as fast, seamless, easy, and secure as possible has become very important in the new digital context. The data insights from payments are also extremely valuable to understand the shopper behavior and to enable a better service. One example is the introduction of omnichannel, i.e. to be able to link payments made at Point-Of-Sale (POS), online and mobile for the same shopper. This information is very useful, in many ways. For example, one could order shoes online and return them to the same brand's store around the corner of his street. Another example is that the merchant can see the full profile of one shopper and make specials offers or can link fraudulent patterns and have a better risk control.

However, payment systems do not always keep up with the needs of merchants and shoppers. One effect is that payments fail. The reason is that during a payment processing there are many different parties involved, that are not always communicating effectively. A payment can be seen as a message sent to the shopper's bank to request money to be transferred to the merchant's account. There are many problems that can occur here, such as network errors, message interpretation, wrong formatting, etc. that cause a payment to fail. Fixing these deviations from the standards is not trivial. In many cases, the processing at the issuing banks (the bank of the shoppers) is treated as a black-box, sometimes due to privacy concerns. The payment response that comes from the issuers contain a response code, but in many cases this code is general and not helpful in identifying the problem that caused the payment to fail. Moreover, there is a lot of other deciding factors (e.g.: card scheme specifics - Visa/MasterCard/Amex -) in the payment flow, there is a lot of diversity (country specific regulations, tens of thousands of different issuers), and a great deal of change in how this processing is being done at every end.

Merchants are aware that payments are failing at times, without fully understanding why. They are cognizant that there is little that they can do about it. The complexity in payment processing allowed dedicated businesses to develop and fulfill the need to solve this complexity. One of these businesses is Adyen B.V., a Dutch-based technology company, that started as a payment gateway to banks and card schemes, on behalf of the merchants. Adyen provides a single-platform to accept payments anywhere in the world through any sales channel. At Adyen, wee see that, on any given day, 10-20% of online card payments may get refused. Approximately half of refusals are for legitimate reasons because of insufficient funds, possible frauds, or expired cards, while the other half are caused by technical or inexplicable flaws in the payment flow. In 12 months time, Adyen solely processed approximately 1.5 billion transactions. Roughly, this means that there are up to 150 million transactions that potentially could have been saved. For companies as Booking.com, Spotify, Uber, any improvement can be very relevant as it brings more revenue without much effort on their side. As for the shoppers, eliminating failing transactions means less frustration.

The main motivation behind this research is to reduce the number of failed transactions and to make a step further in transforming payments from a hindrance to an opportunity.

1.2. Research goal

The ratio between authorized or successful payments over all payments is called *authorization rate* or simply, *auth rate*. The authorization rate is a consistent metric used to calculate performance. Reduction of payment failures is equivalent to increasing the authorization rate.

Adyen, as a gateway or so-called Payment Service Provider (PSP), can observe different patterns of the payments that fail, make suggestions and take actions. For example, if one issuer sends back many failures with the code: “Shopper address not provided”, then Adyen can ask the merchant to ask this information on their webshop and send it over to the issuer. Adyen can also take actions to slightly modify and format the payment request to issuing bank’s specific preferences. For instance, some issuers request the address of the shopper to exist. However, they do not do any further check; so, in case the merchant does not include any address, Adyen can simply attach an empty string field. Another type of action that Adyen takes is to auto-retrying payments to surpass a temporary glitch or transmission issue. Last but not least, Adyen can choose different routes for the payment requests through the payment flow to increase the chances of an payment to be accepted.

These actions are very useful and they might seem logical and simple to apply. In reality, these rules and patterns are not always evident. Furthermore, even if they are identified, they are not consistent in time. In the shopper address example, we do not know beforehand which issuers handle the payments by only checking if the address exists. Even if we have a good idea whether an issuer does this, we do not know when this issuer changes the address check at their end to actually verify the complete shopper address. In the same way, a payment request route might show a better authorization rate today, but this route might not have the same results tomorrow.

The main research goal can be synthesized as: **improve the authorization rate of genuine payments by designing an optimization algorithm for routing transactions in the payment flow, algorithm that supports unknown and unfixed authorization rates, such that payments are more efficiently processed.**

The word ”genuine”, used to describe payments, limits the scope of the algorithm, by excluding the transactions that are supposed to fail (e.g.: insufficient funds or PIN incorrect), but also transactions that are tagged as frauds. The desired algorithm does not directly aim to reduce these failures, because they are managed already by other tools (e.g.: risk detection tools).

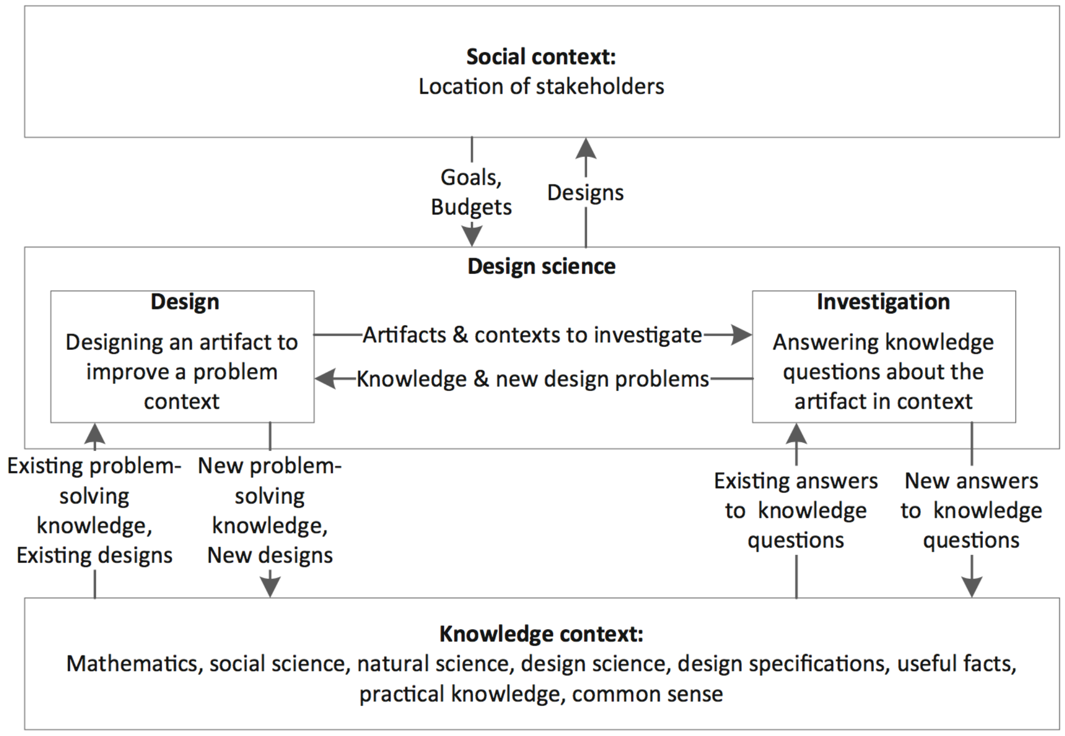
Known algorithms that deal with similar problems are defined by general statistics (section 3.1), randomized experimental design (section 3.2), and reinforcement learning (section 3.3).

As far as the present literature review has gone, none of aforementioned solutions were applied to the payment industry, which leaves a gap of knowledge in the existing algorithms, heuristics and theories. This gap is obvious when trying to apply these algorithms and certain context assumptions are not fulfilled (such as: fixed probabilities or immediate responses), thus resulting in inaccurate outcomes. This research should come as an extension to the current knowledge so that it accommodates the requirements of the payments industry.

1.3. Research methodology and overview

This research is organized and conducted by using the design science methodology explained in [1]. Two of the most important concepts in the aforecited book is the design science framework and the engineering cycle.

Looking at the scheme in figure 1.1, the most important stakeholder is Adyen B.V. that has dedicated most resources into this research. The design is the optimization algorithm. In the investigation phase, we try to explore the payment flow, the current implementation, and especially the subsystems that can be actionable from the point of view Adyen, PSP. The theoretical knowledge (around the literature of the existing algorithms) and the practical knowledge (about how a payment system works) constitute the knowledge context.



Social context:
Location of stakeholders
Goals, Budgets
Designs
Design science
Design
Designing an artifact to improve a problem context
Artifacts & contexts to investigate
Knowledge & new design problems
Investigation
Answering knowledge questions about the artifact in context
Existing problem-solving knowledge, Existing designs
New problem-solving knowledge, New designs
Existing answers to knowledge questions
New answers to knowledge questions
Knowledge context:
Mathematics, social science, natural science, design science, design specifications, useful facts, practical knowledge, common sense

Figure 1.1.: Framework for design science [1]

This current thesis focuses on the design cycle, which is a subset of the engineering cycle, excluding the treatment implementation and implementation evaluation. These final parts are the responsibility of the main stakeholder, Adyen, based on the knowledge and design that result from this research (figure 1.2).

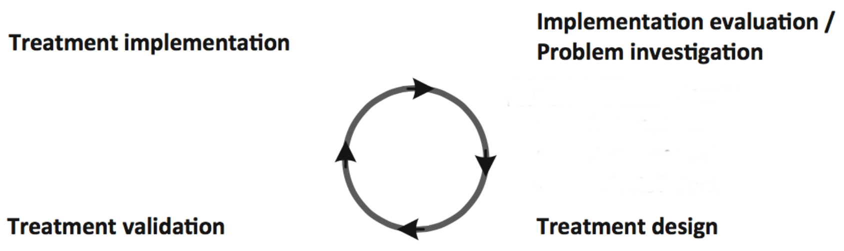


Figure 1.2.: Engineering cycle [1]

The rest of this work is organized as follows:

- problem investigation (Chapters 1, 2, 3) focuses on understanding the problem, identifying the stakeholders, acknowledge the role of Adyen, analyzing the current implementation, defining conceptual and statistical frameworks, and the literature review for relevant existing solutions as resulting from the state-of-the-art;
- treatment design (Chapter 4) describes the algorithm's requirements and context assumptions, determining the available solutions, and finally, proposing designs based on the expectations of contribution to the stakeholders' goals;
- treatment validation (Chapter 5) contains the validation models for the design and context and the sample data used for testing the proposed treatments. The results are used as prediction of what will happen if the algorithm is implemented on the live Adyen payment platform. The most important outcome of this phase is building a design theory based on the validation model;
- the last part (Chapter 6) summarizes the research within three subsections: conclusions, limitations, recommendations, and future work.

2. Background

2.1. Stakeholders

Stakeholders are very important for any type of research, because in the end, they are the ones who directly benefit from the results, but also that might decide to commit resources in order to implement the new design and knowledge. The most important stakeholders are:

- shopper - a person/company that buys products;
- merchant - a company that sells products;
- acquirer - a bank or financial institution that processes credit or debit card payments on behalf of the merchant;
- card scheme - a payment network linked to payment cards; the most relevant card schemes for this research problem are: Visa, MasterCard, American Express, Diners Club, JCB, Discover;
- issuer - a bank or financial institution that issues cards to shopper on behalf of the card schemes; the number of issuers is on the order of thousands;
- payment platform - a technical solution providers that connect the merchant to the acquirers, card schemes and issuer banks; in this case, Adyen is the payment platform under research.

An implicit stakeholder is University of Twente, involved with the supervision and overall contribution to the research. It is important to notice that fraudsters are excluded from the list of the stakeholders, because they are out of the scope of this research and they are not considered to have a relevant impact on the context of the treatment.

Most of the stakeholders are aware of the problem of failing transactions, and may or may not be committed to solve the issues on their side, depending on the level of information they have. For example, payment platforms have usually more data about different merchants, shoppers, acquirers, etc. and can aggregate all this information for more accurate actions. On the other hand, the shoppers have less options in fixing the problem, and can be at most aware of it.

The aforementioned stakeholders have a similar top-goal: **the merchants want to increase the efficiency in processing all genuine transactions and shoppers want all (presumably genuine) payments to go through, so they can get their goods or services.**

Looking more closely to the top-level goal, there is no conflict between stakeholders, because it is a win-win situation for all: shoppers are buying the products they want, merchants are selling their products, and payment platforms, acquirers, card schemes and issuers successfully process payments, thus selling this service.

However, the stakeholders that provide the payment systems have intrinsic conflicts, such as legal, technical, and economic conflicts:

- Legal conflicts. Data confidentiality is an important concern. Each merchant has its own payment data and full control over it. On the other hand, having aggregates over industries has the potential of improving authorization rates by making some patterns evident. Another example of legal conflict is that some banks consider the information of "insufficient funds" as confidential, and they merely send a generic "error" message. Thus, even though the authorization could be optimized by knowing the exact failure cause, banks prefer to keep secret this information, on behalf of the shoppers;

- Technical conflicts. Security is very important and it creates a lot of overhead for the payment process itself. Many checks during the payment process that are designed to assure security, can actually cause technical errors: invalid card, invalid amount, etc. The technical conflict appears when trying to accept as many payments, but without ignoring any security checks;

- Economic conflicts. Unsure payments that have an unknown decline message or that are suspect could be added to a queue, and manually verified by a specialist. This way, trying to increase the authorized genuine payments comes at the cost of the specialist. Another example of economic conflict is to invest more in the IT infrastructure to improve the speed of payment processing, while trying to reduce revenue loses due to failures caused by servers unavailability and long pending times.

2.2. Adyen

The most important stakeholder is Adyen, which fully committed to solving the problem of failing transactions, because having a higher authorization rate will surely bring happier merchants, less frustrated shoppers, and finally, more revenue.

Adyen is a technology company that provides a single platform to accept payments anywhere in the world. Currently, the platform processes payments for over 4,500 customers over the world, including: Uber, Airbnb, Netflix, Booking.com, Mango, Evernote, LinkedIn, Spotify, Groupon, etc (figure 2.1).

Figure 2.1.: Adyen customers

The company has its headquarters in Amsterdam and has other 11 international offices: Europe - London, Paris, Berlin, Stockholm, Madrid; North America - New York, San Francisco; South America - Sao Paolo; Australia: Sydney; Asia - Singapore, Shanghai. The payments platform is founded on a robust core technology, which is developed and maintained in-house. It has regular release cycles (every month). Over 250 payment methods (for example, in the Netherlands, there is IDeal, Visa Pay, Apple Pay, Maestro, etc.) and 187 currencies are supported. In many markets, credit cards account for a small proportion of online payments, and local shoppers favor payment methods such as e-wallets or cash on delivery (figure 2.2).

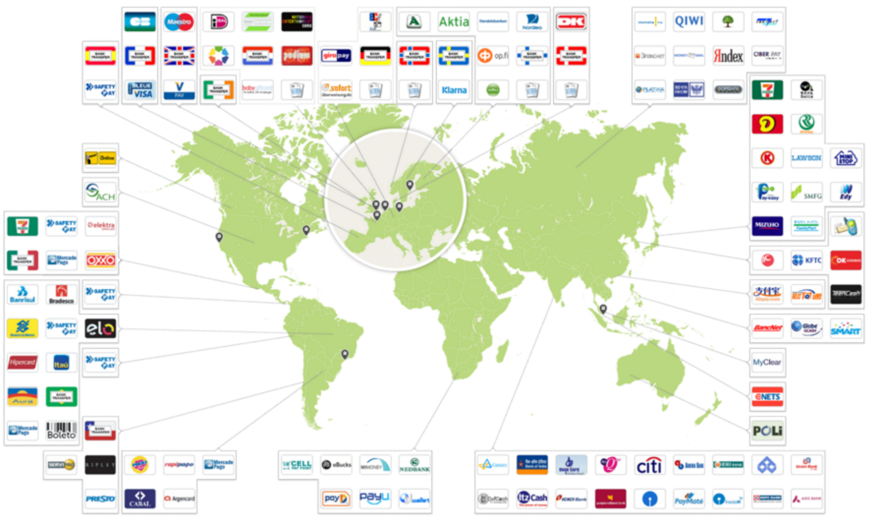


Figure 2.2.: Worldwide offices and payment methods

Adyen is an omnichannel, which means that it includes all payments solutions: social media, e-commerce, mobile, unattended terminals, Point of Sales (POS) and mobile POS (figure 2.3). All these different solutions are important, because they have an impact on the authorization rate of the payments.

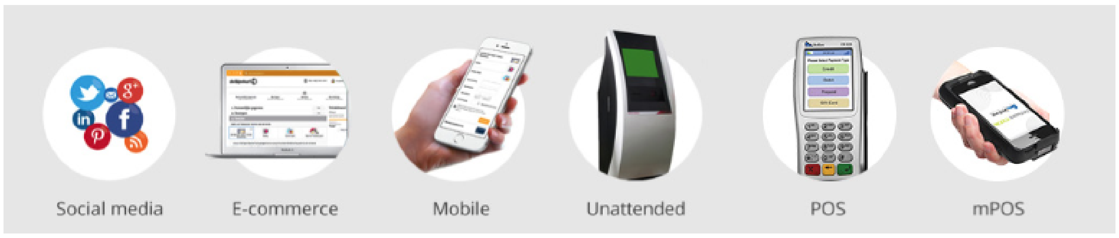


Figure 2.3.: Payment solutions

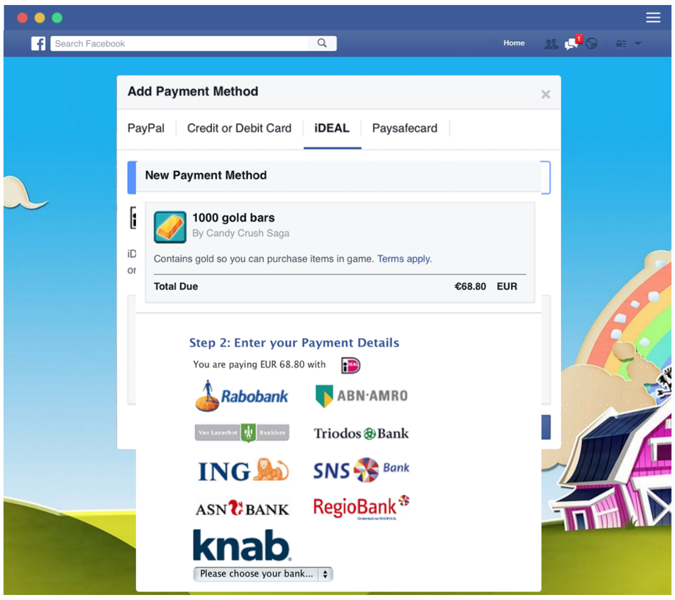
2.3. Conceptual framework

2.3.1. The payment flow

Adyen manages the whole payment flow from checkout, the moment when the shopper decides to buy goods or services, right through to final settlement, the actual money transfer from the shopper's bank account to the merchant's bank account. All using a single platform. This includes:

- Checkout pages for desktop, mobile, and in-store;
- Acquiring services, by processing payments on behalf of the merchant;
- Direct connections to card schemes (e.g.: Visa, MasterCard);
- Complex risk management;
- Comprehensive reports and live monitoring;
- Optimized payments success rate.

When a shopper interacts with a merchant, a payment request is sent to a payment platform, such as Adyen. A payment request is, simply stated, an XML file that contains (confidential) information about the cardholder, merchant, the bank of the shopper, the amount that is requested by the shopper to buy the goods or services, currency, etc. Depending on the implementation, the merchant can host the payment page itself, in which case the merchant is sending the payment request to the payment platform. Otherwise, if the merchant does not want to add this complexity, it can outsource the hosted payment page to the payment platform, in which case, the shopper sends the payment directly to the payment platform. Either way, the type of implementation is (almost) hidden from the shopper. One example of hosted payment page can be seen in figure 2.4, where the shopper can see the product to be bought, the sum and possibility of choosing a payment method (for example, in the Netherlands, iDeal).



Contains gold so you can purchase items in game. Terms apply.

Total Due €68.80 EUR

You are paying EUR 68.80 with

Figure 2.4.: Example of a hosted payment page

Traditionally, since online payments have been introduced, the payment platform was a relatively simple system that had one major role, i.e. to be a gateway for the online payments to the legacy systems of the banks. The system that has only this function is called *Payment Service Provider (PSP)*. But, nowadays the payment platform is much more complex and the PSP has become a component of a much larger system. Next to the gateway, other complementary systems developed, such as: risk management tools, reporting systems and optimization solutions. Newer payment platforms, such as Adyen, merged all of these various systems into a single platform.

The PSP connects to the *acquirer*, which is a bank or financial institution, that processes credit or debit card payments on behalf of the merchant. The acquirer sends the payment requests to the *card schemes* (e.g.: Visa, MasterCard), which are payment networks linked to payment cards. The main role of the card schemes is to set rules and to give guidelines on how card payments should work.

Finally, the payment request reaches the *issuer*, which is the bank of the shopper which issued to him or her the card that is connected to the shopper's bank account. The issuer analyzes the payment request and it does some basics checks, such as verifying the balance of the shopper's account, but also some more complex ones, for example, it verifies the nature of the payment and the merchant, to minimize the probability of charge backs, in case later on the transaction is disputed or it proves to be fraudulent. Nevertheless, the issuer accepts or refuses the payment request, and creates a payment response code to offer extra-information, especially in the case of refusals, such as "Insufficient funds", "Card expired", "Refused with no reason", etc.

The response is sent back on the same pipeline to the card schemes, acquirer, payment platform, merchant and shopper. Based on the response, the merchant can decide if it should send the goods or offer the services. In the same time, the shopper knows whether the transactions was successful or he/she should or should not try again. The architecture framework is described in figure 2.5.

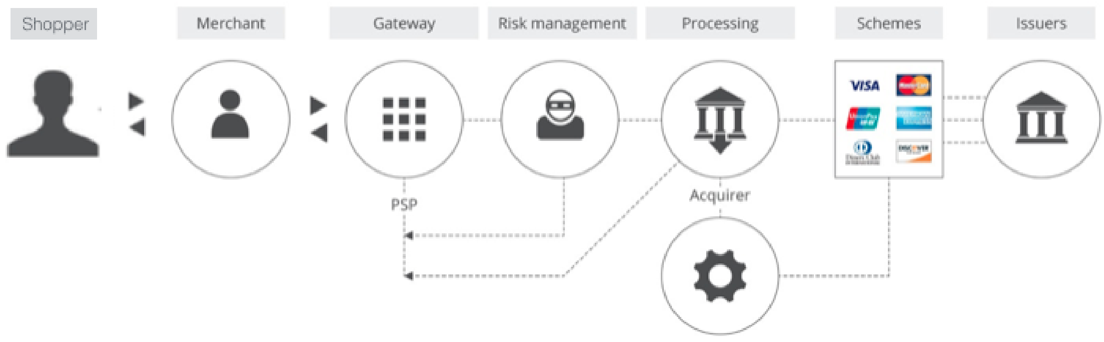


Figure 2.5.: Architecture framework

The components (shopper, merchant, gateway, risk management, acquirer, schemes, issuers), processes (payment flow), constraints (especially card schemes regulations, but also law regulations), and events (payment request, payment response) create the architectural structure around payments. The algorithm to be designed intervenes between the gateway (PSP) and the processing (acquirer).

Understanding the architectural system (the way different parts of the system interacts) is important because it explains why some transactions fail, where exactly they fail, and gives insight on fixing the problem.

2.3.2. Intelligent payment routing

In some countries, Adyen has the possibility to send a payment request to more than one acquirer. Usually, Adyen works with a local acquirer per country and the own Adyen acquirer. Observations and data analyses showed that the choice of an acquirer or the other is relevant for the approval of payments.

The acquirer is the institution that actually formats the payment request following the standards created by the card schemes. Acquirers can ignore any extra information that it receives and can modify some fields to fit these standards.

The framework for routing payments to different acquirers is already integrated in the platform, but the logic on *how* to route payments is missing. The challenge is to decide for every payment the best acquirer it will be sent to. Routing of payments is a good opportunity to take action in increasing the number of authorized transactions. This research focuses on payment routing. In figure 2.6, we see that Adyen (as a PSP) is the one that connects to one or more acquirers.

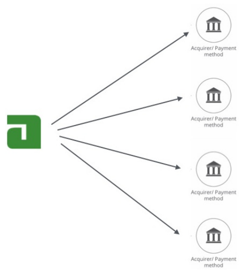


Figure 2.6.: Routing to different acquirers

2.3.3. Authorization rate

Authorization rate is one of the main concerns of both the merchants and the PSP, and also a good indicator of the performance of the solutions proposed by this research.

The observation unit is a transaction. A payment request contains a lot of information, in the order of tens of fields; but, only a handful of them are relevant in the context of the authorization rate. As recommended by the experts at Adyen with a deep understanding of the architectural framework, these are: company, merchant (child of company), transaction variant (card type, such as: MasterCard debit, MasterCard credit, Visa debit, etc.), displayable transaction variant (short version of card type, such as: MasterCard, Visa, American Express, etc.), shopper interaction (explains how the transaction was made, such as: E-commerce or regular online transaction, contAuth or recurring online transaction, accountValidation or transaction with no amount used to check if the account is valid, and POS or Point-Of-Sale), issuer, issuer country, acquirer, creation date (the date when the transaction was made), BIN (Bank Identification Number; usually the first 6 digits of the bank card), currency, and amount.

The variable of interest is the result of a transaction, whether the transaction was a success or a failure. Based on the variable of interest, one can compute the authorization rate:

$$authorization\ rate = \frac{approved\ transactions}{all\ transactions}.$$

The authorization rate over days represents the probability of a transaction to be successful. Previous experiments at Adyen have shown that **in most cases, the authorization rate is unknown and unfixed**, which makes the prediction of the result of future transactions very difficult.

Some information can also be picked up from the detailed response code of a transaction, that mainly says what type of failure was it, such as: “Fraud”, “Invalid amount”, “Canceled”, etc. The population represents all the transactions that run through Adyen's payment platform, whereas a sample population is used that contains just a fraction of these transactions, usually within a time frame.

In table 2.1, one can see the most important variables included in a payment request (the input) and the variables of interest (the output):

Element	**Indicator**	**Example/ Explanation**
Observation unit	Transaction (tx)	A combination of variables + variable of interest
Variables	Company account	Name of the company to which the transaction was made
	Merchant account	Name of the merchant, child of the company
	Tx variant	{mcdebit, mccredit, maestro, visadebit, visaclassic, etc.)
	Displayable tx variant	{mc, visa, amex, etc.}
	Shopper interaction	{Ecommerce, contAuth - recurring Ecommerce, accountValidation - no amount tx, POS - Point Of Sale}
	Issuer name	Name of the issuing bank
	Issuer country code	{es, nl, ro, etc.}
	Acquirer	Name of the acquirer
	Creation date	2016-04-12
	BIN	Usually a 6-digits number, 400000
	Currency code	{eur, usd, etc.}
	Amount	10
Variable of interest	Result of a transaction	1 - Success or 0 - Failure
Secondary variable of interest	Response code of a transaction	{Unknown, Approved, Invalid_amount, Issuer_unavailable, Canceled, etc.}
Sample population	Multiple transactions within a time frame	Usually an aggregation of transactions of the same company, per BIN, per shopper interaction, etc.
Population	All transactions that use Adyen's payment platform	

Table 2.1.: Payment variables

2.3.4. Payment responses

When a shopper tries to buy a product, a payment request is sent to the issuer. The issuer accepts or refuses the request and sends back a response code. Usually, successful transactions have only one response code, while the refused transactions are a lot more explanatory. Moreover, every issuer, card scheme and acquirer have slightly different response codes. Nevertheless, at Adyen, all of the responses codes are mapped to 30 codes. These codes can be split into four categories: success, failure due to discretionary reasons, failure due to technical reasons, failure due to fraud reasons. These codes and categories are explained in figure 2.2.

Response code	**No Codes**	**Category**	**Description**
APPROVED	1	Success	Transactions that are successful
REFERRAL	14	Discretionary	Transactions that are designed to fail in certain situations, such as: transactions cancelled by the shopper or merchant, insufficient funds, incorrect PIN, or CVC, etc.
BLOCK_CARD			
CARD_EXPIRED			
NOT_3D_AUTHENTICAT			
NOT_ENOUGH_BALANCE			
CANCELLED			
SHOPPER_CANCELL			
INVALID_PIN			
PIN_TRIES_EXCEED			
NOT_SUBMITTED			
CVC_DECLINED			
REVOCATION_OF_AUTH			
WITHDRAWAL_AMOUNT_EXCEED			
WITHDRAWAL_COUNT_EXCEED			
UNKNOWN	12	Technical	Transactions that are not designed to fail, but they still due. Causes: implementation deficiencies, invalid input, unknown reason
DECLINED			
ERROR			
INVALID_AMOUNT			
INVALID_CARD			
ISSUER_UNAVAILABLE			
NOT_SUPPORTED			
PENDING			
PIN_VALIDATION_NOT_POSSIBLE			
TRANSACTION_NOT_PERMITTED			
RESTRICTED_CARD			
DECLINED_NON_GENERIC			
ACQUIRER_FRAUD	3	Fraud	Transactions that triggered fraud detection tools of the payment platform, acquirers, card schemes, or issuers.
FRAUD			
FRAUD_CANCELL			

Table 2.2.: Payment response codes

Successful transactions, discretionary failures and fraudulent failures represents a very useful segregation of payment responses. Purely technical failures (invalid input, unsupported transaction, restricted card) and generic response codes (unknown, declined and errors) would not theoretically exist if all of the different processing systems would work perfectly and if they would have a flawless integration between them. One important factor, that causes technical failures to occur, is the change over time in the implementation of each of the systems, separately. These changes are in the form of rules in payment processing imposed by the card schemes or improvements of the payment systems themselves.

Despite the differences between failures (technical, discretionary, fraud), there are some inconsistencies in how each raw response codes are actually tagged by each issuer and acquirer. For example, some issuers might consider "Insufficient funds" as a private information of the card holder, thus sending back generic failure messages: "Unknown" or "Declined".

If we were to concentrate only on technical failures when doing optimization (routing of payments), due to already mentioned inconsistency issues, some acquirers would inevitably and unfairly have more technical failures than the others. This is why, we will take into consideration all failures. This will produce some noise and extra effort in trying to reduce number of failures that, in reality, cannot be saved. On the other side, including all failures will help us reliably compare acquirers, which is our main focus.

2.4. Current implementation

Firstly, Adyen has introduced payment routing by building the core infrastructure in the payment flow. This infrastructure enables getting a list of the available acquirers for specific payments and making the necessary connections to one of these available acquirers. All of this happens in real-time, when each payment request comes in.

Secondly, the intelligent part of payment routing is actually deciding which of the available acquirers is optimal for each payment, given that there are two or more acquirers available. An optimal acquirer is the one that has a higher authorization rate.

2.4.1. Available acquirers

The core part of routing is getting the available acquirers and making sure that once they are declared available, the necessary connections can be made to them.

As described in table 2.1, a payment has many variables. In order to get the available acquirers, the most important variables are the *issuer country* and *company account*. Adyen has contracts with various acquirers in each country, that allows them to process payments made within those countries. For example, in France, there are two acquirers that have contracts and connections with Adyen, whilst in U.S., a specific company (client of Adyen) has costs contracts with two other acquirers in this country. Usually, Adyen offers either one connection to a domestic acquirer (local acquirer within one country), or two connections, a domestic and an international acquirer (an acquirer that is used to process payments in multiple countries). Nevertheless, there are a few cases where there are other combinations (e.g.: two domestic acquirers), but also cases where there are more than two acquirers.

Having multiple available acquirers is helpful because of many reasons, such as:

- assures a robust and reliable connections for payments; acquirers can fail, or can have difficulties processing some payments;
- acquirers can offer different prices and services; using some acquirers can be more expensive than others, and they have different quality of service in terms of uptime, currencies processing, and data-insights;
- acquirers have different authorization rates for different types of payments and markets; for clients, it is very important to have the highest possible authorization rate;

2.4.2. Ordering of available acquirers by authorization rate

Ordering of the available acquirers is applicable for the cases where there are two or more available acquirers. Given that the authorization rate is very important in the world of payments, the ordering is done by authorization rate. However, it is not always clear which acquirer has a higher authorization rate.

Currently, at Adyen, assuming that there are two available acquirers, the ordering of the available acquirers is done by a data analyst, following these 3 steps:

1. gathering of data;
2. data analysis;
3. routing configuration.

In the first step, gathering of data, the data analyst takes data from previous payments. He or she must make sure that there is payment data for both acquirers that will be compared.

Usually, companies have always processed payment through one acquirer and want to see whether using a different acquirer will bring a higher authorization rate. In this case, because only one acquirer has been used to processed payments, there is no available data for the second acquirer. Thus, the analyst needs to route some payments to the second acquirer, to have some data on it. Usually, a static configuration of 95% - 5% of the payment traffic is made, with the majority of payments still going to the traditional acquirer. If there has been a positive experience with the second acquirer in other contexts or the exploration needs to be done fast, then a configuration of 50% - 50% of the payment traffic to the two acquirers is also acceptable.

In the second step, the data analysis, the analyst extracts from the database the payment data of the transactions made trough the two acquirers. The payment data contains many of the variables described in table 2.1. The most important information is the success or failure of transactions by each acquirer. The analyst does the following operations on the data:

- *data cleaning*: some transactions are excluded from the analysis, such as transactions that failed before reaching any of the two acquirers (in most cases, because of a fraud detection);

- *descriptive statistics*: to provide basic summaries about the sample data; the most important information is to make sure that the payments are equally distributed between the two acquirers, for the same variables; for example, some issuing banks have a higher authorization rates than others, shopper interaction *account validation* for 0$ amount transactions has a considerable higher authorization rate, and the payments of various merchants can be very different between each other (because of the nature of their businesses). Thus, it is very important to assure a fair comparison between the two acquirers; in this step, any bias towards an acquirer is eliminated;

- *authorization rate comparison*: the analyst computes the general authorization rates per merchant and acquirer; however, in most cases, the comparison goes on a finer scale, taking into account other variables than just the merchant and acquirer, such as: BIN (the most important one), transaction variant (e.g. Visa/ MasterCard/ Amex), and shopper interaction.

At the third and last step, routing configuration, Adyen uses the authorization rate comparisons from the previous step and draws conclusions on which acquirer has an higher authorization rate.

These conclusions represent the support for creating a new routing configuration. A routing configuration is actually a distribution of payments between the two acquirers (for example, 75% - 25%).

The most valuable routing configurations are those that can be generalized to multiple companies or to an entire country. For example, the conclusion can be that there is enough reasons to believe that in Spain, acquirer A has a higher authorization rate than acquirer B in 95% of the payments. In this case, we create a routing configuration that sends most, if not all, payment requests to acquirer A.

However, it is very difficult to get to such solid conclusions. Usually, the data analyst analysis data per company and BIN and makes routing configuration accordingly, without much generalization to other companies or other BINs.

2.4.3. Problems with the current implementation

There are multiple problems with the current implementation, that refer to the ordering of the available acquirers.

The main problem is the reliability of the result of the authorization rate comparisons, because there is no theoretical or scientific method that applies to this comparison.

More specifically, it is up to the data analyst to decide what actually means that an authorization rate for one acquirer is higher than the authorization rate for the other acquirers. The two main factors that help them conclude a better acquirer are: the number of transactions and the difference in authorization rate. Thus, if there is a "significant" difference between the authorization rate of the acquirers (e.g.: higher than 1% difference) and there are enough transactions (e.g.: higher than 50 transactions for both acquirers) then the analyst reaches a conclusion. Nevertheless, there are all kind of complications and deviations. For example, if 90% of the BINs for one company have only 10-20 transactions and the authorization rate between the acquirers is very large (50%), then the data analyst might still conclude that one acquirer is better. This creates inconsistent thresholds regarding the necessary number of transactions and the authorization rate difference necessary to reliably consider one acquirer better than the other.

The second problem is that the authorization rate of the acquirers changes in time. Thus, it is very difficult to assure that the conclusions made today, about the performance superiority, will still be valid tomorrow.

Thirdly, the whole process of ordering of available acquirers by authorization rate is cumbersome. It is very time consuming and takes a lot of resources from the data team. There is a trade-off between making an analysis for an entire country, applicable to all companies, which is inevitably less accurate, and making tens or hundreds of analyses for each company, which are more accurate, but they need much more effort and maintenance.

2.5. Research questions

In between the top-level research goal defined in section 1.2 and the current implementation problems exemplified in subsection 2.4.3, there are many other research questions. They will get us closer to a full understanding of the problem and will enable finding an adequate design for it.

The classification of the research questions uses the groups defined in chapter 2 of the book [1]. We distinguish two important types of questions that derive from the top-level research goal:

1. **Descriptive**
    a) What is the scope of the routing of payments?
    b) How many payments are affected?
2. **Explanatory**
    a) What are the reasons that cause a payment to fail?
    b) What are the influences of different variables on the payments?
    c) How much does the performance of the acquirers change in time?
    d) What is the expected improvement that a new algorithm will bring?

This research focuses on designing a solution for efficiently routing the payments. Looking more in-depth into the design, there are other types of important questions:

1. **Effect and requirement satisfaction**
    a) What effects does the algorithm have on payments?
    b) What is the execution time when using the new algorithm?

   c) What is the usability of the new algorithm for the payment platform?
   d) Does the algorithm satisfy the requirements of the stakeholders?

2. **Trade-off and sensitivity**
   a) How do different algorithms perform?
   b) How do different parameters of the same algorithm perform?
   c) Can the algorithm be applied to other contexts?

3. Literature review

3.1. Comparing two proportions

Two of the most basic methods of making statistical inferences are hypothesis testing and confidence intervals of the difference between two proportions. In the case of binomial distribution, where the random variable can be 1 or 0, yes or no, success (s) or failure (f) over n experiments a proportion p is the number of successes over the total number of experiments. These two parameters are used to describe the population of random variables.

Hypothesis testing involves having two proportions p_1 and p_2. In our case, these proportions describe the authorization rates for different acquirers. The null hypothesis H_0 assumes there is no differences between the two proportions. The alternative hypothesis H_a assumes there is a difference between the two proportions.

$$H_0 : p_1 = p_2,$$

$$H_a : p_1 \neq p_2.$$

The test can have two outcomes. First, when we have enough evidence to reject the null hypothesis and to accept the alternative hypothesis. Second, when there is not enough certainty to reject the null hypothesis, thus we cannot accept the alternative hypothesis.

If the data is sampled independently and there are enough samples, then the sampling distribution of $\hat{p}_1 - \hat{p}_2$ (proportion estimators) is approximately normal and we can use the Z-methods.

The test statistic is:

$$z = \frac{(\hat{p}_1 - \hat{p}_2)}{\sqrt{\hat{p}(1-\hat{p})(\frac{1}{n_1} + \frac{1}{n_2})}},$$

where,

$$\hat{p} = \frac{s_1 + s_2}{n_1 + n_2},$$

$$\hat{p}_1 = \frac{s_1}{n_1},$$

$$\hat{p}_2 = \frac{s_2}{n_2}.$$

The z value is then compared to the appropriate value from the Z-table, depending on the desired confidence level, and the conclusion is one of the two possible outcomes, i.e. to accept H_0 or H_1.

In any test statistic there are two types of errors that can be made: type I error and type II error. A type I error is when we reject the null hypothesis, in favor of the alternative hypothesis, when the null hypothesis is in fact true. A type II error is if we fail to reject the null hypothesis when the null hypothesis is false.

Type I errors are not relevant in the case of improving authorization, because if the options have equal authorization rate, then there is no loss in sending more transactions to one of them. On the other side, type II errors are crucial, because making these errors means the traffic is sent to the sub-optimal option.

Comparing two proportions with the Z-test is a neat way to find out which acquirer is better. However, there are two important assumptions: the observations are independent and authorization rates for the two acquirers are fixed. The former can be assumed, because payments are in general independent. The latter, in practice, cannot be guaranteed because the authorization rates of the acquirers tend to variate from time period to another.

Moreover, comparing two proportions is not recommended to be used for prediction, but rather for describing an (existing) population, based on sampled data, especially in the cases where the success rates are unfixed.

3.2. Randomized experimental design

Randomized experiments is a designed experiment that allows the random allocation of units to different treatment groups. In many empirical experiments, one can have a reasonable expectation of the result when applying a treatment to a context. However, there are cases in which the context is more of the black-box and one cannot know whether the treatment has a positive effect, negative effect, or any effect at all. Nevertheless, the treatment is still worth trying so that we get this information and potentially improve the context. In other words, [2] suggests that randomized experiments are appropriate when there is little or no knowledge about what is going on, but "treatment (i.e. chemicals) like this seem to have some effect on problems like that; let's see what happens".

Clinical trials were the starting point for this type of experiments and are arguably the most popular applications for randomized experimental designs. There are numerous examples of applications in this domain ([3], [4], [5]). Even though clinical trials do not have much to do with payments, the knowledge on randomized experimental designs from this domain can be extended to payments as well. In the mentioned papers, the basic randomized experiment consists in randomly dividing the patients into two groups: control and test. Doctors have two types of treatments, the real medicine for the test group, and a fake medicine with no effect for the control group. The patients are not aware of which treatment they receive, and in most cases, the doctors are not aware either.

There is a great diversity of types of randomized experiments, such as: constraint randomization (randomization in permuted blocks, stratified randomization), and adaptive randomization (treatment assignment probability changes during the trial).

This research will focus on two popular types of randomized experiments: A/B testing and Play the Winner Rule (adaptive design).

3.2.1. A/B Testing

Marketing and internet advertising is a very common place for randomized experiments that intensively use A/B testing. In most cases, one tries to measure the success of different treatments that try to increase the conversion rate on a website. For example, one can try two colors, two text boxes, or to positions on the website for a Call-to-Action button, such as: "Try it now!". Another example, is when merchants are sending their payment traffic to two payment platforms and compare the two authorization rates. Finally, they might decide to send the majority of their transactions to the better one.

A/B testing algorithms are randomized experiments with two variants, A and B, which are the control and variation in the controlled experiment [6], respectively. The logic behind A/B testing is that one sends a percentage of traffic to variant A, and the remaining of the traffic to variant B (usually the percentage is set to 50%). After reaching to the variants, one can compute the successful rate/conversion rate of the traffic (the number of completed processes over the total number of accesses; e.g.: ratio of paying customers, or authorization rate). After a sufficient number of traffic, one can decide which variant is better, A or B. Even though A/B testing are very often seen in the optimization of web pages, these algorithms have also been tried in the payments industry; for example, to see if a new payment solution is better than an old one (success stories can be found in [7]).

The most common A/B testing algorithm is the fixed-size A/B testing. This mainly requires setting a *significance level*, a *minimum detectable effect* (the smallest effect that will be detected), and a *statistical power* (percent of time the minimum effect size will be detected, assuming it exists). These settings are important to guard against type I and II errors. One can calculate the minimum (fixed) sample size needed to confidently assess the difference between the two populations. After collecting the data, we can decide which of the variant is better.

A different flavor of A/B testing is sequential A/B testing, as presented in [8], [9], [10] and [11]. This algorithm is trying to identify as quick as possible if one variant is better than the other. This advantage can be exploited especially in situations where real-time streams of data are available.

In principle, the same parameters are needed for sequential testing as for fixed-size testing. The main difference is that the test can stop earlier, if the two options diverge enough one from the other. As it results from the cited sources, sequential A/B testing becomes extremely useful especially when one option clearly outperforms the other. In this case, one does not need to continue the experiment until the fixed-size number of samples is reached. In the case of payments, many otherwise sacrificed transactions can be saved by applying sequential A/B testing and the fixed-size one.

No matter what type of A/B testing is conducted, one needs to compare the two proportions with similar methods as the one explained in 3.1 to assess whether one acquirer actually performs better than the other. Thus, the assumptions about observation independence and fixed authorization rates continue to be a challenge even for A/B testing.

3.2.2. Play the Winner Rule

Traditional balanced randomization experiments have an important shortcoming: dealing with sensible (negative) results during the trial. This shortcoming is especially evident in the case of clinical trials, when a possible result of an experimental treatment can go as far as death for the patient.

An adaptive approach of randomized experiments proposes to change the manner in which the treatment is applied during the trial, so that we avoid negative effects on patients as much as possible. For example, we have two groups of patients: one that takes the treatment and the other that take no treatment (or just a placebo). During the experiment is observed that the treatment is performing very well, whereas the placebo is performing very poorly for the other patients, then patients from the second group can be redirected to the first group, so that they can benefit from the treatment early on. In the same way, if the treatment is observed to not work at all, the experiment is designed to be stopped before the term, without great loss.

Likewise, in the online payment domain, a treatment that is tested can cause all payments to fail, thus losing a significant amount of money. Of course, in these extreme cases, the experiment can just be shut off; but, this represents a huge risk for the research, because no conclusions can be drawn, and the treatment that has successfully been applied will stop as well.

Play the Winner Rule (PWR) is a simple, but effective implementation of the adaptive randomized experimental design. There are many forms and applications of this method, as described many specialized papers, such as: [12], [13], and [14]. In these papers, the method tends to successfully apply the treatment more often than in traditional randomization experiment.

Similar to the definitions in [13], assume there are two treatment groups "A" and "B", and there are n observational units - OU - (i.e.: patients, transactions, etc.). We define T_i the treatment applied to the i^{th} observational unit, and Y_i the result of the treatment applied to the i^{th} observational unit (success or failure):

$$T_i = \begin{cases} 1, & \text{for the A treatment, applied to the } i^{th} \text{ observational unit,} \\ 0, & \text{for the B treatment, applied to the } i^{th} \text{ observational unit;} \end{cases}$$

$$Y_i = \begin{cases} 1, & \text{if the treatment applied to the } i^{th} \text{ observational unit was a success,} \\ 0, & \text{if the treatment applied to the } i^{th} \text{ observational unit was a failure.} \end{cases}$$

The success probabilities of the two treatments are $p_A = P(Y_i = 1 \mid T_i = 1)$ and $p_B = P(Y_i = 1 \mid T_i = 0)$. The number of observations units that used treatment A and B are $N_A = \sum_{i=1}^{n} T_i$ and $N_B = n - N_A$, respectively.

In the simplest form, the assignment of observational units (OU) to treatment groups goes like this: first OU goes to a random treatment group, say A. The success probabilities p_A and p_B are updated. The next OU goes to the one with the highest success probability. And the process is continued.

In paper [15], the author discusses about how Play the Winner Rule handles practical and theoretical issues such as stratification, delayed responses, variability, selection of parameters, and inference.

Sticking to the clinical experiments, stratification refers to the segregation of the patients into multiple homogeneous strata and afterwards, each stratum will be divided into the two groups: patients that receive the treatment, and patients that receive the placebo. Variability of the binomial variables is often neglected in adaptive designs, due to their inherent processes that induce randomness. However, it is shown in [15] that Play the Winner Rule for small sample size (e.g.: 10 observations) is extremely variable, thus in these cases, the actual usefulness it questionable. As the sample size increases (>100 observations) the variability of the results decreases.

Delayed responses is a very common problem of any adaptive design. In its most raw form, Play the Winner Rule has all the information up to the ith patient, and based on this knowledge, takes an action for the next $i+1$th patient. Of course, this is not the case in practice. Paper [16] describes a multistage solution that updates the statistics only at certain time points, and not instantaneously and continuously. However, unless there is a very long delay (such as in experiments testing survival rate), the delay causes a slow adaptation, and there will be less patients that benefit from the better treatment.

3.3. Reinforcement learning

According to [17] reinforcement learning focuses on gaining information as interacting with the environment. Say we have a set of possible actions, and each action brings a different reward. The objective is to maximize the total reward in time, but there is a central problem: we do not know the reward each action brings. Naturally, if the rewards were known, then the problem would have been merely trivial by always going for the action with the highest reward.

Another similar research [18] names reinforcement learning as a trial and error approach in which an agent operating in an environment learns how to perform a desired task in that environment. The agent learns by adjusting its policy on the basis of positive (or negative) feedback (reinforcement). This feedback takes the form of a scalar value generated by the agent each time step, high and low values corresponding to rewards and punishments respectively. The mapping from environment states and agent actions to reinforcement values is termed the reinforcement function. The agent converges to the behavior maximizing reinforcement (the optimal policy). In theory an appropriate reinforcement function exists for all task; although, finding such a function is typically hard.

Reinforcement learning is similar to machine learning, but it has three distinguishing characteristics: closed-loops, unknown next action, and unknown reward distribution. First, closed-loops refers to the fact that each action serves as an input for the next action. Second, the order of actions is not given; in the simplest form, one is able to take any of the possible action at any time. Thus, unlike supervised learning, the feedback does not indicate which behavior is correct, but how good or bad the current behavior is relative to others. Taking different actions brings different rewards, and one does not know beforehand the reward that is received. The objective is to eventually converge to the optimal solution, maximizing the reinforcement function.

Some of reinforcement learning implementations are: Markov Decision Processes, multi-armed bandit, dynamic programming, Monte Carlo methods, temporal-difference learning, eligibility traces, approximate solution methods, etc.

Domains where reinforcement learning is applied are rather diverse. Artificial Intelligence is one of the prominent examples, especially in the research of creating mobile robots. These robots need to make many decisions and they should be "optimal“, given the environment in which they find themselves. It is difficult, or even impossible, to program the robots to have a decision ready for any type of event that can occur. Rather, they should learn from the environment, and make the best decision from the available options. In [18] are given many examples on how reinforcement learning has been inspired by behaviors and mechanisms observed on animals, and how they are desired to be adapted and perfected for mobile robots.

Other example applications are shown in [17], such as psychology, neuroscience, and even elevator dispatching. For the last one, the algorithm tries to learn patterns of how passengers take the elevators in a building: if passengers from different floors request the pickups, who should be served first? If there are no pickup requests, how should the elevators distribute themselves to await the next request?

A wide range of applications is given in [19], especially related to economics: experimental consumption, market learning with unknown demand, labor market, investment in innovation or current projects, but also minimizing the delay in a network. Paper [20] mentions domains as packet routing, online auctions, assortment selection and online advertising.

Nevertheless, the most popular utilization of reinforcement learning is within clinical trials ([21], [22], [23]), and gambling ([24], [25], [26]). In clinical trials, usually the primary goals is to use an adaptive strategy to confirm earlier results of the treatments, but also to minimize the number of patients affected by the side effects in trying to get these results.

Gambling serves mostly as a model and it is somewhat different, because it represents the source problem from which the multi-armed bandit implementation of reinforcement learning has emerged. If we consider the unknown and unfixed authorization distribution as characteristics of the intelligent payment routing, then multi-armed bandit implementation makes a good candidate for a solution.

3.3.1. Multi-armed bandit

Multi-armed bandit is an implementation of reinforcement learning, and it mainly focuses on the trade-off between trying the available actions and going with the best one up to the moment. This trade-off is called exploration - exploitation ratio. In a gambling model, bandit problem is pictured as a set of slot machines in a casino, where each slot machine is an arm. An action is defined as the choice of a specific arm. One action can be taken at a time, so each arm-pull represents one step. In the beginning, the player does not know which arm brings more reward, but decides to "explore" the arms and to observe the reward distributions. The final objective is to maximize the reward over a finite or infinite number of slot arm pulls. Due to its adaptive nature, multi-armed bandit can be categorized as an adaptive randomization designed experiment.

Figure 3.1.: Slot machines example used by the multi-armed bandit problem

Multi-armed bandit has multiple versions, depending on the context in which it is applied. Some of them are simple and straightforward, but others are rather complex and try to address very specific problems. Nevertheless, the core concepts stay the same for all versions of bandits.

More formally, we have k different options, actions, or arms (in the gambling example) $a_1, a_2, \ldots, a_k$. Each arm pull is measured in time steps t, where the total number of time steps is called the *horizon* H. Moreover, every arm pull outputs a reward R, that follows unknown distributions corresponding to each arm a. The performance of the arms is measured as an expected reward or preference indicator Q_{t+1}, based on the rewards received up to and including t.

$$Q_{t+1}(a_i) = E[R_t \mid A_t = a_i], \text{ where } i \in \{1, 2, \ldots, k.\} \tag{3.1}$$

The way the expected rewards functions $E[\bullet]$ and ratios between exploration and exploitation are chosen is called a *bandit strategy.* A measure of performance for different bandits is the regret ρ. The regret is the difference between always taking the best decisions and the rewards that the bandit has managed to obtain. The best decision means that at each time step t one picks the best arm, which should bring a maximum reward Q_{max}. In practice, knowing the maximum reward is impossible; but, looking retroactively, it is helpful to use this concept to measure the effectiveness of the algorithm. The regret up to and including time step t is:

$$\rho_t = t * Q_{max} - \sum_{s=1}^{t} Q_s. \tag{3.2}$$

As mentioned in [27] a strategy whose average per time step $\frac{\rho_t}{t}$ tends to zero when the horizon of time tends to infinity is a *zero-regret strategy.* Intuitively, zero-regret strategies are guaranteed to converge to an optimal strategy, not necessarily unique, if enough rounds are played. The following subsections will dive into various bandit strategies, based on different evaluations made in [27] and other previously cited papers.

3.3.2. ϵ-greedy bandit strategy

Based on [17] (similar opinions are shared across the literature) ϵ-greedy bandit is one of the simplest strategy, thus being widely spread. This strategy chooses a random arm with a frequency of ϵ (exploration), and for the rest of arm pulls it goes with the one with the maximum expected reward Q (exploitation). Usually the exploration is done throughout the experiment, but if the horizon is finite, the experiment can start by first doing the exploration. Given that A_t is the action chosen at time t, the strategy of choosing the arm to be played at the next time step *t=1* can be formulated as follows:

$$A_{t+1} = \begin{cases} \text{arg } max_{a_i} Q_t(a_i) & \text{with probability } 1-\epsilon, \\ \text{a random action } a_i & \text{with probability } \epsilon. \end{cases} \tag{3.3}$$

We assume that the reward function is simply the mean of rewards of each arm a_i, where t_i is the total number of arm pulls that have gone through arm i, and $R_{s,i}$ is the result of arm pull s of the arm i:

$$\begin{aligned} Q_{t+1}(a_i) &= \frac{\text{sum of rewards when } a_i \text{ taken up and including } t}{\text{number of times } a_i \text{ taken up and including } t} \\ &= \frac{\sum_{s=1}^{t} R_s * 1_{A_s=a_i}}{\sum_{s=1}^{t} 1_{A_s=a_i}} \\ &= \frac{\sum_{s=1}^{t_i} R_{s,i}}{t_i}, \text{ where } i \in \{1,2,\dots,k\}. \end{aligned} \tag{3.4}$$

If one option is outperforming the other, the simple ϵ-strategy will not have very good results in the long term, because it will constantly trying the low performing option instead always choosing the best arm. To fix this, a different variant of this strategy considers that as the time passes, the expected rewards $Q(a_i)$ become more accurate, so there is no need to have the same amount of exploration, thus it reduces the ϵ. The strategy that has a decreasing ϵ_t is called ϵ-decreasing strategy.

For the ϵ-decreasing strategy is the same, except for ϵ, which is now ϵ_0 and can be selected by the user, and for every round t we have multiple possible ϵ_t functions (the logarithmic version is called the *GreedyMix* strategy). For example:

$$\epsilon_t = \frac{\epsilon_0}{t}, \text{ or } \epsilon_t = \frac{\log(\epsilon_0)}{t}.$$

A different perspective on the ϵ-greedy strategy is the *LeastTaken* strategy, which does the exploration based on how often are the different options tried. Thus, in [28] suggests that the least taken arm should be tried with a probability of $4/(4+m^2)$, where m is the number of times that arm has already been tried.

3.3.3. Pursuit bandit strategy

In [17] and [23], an interesting approach on bandit is elaborated to deal with non-stationary environments, i.e. when the probability of success μ changes in time. The suggestions is to include a recency parameter α that weights recent rewards more heavily than long-past ones. In the same paper, a short mathematical proof shows that the equation (3.4) for each arm a_i is equivalent to:

$$\begin{aligned}
Q_{t+1}(a_i) &= \frac{1}{t_i} * \sum_{s=1}^{t_i} R_{s,i} \\
&= \frac{1}{t_i} * \Big(R_{t,i} + \sum_{s=1}^{t_i-1} R_{s,i}\Big) \\
&= \frac{1}{t_i} * \Big[R_{t,i} + (t_i - 1) * \frac{1}{t_i - 1} * \sum_{s=1}^{t_i-1} R_{s,i}\Big] \\
&= \frac{1}{t_i} * \Big[R_{t,i} + (t_i - 1) * Q_t(a_i)\Big] \\
&= \frac{1}{t_i} * \Big[R_{t,i} + t_i * Q_t(a_i) - Q_t(a_i)\Big] \\
&= Q_t(a_i) + \frac{1}{t_i} * \Big[R_{t,i} - Q_t(a_i)\Big], \text{ where } i \in \{1, 2, \ldots, k\}. \qquad (3.5)
\end{aligned}$$

First, equation (3.5) is more practical because the state consists of only one variable Q_t, and not the list of all $R_1, R_2, \ldots, R_{t-1}$ for the arm i. Second, the time step size is $\frac{1}{t_i}$, which will be denoted as $\alpha_t(a_i)$. This time step size is *incremental* and it decreases from step to step. In the case of a non-stationary environment, when the rewards of the arms change in time, using $\alpha_t(a_i)$ is not appropriate, as new arm pulls that are supposed to be more accurate will have little impact on Q_t. To solve this shortcoming, [17] proposes a *constant* $\alpha \in (0, 1]$. As this time step size increases and comes closer to 1, the more important the latest arm pulls are for deciding the arm assignment for the next arm pull.

$$Q_{t+1}(a_i) = Q_t(a_i) + \alpha * \Big[R_{t,i} - Q_t(a_i)\Big], \text{ where } i \in \{1, 2, \ldots, k\}. \qquad (3.6)$$

Another way to increase the flexibility of the exploitation function $Q(a)$ is to manipulate the initial values $Q_1(a_i)$, if there exist information prior to starting the algorithm about a more rewarding arm. For example, setting $Q_1(a_1)$ = +5, $Q_1(a_2)$ = -5 is highly biased towards the first arm, which can be a advantage for the exploitation, following that for the next arm pulls, it will adapt to the optimal value, but without sacrificing transactions in the beginning if one arm is known to be better, *at that time*, than the other. Nevertheless, this prior values is left to the user to be set, with the default of 0.

3.3.4. Upper-Confidence-Bound bandit strategy

If there are more than two arms, then another improvement opportunity arises. The bandit strategies, presented up to the moment, select a random arm in the exploratory phase. But, what if there are ten arms in total, one of them is currently used for exploitation, two are performing very poorly, and two others are performing almost as good as the exploitation arm. In this case, we would want to explore especially the two challenger arms and very rarely explore the two bad performing arms. With this in mind, one can take into account both the expected reward and the potential for actually being optimal, and not only selecting a random arm in the exploration phase. Upper-Confidence-Bound (UCB) strategy has many versions, as presented in papers as [23], [27], [25], but the version presented both in [17] and [29] is very comprehensive:

$$A_{t+1} = \arg {}_a max \left[Q_t(a_i) + c * \sqrt{\frac{\log t}{N_t(a_i)}} \right]. \tag{3.7}$$

In equation (3.7), compared to equation (3.3), we have $N_t(a_i)$ for the number of times action has been selected prior to time t, and $c > 0$ to control the degree of exploration. However, for non-stationary problems, one needs to take into account the more complicated method of the pursuit bandit strategy (see section 3.3.3). An adaption of *UCB* for non-stationary problems is presented in [24].

3.3.5. Gradient bandit

The gradient bandit, as presented in [17], does not use estimated action values to select the best action, but instead takes a different approach: this bandit uses a numerical preference $Q_t(a)$ for each action a at time t. The preference depends on the rewards of the action, but does not have any particular meaning; the action's preference is important only when comparing it to another action's preference.

The action probabilities are determined based on Gibbs or Boltzmann distribution. This distribution is often used in other bandit algorithms (e.g.: Gittins index, SoftMax). Based on action probabilities of being the preferred one p_t, A_t the action for time t is selected. The probabilities are computed as follows:

$$p_t(a_i) = \frac{e^{Q_t(a_i)}}{\sum_{j=1}^{k} e^{Q_t(a_j)}}, \text{ where } i \in \{1, 2, \ldots, k\}.$$

The important aspect of the gradient bandit is the preference update after each time t. Same as the pursuit bandit explained in section 3.3.3, it takes into account the *step size*, $\alpha > 0$, so that the preference weights more the most recent transactions than the older ones. More, the preference uses the idea of stochastic gradient ascent for updates. Considering R_t the reward after time t and $\bar{R}_t$ the average of all rewards up through and including time t. The preferences can be initiated with same values: $Q_1(a_i) = 0, \forall i \in \{1, 2, \ldots, k\}$. The update of the preferences is done as follows:

$$Q_{t+1}(a) = \begin{cases} Q_t(a_i) + \alpha(R_t - \bar{R}_t)(1 - p_t(a_i)) & a_i = A_t \text{ the selected arm,} \\ Q_t(a_i) - \alpha(R_t - \bar{R}_t)p_t(a_i) & \forall a_i \neq A_t \text{ other arms.} \end{cases} \tag{3.8}$$

It is worth noticing that the gradient ascent is slightly changed, because it has usually only measured the effect of the increment R_t on the preference Q_t. Nonetheless, [17] proposes the version of gradient ascent described in equation (3.8) that uses the baseline $\bar{R}_t$ as an extra factor. If the reward is higher than the baseline (the averaged rewards), then the preference of the selected reward is increased; otherwise, if the reward is lower than the baseline, then the preference is decreased.

3.3.6. Exponential-weight algorithm for exploration and exploitation

Many of the bandit strategies have different assumptions about the statistics of the arms. Exponential-weight algorithm for exploration and exploitation (henceforth referred to as *Exp*) tries to detach from these assumptions and admit not having complete control over the rewards. In the paper [30] various forms of *Exp* are presented in full detail.

One of the *Exp* forms is *Exp3.1*. This version is the most appropriate for the problem raised by this research, both from a implementation point of view and because of the input assumptions.

Using the notations defined in (3.1) and (3.2) *Exp3.1* assures that the regret ρ is bounded by $O(\sqrt{kQ_{max}lnk})$ and it holds uniformly over time, where k is the number of arms and Q_{max} maximum reward as if we were to choose the best arm at each time t.

An interesting aspect of *Exp3.1* is that it runs in Γ epochs that divide the time t. The epochs are used to restart some indicators so that we do more exploration at the beginning of each epoch. As the time passes, the epochs are longer and longer.

In paper [30] is given a pseudocode for this algorithm. The first step is an epoch initialization:

$$\Gamma = 0.$$

Exp is aiming at securing a boundary for the regret. However, to measure the regret, the maximum expected reward Q_{max} is needed, which in most cases is difficult to have a value for. Thus, *Exp31* proposes an approximation for Q_{max}, computed per epoch, as follows:

$$q_\Gamma = \frac{k * lnk}{(e-1)4^\Gamma}. \tag{3.9}$$

As in the case of the gradient bandit (subsection 3.3.5), probabilities $p_t(a_i)$ are computed for selecting A_t, but with a different function:

$$p_t(a_i) = (1 - \gamma_\Gamma)\frac{\omega_t(a_i)}{\sum_{j=1}^{k} \omega_t(a_j)} + \frac{\gamma_\Gamma}{k}. \tag{3.10}$$

Exp uses γ_Γ as a parameter for more control over the probabilities defined at (3.10). This parameter can be tuned as:

$$\gamma_\Gamma = \min\left\{1, \sqrt{\frac{k * lnk}{(e-1)g_\Gamma}}\right\}.$$

To compute (3.10), ω is a variable similar to the expected reward Q, that it is used to calculate the probabilities of an arm to being the best one, based on previous results:

$$\omega_1(a_i) = 1 \text{ for i = 1, ..., k.}$$

The reward is corrected by the probability of the selected action of actually being the optimal:

$$\hat{R}_t = \frac{R_t}{p_t(A_t)}.$$

The variable ω is updated only for the selected arm A_t, as follows:

$$\omega_{t+1}(A_t) = \omega_t(A_t)exp(\frac{\gamma_\Gamma \hat{R}_t}{k}).$$

The expected reward is also updated:

$$Q_{t+1}(A_t) = Q_t(A_t) + \hat{R}_t.$$

If the following condition is fulfilled, then the epoch Γ is incremented. Otherwise, the epoch stays the same. In any case, the steps from (3.9) on are then repeated:

$$maxQ_t > q_\Gamma - \frac{k}{\gamma_\Gamma}.$$

3.3.7. Thompson sampling

Thompson sampling is a simplistic heuristics for exploration and exploitation trade-off. This strategy focuses on giving each arm a probability of being the optimal by using a *Beta distribution*. As always, the arm with the highest probability if being optimal is chosen. In paper [31], Thompson sampling is presented in much more detail, along with an example application for an online advertising application to decide which ads should be shown based on the highest click-through rates.

In the case of a Bernoulli bandit (rewards are 1 or 0), we define $S_t(a_i)$ and $F_t(a_i)$ the number of successes of arm a_i up to time t and the number of failures, respectively. The probabilities of each arm of being the optimal one are computed using the Beta distribution, as follows:

$$p_t(a_i) = Beta[S_t(a_i), F_t(a_i)].$$

For the selected arm (based on the highest probability) A_t, the reward is observed, and for $S_{t+1}(A_t)$ or $F_{t+1}(A_t)$ are incremented.

Originally, Thompson sampling does not have any parameter. However, in the above mentioned paper it is shown that better performance can be achieved by reshaping the posterior of the Beta distribution, or tweaking the Beta prior parameters.

3.3.8. Other strategies

There are an impressive number of bandit strategies developed over time that try to target specific problems. For the sake of completeness, this subsection enlists a couple of other strategies.

John Gittins is one of the pioneers of solving multi-armed bandit problems using index policies. The main idea is to compute a number (called an index) for each of the bandit arms and assign arm pulls to the arm with the greatest index. This method is very useful for job scheduling, and it originates from the book [32]. This book defines k independent reward-producing Markov Decision processes that can be continued (arm pull) or frozen (not an arm pull) by naming it a simple family of alternative bandit processes, where simple means that the bandit arms are available at all times. More, there is no cost for transitions from one arm to the other. In the simplest form, the Gittins proposes a function $G(a_i)$ that uses a discount factor β, a stopping time τ, and for Bernoulli bandits it depends on the beta distribution for the expected reward function. The function tries to find the optimal strategy for maximizing the total expected discounted reward. Formally, the index is calculated as:

$$G_i = \frac{E\big[\sum_{t=0}^{\tau-1} R_{i,t} * \beta^t\big]}{E\big[\sum_{t=0}^{\tau-1} \beta^t\big]}.$$

SoftMax strategy and probability matching variants is a popular strategy that uses Boltzmann (also called Gibbs) distribution. This uses a probability

$$p(a_i) = \frac{e^{Q(a_i)/\tau}}{\sum_{i=j}^{k} e^{Q(a_j)/\tau}}.$$

The choice of τ is left to the user. *SoftMax* can be adapted to a *decreasing SoftMax*, by using for example $\tau = \frac{\tau_0}{t}$, in the same manner as for ϵ-decreasing strategy (see subsection 3.3.2). Another adaptation if this strategy is *GaussMatch* specific for the case where the rewards follow a Gaussian distribution. More details about *SoftMax, and GaussMatch* can be found in [17], [23], [25], [26], and [27].

The paper [33] presents an extension of the bandit strategy for the cases when the arms depend on each other. The authors propose a model that clusters the arms and manage to reduce the problem to a traditional bandit problem.

4. Treatment design

4.1. Context assumptions

The context assumptions are the given facts about the environment in which the design will act upon. Thus, the routing algorithm for payments need to take into consideration and to integrate into the payment flow. To a certain extent, these assumptions define the scope of the to-be-designed solution.

The most important context assumptions are:

1. **Assumptions on payments**
    a) Infinite horizon. The total number of transactions is unknown beforehand and it is considered infinite;
    b) Batch and continuous transactions. Most of the merchants have the transactions that come continuously, as a stream, but there are a few merchants that have all their transactions in batches. A batch of payments means that all the payments from the previous week, month or quarter of year come all together;
    c) Binomial rewards. The reward function of one payment is not a continuous number, but rather binomial with two possible outcomes: 1 - Success or 0 - Failure;
    d) Delayed responses. There is a time delay between sending a payment and receiving a response, because network transfer time and external processing time by card schemes and issuers.
2. **Assumptions on acquirers and authorization rates**
    a) Unknown authorization rate distribution. This propriety is general across all bandit problems, because if the authorization rate was known, choosing the best action for each transaction would have been trivial, by always going to the acquirer with the higher authorization rate;
    b) Non-stationary authorization rate. The experience at Adyen shows that authorization rate changes in time for the same acquirer, because of unknown reasons to Adyen;
    c) Independent acquirers. The authorization rate and the processes of one acquirer does not influence the other arm, no matter what action is taken;
    d) Acquirers availability. It is assumed that the acquirers are available at all times;

e) No transitional costs. Going from one acquirer to the other does not involve any cost;

f) Small number of acquirers. The number of acquirers is usually two, although it can be three, four, or sometimes more;

g) Acquirers are available at all time. As long as an acquirer is on the list of possible acquirers, it is also assumed that the acquirer can be reached out and used for processing of payments;

h) Acquirers do not treat payments independently. For example, if there is a history of *n* failed transactions, the *n+1* transaction might be also refused, if there is any sign of doubt about it (for example, if the shopper address is missing). However, the same transaction could have been accepted in other circumstances, i.e. if the history of the shopper, merchant or issuing bank would have been cleaner (less failed transactions in the past).

4.2. Treatment requirements

Using the definitions of treatment requirements defined in [1], we consider them as desired proprieties of the solution that should contribute to the stakeholders goals.

The requirements have been defined by both experts at Adyen and by external stakeholders. The most important stakeholders at Adyen in defining the requirements are the data analysts and product owners (for dew the expected behavior, input, and output), software engineers (for making sure that the implementation is realizable), and account managers (the actual users of the new algorithm, responsible for setting it up for the merchants and monitor the results for them). On the other hand, in the case of external stakeholders, the merchants (the buyers of the payment services) are the most important ones, because they are the beneficiaries of the treatment.

The requirements can be functional (*f*) or non-functional (*nf*). A functional requirement is a desired function of the solution or algorithm, whereas a non-functional requirement is actually a desired propriety when the solution is introduced in its context.

To make everything more concrete, it is important to have ways of measuring a requirement fulfillment by using indicators or metrics. This is usually needed for non-functional requirements, but indicators can be defined for functional requirements as well. This procedure is called operationalization. The priority of the requirements are determined by the means of importance and likelihood. These treatments requirements are described in table 4.1.

Requirement	**Type**	**Description**	**Indicators**	**Priority**	**Main stakeholder**
Performance	f	The treatment should increase the number of authorized transactions	Auth. rate	High	Merchants
Low Risk	f	In worst case scenario, the eventual negative impact should be low and controllable	Auth. rate	High	Merchants
Automation	f	The treatment needs a minimum user interaction to configure and control the treatment	Avg. no. of user interactions	Normal	Adyen
Utility	nf	The treatment is meant for cases where there are two or more available acquirers	No. of merchants	Normal	Adyen
Usability	nf	The configuration and monitoring interfaces should be simple to use; the treatment mechanisms should be easy to explain to the beneficiaries (merchants)	Effort to use, effort to learn	Normal	Adyen
Flexibility	nf	The treatment should allow configurations per different variables (e.g.: merchants, BINs); it should be configured and come into effect in real-time;	No. of var.; effect delay	High	Adyen
Reliability	nf	There should be no system failures or deadlocks in the acquirer selection	System errors count	High	Adyen
Interoperability	nf	Compatible with the current implementation; ability to set proportion of traffic between treatment and current implementation;	Compatibility effort	Normal	Adyen
Time efficiency	nf	Payments should happen very fast; the treatment should add an insignificant time overhead	Time overhead due to the algorithm	Normal	Adyen
Space efficiency	nf	The main memory used by the treatment should be not be excessive	Main memory used by the algorithm	Normal	Adyen
Accuracy	nf	Payments are financial operations, so the treatment should handle them consistently and accurately, where all numbers match; e.g.: total no.payments in equals total no. payments out.	Accuracy checks	Normal	Adyen
Security	nf	The algorithm should, in no way, compromise or weaken the security of the transactions in the payment flow	Security checks	Normal	Adyen
Compliance	nf	It should not affect various compliance rules from the card schemes; e.g.: sending only failing payments to one acquirer might attract penalties for bad traffic from the card schemes	Compliance checks	Normal	Adyen

Table 4.1.: Treatment requirements

4.3. Proposed treatments

The current implementation is not fully adequate for the research problem, as explained in section 2.4. There are many alternatives to the current implementation, as shown in the literature review (section 3), but given the scope of this research, not all of them will be tested and validated.

The presented algorithms need to be tailored to and best fit the context assumptions (section 4.1) and the treatment requirements (section 4.2), the list of possibilities can be divided into two lists: high expectation designs, and low expectation designs.

4.3.1. High expectation designs

There are 5 high expectation designs: play the winner rule 3.2.2, a combination between the greedy bandit (subsection 3.3.2) and the pursuit bandit (subsection 3.3.3), the gradient bandit (subsection 3.3.5), Exp31 (subsection 3.3.6), and Thompson sampling (subsection 3.3.7).

These high expectation designs cover most of the context assumptions and treatment requirements. However, there are couple of assumptions that are not particularly managed by any of these algorithms and they might bring a performance shortfall. These are the assumptions on delayed responses and batch transactions. The algorithms are designed to work best when the decision on *t+1* transaction is taken by using the information up to the *t* transaction.

Most of high expectation algorithms are part of reinforcement learning - multi armed bandit, that is why, all of these algorithms have common structure. The algorithm can be divided into three phases:

1. *Initialization*: the variables used by the algorithm are assigned initial values. They can be either some default values or user specified. The initialization can be done all at once in the beginning, but it is more appropriate to have a lazy initialization. This means that for each new payment to be routed, we look whether the initialization has already been done for the relevant combination of payment characteristics (e.g.: merchant, BIN). If not, then we do the initialization.

2. *Send payments*: the payments are sent based on the routing rules set by the algorithm.

3. *Update*: the update step refreshes the algorithm variables based on the result of the payment (success or failure), and thus updating the routing rules used for the future payments.

First, update step can only happen once the response have been received, but in between payment request and the actual update of the variables based on the payment responses, other payments can come in. Inevitably, these later payments will be routed using outdated information.

Second, time efficiency means that the variables should be kept in the main memory. On the other hand, the payment flow is a stateless process, where the only allowed state are static caches of the data found in the database. These caches work as a snapshot of that data. The variables used by the algorithm can only be stored and used through the caches. Once a payment response is received, then the flow is more relaxed, and it is possible write to the database. However, for each new payment, it is impossible to access the database to read the algorithm variables, due to time efficiency. The middle ground is to use the cache as a (outdated) snapshot of the database variables. The caches are refreshed at a fixed time (e.g.: 15 minutes, 1 hour, 4 hours, 1 day). Having a smaller time frame means that the algorithm variables are more up to date; having a larger time frame means that the execution time overhead brought by the algorithm is smaller.

4.3.2. Low expectation designs

Comparing two proportions (subsection 3.1) is very similar to the current implementation. However, it offers an additional layer of statistical rigidity by clearly specifying the minimum sample sizes needed to make a fair comparison and it defines the minimum difference of authorization rate to be able to conclude that one acquirer consistently performs better than the other acquirer, not only due to chance. Nevertheless, there are two important drawbacks of this approach. The authorization rates need to be constant, which is not the case for payments (see context assumptions). This makes it impossible to make predictions. . Second, while data is gathered for the minimum sample size, this method cannot offer any information on the best acquirer, so the risk of losing transactions while learning is very large.

A/B Testing is a bit more advanced than comparing two proportions. It can cope better with setting the minimum sample size and it can assess quicker the best acquirer, especially if the difference in authorization rate is large. In any case, it is not very appropriate to make predictions, because the differences of performance is done with Z-test, and this has the same shortcomings as in the case of comparing two proportions.

Upper-Confidence-Bound bandit strategy leverages cases where there are more than two acquirers. For payments, in almost all cases, there are two acquirers. Thus, the extra complexity brought by this strategy does not pay-off.

Gittins index has a strong theoretical support, but in practice, it is very difficult to implement due to the resources needed to compute the index for every new payment. More, the infinite horizon context assumption (undefined total number of payments) it is not genuinely supported by the Gittins index. As for the *SoftMax strategy* and *GaussMatch*, most of their logic are incorporated by *Exp31*.

5. Treatment validation

5.1. Validation models

The main two objectives of validating the algorithms in the context are: selecting the best algorithm to be implemented on the platform and building a design theory around the algorithm. The design theory should assure that the assumptions and requirements are covered by the algorithm, so that the results of the simulation can be generalized to the real context. More, the design theory should commit to an improvement expectation of the algorithm. The validation can also bring to surface patterns and other information about the data.

As described in subsection 4.3, there are multiple possible solutions. Ideally, the best way to validate the performance is to implement all of the possible algorithms in the real world context (the payment platform). However, this is not feasible, and a more practical approach is to have models of the algorithms' implementation applied to a model of the context.

A great deal of complexity of the real payment platform is not included in these models:

- mechanisms used to send information between parts of the payment platform and outside (network, RPL protocol, etc.). The models use direct function calls;
- database and cache implementations. Everything is kept in memory;
- other irrelevant sub-systems (risk system, payment request parser, etc.). The models simulate only the strictly necessary sub-systems (payment generation and acquirers);
- fallback procedures. On the payment platform, everything needs to have a safe fallback in case of any kind of failure or inconsistent state. On the real payment platform, this might result in situation in which some payments will ignore the recommendation of the treatment, or the treatment fails before making any recommendation;
- control on the application of the treatment. This can be done by wrapping everything around a on/off feature, and controlling exactly when a payment uses the treatment by creating filters by variables (e.g.: only for some merchants, only for some BINs, only for 50% of the payments of these filters);

5.1.1. Model of the context

The model of the context needs simulate the following:

1. generation of payment requests;
2. acquirer selection out of the available acquirers;
3. payment responses for the generated payment requests.

Payment generation is very simplistic and it only contains the necessary variables (issuer country, merchant, BIN). The available acquirers are given, and the acquirer selection is the result of the algorithm.

The most difficult part is to simulate the payment response, as it is impossible to replicate a true acquirer that handles real payments. However, it is possible to simulate the authorization rate distributions of real acquirers by using historical data. To keep the non-stationary assumption of the distributions, the authorization rates will be calculated per day. These distributions will determine whether it is a success or a failure. More, the number of payments generated per day will be the same as the one on the real platform for the respective day for which the authorization rate distribution is simulated.

The payment responses are also simulating the delay to which they are received by the payment platform. The delay is not fixed a time, but probabilistic, similar to the real world. So, the *t+1* payment might or might not have an updated view of the algorithm variables up to the *t, t-1, t-2, ...* payment. In paper [15], two types of simulated delays are defined: deterministic (when a payment request comes in, we get the response of this payment after 2, 3, 4, ... other payment requests) or stochastic (when a payment request comes in, we get the response for this payment with a probability of 1/3, 1/4, ... before each new payment request). Thus, in the current simulations, we used the stochastic approach, fixing a probability of 60% that a payment response is received after each payment request. This means that there will be cases in which the number of payment responses will be behind compared to the number of payment requests that have been made.

Another important aspect is that all experiments are run for 1,000 times, so that the results are as accurate as possible and we limit the impact of the randomness on the algorithms' performance.

5.1.2. Model of the algorithms

The algorithms are build in Java and they use the pseudocodes described in appendix A. The algorithms are isolated from each other, but they use the same sample data for benchmarking.

To measure the performance of the algorithms we define the following:

- *Improvement*: the difference between the authorization rate of the simulated payments handled by the algorithms and the real authorization rate from the sample data. A negative improvement means that the algorithm does more harm than good;

- *Regret*: the difference between the authorization rate of the best acquirer and the authorization rate of the simulated payments handled by the algorithms; of course, the best acquirer is not known beforehand and it is determined by looking backwards at the real data. It is not realistic to consider an authorization rate of 100% as being the maximum, because that cannot be reached by only selecting the best acquirer. The gap between the authorization rate of the best acquirer per day and the real authorization rate per day is the actual improvement opportunity;
- *Weight*: some types of BINs appear more often in the data set than other BINs; this is why, it is important to add some weights to the results to make sure we make a fair comparison between the acquirers;
- *Weighted improvement*: importance * % improvement. This measurement is reliable and can be used to compare the algorithms' performance.

5.2. Sample data

The sample data is real payment data from one merchant for E-commerce and ContAuth (recursive E-commerce) shopper interactions. We use data from three countries, same merchant, to increase diversity of the validation by making sure that the results are consistent across countries. The sample has two sets, presented in table 5.1.

Country	**Period**	**No. payments**	**No. BINs**
US	April - June 2016	26.7 mil	15,919
FR	January - June 2016	1.8 mil	653
ES	January - June 2016	2.2 mil	702

Table 5.1.: Sample data

The algorithm is supposed to run per country, merchant, and BIN, as they include much of the information of a payment. Of course, the design should and does allow later variables addition, if needed.

5.2.1. Data description

Three of the most important variables that describe our data are: volume, variation and authorization rate difference between acquirers. The volume is the number of payments done by a BIN in a day. The variation is defined as the probability of an acquirer to have the highest authorization rates two days in a row. For example, if the variation of a BIN is 50%, it means that if at transaction t acquirer A is the best, then at $t+1$, there are 50% chances that acquirer A will be the best, whereas there are 50% chances that acquirer B will be the best. These characteristics were selected in consultation with data experts at Adyen. However, the daily number of transactions is the decisive factor, for two reasons: the impact on the business and the influence on the algorithms.

In figure 5.1, we can see that the cumulative density of BINs by the daily number of transactions. Most of the BINs (86.18%) have between 0-10 transactions per day. As the daily number of transactions increases, the number of BINs decreases, up to one BIN that has nearly 10,000 transactions per day.

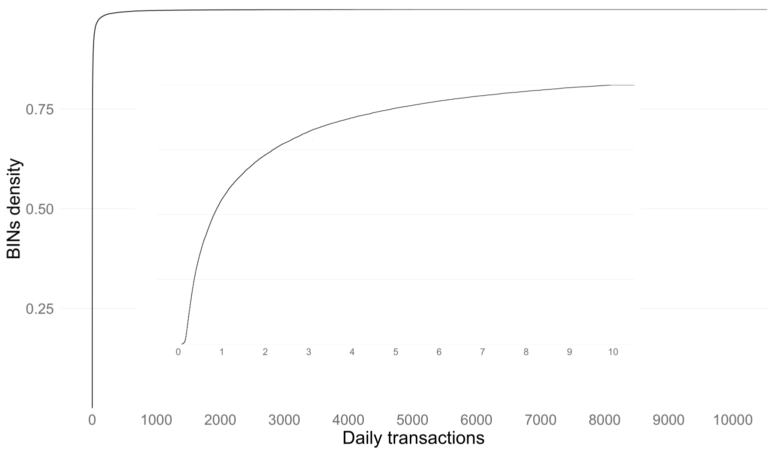


Figure 5.1.: Cumulative density of BINs by daily number of transactions

Figure 5.2 looks more in-depth in the small BINs. We can see here the same distribution of the small BINs in all the three sampled countries: France(FR), Spain(ES), and United States(US).

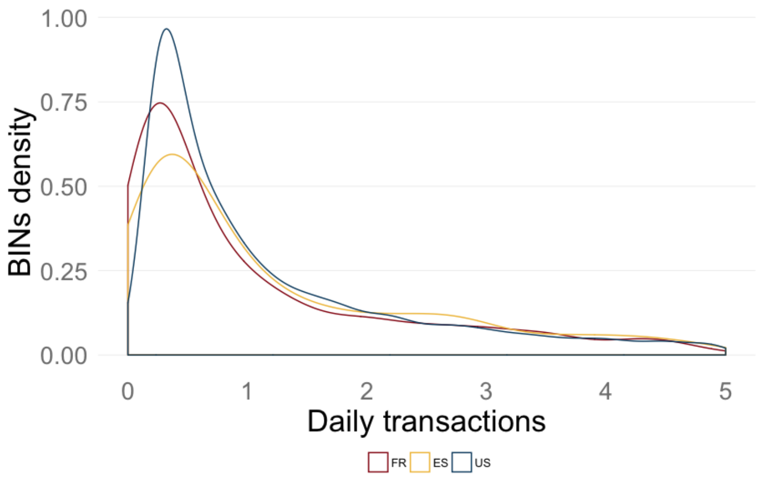


Figure 5.2.: Distribution of BINs by daily number of transactions - less than 5 tx/day

However, the majority of small BIN counts for less then 10% of the total number of transactions for the sampled data. In figure 5.3, we see that the most influential BINs in terms of total number of transactions are 48 BINs (only 0.28% of all BINs) that count for more 38% of all transactions. These can be considered as "outliers", but in our case, they are very important, and we might want to cluster them differently. Some of these BINs have around 6,000 thousands - 10,000 transactions per day. More than 50% of transactions are done via BINs that have between 11 - 1,000 transactions per day.

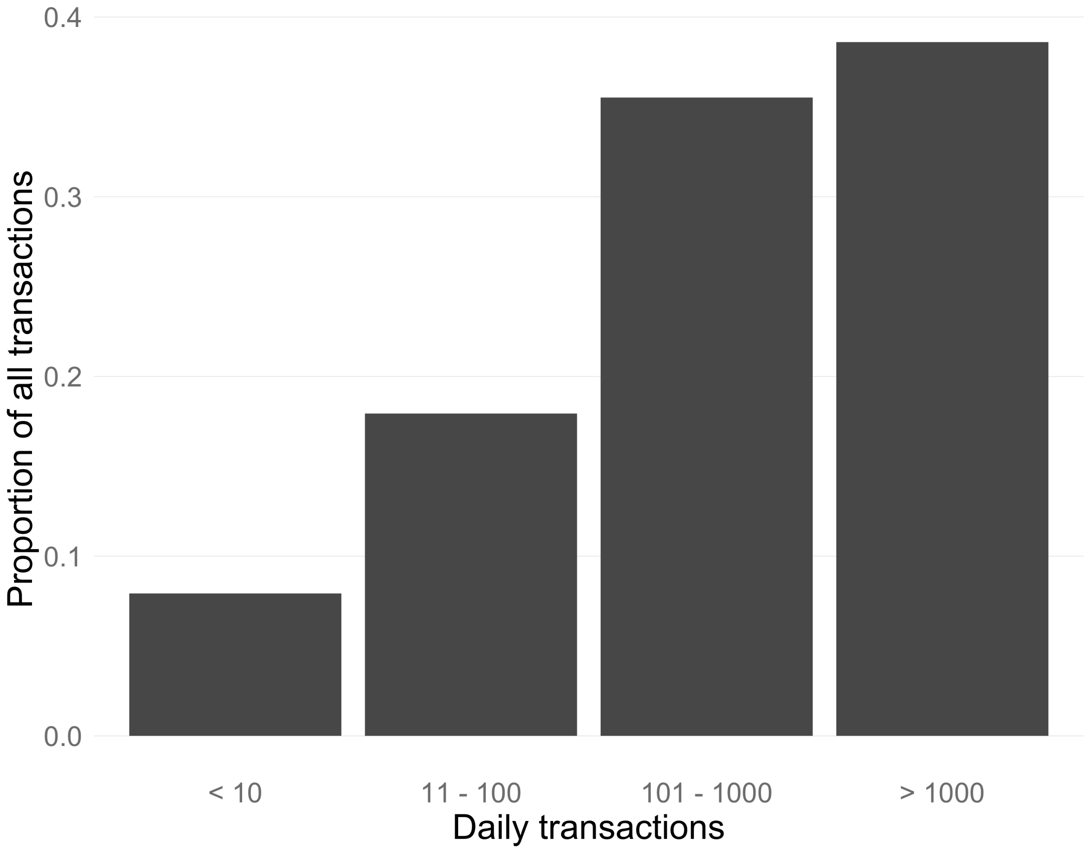


Figure 5.3.: BINs impact on total number of transactions based on the daily number of transactions

Figure 5.4 shows a more general view on the BINs by variation and authorization rate difference between acquirers. We can observe that most of the BINs cluster around variation of 50%. At the same time, small BINs tend to have a larger difference in authorization rate than larger BINs. This behavior is normal, because if one BIN has a few transactions, from which one of them fails, then one acquirer will present a much higher authorization rate than the other. As already shown, the number of BINs between 0-5 tx/day is much higher than the others. The figures are dominated by the blue color, because US BINs are much more frequent in the sampled date, but the identified trends are similar for all three countries.

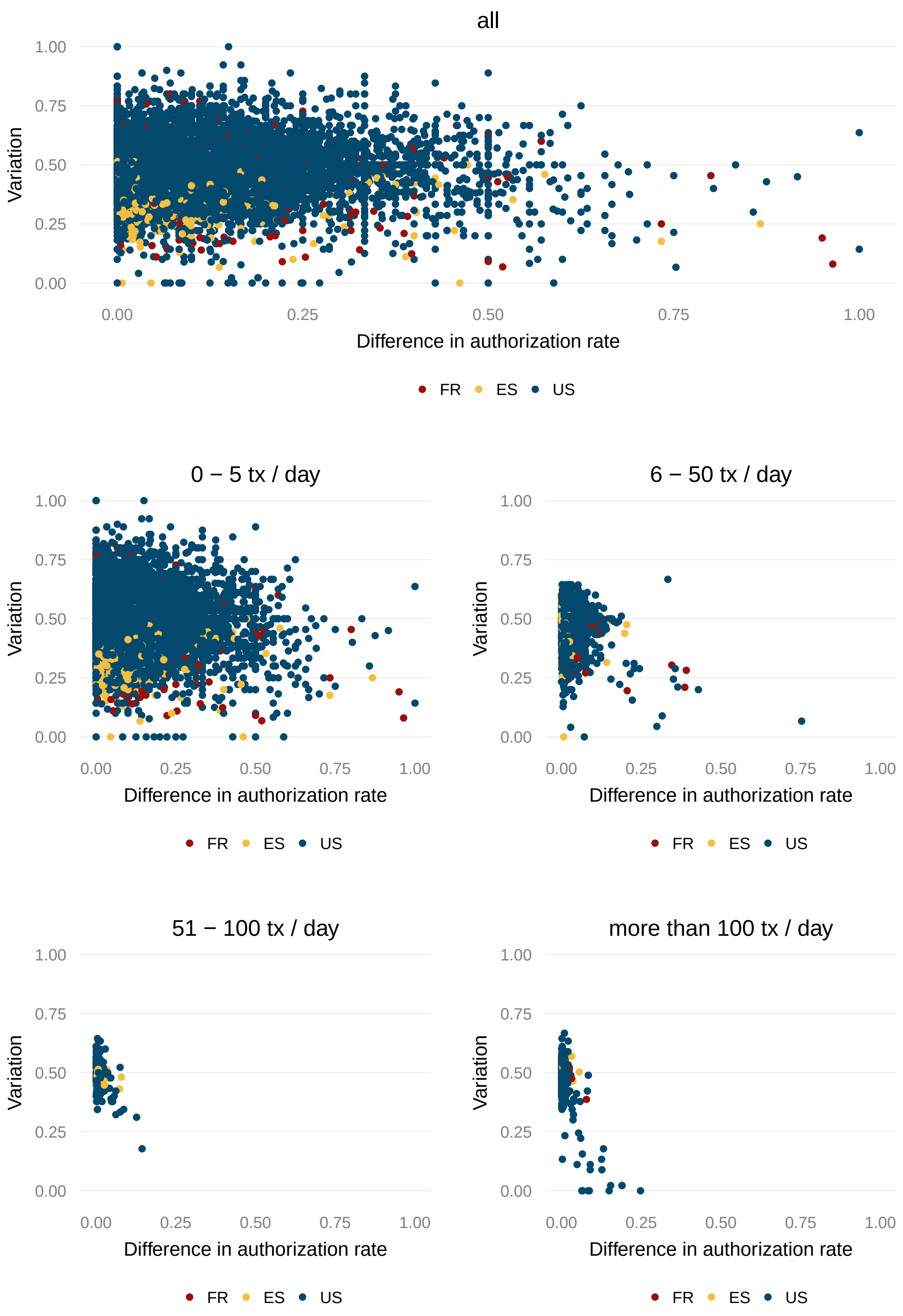


Figure 5.4.: Distribution of BINs by daily number of transactions, variation, and difference in authorization rate

5.2.2. Clustering

The generation of millions of payments is very demanding. An alternative is to extract several BINs and infer the results over all the data. However, the BINs are not homogeneous, and it is important to assure representativeness of the extracted BINs. To do this, we first cluster the data based on the three characteristics: volume, variation and authorization rate difference between acquirers.

There is a second use of the layers: to find and validate patterns of the algorithms behavior. On one hand, the patterns should help in comparing the algorithms' performance on different types of BINs. For example, *Play the Winner Rule* algorithm is expected to perform better for BINs with that have small variation, because once the better acquirer is identified, exploration is non-existing, as long as the selected acquirer is not failing too much. One the other hand, if the variation is high, we expect that the other algorithms perform better because they have more complex mechanisms to explore change and to exploit the rightful acquirer. On the other hand, the identified patterns should help in deciding the right parameters for each of the algorithms. For example, the threshold for the *Play the Winner Rule* algorithm, for which the algorithm performs best, is expected to be high for large BINs, but for small BINs, it is expected to be very low.

One of the most used multivariate clustering analysis in data mining [34] is *k-means*. We have used k-means to reduce the number of selected BINs to run simulations, but in the same time, to make sure that we have variety of BINs in our selection. The clustering has been done on normalized/scaled variables (mean is subtracted and divided by the standard deviation).

In figure 5.5 we plot the within groups sum of squares by the number of clusters.

There is no "good" way to choose the right number of clusters. However, there is a popular heuristics, called the Elbow method, that looks at the variance explained by various number of clusters. When there is little variance explained by adding a new cluster, then the Elbow method suggests to stop.

After running k-means analysis on all BINs, we have the number of clusters to be 10, because here we see the error (within groups sum of squares between points and the center of the cluster) to stop from the fast decrease. This method is based on the one presented in [35], which is actually a slightly modified version of the Elbow method, by using within cluster variance.

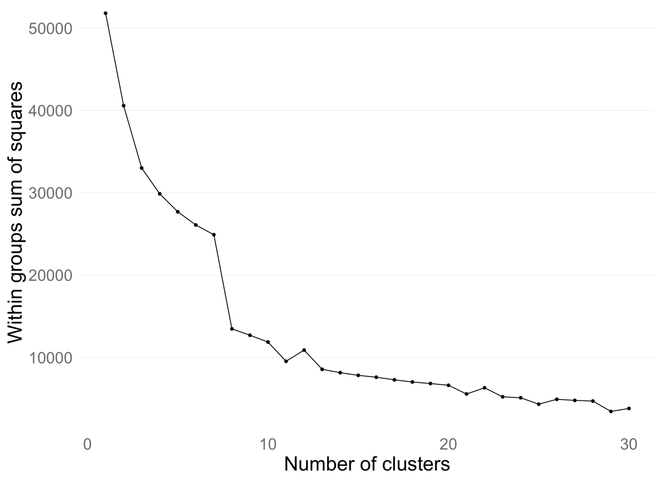


Figure 5.5.: K-means analysis of clusters

Figure 5.6 presents the BINs divided into 10 clusters in a 3D plot for the three (normalized) variables: variation, authorization rate difference, and daily number of transactions.

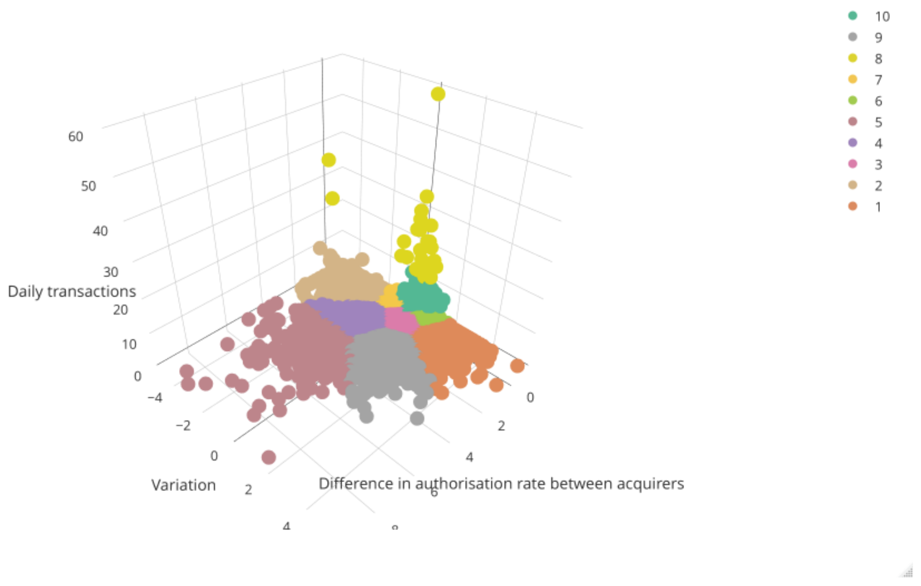
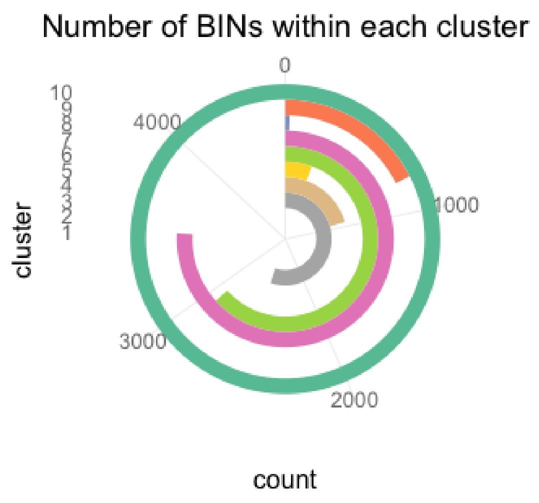


Figure 5.6.: All BINs divided into 10 clusters with kmeans

In figure 5.7, we can see the number of BINs per cluster. Cluster 10 contains the most BINs and more than 50% of the total transactions. Cluster 8 has the least BINs, but the second most transactions, because it has the biggest BINs in terms of number of transactions. Clusters 1-5 and 9 have many BINs, but not so many transactions, because they mostly contain small BINs.



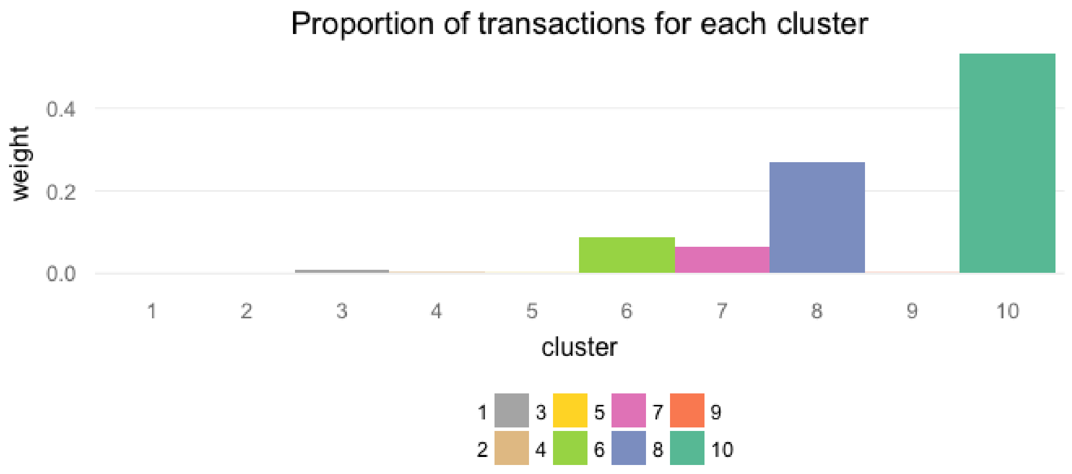


Figure 5.7.: Number of BINs and proportion of transactions per cluster

Out of each cluster, we have selected 2 BINs to simulate and apply the proposed treatments. More than half of the clusters have very small BINs (around 1 transaction per day), and there is little sense to apply any of the proposed algorithms and expect improvements. Thus, we have selected the BINs with the most number of transactions within each cluster. More, larger BINs are more important, as they count for more transactions. Table 5.2 lists the selected two BINs within each cluster.

Cluster	BIN	Daily tx	Variation (%)	AuthRate Difference (%)	Country
1	468xxx	192.97	66.67	0.91	US
1	457xxx	17.48	64.29	5.21	US
2	438xxx	1711.03	0	14.93	US
2	484xxx	999.19	13.33	0.27	US
3	435xxx	20.62	44.44	12.91	US
3	529xxx	14.76	46.41	9.36	FR
4	449xxx	29.4	28.89	24.45	US
4	540xxx	23.9	38.89	15.71	US
5	513xxx	29.08	20.99	38.66	FR
5	513xxy	13.84	28.18	39.12	FR
6	474xxx	606.76	58.89	0.38	US
6	470xxx	550.22	58.89	2.05	US
7	435xxy	405.62	34.44	3.33	US
7	454xxx	391.08	35.56	0.30	US
8	434xxx	10034.18	48.89	0.27	US
8	414xxx	6016.99	0	8.59	US
9	448xxx	5.1	66.67	33.33	US
9	478xxx	3.11	50	33.33	US
10	542xxx	1798.2	43.33	0.16	US
10	542xxy	1784.35	46.67	0.77	US

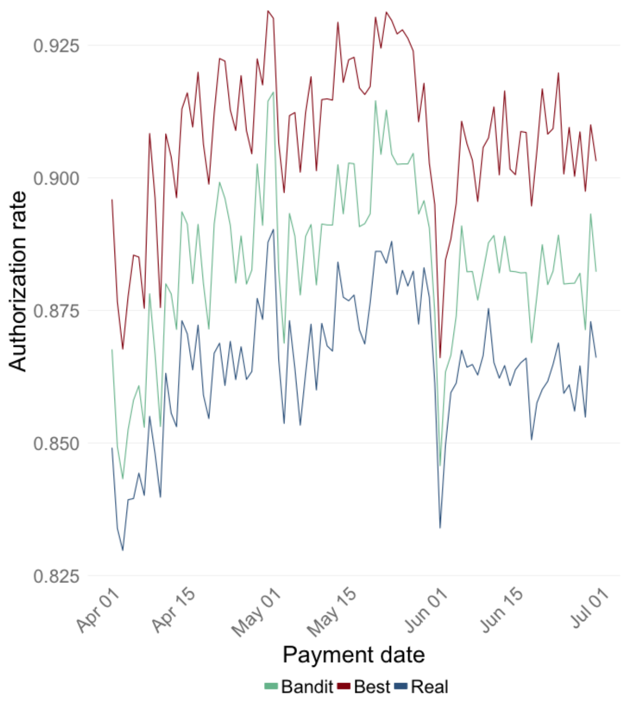
Table 5.2.: Sampled BINs for simulations

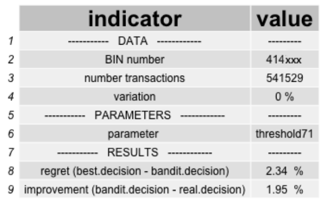
5.3. Results

In this section we present the results of the five proposed algorithms (section 4.3) on the sample data (section 5.2). Based on these results, we measure the performance of the treatments. These statistics are used to compare the proposed algorithms and conclude on the most appropriate choice for the identified problem. More, we learn more about the treatments themselves, especially about the right parameters that can be used. These results bring us closer to fulfilling the three functional requirements (section 4.1).

5.3.1. Performance

We have applied the proposed algorithms for each of the 20 selected BINs. In figure 5.8, we can see the results of Play the Winner Rule applied to BIN 414xxx (part of cluster 8), which is an US BIN with large volume, no variation, and small authorization rate difference. This result uses the optimal parameter (failure threshold), or the parameter for which the best results were obtained. The impact of different parameters is analyzed in the next subsection.





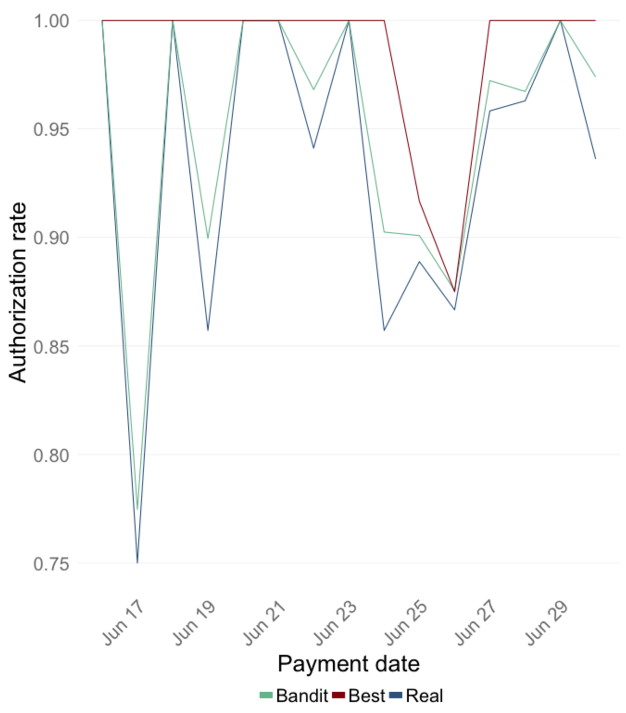
	indicator	value
1	---------- DATA -----------	--------
2	BIN number	414xxx
3	number transactions	541529
4	variation	0 %
5	---------- PARAMETERS -----------	--------
6	parameter	threshold71
7	---------- RESULTS -----------	--------
8	regret (best.decision - bandit.decision)	2.34 %
9	improvement (bandit.decision - real.decision)	1.95 %

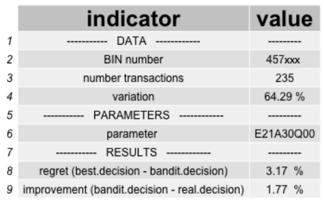
Figure 5.8.: Play the winner on large volume, no variation, small authorization rate difference BIN

The real line represents the authorization rate per day from real payments. The best line represents the authorization rate per day, if we were to choose the acquirer with the highest authorization rate for the respective day. The bandit line represents the authorization rate of the algorithm. The difference between the bandit line and the real line is the improvement. If the former is below the latter, then the bandit had a negative impact. The difference between the best line and the bandit line is the regret. If the bandit line is close to the best line, then we have fulfilled most of the improvement goal.

In the above case, we have an improvement throughout the period. On average, the improvement is 2.34% authorization rate. There is still 1.95% improvement possibility left (regret). The threshold used for this experiment is 71 failures.

In figure 5.9, we applied *Greedy and Pursuit bandit* on the BIN 457xxx (part of cluster 1), which has fewer transactions, higher variation and smaller authorization rate difference.





	indicator	value
1	---------- DATA -----------	---------
2	BIN number	457xxx
3	number transactions	235
4	variation	64.29 %
5	---------- PARAMETERS -----------	---------
6	parameter	E21A30Q00
7	---------- RESULTS -----------	---------
8	regret (best.decision - bandit.decision)	3.17 %
9	improvement (bandit.decision - real.decision)	1.77 %

Figure 5.9.: Greedy and Pursuit bandit small volume, high variation, small authorization rate difference BIN

Here, we have an improvement of 3.17% and a regret of 1.77%. The bandit line is closer to the real line, though, it never goes under it, which means that we do not expect it to have a negative impact. This is important to know when assessing the low risk requirement.

This process has been repeated for all sampled BINs and for all proposed algorithms.

In figure 5.10 we can see the performance improvement brought by the five proposed algorithms for each cluster. For cluster 4, the majority of the algorithms managed to bring an important improvement. Clusters 1, 6, and 10 have a small authorization improvements. As for cluster 9, the improvements are negative for some algorithms, which means that they actually have a negative impact on the performance of those BINs, if the algorithms were to be applied.

From these results, a few patterns can be identified. First, Play the Winner Rule shows better improvements for clusters that have a larger variation (clusters 1 and 9). This can be explained by the fact that it can change to the other acquirer quicker, because it just counts the number of failures without keeping track of any other information. In the case of high variation, the algorithms retain more information from the past transactions, thus they take some more time and transactions until they propose the other acquirer to be explored.

Second, the algorithms Exp31, Gradient bandit and Greedy, and Pursuit bandit have, in most cases, improvements that go in the same direction. However, Exp31 consistently presents lower authorization rate improvements.

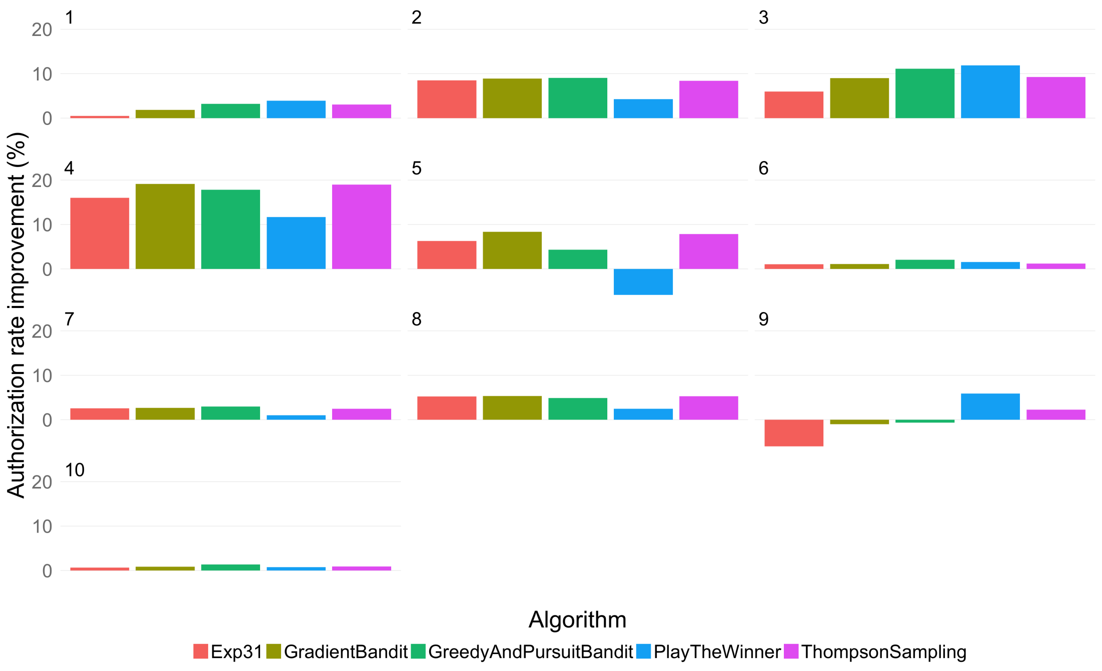


Figure 5.10.: Authorization rate improvement per cluster

An important question that the stakeholders ask is ”What authorization rate improvement are we expecting on average?”. This question can be answered by weighting the authorization rate improvement by the total number of transactions. Nevertheless, not all clusters are equally important. Results in figure 5.11 Try to answer stakeholders‘ question. Expected improvements are 2.38% for Exp31, 2.57% for Gradient bandit, 2.84% for Greedy and Pursuit bandit, 1.54% for Play the Winner Rule, and 2.58% for Thompson Sampling. Greedy and Pursuit bandit shows the best performance; followed by Gradient bandit and Thompson sampling.

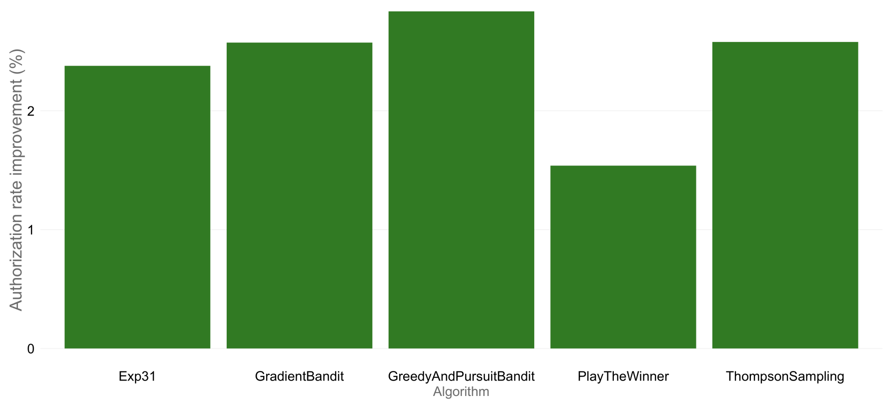


Figure 5.11.: Authorization rate improvement weighted by the total number of transactions

5.3.2. Automation

One other important functional requirement (see table 4.1) is to minimize the user interactions when trying to configure a route. The proposed algorithms manage to do this to a certain extent, but some of them still have parameters that need to be set. In the literature, setting the parameters is usually left to the user, depending on the context of the application. In our context, it is important to offer default values of the parameters, and if needed, to allow the user to manually change them.

Thompson sampling and Exp31 algorithms do not have any parameters to be set. Thus, the analysis is only applicable for Play the Winner Rule (failure threshold), Greedy and Pursuit bandit (exploration rate ϵ, recency rate α and initial preference indicators Q) and Gradient bandit (recency rate α and initial preference indicators Q). The initial preference indicators (where applicable) are not analyzed, because their use is quite clear and they should be used only when there is already good information about the performance of the available acquirers. Otherwise, their default values are 0.

We analyzed the effects that different parameters have on the algorithms to see whether a pattern can be observed based on which we could define a heuristics. More, it is important to see what are the expected costs in terms of performance.

An overview of the impact of different parameters on the authorization rate can be seen in table 5.3. This table tells us that, in the case of Play the Winner Rule, we expect, on average, to have a difference of 0.20% authorization rate between the best performing failure threshold compared to the worst failure threshold. This amplitude gives us insight on how important it is to focus on getting the best performing parameters. For example, looking at figure 5.11, for Play the Winner Rule, we expect an improvement of 1.54% in authorization rate, for the case where we have selected the best performing parameters. On the other hand, the amplitude of the failure threshold is 0.20%. So, if we were to select the worst failure threshold instead of the best one, we expect the lowest authorization rate to be 1.34%.

Algorithm	**Parameter**	**Average amplitude of the impact on the authorization rate**
Play the Winner rule	Failure threshold	0.20%
Greedy + Pursuit bandit	Exploration rate	1.33%
Greedy + Pursuit bandit	Recency rate	1.20%
Gradient bandit	Recency rate	0.39%
Thompson sampling	-	-
Exp31	-	-

Table 5.3.: Impact on authorization rate for different parameters

As one can observe, Greedy and Pursuit bandit is the most sensible algorithm to different values of the parameters. More, it has two parameters that might be set to the wrong values. In simplest terms, one can expect a lowest authorization rate average of 0.31% (2.84% - 1.33% - 1.20%). Despite its highest authorization rate improvement, Greedy and Pursuit bandit can easily turn into the worst performing algorithm in case the exploration and recency rates are not carefully set for each BIN.

Thus, we will have a closer look on the behavior of the parameters for different cluster of BINs.

Play the winner rule. Failure Threshold. In figure 5.12 we observe the impact of different values for the failure threshold parameter on the authorization rates of different types of BINs.

This figure confirms that setting one threshold or another has little effect on the overall authorization rate. The main reason is that there is usually little difference in authorization rates between the two acquirers and the algorithm is too simplistic to exploit the acquirer that is slightly better, no matter what threshold we set. There is one exception: BIN 448xxx (cluster 9) which is a small BIN with a high variation. For this BIN, a higher threshold has better results.

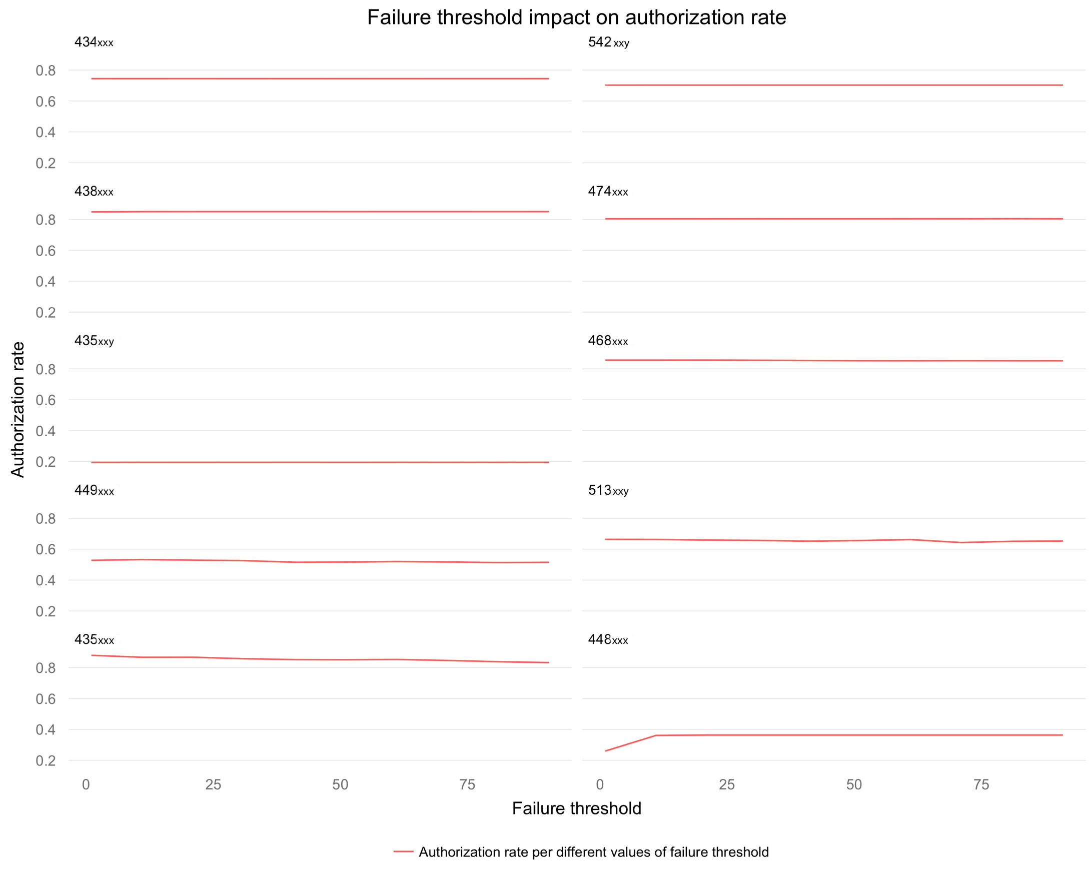


Figure 5.12.: Play the winner rule - failure threshold impact on the authorization rate

Greedy and pursuit bandit. Exploration rate and recency rate. In figure 5.13 we can see the impact of exploration and recency rates on the authorization rate. In the case of exploration rate lines, we keep fixed recency rates; in the case of recency rate lines, we keep fixed exploration rates. These fixed values are set differently for each BIN. We are interested more in the dynamics of the parameters lines, than on the interaction between the two for different values.

The impact of the exploration rate is stable for BINs 468xxx (cluster 1), 474xxx (cluster 6), 435xxy (cluster 7), 434xxx (cluster 8), and 542xxy (cluster 10). All of these BINs are medium to large BINs. Usually, for larger BINs, the authorization rate differences are smaller, thus there is little effect on always exploring or actually exploiting the slightly better acquirer. On the other hand, for the rest of the BINs, we see a decreasing trend in authorization rate as the exploration rate increases. Most of these BINs are small BINs (with the exception of BIN 438xxx). For these BINs, the differences in authorization rates are higher, and too much exploration causes to go more often with the worst acquirer. BIN 438xxx is the exception because it has 0% variation,

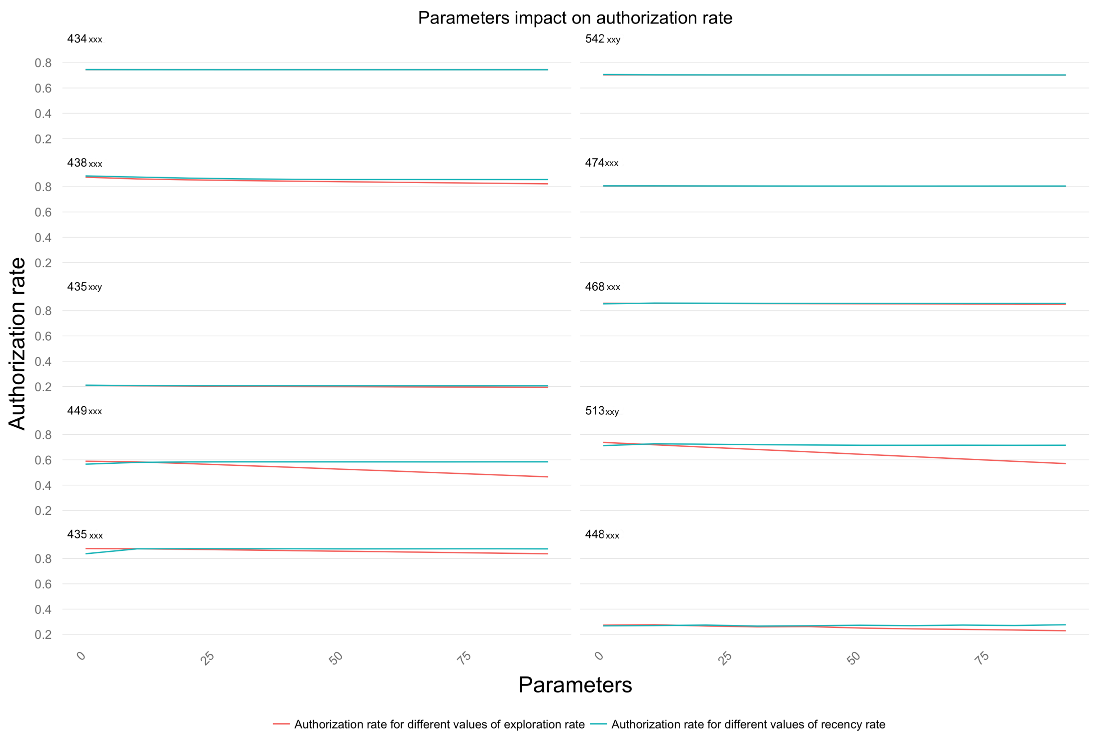


Figure 5.13.: Greedy and pursuit bandit - exploration rate and recency rate impact on the authorization rate

which means that an acquirer is always better than the other, thus excessive exploration has a negative effect.

Compared to the exploration rate, recency rate is rather stable. For some BINs (e.g. BIN 449xxx, BIN 513xxy, BIN 435xxx), wee see that a higher recency rate is better for the authorization rate. The same behavior can be seen at all BINs, but at a smaller impact scale. This makes sense, because having a very small recency rate means that the latest transactions have a very small weight when calculating the score. A small recency rate is fine for very low variation, but in our case, most BINs have a variation between 40% - 60%, thus a higher recency rate is beneficial.

Gradient bandit. Recency rate. In figure 5.14 we can see the impact of the recency rate α on the authorization rate.

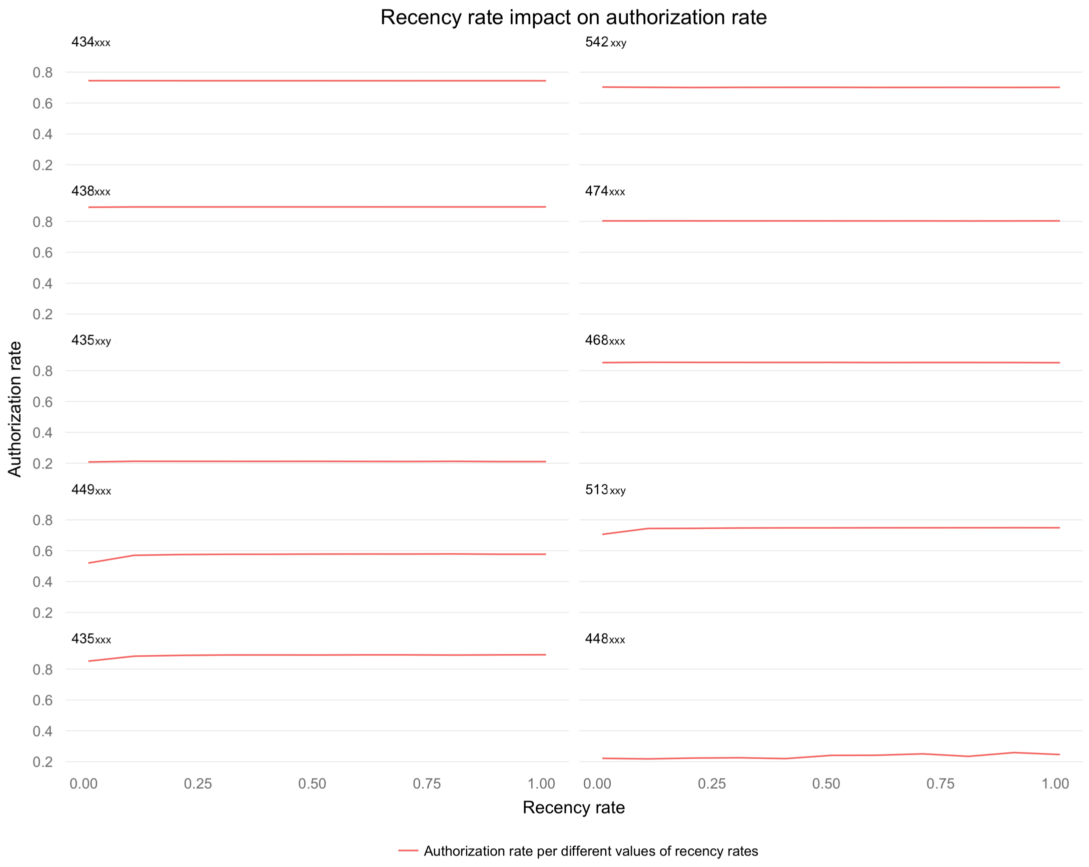


Figure 5.14.: Gradient bandit - recency rate impact on the authorization rate

The main observation is that the impact of different values is not big. Still, same small impact trends as in Greedy and Pursuit bandit can be observed for BIN 449xxx, BIN 513xxy, and BIN 448xxx. The small effect is due to the fact that the Gradient bandit has other regulating mechanisms to keep the preference indicator in check, i.e.: the average reward $\bar{R}$ and the probabilities $p(a_i)$. More, it is the only algorithm that updates preferences of both acquirers Q for every transaction.

6. Discussion and conclusions

6.1. Discussion

6.1.1. Remarks on the research goal and research questions

Throughout the paper, we analyzed ways to reach our research goal and to answer all of the question that derive from it. In this section, we come back and discuss these aspects.

1. **Descriptive**
   a) What is the scope of the routing of payments?
   The scope of the routing of payments is wherever there are multiple available acquirers to connect to, which is country specific. It is only applicable to credit cards networks Visa and MasterCard (not for other payment methods, such as iDeal). For Adyen, this is the case the following countries: Spain, France, Brazil, and US. However, not all merchants want to use all of the available acquirers due to different costs. At the moment, there are 21 merchants that allowed routing of payments between multiple acquirers. It is not excluded for more merchants to sign-in for routing, or for the number of countries that Adyen has multiple acquirer connections to increase.

   b) How many payments are affected?
   For the countries mentioned above, only card payments, there are approximately 2.35 mil payments per day that could eventually have the intelligent payment routing applied to.

2. **Explanatory**
   a) What are the reasons that cause a payment to fail?
   Payments fail due to discretionary, technical and fraud reasons. The technical failures are the ones that need to be fixed, and they count, on average, for 5-10% of all transactions.

   b) What are the influences of different variables on the payments?
   For card payments, the card networks decide the format of the messages. There are approximately 150 data fields (no sub-fields included) that eventually influence the decision of the issuers. However, for routing purposes, we only consider a handful of them: issuer country, merchant, and BIN.

   c) How much does the performance of the acquirers change in time?
   The variation is defined as the chances to have the same best acquirer (in

terms of auth rate) for two days in a row. In France, the average variation is 45.71%, in Spain 38.33%, and in US, it is 48.05%.

   d) What is the expected improvement that a new algorithm will bring?
   If we apply *Greedy and Pursuit bandit* we expect an average 2.84% authorization rate increase. *Play the Winner Rule* has the lowest expected authorization improvement of 1.54%, but in the same time, it is the simplest algorithm (to both understand and implement).

1. **Effect and requirement satisfaction**

   a) What effects does the algorithm have on payments?
   Any of the algorithms make recommendation for the best acquirer to go for each transaction. It will modify the payment request, acquirer field, to the one selected.

   b) What is the execution time when using the new algorithm?
   To make the it as fast as possible, the parameters needed for the algorithm to decide are kept in cache. It is not expected to bring any significant delay during a payment.

   c) What is the usability of the new algorithm for the payment platform?
   The algorithm is easily implementable, based on the given pseudocodes. User interaction is minimal. Users need only to enable/disable the algorithm. They can set the parameters manually or use the default ones. Payments that use this algorithm can be flagged, thus it enables real time performance monitoring. More, the traffic can be A/B tested with the current implementation.

   d) Does the algorithm satisfy the requirements of the stakeholders?
   The requirements are listed in 4.1. The most important requirement is the performance and low risk. As it results from the results, the expected improvement is positive, which means that the algorithms can safely be implemented and tested. The other requirements become more important in the implementation phase, which will eventually be covered by Adyen.

2. **Trade-off and sensitivity**

   a) How do different algorithms perform?
   There are differences between algorithms, as *Play the winner rule* and *greedy and pursuit bandit* perform best, especially because data shows a high variation. *Gradient bandit*, *Exp31*, and*Thompson sampling* perform similarly.

   b) How do different parameters of the same algorithm perform?
   Parameters affect the performance of the algorithms. In table 5.3, we see that the impact is around 2.5% authorization rate, which means that, in some cases, choosing the unfitting parameters might make the improvement of the algorithm negative.

   c) Can the algorithm be applied to other contexts?
   The algorithms are highly adaptable to other contexts. In payments, we can

apply them for payment flagging (e.g.: include or not include shopper address) and retrying of transactions (e.g.: retry transactions if the response code is "Unknown"). But, one can make use of the algorithms and these results for other domains where there is unknown and unfixed probability distribution.

6.1.2. Research contribution

The current research gives the data used in the payment industry a whole new look and functionality. Payment data is commonly used to either exchange data between financial institutions, shoppers, and merchants, or to make reports about amounts, number of transactions, authorization rates, etc. However, we have showed that there is much more that can be done with the data, than just looking at aggregates or case by case payments. The complexity of the payment network brings to the surface the need of a data-driven, dynamic approach in taking various decisions. This paper tries to frame routing as a reinforcement learning application, especially as a case of the multi-armed bandit problem. As far as this literature review has gone, this has not been tried in the payment industry before.

The most interesting characteristic of payments is the unknown and unfixed nature of the authorization rates. This enabled a comparison between five different bandit strategies on real-data that lead to better understanding the performance expectations of the algorithms under these particular circumstances.

More, the application quantified the impact of various parameter values of the bandit strategies on the results and raised the problem of automatically decide the parameters, which has partially solved by using heuristics and the expert opinions based on the logic of payments.

6.1.3. Limitations and recommendations

There are some limitations to the treatment's approach:

1. no information about why an acquirer works better than the other, we just see the preferred one; to understand better the variables that lead to a failure or a success, we need a different approach, such as logistic regressions or decision trees;

2. little control on the routed payments; in Brazil especially, merchants have strict contracts with the acquirers about the volume and costs; or, Adyen might have strategic needs to route more payments to particular acquirer; the proposed treatment tries to find the best acquirer in terms of performance;

3. the performance expectations do not cover all the improvement opportunity; as mentioned, 5-10% of payments technically fail, but routing of payments with the proposed treatments expect an increase in authorization rate of 2.84%; however, by applying the same algorithm for flagging, retrying of payments, or other parts of the payment flow, we might be able to reach a higher improvement;

4. there is no available theory for setting the parameters (e.g.: exploration rate and recency rate), but merely heuristics based on observations on which parameters work better for which test cases.

The main recommendation for the treatment implementation on the real platform is to apply and compare Play the Winner Rule and Greedy and Pursuit bandit, because the first one is very simple to implement, and the second one has the highest expected results. The second recommendation is to implement one of the other three proposed algorithms for the cases in which one acquirer is always better than the other, because they have more stable results and one or no parameters to be set.

6.1.4. Future work

The future work should concentrate in covering the limitations of the treatments, especially shortcomings of the multi-armed bandit strategies. These are:

1. connect multi-armed bandit algorithms with other types of learning based on the input variables to explain the contribution of the features; or, another way of understanding why is to integrate the response codes of a payment, so that it does not use just a binomial model (1-success, 0 - failure)

2. incorporate more complex rules in the decision of the best option, for example, minimum or maximum number of transactions through one acquirer;

3. a method to compute and use the optimal parameters (highest performance).

Adyen can also modify the payment request itself (by tweaking various fields), procedure called payment flagging. There are many similarities between payment flagging and intelligent payment routing. Based on analogy, the same research could be used in payment flagging applications.

6.2. Conclusions

Multi-armed bandit is a neat solution to optimize unknown and unfixed probability distributions. The main research goal can be achieved by routing payments to different acquirers. It is expected to bring a 2.84% authorization rate improvement and it definitely reduces the effort of setting routing rules, compared to the current implementation.

Intelligent payment routing can be easily implemented and tested on the payment platform at Adyen and leverages the connections to multiple acquirers. Even more, the proposed treatments can be used in many other applications, such as payment flagging or retry of payments.

Bibliography

[1] Roel J Wieringa. *Design science methodology for information systems and software engineering*. Springer, 2014.

[2] Ian Hacking. Telepathy: Origins of randomization in experimental design. *Isis*, pages 427–451, 1988.

[3] Thomas J Meyer and Melvin M Mark. Effects of psychosocial interventions with adult cancer patients: a meta-analysis of randomized experiments. *Health psychology*, 14(2):101, 1995.

[4] Marvin Zelen. A new design for randomized clinical trials. *New England Journal of Medicine*, 300(22):1242–1245, 1979.

[5] Joseph L Fleiss. *Design and analysis of clinical experiments*, volume 73. John Wiley & Sons, 2011.

[6] Ron Kohavi and Roger Longbotham. Online controlled experiments and a/b tests. *Encyclopedia of Machine Learning and Data Mining, edited by Claude Sammut and Geoff Webb*, 2015.

[7] Brian Christian. The a/b test: inside the technology that's changing the rules of business. `http://www.wired.com/2012/04/ff_abtesting/`, 04 2012. Accessed: 03-11-2015.

[8] Evan Miller. Simple sequencial a/b testing. `http://www.evanmiller.org/sequential-ab-testing.html`, 11 2015. Accessed: 02-11-2015.

[9] Audun M. Øygard. Rapid a/b-testing with sequential analysis. `http://auduno.com/post/106141177173/rapid-ab-testing-with-sequential-analysis`, 12 2014. Accessed: 05-11-2015.

[10] Leo Pekelis, David Walsh, and Ramesh Johari. The new stats engine. `http://pages.optimizely.com/rs/optimizely/images/stats_engine_technical_paper.pdf`. Accessed: 18-12-2015.

[11] Alexander Tartakovsky, Igor Nikiforov, and Michèle Basseville. *Sequential analysis: Hypothesis testing and changepoint detection*. CRC Press, 2014.

[12] Marvin Zelen. Play the winner rule and the controlled clinical trial. *Journal of the American Statistical Association*, 64(325):131–146, 1969.

[13] David Tolusso and Xikui Wang. Some properties of the randomized play the winner rule. *Journal of Statistical Theory and Applications*, 11(1):1–8, 2012.

[14] LJ Wei and S Durham. The randomized play-the-winner rule in medical trials. *Journal of the American Statistical Association*, 73(364):840–843, 1978.

[15] William F Rosenberger. Randomized play-the-winner clinical trials: review and recommendations. *Controlled Clinical Trials*, 20(4):328–342, 1999.

[16] Q Yao and LJ Wei. Play the winner for phase ii/iii clinical trials. *Statistics in medicine*, 15(22):2413–2423, 1996.

[17] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction.* MIT press, 1998.

[18] Jeremy Wyatt. Exploration and inference in learning from reinforcement. 1998.

[19] Dirk Bergemann and Juuso Valimaki. Bandit problems. 2006.

[20] Omar Besbes, Yonatan Gur, and Assaf Zeevi. Stochastic multi-armed-bandit problem with non-stationary rewards. In *Advances in Neural Information Processing Systems*, pages 199–207, 2014.

[21] William H Press. Bandit solutions provide unified ethical models for randomized clinical trials and comparative effectiveness research. *Proceedings of the National Academy of Sciences*, 106(52):22387–22392, 2009.

[22] Michael N Katehakis and Cyrus Derman. Computing optimal sequential allocation rules in clinical trials. *Lecture notes-monograph series*, pages 29–39, 1986.

[23] Volodymyr Kuleshov and Doina Precup. Algorithms for multi-armed bandit problems. *arXiv preprint arXiv:1402.6028*, 2014.

[24] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *Machine Learning: ECML 2006*, pages 282–293. Springer, 2006.

[25] Jean-Yves Audibert, Rémi Munos, and Csaba Szepesvári. Exploration–exploitation tradeoff using variance estimates in multi-armed bandits. *Theoretical Computer Science*, 410(19):1876–1902, 2009.

[26] Steven L Scott. A modern bayesian look at the multi-armed bandit. *Applied Stochastic Models in Business and Industry*, 26(6):639–658, 2010.

[27] Joannes Vermorel and Mehryar Mohri. Multi-armed bandit algorithms and empirical evaluation. In *Machine learning: ECML 2005*, pages 437–448. Springer, 2005.

[28] Malcolm John Alexander Strens. *Learning, cooperation and feedback in pattern recognition.* PhD thesis, Citeseer, 1999.

[29] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

[30] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. The non-stochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002.

[31] Olivier Chapelle and Lihong Li. An empirical evaluation of thompson sampling. In *Advances in neural information processing systems*, pages 2249–2257, 2011.

[32] John Gittins, Kevin Glazebrook, and Richard Weber. *Multi-armed bandit allocation indices*. John Wiley & Sons, 2011.

[33] Sandeep Pandey, Deepayan Chakrabarti, and Deepak Agarwal. Multi-armed bandit problems with dependent arms. In *Proceedings of the 24th international conference on Machine learning*, pages 721–728. ACM, 2007.

[34] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA., 1967.

[35] Cyril Goutte, Peter Toft, Egill Rostrup, Finn Å Nielsen, and Lars Kai Hansen. On clustering fmri time series. *NeuroImage*, 9(3):298–310, 1999.

Appendices

A. Pseudocodes

A.1. Play the winner rule

Pseudocode:

```
set failure threshold;
F = 0 ;                                              /* init failure count */
$A_1$ = random ($a_i$) ;                              /* choose a random acquirer */
for $t$ = 1, 2, 3, ... do
    send payments to $A_t$;
    if $F >= failure\ threshold$ then
        F = 0 ;                                      /* reset failure count */
        $A_t$ = switch ($a_i$) ;                      /* switch to a different acquirer */
    end
end
updateDBFailureCountASync;
refreshCacheASync;
/* when a payment response is received,                                    */
/* update failure count in the database                                    */
define updateDBFailureCountASync begin
    if $R_t = 0$ then increment failure count
        F++;
    end
end
```

The function *updateDbFailureCountASync* runs asynchronously, and for each payment response received, and in case it is a failure, then it increments the F failure count in the database. However, the cache that actually gives the failure count is only refreshed in a given time frame.

A.2. Greedy and pursuit bandit

Pseudocode:

```
set preference indicator $Q_1$;
set exploration rate $\epsilon$;
set recency rate $\alpha$;
for $t$ = 1, 2, 3, ... do
    switch randomly generated number do
        case $\epsilon$ %
            $A_t$ = random($a_i$) ; /* refresh cache every given time frame */
        end
        case 100-$\epsilon$ %
            $A_t$ = arg $max_a Q_t(a_i)$ ; /* choose acquirer with max preference
            */
        end
    endsw
    send payments to $A_t$;
end
updateDBPreferenceASync ;          /* on responses, update database */
refreshCacheASync;
/* when a payment response is received,                               */
/* update the preference indicators in the database                   */
define updateDBPreferenceASync begin
    for $i$ = 1, 2, ... $k$ do
        $Q_{t+1}(a_i) = Q_t(a_i) + \alpha * [R_t - Q_t(a_i)]$;
    end
end
```

A.3. Gradient bandit

Pseudocode:

```
set preferences $Q_0(a_i)$;
computeProbabilities;
for $t$ = 1, 2, 3, ... do
    $A_t$ = drawAcquirerUsingProbabilities;
    send payments to $A_t$;
end
updateDbASync ;                    /* on responses, update database */
refreshCacheASync;
/* when a payment response is received,                               */
/* update the preferences and probabilities                           */
/* in the database                                                    */
define updateDbASync begin
    for $i$ = 1, 2, ..., $k$ do
        $\bar{R_t}$ = average of all R up to transaction t;
        if $A_t = a_i$ then
            $Q_{t+1}(a_i) = Q_t(a_i) + \alpha * (R_t - \bar{R_t})(1 - p_t(a_i))$;
        else
            $Q_{t+1}(a_i) = Q_t(a_i) - \alpha * (R_t - \bar{R_t})p_t(a_i)$;
        end
        computeProbabilities;
    end
end
/* calculate probabilities using Boltzmann distribution               */
define computeProbabilities begin
    for $i$ = 1, 2, ..., $k$ do
        $p_t(a_i) = e^{Q_t(a_i)}[1/e^{sumQ_t(a)}]$
    end
end
```

A.4. Exp31

Pseudocode:

```
$\Gamma$ = -1 ;                                          /* init epoch count */
startNewEpoch;
computeProbabilities;
for $t$ = 1, 2, 3, ... do
    $A_t$ = drawAcquirerUsingProbabilities;
    send payments to $A_t$;
end
updateDbASync ;                        /* on responses, update database */
refreshCacheASync;
/* when a payment response is received,                                 */
/* update the preference indicators in the database                     */
define updateDbASync begin
    $\Gamma$++ $\hat{R}_t = R_t/p_t(A_t)$;
    computeProbabilities;
    $Q_{t+1}(A_t) = Q_t(A_t) + \hat{R}_t$;
    if $maxQ_t > q_\Gamma - (k/\gamma_\Gamma)$ then
        startNewEpoch;
    end
end
define startNewEpoch begin
    $q_\Gamma = (k * lnk)/[(e-1)4^\Gamma$;
    $\gamma_\Gamma = \min\left[1, \sqrt{(k * lnk)/[(e-1)g_\Gamma]}\right]$;
    $\omega_t(a) = 1$;
end
define computeProbabilities begin
    $\omega_{t+1}(A_t) = \omega_t(A_t)exp(\gamma\hat{R}_t/k)$;
    for $i$ = 1, 2, ..., $k$ do
        $p_t(a_i) = (1-\gamma)[\omega_t(a_i)/(sum\omega_t(a)] + (\gamma/k)$
    end
end
```

A.5. Thompson sampling

Pseudocode:

```
$S_0(a_i) = 0; F_0(a_i) = 0$ ;              /* init success and failure counts */
computeProbabilities;
for $t$ = 1, 2, 3, ... do
    $A_t$ = drawAcquirerUsingProbabilities send payments to $A_t$;
end
updateDbASync ;                            /* on responses, update database */
refreshCacheASync;
/* when a payment response is received,                                     */
/* update the success, failure counts, and probabilities                    */
/* in the database                                                          */
define updateDbASync begin
    if $R_t = 1$ then increment success count
        $S_t(A_t) + +$;
    else increment failure count
        $F_t(A_t) + +$;
    end
    computeProbabilities;
end
/* calculate probabilities using Beta distribution                          */
define computeProbabilities begin
    for $i$ = 1, 2, ..., $k$ do
        $p_t(a_i) = Beta[S_t(a_i), F_t(a_i)]$
    end
end
```

B. Simulation results

B.1. Play the Winner Rule

BIN	Cluster	Improvement	Regret	Weight	Best parameter (threshold)	Weighted improvement
457xxx	1	2.76	2.19	0.14	1	0.00
468xxx	1	1.14	1.22	0.14	21	0.00
438xxx	2	3.64	3.83	3.38	51	0.12
484xxx	2	0.61	1.46	3.38	31	0.02
435xxx	3	4.57	3.43	0.87	1	0.04
529xxx	3	7.31	6.29	0.87	1	0.06
449xxx	4	8.05	9.55	0.18	11	0.01
540xxx	4	3.61	5.08	0.18	1	0.01
513xxx	5	-1.38	10.12	0.05	1	0.00
513xxy	5	-4.46	10.47	0.05	1	0.00
470xxx	6	0.9	1.16	8.87	71	0.08
474xxx	6	0.67	0.61	8.87	81	0.06
435xxy	7	0.32	1.96	6.22	81	0.02
454xxx	7	0.66	1.25	6.22	91	0.04
414xxx	8	1.95	2.34	26.90	71	0.52
434xxx	8	0.52	0	26.90	81	0.14
448xxx	9	18.18	0	0.08	21	0.01
478xxx	9	-12.31	14.39	0.08	61	-0.01
542xxx	10	0.43	0.1	53.31	91	0.23
542xxy	10	0.32	0.36	53.31	91	0.17
Total						**1.54**

Table B.1.: Play the Winner Rule full results

B.2. Greedy and Pursuit bandit

BIN	Cluster	Improvement	Regret	Weight	Best parameters (exploration - recency rates)	Weighted improvement
457xxx	1	1.77	3.17	0.14	0.21-0.3	0.00
468xxx	1	1.42	0.94	0.14	0.1 - 0.11	0.00
438xxx	2	7.03	0.43	3.38	0.1-0.01	0.24
484xxx	2	2.02	0.05	3.38	0.1-0.01	0.07
435xxx	3	4.41	3.59	0.87	0.1-0.21	0.04
529xxx	3	6.68	6.91	0.87	0.21-0.3	0.06
449xxx	4	13.76	3.84	0.18	0.01-0.3	0.02
540xxx	4	4.04	4.65	0.18	0.11-0.3	0.01
513xxx	5	2.42	6.32	0.05	0.01-0.3	0.00
513xxy	5	1.92	4.09	0.05	0.1-0.11	0.00
470xxx	6	1.28	0.78	8.87	0.1-0.11	0.11
474xxx	6	0.79	0.49	8.87	0.1-0.01	0.07
435xxy	7	1.96	0.32	6.22	0.1-0.01	0.12
454xxx	7	1.03	0.88	6.22	0.1-0.01	0.06
414xxx	8	4.24	0.05	26.90	0.1-0.01	1.14
434xxx	8	0.62	0	26.90	0.1-0.01	0.17
448xxx	9	9.47	0	0.08	0.11-0.3	0.01
478xxx	9	-10.12	12.2	0.08	0.71-0.3	-0.01
542xxx	10	0.64	0	53.31	0.1-0.01	0.34
542xxy	10	0.71	0	53.31	0.1-0.01	0.38
					Total	**2.84**

Table B.2.: Greedy and Pursuit bandit full results

B.3. Gradient bandit

BIN	**Cluster**	**Improvement**	**Regret**	**Weight**	**Best parameter (recency rate)**	**Weighted improvement**
457xxx	1	1.01	3.93	0.14	0.91	0.00
468xxx	1	0.86	1.51	0.14	0.11	0.00
438xxx	2	7.97	0	3.38	1.01	0.27
484xxx	2	0.93	1.14	3.38	0.81	0.03
435xxx	3	5.87	2.13	0.87	1.01	0.05
529xxx	3	3.11	10.48	0.87	0.51	0.03
449xxx	4	12.73	4.87	0.18	0.81	0.02
540xxx	4	6.39	2.3	0.18	0.41	0.01
513xxx	5	4.26	4.48	0.05	0.81	0.00
513xxy	5	4.11	1.9	0.05	0.91	0.00
470xxx	6	0.59	1.47	8.87	0.91	0.05
474xxx	6	0.53	0.76	8.87	1.01	0.05
435xxy	7	2.16	0.13	6.22	0.11	0.13
454xxx	7	0.52	1.39	6.22	0.21	0.03
414xxx	8	4.78	0	26.90	0.61	1.29
434xxx	8	0.56	0	26.90	0.01	0.15
448xxx	9	7.73	0	0.08	0.91	0.01
478xxx	9	-8.75	10.83	0.08	0.41	-0.01
542xxx	10	0.44	0.1	53.31	0.11	0.23
542xxy	10	0.41	0.26	53.31	0.01	0.22
					Total	**2.57**

Table B.3.: Gradient bandit full results

B.4. Exp31

BIN	**Cluster**	**Improvement**	**Regret**	**Weight**	**Best parameter (NA)**	**Weighted improvement**
457xxx	1	-0.11	5.05	0.14	NA	0.00
468xxx	1	0.62	1.75	0.14	NA	0.00
438xxx	2	7.89	0	3.38	NA	0.27
484xxx	2	0.62	1.44	3.38	NA	0.02
435xxx	3	3.44	4.56	0.87	NA	0.03
529xxx	3	2.52	11.07	0.87	NA	0.02
449xxx	4	11.95	5.64	0.18	NA	0.02
540xxx	4	4.04	4.65	0.18	NA	0.01
513xxx	5	2.86	5.89	0.05	NA	0.00
513xxy	5	3.4	2.6	0.05	NA	0.00
470xxx	6	0.56	1.5	8.87	NA	0.05
474xxx	6	0.5	0.79	8.87	NA	0.04
435xxy	7	2.09	0.19	6.22	NA	0.13
454xxx	7	0.47	1.43	6.22	NA	0.03
414xxx	8	4.73	0	26.90	NA	1.27
434xxx	8	0.52	0	26.90	NA	0.14
448xxx	9	4.67	0	0.08	NA	0.00
478xxx	9	-10.69	12.77	0.08	NA	-0.01
542xxy	10	0.37	0.16	53.31	NA	0.20
542xxx	10	0.28	0.39	53.31	NA	0.15
					Total	**2.38**

Table B.4.: Exp31 full results

B.5. Thompson sampling

BIN	**Cluster**	**Improvement**	**Regret**	**Weight**	**Best parameter (NA)**	**Weighted improvement**
457xxx	1	2.17	2.77	0.14	NA	0.00
468xxx	1	0.89	1.48	0.14	NA	0.00
438xxx	2	7.96	0	3.38	NA	0.27
484xxx	2	0.45	1.62	3.38	NA	0.02
435xxx	3	5.8	2.2	0.87	NA	0.05
529xxx	3	3.44	10.15	0.87	NA	0.03
449xxx	4	12.5	5.09	0.18	NA	0.02
540xxx	4	6.48	2.21	0.18	NA	0.01
513xxx	5	4	4.74	0.05	NA	0.00
513xxy	5	3.85	2.16	0.05	NA	0.00
470xxx	6	0.59	1.46	8.87	NA	0.05
474xxx	6	0.63	0.66	8.87	NA	0.06
435xxy	7	1.99	0.3	6.22	NA	0.12
454xxx	7	0.49	1.41	6.22	NA	0.03
414xxx	8	4.77	0	26.90	NA	1.28
434xxx	8	0.5	0	26.90	NA	0.13
448xxx	9	5.45	0	0.08	NA	0.00
478xxx	9	-3.17	5.25	0.08	NA	0.00
542xxx	10	0.47	0.06	53.31	NA	0.25
542xxy	10	0.45	0.23	53.31	NA	0.24
					Total	**2.58**

Table B.5.: Thompson sampling full results