Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Computer Science

DIPLOMA THESIS ASSIGNMENT

Student: Linka Jan

Study programme: Open Informatics
Specialisation: Artificial Intelligence

Title of Diploma Thesis: Data-driven user engagement optimization for mobile applications

Guidelines:

User engagement optimization aims to maximize the number and the length of user interactions with a mobile application. The goal of this thesis is to explore how driven techniques based on machine data analysis can be used to automate the optimization of user engagement. The student should perform the following steps:

1) Familiarize yourself with the problem of user engagement optimization for mobile applications
2) Survey suitable methods for automated user engagement optimization
3) Formalize the problem of engagement optimization in way suitable for the application of data-driven techniques
4) Design and implement a suitable data-driven engagement optimization technique
5) ntegrate the implemented technique with a mobile application
6) Evaluate the technique on a realistic sample data from a mobile application

Bibliography/Sources:

[1] D. Silver, L. Newnham, D. Barker, S. Weller, and J. McFall. Concurrent Reinforcement Learning from Customer Interactions. Proceedings of the 30th International Conference on Machine Learning. in PMLR vol. 28, no. 3, pp. 924-932, 2013.

[2] B. Smith, G. Linden. Two Decades of Recommender Systems at Amazon.com. In IEEE Internet Computing, vol. 21, no. 3, pp. 12-18, 2017.

[3] A. Dosovitskiy, V. Koltun: Learning to Act by Predicting the Future. In CoRR, vol. abs/1611.01779, 2016.

Diploma Thesis Supervisor: doc. Ing Jakob Michal, Ph.D.

Valid until the end of the winter semester of academic year 2018/2019

[redacted]

prof. Dr. Michal Pěchouček, MSc.

Head of Department

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická
Z.S.
-4-

[redacted]

prof. Ing. Pavel Ripka, CSc.

Dean

Prague, September 09, 2017

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science and Engineering

Diploma Thesis

Data-driven user engagement optimization for mobile applications

Bc. Jan Linka

Supervisor: doc. Ing. Michal Jakob, Ph.D.

Study Programme: Open Informatics

Field of Study: Artificial Intelligence

May 25, 2018

Acknowledgements

I would like to express my great appreciation for doc. Ing. Michal Jakob, Ph.D., my supervisor, for his patient guidance, encouragement and useful critique. I would also like to extend my thanks to Ing. Pavol Žilecký and Ing. Jan Nykl for making it possible to run experiments on existing mobile application server.

Declaration

Abstract

Mobile application are large industry field that is highly competitive. This leads to a necessity of use of techniques for user engagement optimization. Smartphones are also able to use remote data processing which allows for automation of the optimization by machine data analysis techniques.

In this thesis, is for this reason formalized a milestone reaching problem and solution methods based on reinforcement learning are proposed. The proposed methods are tested in simulated environment and in experiments in real mobile application.

Abstrakt

Mobilní aplikace jsou velké průmyslové odvětví s vysokou konkurencí, které proto musí využívat techniky pro optimizaci angažovanosti uživatelů. Chytré telefony však mohou využívat vzdáleného zpracování dat, které umožňuje tuto optimalizaci automatizovat na základě technik strojové analýzy dat.

V této práci je proto formalizován problém dosažení milníku a navrženy způsoby jeho řešení založené na metodách posilovaného učení. Navržené metody jsou otestovány v rámci simulace a v experimentech v reálné mobilní aplikaci.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Smartphones are enormous field with 2 billion active users [7] that offers many opportunities for mobile application developers. The portable design leads to small screens, high distraction level and short sessions. Furthermore, there is competition with over 3.5 million other published applications [4]. However, the modern smartphones also offer many advantages over traditional desktop software. The most of the devices are connected to the internet allowing the developer to communicate directly with the user over notifications and in-app messages. Another advantage is the wide array of sensors available to track the location or movement of the device.

Average Android smartphone user launches less than half of installed applications and uses only 10 applications daily [8].

This short attention span and high competition create a need for the user engagement optimization. The internet connection on the other hand enables data gathering from all of the users. This opens window to data-driven techniques and many commercial products are offered for this optimization. However, very few of them seem to leverage machine learning and even fewer publish their methods. This creates an opportunity to formalize and build such technique.

1.2 Aim of the Thesis

The aim of the thesis is to explore how data-driven techniques based on machine data analysis can be used to automate the optimization of user engagement by designing and implementing a suitable data-driven technique.

To achieve this goal following steps will be performed. Firstly, the definition and possible metrics of user engagement will surveyed. Then commercially available user engagement solutions will be surveyed for the popular black-box solutions.

With information from the surveys a problem for user engagement optimization will formalized and optimization technique for solving the problem will be designed.

Finally, the designed technique will be implemented and evaluated.

1.3 Structure of the Thesis

In the Chapter 2 the domains of user engagement, mobile applications and commercial products for user engagement are surveyed. A promising problem related to user engagement optimization is also described. In Chapter 3 a narrow part of user engagement problem is formalized. Techniques for this problem are proposed in Chapter 4 and their implementation is described in Chapter 5. The performance of proposed techniques are evaluated in Chapter 6. Finally, Chapter 7 presents the conclusion from the evaluation results and proposes future work.

Chapter 2

Preliminaries

In this chapter, user engagement is defined and types of user engagement optimization are described. The domain of mobile application including its monetization in relation with user engagement is also described. Furthermore, the available communication channels are described. A Survey of commercial products for mobile optimization is also presented. Finally, a problem with promising relation to user engagement is described including some algorithms for solving it.

2.1 Defining User Engagement

In mobile applications domain, *user engagement* is a term used to denote various concepts of interaction between the user and the application. Bouvier et al. [10] define it for digital gaming as "the willingness to have emotions, affect and thoughts directed towards and determined by the mediated activity". To present a definition from the industry, Taige Zhang [28] from company Kissmetrics defines engagement as "the key performance indicator (KPI) for your app's core behaviors" with examples such as search queries, videos played per session, likes and comments on a post or simply time spent in a game. Andy Carwell [9] on the other hand defines engagement as a depth of the interaction in a contrast to the retention which is the breadth. In Fabric Answers [1] mobile analytics platform, the engagement refers to number of active users, the number of user sessions and their length.

All applications also have a notion of *a user lifetime.* It is the user retention and engagement, but is usually measured as retention [22].

As the previous engagement definitions are focused on value to the user which might be hard to measure, one of the alternatives for the developer is focus on a lifetime value (LTV) which is the total monetary value of the specific user to the developer. The engagement and lifetime value are linked, because only engaged user will contribute towards monetization and type of monetization model can point to the region of engagement that is important for optimization.

[1]`https://fabric.io`

2.2 User Engagement Optimization

User engagement with a mobile application is ultimately determined by perceived value for the user. The types of optimization methods described in following sections are streamlining the user experience, showing value, providing goals and providing content.

- Streamlining of the user experience is the most powerful method for applications that are not based on novelty. Automated methods in this include layout A/B testing and automated help for so called tripwires, when user stops using the application because of unwillingness or inability to do a required action.
- Showing the value is also possible optimization to improve the perceived value. Automated methods can be used to select a value to advertise and choose right context to do so.
- Providing goals works for keeping engagement with repetitive functionality or to showcase available features to the new user. Automated methods can be used for selecting the best encouragement towards the goal or for recommending the best goal to the user.
- Providing content is the main goal of social, news and other types of applications with growing content. Automated methods can be used to recommend the right content for the user.

2.3 Mobile Applications

2.3.1 Applications, Games & Gamification

Mobile applications are divided into many categories leading to different interaction patterns. The category with the most tools to affect user are games. This advantage led to inclusion of the game elements to many applications from other categories. This phenomenon is known as gamification.

2.3.2 Gamification

Gamification is defined by Deterding et al. [16] as "the use of design elements characteristic for games in non-game context" where gaming means playing structured by rules and competitive strife toward goals. The design elements interesting for improvement of engagement are for example badges, leaderboards, user levels, time constrains, limited resources or turns. The key feature of the elements is that they provide clear goals for the user.

Gamification can be successfully used for interactions that user considers to be valuable but not as engaging. This is very useful for education application such as language learning Duolingo and Memrise, multi-field learning Khan Academy, or to health and fitness applications such as Simple Habit.

Another advantage of virtual rewards is using them as reward for daily launch or simple action in the application. The simplest example is the streak counter that shows current count of consecutive days where the user took the action. This approach combined with social pressure is so effective in messaging application Snapchat that it is considered to be harmful by tech critic Tristan Harris [20].

2.3.3 Monetization Models

There are applications that serve simply as a tool for offline business or a just a digital storefront for selling content, however many of them rely on revenue generated by the usage of the application. Some possible revenue models are described as they promote different type of interaction and the income generated by a user might be a metric that is optimized by improving engagement.

2.3.4 Mobile App Monetization Models

Some common models are displaying third party advertisement, one time payment, subscriptions, various forms of data gathering, paid downloadable content, paid consumable content or often some combinations of these.

- Third party advertisement is the most common model where the most used platform AdMob only is currently used in over of quarter of applications published on the Google Play Store. As ad revenue comes from impressions (displaying the advertisement) or clicks, the revenue can be generated simply by making the user repeatedly open the application.
- One time payment is another common model for the application. With this model the user engagement is mostly important for attracting more users for example through favorable reviews.
- The subscription model is related to one-time payment model, however, it is a natural fit for applications that provide continually updated content. Many applications also require backend server computations and continuous security and compatibility updates. In case one time payment these costs must be paid by continuous growth of the number of users or by releasing paid upgrades. Under those requirements, subscription becomes more in line with the expenses for the publisher, however it might not be favored by the users used to the one time payment model. This model requires to persuade the user about the usefulness of the application in order to subscribe and keep the subscription.
- Data gathering and subsequent data monetization is another revenue building model as user generated data are useful as anonymous dataset used for rating and recommendations based on item similarity such as products, media or even real life locations. However, data gathering might also be used for building rich profile for advertisers that leverage it trough other channels in ethically questionable ways. In contrast to hidden gathering of data about user, the generation of useful dataset by explicit user input usually requires strong interest from the user.
- Purchase of additional downloadable features or game levels is a model that specifically requires reactivation of the user when the new content is released.

- So called consumables for games and gamified applications can provide a profitable monetization model. The game adds artificial waiting times or other barriers that can be skipped each time by paying with real-world money. This can lead to gaining an edge over other players and is made profitable by so called 'whales', which are users that keep playing a single game for a long time. Over the course of playing, the accumulated sum of their in-app purchases can often greatly exceed the sum they would be willing to spend on the game in the up-front single payment. This model however requires strong game mechanism design considerations such as in-game currencies and presenting the player with its benefits. These requirements and possibilities makes the publishers bring a whole different set of techniques for user engagement that are not suitable for other monetization models.

2.3.5 Combining Monetization Models

Most applications combine previously described models to target various type of the users. Some examples from the list of Google Play Store Editor's Choice which showcases handpicked applications that are the best in innovation, creativity, and design are following:

- SkyView has third party advertisement and version that removes it by one time payment.
- Memrise has a free version with an advertisement for bonus content available only with a subscription that is far more expensive than usual applications available as one time purchase.
- Goodreads lets users create a database of their books and rate them to create recommendation data and also uses affiliate links for purchases.
- Star Wars^{TM}: Galaxy of Heroes is example of free to play game that offers consumable purchases. The gameplay involving direct competition between players can lead to non-paying players to being easily defeated by paying players which motivates the both of them to buy additional consumables to stay competitive.

2.4 Retention

Retention is defined as a fraction of users that still use the application on specific day or in a specific week after they have installed the app.

As it is very common for mobile application to offer a version that is free to install, the majority of applications will see only a third of the users launching it in the day after the installation, as users many of them before committing. This makes the new users a prime target for engagement optimization.

Interesting data were released by Andrew Chen from company Quettra in 2015 [14]. As shown in Figure 2.1, there is a rapid drop at day 1 for all the applications. However, in these measurements from 125 million devices in five months since January 2015, the best performing non-preinstalled applications retained a much higher percentage of users even on day 90. As this metric measures the fraction of user that are active on the given day, it is not as useful for application in categories that are designed to be used less often. Especially for those case, weekly activity window for measurement can be useful.

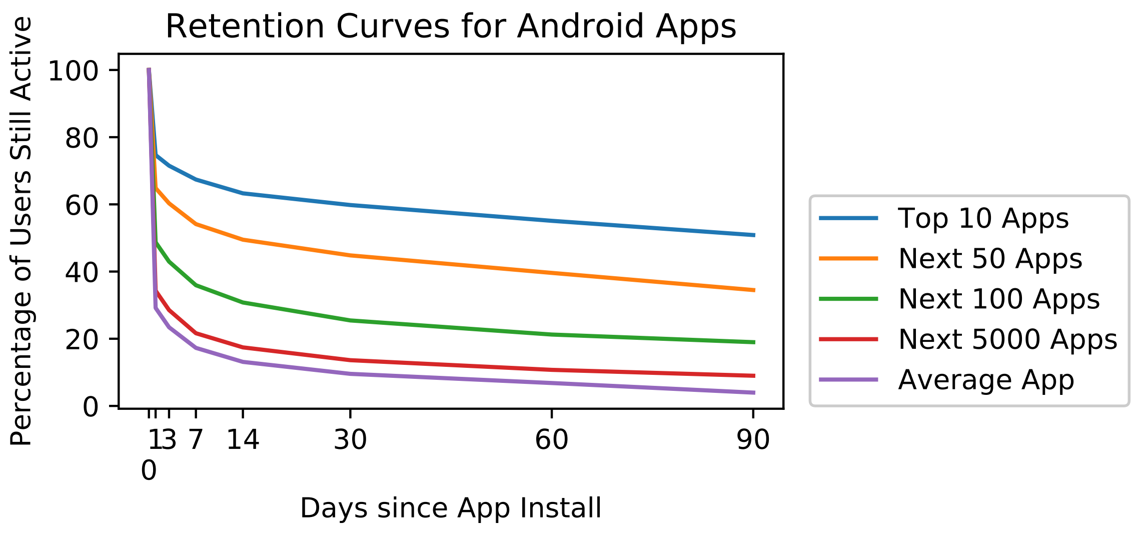


Figure 2.1: Retention Curves for Android Apps - Andrew Chen, 2015 [14]

In most cases, to be counted as active for the retention, the only action the user has to do is to launch the application. While this is useful value and means that user still at least knows that the application is installed, the narrower definition can be better optimization target. Example of this is "listener retention" as used at Soundcloud [9] that measured only users that listened to the content, which is the core action for their application and thus also better reflection of the user engagement.

2.5 Communication Channels

At the moment of writing (2018) the only major platforms for mobile applications are Android and iOS. The both platforms have support for their native applications and allow use of analogous sensors and services. However, the sensors and other APIs are restricted due to battery life and privacy concerns, but some of them are available through Google's or Apple's proprietary channels for registered developers.

Both platforms offer many communication channels towards the user. Any application has internet access that might be unstable or limited resulting in important software architecture considerations. However, due to history in web application, it is not uncommon to utilize connection to a remote server that can handle data from all the users. This connection allows to show downloaded content such as in-app messages and the push notification that are powerful channel with some restrictions. In some cases the email address or telephone number might be available to the application.

2.5.1 Email & Phone Number

Email and Phone numbers are two identifiers and communication channels that are not available to the application in the default case, but might be acquired as a part of creation of an account for the application. The email is provided by popular identity providers such as Google, Facebook or Twitter accounts, or as part of email and password authentication. The phone number also provide verifiable identifier that is cross-platform.

While both of the channels have the possibility to contact a user even without the application installed, this is not that common for mobile only applications and might be limited by a law.

2.5.2 In-app Messages

A message from the application operator can downloaded and displayed inside the application for example as an overlay or inside a content feed that is important part of many applications working with user generated content. The content and graphical representation is under full control of the operator and can use the same components as the rest of application user interface.

The main disadvantage of this channel is that it can target only users using the application at the moment. While this is not as much of a problem for games with longer sessions, a context dependent application might be launched only for a brief time and on-the-go.

2.5.3 Notifications

Both platforms provide push messages that can be send over internet connection to the device even when the application is not running. This message is presented to the user as a notification. The main components of a notification is the title and the text content. If the user is currently using the phone it is displayed over any currently running application for a brief time. On a inactive device, it is displayed on the lock screen. The notification offers interaction by clicking the body or one of the optional buttons. This can be used for opening the application on any screen inside the application and thus guiding the user directly to some functionality or just to do some action without opening the application.

The battery life concerns limited the background computation on iOS since first versions, meaning that push messages must be delivered trough Apple Push Notification service (APN) that can forward the notifications through Apple's servers to the device. While older Android versions allowed custom solutions for message delivery as the background work was not restricted, the recent versions restrict it only to short service windows. However, Google provides Firebase Cloud Messaging service (FCM) that also allows forwarding the notifications through the service that can ignore the battery optimization restrictions.

The important difference between the platforms is that notifications are the core part of mobile Android OS. The improvements of the notification tray and interactivity of the notifications play prominent role in updates of the OS and are even synchronized to Wear OS wearables connected the Android OS device. Unopened notifications are shown as an icon in the status bar reminding the user it can be accessed from the notification drawer by pulling it from the top of any screen. The iOS also has notification drawer, however without the icons in status bar, the visibility is much lower. This is noticed by the reviews [23] and more importantly by analytics on opened notifications. Other important difference is, that on Android, every application can show the notifications as default, but on iOS, the application must request an explicit permission from the user first.

The various industry analytics report show very varied numbers for the share of the users opening the messages, also known as click-through rate (CTR). While some sources such as Kahuna in 2014 [13] claimed CTR of 10-40% based on application category, Leanplum in 2016 [17] presented open rate of 1.77% on iOS and 3.48% for Android and Phiture [11] the rate of 0.1-10% for iOS and 0.5-15% based on level of personalization of the message for the user.

2.6 Recommendations

Delivering relevant content to the user is an important part of user engagement in shopping and media consumption applications. There is a variety of published solutions for this problem. Ordered by number of items from which the recommended item is selected, Li et al. [21] propose usage of *contextual bandit problem* algorithms for selection of the highlighted news item on a front page from small pool. The classical solutions use *collaborative filtering* to recommend items by learned similarity between users or the items themselves. The item-based solution proved to be important for Amazon store [24]. Current developments in this area involve training of *deep neural networks*, for example Covington et al. [15] from video service Youtube created successful two stage architecture, which however requires massive computational power and data set size that is not available to majority of application developers.

2.7 Mobile Optimization Industry

Acquisition, retention and engagement optimization for mobile applications is an industry field with wide variety of companies offering their services to the mobile application developers. The services range from more narrow such as message scheduling, re-activation messages for inactive users, message personalization, or layout A/B testing to more data analytics based segmentation of user types or prediction of certain events.

The offered services are however available only as the interface, with optimization approaches not known by public. Despite this, a survey of commercially available services should show problems that have impact in real applications.

The following commercial offers are surveyed for available user engagement features:

- Mixpanel[2] is a popular analytics solution that provides many data-driven features. It offers automatic segmentation, prediction for future analytics events based on single user history and anomaly analysis. It also offers sending of messages based on users actions through push notification, email and in-app messages. Finally, A/B testing of layouts is also offered.

[2] `https://mixpanel.com`

- Firebase[3] is a Google platform that offers wide variety of products for mobile application developers. In addition to tools for building applications such as real time database, it integrates Google Analytics and can use the user event data for prediction of the future events. Tools for A/B testing of layouts and messages are offered with ability to show experiments only to certain segments defined by analytics events.
- Flurry[4] is a popular analytics solution owned by Yahoo. It offers usage analytics and scheduling of push notification messages triggered by received event from the application. It also allows creation of user segments, however the functionality is not advertised as automatic or self-learning.
- Localytics[5] is an analytics solution that offers prediction of user churn and is focused on precise targeting of users with messages. An uncommon feature of their targeting is the possibility to use location to trigger sending of the message.

The commercial services mostly use engagement optimization as a term for session length from analytics view or as a term for sending messages through various channels such as push notifications while combined with event tracking to precisely target the messages in case of specific product. Another popular option is prediction of future event, especially of user stopping using the application.

Another surprising use of data to optimize engagement is automated crash reporting. As shown in report by Apteligence [2] users are up to 8x less likely to return to the application the next day after a crash. This degradation of user experience also leads to Google Play Store lowering the promotion of applications with high crash rate.

The importance of crash reporting for the applications can be inferred from inclusion of Crashlytics[6] service library in over 43% of Top 500 Android applications [3]. The data-driven techniques are however targeting the developers of the applications by prioritizing common crash causes and automated hints towards possible primary causes of the crash.

2.8 Multi-armed Bandit Problem

The multi-armed bandit problem (MAB) as described by Sutton & Barto [25] involves concurrent learning and exploitation of stochastic environment that occurs for example when deciding between two advertising layout in case of A/B testing for higher click through rate.

The multi-armed bandit problem is formalized with following abstraction: We are given a slot machine with N arms. At each time step $t = 1, 2, 3, ...$ one of the arms is pulled and the agent immediately receives reward chosen from a stationary probability distribution dependent on arm selected. The objective is to maximize the expected total reward over some time period T. Rewards for repeated selection of the same arm are independent and identically distributed and independent of the selection of the other arms.

An algorithm for MAB has no beforehand knowledge about the distribution associated with the arm. If the distribution was known, the optimal algorithm would simply always

[3]`https://firebase.google.com/`
[4]`http://www.flurry.com`
[5]`https://www.localytics.com`
[6]`https://try.crashlytics.com/`

pull the arm associated with the largest expected reward. Therefore, the algorithms used for this problem balance between exploration pulls that try to improve the model of the distributions that might lead to higher expected reward in the long run and the exploitation pulls that try to maximize reward at the current step.

2.8.1 Contextual Bandit Problem

The contextual bandit is extension of MAB problem that adds context x_t at each time step t. Probability distribution from which is the reward chosen depends on the context in addition to the arm selection.

2.8.2 ϵ-greedy Algorithm for MAB

The simplest model for MAB works with an assumption that if the expected reward for each arm a was $\mu_a = \mathbb{E}[r_t|a]$ and was known, then $\mu_* = \max_a \mu_a$ could be simply chosen. The simple estimation of the expected reward of pulling arm a at time t can be created by averaging the rewards actually received from pulling each arm:

$$\overline{\mu_a(t)} = \frac{\sum_{i=1}^{t-1} r_i \mathbb{1}_{a(i)=a}}{\sum_{i=1}^{t-1} \mathbb{1}_{a(i)=a}}$$

where $\mathbb{1}_{a(i)=a}$ is 1 if arm a was chosen at time i and 0 if it was not. The default expected reward in case of denominator being 0 can be set to 0.

With this model of $\overline{\mu_a}(t)$ the ϵ-greedy algorithm chooses with probability 1 - ϵ the arm $a(t) = argmax_a \overline{\mu_a}(t)$ with ties broken randomly. This is called *a greedy selection* as it exploits the knowledge at the time to maximize immediate reward. In order to explore other than the first successful arm, at each time step, a random arm a from the N arms is selected with the probability ϵ instead of the greedy selection. The pseudocode of this algorithm based on described by Sutton & Barto [25] can be seen in Algorithm 2.1.

Algorithm 2.1 ϵ-greedy Algorithm for MAB

```
for each arm a do
    $\overline{\mu_a} = 0$
    $n_a = 0$
end for
for $t = 1, \dots, T$ do
    if Draw from $[0, 1) < \epsilon$ then
        Select $a(t)$ = random arm $a$
    else
        Select $a(t) = argmax_a \overline{\mu_a}$
    end if
    Observe reward $r_t$
    Update $n_a = n_a + 1$
    Update $\overline{\mu_a} = \overline{\mu_a} + \frac{1}{n_a}(r_t - \overline{\mu_a})$
end for
```

2.8.3 Thompson Sampling

Thompson sampling algorithm for MAB is based on the idea of randomly choosing an arm based on the probability of it being optimal. This idea of probability matching described by William R. Thompson [27] led to commonly referring to this algorithm as Thompson sampling.

To apply this algorithm on MAB problem, the problem is redefined in Bayesian context. The past observations are modeled by likehood function $P(r \mid a, x, \theta)$ where r is reward, a is selected arm, x is the context and θ are hidden parameters.

While the probability matching approach is not optimal in the Bayesian decision making about single decision and thus suboptimal compared to deterministic strategy in single exploitation step, it leads to efficient balance of exploration and exploitation.

2.8.4 Thompson Sampling for Bernoulli Bandits

In cases where the reward for a MAB is binary only, e.g. success or failure, a Beta probability distribution can be efficiently used with Thompson Sampling to achieve efficient exploration and strong automatic exploitation during later time steps. This is possible, because a Beta distribution is conjugate prior probability distribution for Bernoulli distribution that is used as model for the success rate of each arm.

The algorithm used for example by Chapelle & Li [12] shown in 2.2 is given prior knowledge as α and β parameters as $(1, 1)$ in unbiased case or larger to set direction and magnitude of the bias.

Algorithm 2.2 Thompson Sampling for Bernoulli Bandit

```
Given: $\alpha, \beta$
for each arm a do
    $S_a = 0$                                    ▷ Success counter
    $F_a = 0$                                    ▷ Failure counter
end for
for $t = 1, \ldots, T$ do
    for each arm a do
        Draw $\theta_a$ from Beta$(S_a + \alpha, F_a + \beta)$
    end for
    Select $a(t) = argmax_a \, \theta_a$
    Observe reward $r_t$
    if $r_t$ is success then
        $S_{a(t)} = S_{a(t)} + 1$
    else
        $F_{a(t)} = F_{a(t)} + 1$
    end if
end for
```

2.8.5 Delay in Multi-armed Bandit Problem

In many real-world situations that otherwise fit the multi-armed bandit problem, there is a slight delay between selection of the arm to pull and observability of the reward. Example of this is a selection and a display of a layout of a web page and user clicking or not clicking on tested button while visiting the page.

This delay is at best of our knowledge not handled by specific algorithm, but treated by many authors only as a constraint leading to choice of algorithm with best result at required maximum delay of solved situation. This comparison is explored for example by Chapelle & Li [12] and Joulani et al. [19]. Hill et al. [18] showed that in live production system an algorithm not considering delay can be successful even at a delay as big as tens of thousands selections.

Chapter 3

Milestone Reaching Problem

In this chapter *a milestone reaching problem* is formalized in a way which can be optimized by data-driven algorithms and which affects the user engagement in ways described below.

3.1 User Journey

For this problem, a user journey is defined as a sequence of milestones reached by the user that are sufficient to turn the user from beginner to a regular user. Each milestone can be reached by some specific action by the user. The actions might provide immediate benefits, show possible future benefits or just strengthen a habit.

3.1.1 Milestones

A milestone is a domain specific and might not be directly observable by the application or only after some time. An early milestone reaching actions taken by the user might be for example watching a first video, creating an account, tracking a first fitness activity with the application or simply returning to the application in a day after she installed it. Later milestones might serve as exploration of non-core actions such as adding friends in non-social application, switching to different target in a case of fitness applications or setting up some custom content preferences. More complex milestones, such as doing an activity streak for a week, can have required user activity in each day, however it is possible for example to only define sixth and seventh day activity as the targets for optimization to let the user to self-select towards the milestone over the first days.

3.2 Milestone Reaching Optimization Components

In this section we describe how we formalize milestone reaching in terms of messages, activation conditions, rewards and context vectors. The user attributes, interaction history and location are also described as user data sources for creation of activation conditions and context vectors.

3.2.1 Messages

A message in a milestone reaching problem is the way of communication with the user that is used to affect her and trigger the reaching of the milestone. To use mobile notifications as a channel for delivering the message to the user, it can contain short text title and short text description. As the interface for displaying the notification is generally controlled by the OS, the support for emoji becomes an important part of the customization of the message.

Emoji are ideograms or smileys that can be displayed inside the text. Each emoji is only defined as Unicode characters defined by short text description. This means each vendor can supply their own graphics, which can in rare cases lead to slight shift in meaning of displayed message in comparison to one intended and previewed by the sender.

The content of the message is created using domain knowledge of the operator and should promote a way of achieving the milestone or just remind of its existence.

3.2.2 User Data

The user attributes describe the user in general and application domain specific ways. The general attributes are for example gender, age bracket, country or general interest in some area such as technology or sport. The domain specific attributes are other preferences directly stated by the user such as preferred movie genre or reported fitness skill level.

Interaction history is a summary of action taken by the user while using the application, e.g. preferred movie genre based on movies watched using the application, performance level based on measured fitness activity or interest level based on the interaction with previous messages.

Location is another type of data source for the problem. It can be detected by the variety of sensors in a modern mobile device, but the acquisition is limited by the privacy and battery life concerns.

- Geofencing wakes the application up if it enters or leaves a location, making it a viable activation condition. However, without specialized hardware at the location such as a beacon, the assumed accuracy is only about 50 meters. This is enough to detect that the user is at home, at work or at some other preset location or at neither. Geofencing re-uses location sources such as Wi-Fi scans which leads to low battery impact.

- Activity recognition can differentiate different types of movement such as walking, driving a car, or not moving. This, however, requires many sensor readings. This makes the activity type based activation conditions viable only if the application is already actively using fine location sensors such as GPS.

- Coarse location of the device can also be utilized especially in a fusion with other data sources about the environment such as weather situation, traffic conditions or local events. This can be used both as a activation condition or similarly to an user attribute.

3.2.3 Activation Conditions

An activation condition fulfillment leads to including user into a milestone reaching problem. It combines an event such as user's action inside the application, detection of real-world situation or simply a physical clock reaching a time with good change to read the message, with a conditions on user attributes or interaction history such as having no friends added inside the application yet. Each activation condition fulfillment is treated as a logical time step leading to a selection of the best message to send as the trigger towards the milestone.

3.2.4 Context Vector

At each logical time step a context vector is gathered as a representation of the user affected by the environment approaching the milestone. It contains relevant subset of user attributes, interaction history summaries and addition environment data fused with the location.

3.2.5 Reward

The reward is granted if the user passes the milestone or a proxy for a milestone passing in the future if it is hard to trace from a message. However, this reward can be observed only after some physical time as the user needs time to react to the message, which can be minutes in case of digital only milestone, or hours or even days in cases where milestones require real-world action. It is assumed that reaction during any time in the window is equally valuable.

One of the effects of the delayed observability of the rewards is that the activation condition fulfillment happening only in batches with large enough interval between lead to the rewards being observable also in batches.

3.3 Milestone Reaching Problem Specification

We define milestone reaching as the following problem:
We are given a set $M = \{m_1, m_2, \ldots, m_k\}$ possible messages for users and physical time window W for each user to reach the milestone since the activation condition fulfillment. After each fulfillment at each logical time step $t = 1, 2, 3, ...$, one of the k messages M must be selected by a message selection function $\psi_t(M, H_t, x_t)$ and sent. When sent, message yields a hidden real-valued reward r_t from a stationary probability distribution that depends on selected message m_t and the context vector x_t. The reward r_t becomes observable at time step $t + \delta_t$ where the delay δ_t is the effect of the time window W on reward observability. The delay $\delta_t \in \mathbb{N}_0$ is unknown at time t but observability at the end of the fixed size time window W guarantees that $\forall t_1, t_2 : t_1 \leq t_2 \Rightarrow t_1 + \delta_{t_1} \leq t_2 + \delta_{t_2}$

The goal is to maximize expected total reward in time T, i.e.

$$\mathbb{E}[\sum_{t=1}^{T} r_t(x_t, m_t)].$$

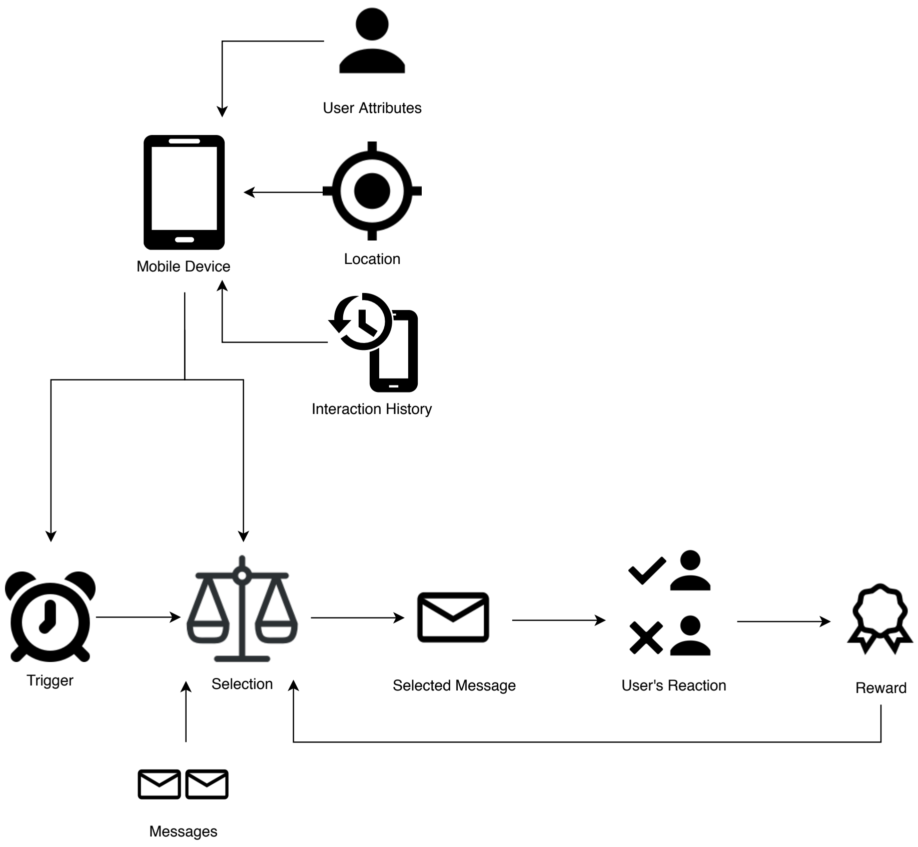


Figure 3.1: Milestone Reaching Problem Diagram

An algorithm for milestone reaching problem needs to choose at every time t a message using current context vector x_t and observable history of rewards received with corresponding message selection and context vector $H_{t-1} = \{r_z(x_z, m_z) \mid z \leq t + \delta_t\}$ to learn optimal message selection series ψ^* which maximizes the expected total reward.

3.4 Milestone Reaching Problem Properties

It is not generally true that a user being sent a message repeatedly or receiving different messages in succession are events that are independent with relation to successful milestone reaching. However, if we assume that user that has received multiple messages related to a milestone and have not reached it is not interested in the milestone enough to be persuaded by the automated messages, then we can remove such user from the optimization process.

Given the upper limit for repeated sends, if the number of the users is large enough compared to the limit and the number of already received messages is a part of the context, we assume that dependency becomes negligible.

Chapter 4

Milestone Reaching Optimization Algorithm

In this chapter, multiple algorithms for solving milestone reaching problem are proposed and their conditions and relations to algorithms for multi-armed bandit problem and its extensions. Trade-offs of some methods are also compared.

4.1 Relation to Different Problems

The milestone reaching instance has time window W before it becomes observable. The time window is defined by physical clock value, i.e. 24 hours, leading to a delay δ_t at each time step defined in time steps. The ratio between the delay δ and final time step T fundamentally affects the class of methods usable for the problem. If the delay δ is close to T, any informed selection can be only done for few final steps. However, if the delay δ is much smaller than T, as is a case if time window has size of a day and final time steps comes after many days, the balance between the exploration for the best action and exploitation of current knowledge becomes important.

As the pattern of the user journey through the application generally does not change too fast, the following methods target the instances that require the balance between the exploration and exploitation. As this is common in the field of reinforcement learning, proposed methods incorporate various algorithms used in this field.

Multi-armed bandit problem as described in Chapter 2 also aims at optimal selection in stochastic environment. The independence claim required in bandit problem is also present in milestone reaching problem under the previously stated repeat limit.

4.2 ϵ-greedy Algorithm with Context Buckets

The first method for solving milestone reaching problem is based around the idea of ϵ-greedy algorithm for multi-armed bandit problem as presented by Sutton & Barto [25]. It assumes that we are given mapping from the context vector x to few buckets. The mapping can be created by some unsupervised algorithm, by creation of the buckets by the operator and learning the mapping by supervised methods or simply by manual decision by the operator with the domain knowledge.

The core idea of the algorithm is the estimation of reward by averaging the rewards actually received from sending the messages.

Given the messages for each of the buckets, average reward for each of the messages is kept. After each activation condition fulfillment, the estimates of the rewards are updated by any rewards that became observable since last fulfillment. Then, a context bucket is selected by the mapping from the context vector. From the bucket, either the message with largest expected reward is selected, or, with probability ϵ a random action from the bucked is selected instead. This message is sent and the corresponding reward will be observable after some future trigger. The pseudocode is described in Algorithm 4.1.

The ϵ based exploration leads to a flat rate of exploration after the greedy actions stop lowering the estimate of expected reward for at the moment largest reward. This means that large ϵ such as 0.1 limits the total received reward as fraction of selections is still assigned to messages that might be clearly inferior. On the other hand low ϵ can lead to slow start, as first few selections can lead to hundreds of selections on message with lower actual value.

While delay δ generally lowers the accuracy of the reward estimations, a delay before first observed reward actually forces random exploration that can prevent slow start with low ϵ.

Algorithm 4.1 ϵ-greedy Algorithm

```
Given: messages M divided into $b$ buckets $M_1, \dots, M_b$, $\epsilon \in (0,1)$, mapping from context
vector $x$ to buckets $M_\beta$
$H \leftarrow \{\}$
$O \leftarrow \{\}$
for each message m do
    $\overline{\mu_a} \leftarrow 0$                                        ▷ Expected value
    $n_m \leftarrow 0$                                         ▷ Selection counter
end for
for $t \leftarrow 1, \dots, T$ do
    for all $r_\tau$ in $H$ and not in $O$ in ascending order do
        Observe reward $r(\tau)$ and $O \leftarrow O \cup r(\tau)$
        Update $n_{m(\tau)} \leftarrow n_{m(\tau)} + 1$
        Update $\overline{\mu_{m(\tau)}} \leftarrow \overline{\mu_{m(\tau)}} + \frac{1}{n_{m(\tau)}}(r_\tau - \overline{\mu_{m(\tau)}})$
    end for
    Select the bucket $M_\beta$ by mapping from $x_t$
    if Draw from $[0,1) < \epsilon$ then
        Select $m(t) \leftarrow$ random message $m$ from $M_\beta$
    else
        Select $m(t) \leftarrow argmax_m \overline{\mu_m}$ where $m$ is in $M_\beta$
    end if
    Send message $m(t)$ and $r(t)$ will be observable in history at $t + \delta_t$
end for
```

4.3 Thompson Sampling Algorithm with Context Buckets

This algorithm is based around the idea of Thompson sampling. It assumes we are given the mapping from context vector to the buckets as described for previous ϵ-greedy algorithm.

A limitation of this method is that it only works with binary reward. However, binary reward is also the most straightforward representation of the success or failure of passing the milestone by the user.

Given the messages for each bucket, a count of successful and unsuccessful trials of sending for each message is kept. The algorithm as seen in pseudocode form in Algorithm 4.2 utilizes the increasing certainty about success rate of each message to automatically decrease the exploration rate.

Algorithm 4.2 Thompson Sampling

```
Given: messages M divided into $b$ buckets $M_1, \dots, M_b$, $\alpha, \beta$, mapping from context vector
$x$ to buckets $M_\beta$
$H \leftarrow \{\}$
$O \leftarrow \{\}$
for each message m do
    $S_m \leftarrow 0$                                              ▷ Success counter
    $F_m \leftarrow 0$                                              ▷ Failure counter
end for
for $t \leftarrow 1, \dots, T$ do
    for all $r_\tau$ in $H$ and not in $O$ in ascending order do
        Observe reward $r(\tau)$ and $O \leftarrow O \cup r(\tau)$
        if $r(\tau)$ is success then
            $S_{m(\tau)} \leftarrow S_{m(\tau)} + 1$
        else
            $F_{m(\tau)} \leftarrow F_{m(\tau)} + 1$
        end if
    end for
    Select the bucket $M_\beta$ by mapping from $x_t$
    for each message $m$ in $M_\beta$ do
        Draw $\theta_m$ from Beta$(S_m + \alpha, F_m + \beta)$
    end for
    Select $m(t) = argmax_m \, \theta_m$
    Send message $m(t)$ and $r(t)$ will be observable in history at $t + \delta_t$
end for
```

Chapter 5

Implementation

In this chapter the software architecture used to implement and deploy proposed *milestone reaching problem* solutions is described. It is divided into implementation of the message optimization module and additional functionality required from the application itself and its backend server. The simplified data flow is shown in Figure 5.3.

5.1 Message Optimization Module

The optimization module is a library that offers selection of the best action over longer timeframe for which it includes internal model and rewards persistence. It is designed for solving *milestone reaching problem* by selection of the messages, however it can be used for selection of other actions.

The module requires Java 8 runtime environment and access to SQL database with prepared tables. The initial inputs consist of a list of messages, parameters of selected algorithm and a provider of a connection to the database in form of `java.sql.Connection` [1]. During the optimization process, it requires being called back when any reaction to the message becomes serializable and when the batch of responses to the selected messages is finished. With the previous inputs, a recommended message can be requested by a trigger event coming from the hosting backend system.

Internally, the module takes cares of restarts and updates of the deployed system by eagerly serializing the state of any running experiment and by tracking the status and origin of each of the messages selected by the experiments.

The module is written in Kotlin [6] 1.2 language and can be used as a library jar by JVM languages such as Java. As the experiments and messages are identified by *universally unique identifier* (UUID) it requires database with support for this data type such as PostgreSQL[1].

The test deployment uses Java 8 language at JRE 1.8 and with connection to PostgreSQL 10.3 database through PostgreSQL JDBC Driver 42.1.4.

[1]`https://www.postgresql.org/`

5.2 Mobile Application Integration

The deployment to an existing Android application requires a delivery method, a notification display implementation and a method for upload of user's reaction and the context composed from location, interaction history and user attributes. Each requirement is constrained by Android battery saving and privacy measures.

5.2.1 Notification Delivery

Notification delivery is effectively forced to use Firebase Cloud Messaging (FCM) due to battery life optimization by the Android OS [5]. The application must be registered as listening to this proprietary Google service. The registration yields a registration token, which must be forwarded to the backend as it is required to target the device. This communication channel allows pushing of short JSON payload to the device. This payload is used to define the notification message and is used at the same time to deliver additional data that can be displayed as in-app message.

5.2.2 Notification Display

When the data message is delivered, a background service owned by the application is invoked without displaying any user interface. The service has brief time period to compose and present the notification message data to the OS. The delivery status is forwarded to the sender by FCM itself.

Example of notification displayed by Android 8 phone is show in Figure 5.1 as displayed on the phone lockscreen and in expanded state if user opens the notification drawer in Figure 5.2.

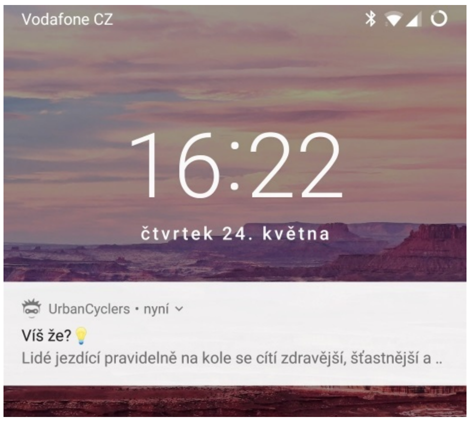
To utilize the click action of the notification a deep linking is implemented allowing the direct opening of a screen that is otherwise reachable only deep in the hierarchy of the entry point of the application launched from the system launcher. As the application is not running while the notification is displayed, an ID of the message must be forwarded during the open event.

5.2.3 Reaction and Context upload

If the user successfully opens the application by the notification, this event must be forwarded to the backend server by the application itself. The communication with the test deployment backend server is over REST style API. As the application can be stopped or killed by the OS at any time, any data must be persisted on the device. Then the communication is scheduled by the system work API that can use network connection and battery status to determine when to start.

5.3 Application Backend Integration

Application backend server is the component that offloads battery intensive computations, allows synchronization between the devices of single user and centralized interaction between the users such as leaderboards or public comments.


Vodafone CZ

16:22

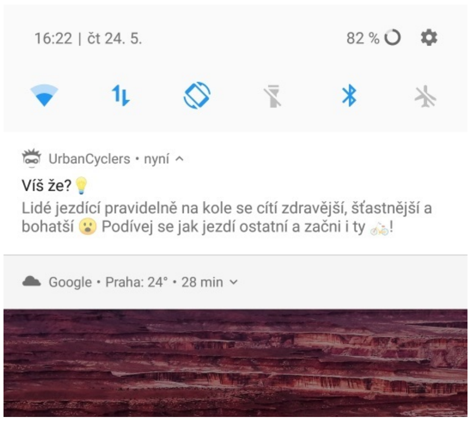
čtvrtek 24. května

UrbanCyclers • nyní

Víš že?💡

Lidé jezdící pravidelně na kole se cítí zdravější, šťastnější a ..

Figure 5.1: Example of notification on Android 8 lockscreen


16:22 | čt 24. 5. 82 %

UrbanCyclers • nyní

Víš že?💡

Lidé jezdící pravidelně na kole se cítí zdravější, šťastnější a bohatší 😮 Podívej se jak jezdí ostatní a začni i ty 🚲!

Google • Praha: 24° • 28 min

Figure 5.2: Example of expanded notification on Android 8

The server instantiates the action optimization module experiments and sends the messages to the device through FCM. It receives the reactions to the messages from the application and gathers user attributes, interaction history and in case of the test deployment, the location of the device.

In combination with timers the received data provides the information about activation conditions fulfillment and context about the user and environment to the optimization module experiments.

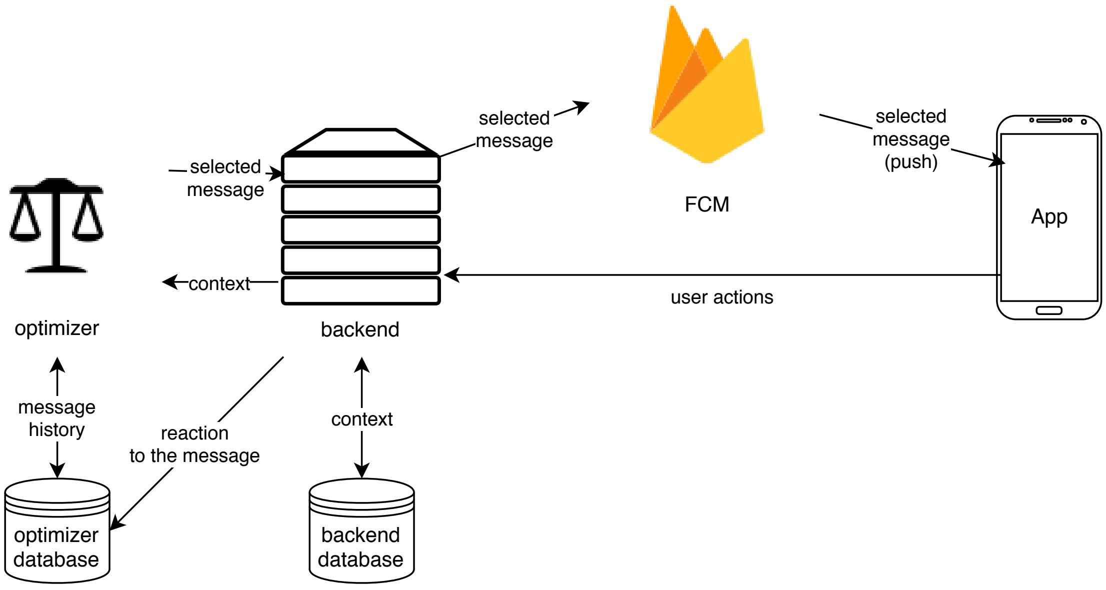


Figure 5.3: Simplified data flow between implementation components

Chapter 6

Evaluation

In this chapter the proposed algorithms are evaluated. In the first section the simulated environment is used to explore the properties of the algorithms under various conditions of the environment. In the second section the experiments using the application UrbanCyclers on real user base are described and evaluated.

6.1 Simulation Results

The simulations were performed to test the performance of ϵ-greedy sampling and Thompson sampling algorithms under delay.

The test environment has two messages with success rate of 15% and 20% respectively. This represents environment with two moderately successful messages with clear difference of preferred message. Each experiment configuration run for 1000 time steps and was repeated 4000 times. The configurations made the reward observable immediately, each 50, 100 and 200 selections.

6.1.1 ϵ-greedy Simulation

Figure 6.1 shows the average reward for ϵ-greedy sampling algorithm with $\epsilon = 0.1$. The horizontal lines shows the first selection where any learned knowledge could be used for the corresponding configuration. The same averaged series are also shown in Figure 6.2 with each step series shifted so that the exploitation of observed rewards start at step 0.

This data show that the delayed observability actually can enable the algorithm to escape the possibility of greedily selecting the lower actual value message.

6.1.2 Thompson Simulation

Figure 6.3 shows the average reward for Thompson sampling algorithm with $\alpha = 1$ and $\beta = 1$. The horizontal lines again shows the first selection where any learned knowledge could be used for the corresponding configuration.

Figure 6.4 presents the series shifted with first observable batch to step 0. It shows the undesirable effect of the delay to estimate distribution update. The data from first batch are unable to offset the delayed updates during later steps leading to lower accuracy.

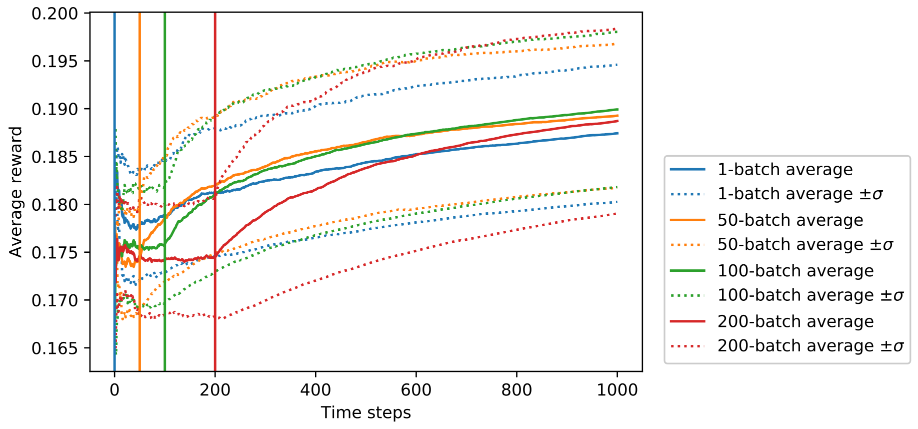


Figure 6.1: Effect of delay from batching on ϵ-greedy sampling algorithm

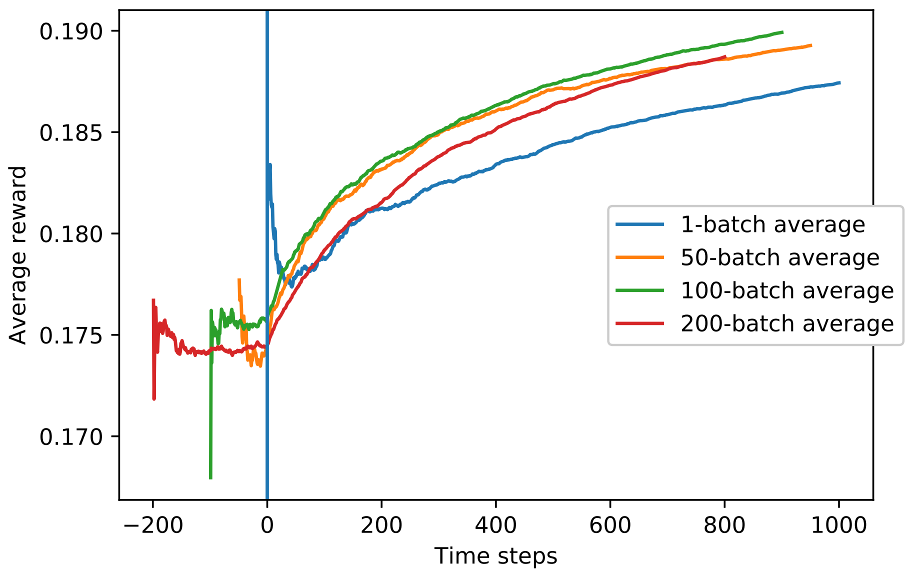


Figure 6.2: Effect of delay from batching on ϵ-greedy sampling algorithm after first batch

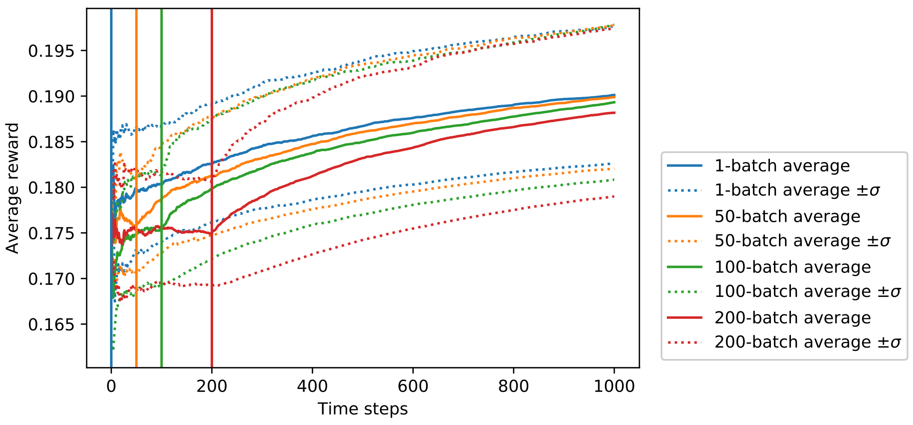


Figure 6.3: Effect of delay from batching on Thompson sampling algorithm

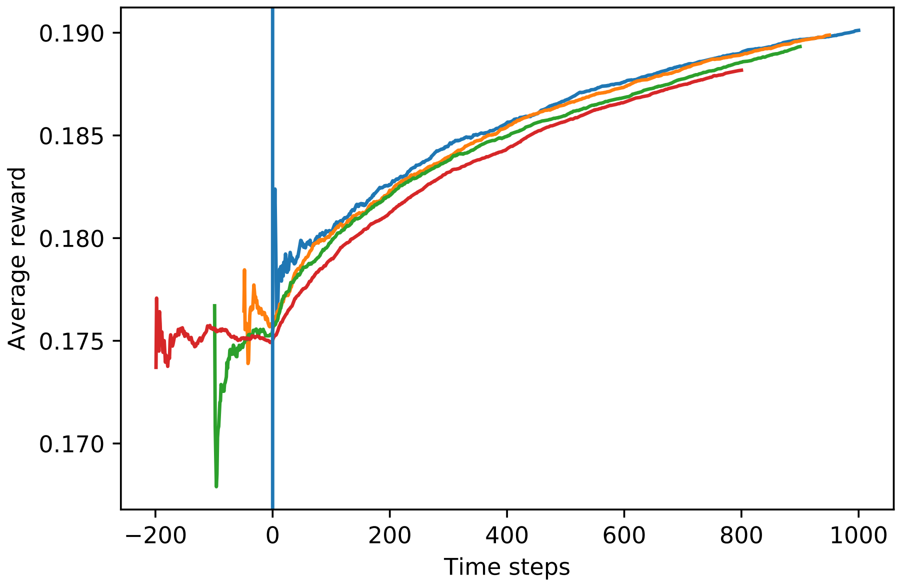


Figure 6.4: Effect of delay from batching on Thompson sampling algorithm after first batch

6.1.3 Simulation Comparison

Figure 6.5 shows comparison of ϵ-greedy sampling algorithm with Thompson sampling algorithm from 3000 runs of 3000 steps. The $\epsilon = 0.1$ and $\alpha = \beta = 1$. The average reward is shown in band of one standard deviation around it represented by the dotted line. While the ϵ-greedy sampling shows advantage at start, the adaptive estimate by Thompson sampling allows better exploitation in later steps.

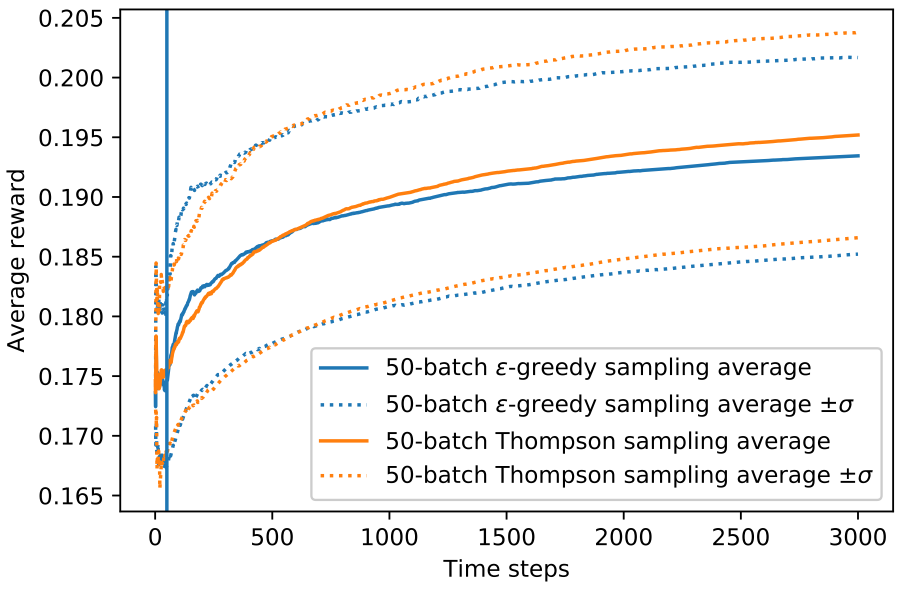


Figure 6.5: Comparison between ϵ-greedy and Thompson sampling algorithms

6.2 Experimental Results

6.2.1 The Application

The experiments were integrated to and executed at Android version of existing commercial application UrbanCyclers[1]. It targets cyclists and has focus on cycling as a sustainable transport option in urban areas. This cycling focus places the core user activity in the real world and makes the mobile device with the application an assistant and a motivator enabled by sensor data read about the real-world situation.

The main features are centered about navigation, gamification elements such as ride leaderboards and badges, and social elements especially based around reporting of places that are difficult or dangerous to ride. A map from the application with some reports is shown in Figure 6.6.

[1]`https://play.google.com/store/apps/details?id=com.umotional.bikeapp`

The monetization model includes a subscription for unlocking premium features, but base features are available for free as all active users provide anonymous data that can be used for improvements of the cycling infrastructure.

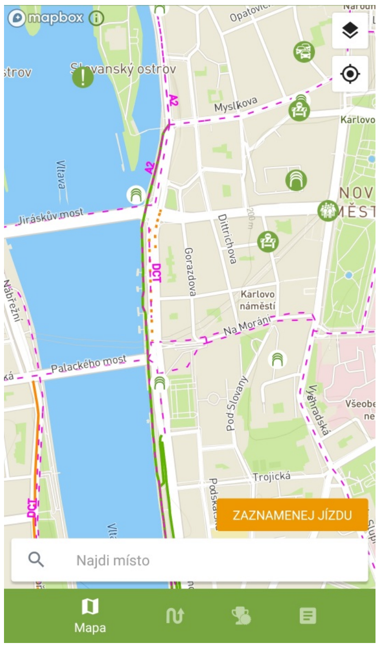


Figure 6.6: Map with reports in UrbanCyclers application used for the experiments

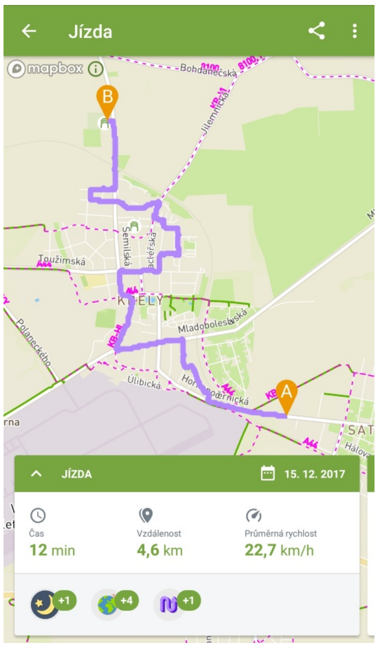


Figure 6.7: Tracked ride in UrbanCyclers application used for the experiments

6.2.2 The Milestone

The most important milestone for the UrbanCyclers app user is the first ride with the application. The experiment targeted users that installed the application and signed up for an account but did not use it for a ride up until the next day after their sign up. Example of successfully tracked ride inside the application is shown in Figure 6.7.

6.3 ϵ-greedy Sampling Algorithm Experiment

6.3.1 Experiment Design

The experiment targeted users with no tracked ride in the first day after sign up. The deployed algorithm was ϵ-greedy with context buckets. The context mapping selected between two buckets by the cycling frequency reported by the user during the sign up. The self-reported high frequency group received a message about gamification features selected between *leaderboards* and virtual *badges* received for reaching certain bike riding goals. The rest of users received a message selected between *weight* loss and general *health* improvements.

The selected reward was a unit reward for each opened notification by a user. The time windows for the user to open the message was 24 hours. This was assumed to be a proxy of a ride in next week after the message.

The used $\epsilon = 0.01$ meaning exploration rate of of 1%.

6.3.2 Analysis of Results

The experiment ran for 16 days in spring 2018 and the messages were sent to 743 users from which 202 were self-reportedly high frequency riders with 541 remaining in low or average categories. This led to rewards being observable approximately after 45 time steps.

Results of each type of message are shown in Table 6.1 and Table 6.2. The number of messages that were opened by the users in Table 6.2 shows some preference for the *weight* and *leaderboards* messages, however with p-value from Fisher's exact test [26] at 0.58 for *weight* vs. *health* and 1.0 for *leaderboards* vs. *badges* respectively, the null hypothesis of the type of message not affecting the outcomes cannot be rejected with the current number of samples. The difficulty of learning of superior reward is also supported by the balance of the number of selections for the *weight* and *health* messages that is caused by changes between the highest expected message value estimates. On the other hand, high imbalance between selections of *leaderboards* and *badges* messages makes the rejection of null hypothesis hard by itself.

	Health	Weight	Leaderboards	Badges
Opened	47	56	27	1
Not opened	214	224	164	10
Success ratio	0.18	0.20	0.14	0.09

Table 6.1: ϵ-greedy Sampling - Users opening messages by message type

Results of each type of message in numbers of user that have at least one ride in seven days after delivery is in Table 6.2. At p-value of 0.58 for *weight* vs. *health* and 0.51 for *leaderboards* vs. *badges* the hypothesis no effect of type of message cannot be rejected.

The phi coefficient of the same user opening the messages and having any rides computed from Table 6.3 is 0.22 which shows low relation. However, the message being displayed as a notification can be read and affect the user without being opened, similarly to relations between clicks and impressions for online advertising. With same preferred message for both rides and clicks the proxy reward might still work through impressions.

	Health	Weight	Leaderboards	Badges
Some rides	36	50	57	2
No rides	225	230	134	9
Success rate	0.14	0.18	0.30	0.18

Table 6.2: ϵ-greedy Sampling - Users with any rides by message type

	Opened	Not opened	Total
Some rides	50	95	145
No rides	81	517	598
Total	131	612	743

Table 6.3: ϵ-greedy Sampling - Users with any rides vs. opening any message

6.4 Thompson Sampling Algorithm Experiment

6.4.1 Experiment Design

The experiment with Thompson sampling algorithm with context buckets also targeted users with no tracked ride in the first day after sign up. The same context mapping as in previous experiment selected between two buckets by the cycling frequency reported by the user during their sign up. The messages were again targeting general *health* benefits, *weight* loss, availability of competition in *leaderboards* or personal targets in *badges.*

The selected reward was also a unit reward for the opening of the notification by the user and the time windows for user to open the message was 24 hours.

6.4.2 Analysis of Results

The experiment ran for 15 days in spring 2018 and the messages were sent to 989 users from which 290 were self-reportedly high frequency riders with 699 remaining in low or average categories. This led to rewards being observable approximately after 65 time steps.

Results of each type of message are show in Table 6.4 and Table 6.5. The numbers of message that were opened by the users presented in Table 6.4 show promising preference for *weight* and *badges* messages. At respective p-values from Fisher's exact test of 0.22 and 0.10, this experiment has the better chance at rejecting the null hypothesis of no difference between the messages, but both p-values are still far above the 0.05 baseline.

	Health	Weight	Leaderboards	Badges
Opened	38	91	7	35
Not opened	201	369	73	175
Success rate	0.16	0.20	0.09	0.17

Table 6.4: Thompson Sampling - Users opening messages by message type

Results of each type of message in numbers of user that have at least one ride in seven days after delivery is in Table 6.5. The p-value of 0.37 for *weight* vs. *health* means the hypothesis no effect of those types of message cannot be rejected. With p-value of 0.012 for *leaderboards* vs. *badges* the null hypothesis could be rejected by the test, however the high difference of the success rates between the experiments and lower number of samples in this context bucket might hint at broken identical distribution assumption.

The phi coefficient of the same user opening the messages and having any rides computed from Table 6.6 is 0.18 which again shows low relation.

	Health	Weight	Leaderboards	Badges
Some rides	102	180	46	85
No rides	137	280	34	125
Success rate	0.43	0.39	0.58	0.40

Table 6.5: Thompson Sampling - Users with any rides by message type

	Opened	Not opened	Total
Some rides	104	309	413
No rides	67	509	576
Total	171	818	989

Table 6.6: Thompson Sampling - Users with any rides vs. opening any message

6.5 Summary

Simulations have shown that ϵ-greedy sampling and Thompson sampling algorithms are able to identify and exploit the message with higher value even under delay from batched observability of message results.

Experiments deployed to a mobile application have shown that the different messages aimed at directing the user towards the milestone of the first bike ride have different rate of being opened and also the rate of the first successful ride after different messages being received is different. However, the difference have not proven to be statistically significant. Furthermore, the opening of the message was shown not to be correlated with the first ride of the same user within the 7 days of delivery. To show more conclusive results, the experiments could be run for longer time to gather more data, set up with the first ride as direct reward, or with different messages.

Chapter 7

Conclusion

The primary aim of the thesis was to explore how data-driven techniques based on machine data analysis can be used to automate the optimization of user engagement. In order to reach this goal following steps were performed.

First, the milestone reaching problem was formalized based on survey of actions proposed by industry experts as effective ways of increasing user engagement. The user is encouraged to reach a milestone by the most effective message.

To solve the formalized problem, two algorithms were proposed - ϵ-greedy sampling with context buckets and Thompson sampling with context buckets. Their relations to the algorithms for reinforcement learning multi-armed bandit problem was described.

The implementation of proposed algorithms as a message optimization module that allows their deployment due to included persistence was described. The required features of the application itself and its back end server were also described.

The simulations demonstrated that proposed algorithms work in environment with messages with distinguishable success rate.

The experiments run with over 1700 users of real mobile application tested the hypotheses of different messages leading to different engagement rate as measured by opened messages and physical world actions. The hypotheses were not proven to be statistically significant at the sample size and the link between the opening of the message and the physical world action of bike ride was dismissed.

Future work includes a larger experiments to accept or reject the difference of efficiency of the messages and design of algorithms with automatic handling of context vector either by segmentation or by creation of a metric for the context space. The investigation of algorithms based on classification or collaborative filtering for solving the milestone reaching problem is also possible.

Bibliography

[1] Connection (Java Platform SE 7), . Available at: https://docs.oracle.com/javase/7/docs/api/java/sql/Connection.html.

[2] Data Report: Crash & Churn Edition, November 2016. Available at: https://www.apteligent.com/2016/11/data-report-crash-churn-edition/.

[3] Development tools - Android library statistics - AppBrain, . Available at: https://www.appbrain.com/stats/libraries/dev.

[4] Number of Google Play Store apps 2018 | Statistic, . Available at: https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/.

[5] Optimize for Doze and App Standby, . Available at: https://developer.android.com/training/monitoring-device-state/doze-standby.

[6] Reference - Kotlin Programming Language, . Available at: https://kotlinlang.org/docs/reference/index.html.

[7] Smartphone users worldwide 2014-2020 | Statistic, . Available at: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/.

[8] Spotlight on Consumer App Usage, . Available at: http://files.appannie.com.s3.amazonaws.com/reports/1705_Report_Consumer_App_Usage_EN.pdf.

[9] BALFOUR, B. Solving Mobile Growth & Retention with Andy Carvell, ex Growth at SoundCloud, July 2017. Available at: https://brianbalfour.com/essays/mobile-growth-retention-soundcloud.

[10] BOUVIER, P. et al. Identifying Learner's Engagement in Learning Games: a Qualitative Approach based on Learner's Traces of Interaction. In *5th International Conference on Computer Supported Education (CSEDU 2013)*, s. 339–350, Aachen, Germany, May 2013. Available at: https://hal.archives-ouvertes.fr/hal-00854579.

[11] CARVELL, A. RRF: a framework for building impactful notifications, April 2017. Available at: https://mobilegrowthstack.com/rrf-a-framework-for-building-impactful-notifications-73c7b91c45a7.

[12] CHAPELLE, O. – LI, L. An Empirical Evaluation of Thompson Sampling. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS'11, s. 2249–2257, USA, 2011. Curran Associates Inc. Available at: `http://dl.acm.org/citation.cfm?id=2986459.2986710`. ISBN 978-1-61839-599-3.

[13] CHEN, A. New data on push notifications show up to 40% CTRs, the best perform 4X better than the worst (Guest post), September 2014. Available at: `http://andrewchen.co/new-data-on-push-notification-ctrs-shows-the-best-apps-perform-4x-better-than-the-worst-heres-why-guest-post/`.

[14] CHEN, A. New data shows losing 80% of mobile users is normal, and why the best apps do better, June 2015. Available at: `http://andrewchen.co/new-data-shows-why-losing-80-of-your-mobile-users-is-normal-and-that-the-best-apps-do-much-better/`.

[15] COVINGTON, P. – ADAMS, J. – SARGIN, E. Deep Neural Networks for YouTube Recommendations. s. 191–198. ACM Press, 2016. doi: 10.1145/2959100.2959190. Available at: `http://dl.acm.org/citation.cfm?doid=2959100.2959190`. ISBN 978-1-4503-4035-9.

[16] DETERDING, S. et al. From Game Design Elements to Gamefulness: Defining Gamification. In *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments, MindTrek 2011*, 11, s. 9–15, September 2011. doi: 10.1145/2181037.2181040.

[17] FLEIT, B. What Happens When You Analyze 1.5 Billion Push Notifications?, September 2016. Available at: `https://www.leanplum.com/blog/analyze-1-5-billion-push-notifications/`.

[18] HILL, D. N. et al. An Efficient Bandit Algorithm for Realtime Multivariate Optimization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, s. 1813–1821, New York, NY, USA, 2017. ACM. doi: 10.1145/3097983.3098184. Available at: `http://doi.acm.org/10.1145/3097983.3098184`. ISBN 978-1-4503-4887-4.

[19] JOULANI, P. – GYORGY, A. – SZEPESVARI, C. Online Learning under Delayed Feedback. In *International Conference on Machine Learning*, s. 1453–1461, February 2013. Available at: `http://proceedings.mlr.press/v28/joulani13.html`.

[20] KLEIN, E. How technology is designed to bring out the worst in us, February 2018. Available at: `https://www.vox.com/technology/2018/2/19/17020310/tristan-harris-facebook-twitter-humane-tech-time`.

[21] LI, L. et al. A Contextual-bandit Approach to Personalized News Article Recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, s. 661–670, New York, NY, USA, 2010. ACM. doi: 10.1145/1772690.1772758. Available at: `http://doi.acm.org/10.1145/1772690.1772758`. ISBN 978-1-60558-799-8.

[22] MONEREO, I. Insights for evaluating lifetime value for game developers, March 2018. Available at: `http://services.google.com/fh/files/blogs/insights_for_evaluating_lifetime_value_for_game_developers.pdf`.

[23] RUDDOCK, D. Switching to the iPhone, part three: Everything I hate about the iPhone X, December 2017. Available at: `https://www.androidpolice.com/2017/12/22/switching-iphone-part-three-everything-hate-iphone-x/`.

[24] SMITH, B. – LINDEN, G. Two Decades of Recommender Systems at Amazon.com. *IEEE Internet Computing*. 2017, 21, 3, s. 12–18. ISSN 1089-7801.

[25] SUTTON, R. S. – BARTO, A. G. *Reinforcement Learning: An Introduction.* Cambridge, Massachusetts : The MIT Press, second edition, 2018. ISBN 978-0-262-19398-6.

[26] The SciPy community. scipy.stats.fisher_exact — SciPy v1.1.0 Reference Guide, May 2018. Available at: `https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.fisher_exact.html`.

[27] THOMPSON, W. R. On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of Two Samples. *Biometrika*. 1933, 25, 3/4, s. 285–294. ISSN 0006-3444. doi: 10.2307/2332286. Available at: `http://www.jstor.org/stable/2332286`.

[28] ZHANG, T. A Simple Framework for Building User Engagement Features. Available at: `https://blog.kissmetrics.com/user-engagement-features-framework/`.

Appendix A

CD content

- `source` - contains the Gradle project and source for building the message optimization module and running simulations, see README.txt for the instructions.
- `text` - contains the text of this diploma thesis in pdf format.
- `text source` - contains LaTeX files for the text of this diploma thesis.