



TECHNISCHE  
UNIVERSITÄT  
WIEN

DIPLOMARBEIT

# Simulating a Reinforcement Learning Model for Application in Just-In-Time Adaptive Intervention Recommender Systems

zur Erlangung des akademischen Grades

**Diplom-Ingenieurin**

im Rahmen des Studiums

**Technische Mathematik**

ausgeführt am

Institut für Analysis und Scientific Computing  
der Technischen Universität Wien

unter der Anleitung von

**Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Felix Breitenecker**

und

**Univ.Lektor Dipl.-Ing. Dr.techn. Bernhard Hametner**

durch

**Katharina Brunner, BSc**

Matrikelnummer: 01204726

Markomannenstraße 18/5

1220 Wien

Wien, 13. Dezember 2021

---

Datum

---

Felix Breitenecker

---

Katharina Brunner

## Kurzfassung

Eine einheitliche Gesundheitsversorgung und offene Zugänge zu Gesundheitsdiensten für alle Bevölkerungsschichten sind weltweit noch immer eine Utopie. In einigen Gebieten dieser Welt übersteigt die Reichweite der Mobilfunknetze die der lokalen Gesundheitsinfrastruktur. Um diesem Problem zu begegnen, sind in den letzten Jahren Initiativen zu “Digital Health“ entstanden, die mögliche Lösungen versprechen. Ein wichtiger, aber komplexer Sektor im Bereich von Digital Health ist die Entwicklung von “Recommender Systems“, also Empfehlungssystemen, bei denen es sich um Machine Learning-basierte mehrkomponentige Anwendungen handelt, die künstliche Intelligenz nutzen, um Einzelpersonen personalisierte unterstützende Interventionen bereitzustellen, die auf Just-in-Time-Interventionsbereitstellung und Adaptivität basieren.

Diese Arbeit befasst sich mit einem Machine Learning Algorithmus namens “Thompson Sampling with Restricted Context“ (TSRC) und untersucht, ob er ein Anwärter für die Engine in einem adaptiven Just-in-Time-Empfehlungssystem ist. Zunächst wird ein Überblick über die Struktur von medizinischen Empfehlungssystemen gegeben, der die Beschreibung von Schlüsselementen und eine Erörterung der aktuellen Anwendungen beinhaltet, die mit diesem Konzept arbeiten.

Mathematisch kann das Problem von Empfehlungssystemen als kontextuelles mehrarmiges Banditenproblem interpretiert werden. Die Standardalgorithmen, die dieses Problem lösen, werden vorgestellt und ihre Vor- und Nachteile diskutiert, bevor argumentiert wird, warum der Thompson-Sampling-Ansatz für diese Diplomarbeit gewählt wurde.

Anschließend wird Thompson Sampling als Paradigma des Machine Learning untersucht und der TSRC-Algorithmus als Erweiterung der traditionellen Heuristik vorgestellt, der aufgrund seiner Einschränkung von Kontextvariablen in Situationen nützlich sein kann, in denen Kontextinformation fehlt, zum Beispiel im Falle eines technischen Ausfalls während der Datenaufzeichnung.

Um die Leistung des TSRC-Algorithmus bei der Auswahl unterstützender Interventionen zu analysieren, wird ein Reinforcement Learning Modell entworfen und in MATLAB implementiert. Es umfasst das Modell eines Empfehlungssystems, virtuelle Klient\*innen und die Implementierung des TSRC-Algorithmus.

Anschließend werden Simulationen mit dem Modell des Empfehlungssystems und ver-

schiedenen Modell-Klient\*innen durchgeführt, und die Reaktion des TSRC-Algorithmus auf Sparsität von Kontext und den Fall fehlender Daten untersucht, die sich beide auf eingeschränkte Kontextvariablen beziehen. Alle Simulationsergebnisse deuten stark darauf hin, dass der TSRC-Algorithmus ein Anwärter für adaptive Just-in-Time-Empfehlungssysteme ist, und es wird ein Ausblick auf weiterführende Forschungsbereiche zu diesem Thema gegeben.

# Abstract

Globally speaking, consistent healthcare and easy access to health services for all citizens is still a utopian concept. In some areas of this world the coverage of mobile networks surpasses the local health care infrastructure. In order to combat this issue, digital health initiatives have emerged in recent years, promising possible solutions. An important but complex sector in digital health is the development of treatment recommender systems, which are machine learning driven multi-component applications that utilise artificial intelligence to deliver personalised supportive intervention to a client, based on just-in-time intervention delivery, and adaptiveness.

This thesis looks at a machine learning algorithm called Thompson Sampling with Restricted Context, or TSRC, and investigates whether it is a contender for the engine in a just-in-time adaptive recommender system. First, an overview of the framework for medical recommender systems is given, which includes a description of its key elements and a discussion of the current applications working with this concept.

Mathematically, the problem faced by recommender systems can be interpreted as a contextual multi-armed bandit problem. The standard algorithms solving this problem are presented, and their advantages and disadvantages are discussed before arguing why the Thompson Sampling approach is selected for this thesis.

Subsequently, Thompson Sampling is investigated as a machine learning paradigm, and the TSRC algorithm is presented as an extension of the traditional heuristic, which, due to its restricted context policy may be equipped to handle cases where contextual information is missing, for example in the case of a technical failure to record data.

In order to analyse the TSRC algorithm's performance in choosing supportive interventions, a reinforcement learning-based model is designed and implemented in MATLAB. It includes a model recommender system together with virtual model clients, and the implementation of the TSRC algorithm.

Thereafter, simulations are performed with the model recommender system and different clients, and the TSRC algorithm's response to contextual feature sparsity and cases of missing data, both relating to restricted context, are investigated. All simulation results strongly suggest that the TSRC algorithm is a contender for just-in-time adaptive recommender systems, and an outlook containing future research into the topic is provided.

# Acknowledgement

*< I'm the Greek economy of cashing intellectual cheques and I'm trying to progress. >*

The 1975 | Loving Someone

First and foremost, I want to thank my supervisors Ao.Univ.Prof.i.R. Dipl.-Ing. Dr.techn. Felix Breitenecker, who has paved the way for me to write my thesis as part of an internship, and Univ.Lektor Dipl.-Ing. Dr.techn. Bernhard Hametner, who has been patient and kind in answering all my questions no matter how silly, and who has encouraged me in working autonomously, shaping my ideas, and guiding this thesis.

Furthermore, I am grateful for receiving financial support from the Austrian Research Promotion Agency (FFG) in form of the FEMtech internship at the Austrian Institute of Technology (AIT), which has enabled me to work full-time for six months in a company that is at the forefront of biomedical research, so that I could gather invaluable experience. I cherish the support and encouragement I have received from my co-workers at AIT, fellow interns and researchers alike, who have made me feel very welcome and who have always had an open ear for my thesis-related troubles, and I am glad to have gotten to know them.

At last, I want to thank my family for being unquestioningly supportive and for offering their home and garden as a refuge every so often. But most importantly, I want to thank my partner for being my rock, never once yielding to the waves of perfectionism, impatience, and despair that I let crash down upon him like Neptune his sea. He has been the most patient rubber duck debugging buddy, the most encouraging thesis proofreader, my most consistent supporter and number one fan. This thesis would not have been possible without his help.

# Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Wien, am 13. Dezember 2021

---

Katharina Brunner

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Just-in-Time Adaptive Interventions</b>	<b>4</b>
2.1	Why Just-In-Time Adaptive Interventions? . . . . .	4
2.2	Design and Concept of a Just-In-Time Adaptive Intervention . . . . .	5
2.2.1	Decision Points . . . . .	7
2.2.2	Tailoring Variables . . . . .	8
2.2.3	Intervention Options . . . . .	9
2.2.4	Decision Rules . . . . .	10
2.2.5	Proximal Outcome . . . . .	11
2.2.6	Construction and Evaluation of a Just-in-Time Adaptive Intervention	11
2.3	Applications in Mobile Health . . . . .	15
2.3.1	State-of-the-Art Mobile Health Applications . . . . .	17
2.3.2	Challenges . . . . .	22
<b>3</b>	<b>The Contextual Multi-Armed Bandit Problem</b>	<b>25</b>
3.1	The Multi-Armed Bandit Problem . . . . .	25
3.2	Regarding Context in a Multi-Armed Bandit . . . . .	29
3.3	The Contextual Multi-Armed Bandit Approach for Decision Rules in a Clinical Setting . . . . .	32
3.4	Contextual Multi-Armed Bandit Algorithms . . . . .	34
3.4.1	The e-Greedy Strategy . . . . .	36
3.4.2	The Upper Confidence Bound Strategy . . . . .	41
3.4.3	The Thompson Sampling Strategy . . . . .	48

---

3.5	Applications of Multi-Armed Bandits and Contextual Multi-Armed Bandits in Mobile Health . . . . .	54
<b>4</b>	<b>A Choice of Thompson Sampling Algorithm</b>	<b>58</b>
4.1	Markov Decision Processes . . . . .	58
4.2	Contextual Multi-Armed Bandits versus Markov Decision Processes . . . . .	60
4.3	Thompson Sampling as a Reinforcement Learning Algorithm . . . . .	62
4.4	Contextual Bandits with Restricted Context . . . . .	65
4.4.1	Handling Restricted Context in a Contextual Multi-Armed Bandit . . . . .	65
4.4.2	Combinatorial Bandits . . . . .	67
4.5	Thompson Sampling with Restricted Context . . . . .	70
<b>5</b>	<b>Methods of Implementation</b>	<b>76</b>
5.1	The Model Just-In-Time Adaptive Intervention . . . . .	76
5.2	Modelling the Client . . . . .	80
5.3	The Code Architecture . . . . .	84
5.3.1	The Weather Generator Function <code>weatherGenerator.m</code> . . . . .	85
5.3.2	The Function <code>TSRC.m</code> . . . . .	87
5.3.3	The Monte Carlo Simulation Function <code>monteCarloTSRC.m</code> . . . . .	88
<b>6</b>	<b>Simulation</b>	<b>94</b>
6.1	The Weather Feature . . . . .	94
6.1.1	Comparing Different Weather Scenarios for a Weather-Sensitive Client	95
6.1.2	Comparing a Weather-Sensitive Client to a Weather-Insensitive Client	101
6.2	The Fitness Feature . . . . .	104
6.2.1	Comparing Different Fitness Levels . . . . .	105
6.2.2	Comparing a Fitness-Sensitive Client to a Fitness-Insensitive Client .	109
6.3	The Availability Feature . . . . .	112
6.3.1	Comparing Different Levels of Availability . . . . .	112
6.3.2	Comparing an Availability-Sensitive Client to an Availability-Insensitive Client . . . . .	116



## Contents

---

6.4	The Motivation Feature . . . . .	119
6.4.1	Comparing Clients Motivated for Different Activities . . . . .	119
6.4.2	Comparing Different Models for Motivation . . . . .	123
6.5	Investigating Feature Sparsity . . . . .	131
6.6	The Case of Missing Data . . . . .	140
<b>7</b>	<b>Conclusion and Outlook</b>	<b>145</b>
	<b>List of Figures</b>	<b>149</b>
	<b>List of Tables</b>	<b>152</b>
	<b>List of Algorithms</b>	<b>154</b>
	<b>Index</b>	<b>155</b>
	<b>Bibliography</b>	<b>157</b>

# 1 Introduction

In 2020, the Seventy-third World Health Assembly endorsed the global strategy on digital health 2020 – 2025 proposed by the World Health Organization with the purpose to strengthen health systems in its member states through the application of digital health technologies for consumers, health professionals, health care providers, and industry towards empowering patients and achieving the vision of health for all citizens. Digital health is expected to revolutionise the public health sector on a global scale, providing healthcare solutions even in the least-developed countries through the use of smart and connective devices that process health data according to the principles of advanced computing, big data analytics, artificial intelligence including machine learning, or robotics [88].

An emerging research domain within the digital health framework is the development of personalised recommender systems, with the general aim to support the self-management of chronic illnesses. Two principles of support provision are merged to optimise patient care: just-in-time assistance, and adaptiveness. By employing machine learning techniques, the recommender system identifies the best supportive intervention for the customer out of a pool of possible options and delivers it at a time where the customer is most receptive to it. This digital health practice can be accompanied by traditional forms of treatment, or it may be used as a standalone treatment regime. Recommender systems of this kind already exist, but most of them are still in the development phase. There are many aspects that need improvement before such a system is applicable on a large scale, such as long-term effect research, or health data encryption.

The research on this topic encompasses many scientific fields, including applied mathematics. In order to save time and resources, modelling and simulation provide the opportunity to test just-in-time adaptive recommender systems *in silico* to investigate whether a clinical trial is expedient in terms of efficiency and effectiveness of treatment. The effort going into such simulations can be extensive.

Before the simulation environment can be implemented, an appropriate machine learning algorithm must be settled on. Due to machine learning paradigms providing a variety of possible learning structures all equally likely to be eligible for the job, the process of finding the right algorithm can take time. Generally, machine learning algorithms are quite young, with a few exceptions. For example, Thompson Sampling is a heuristic designed in 1933 by

William R. Thompson [80]. It is widely applicable throughout machine learning problems in its basic form, and recently developed extensions of the original algorithm are able to solve advanced, more specific problems.

Mathematically, the problem posed by a just-in-time adaptive recommender system is typically modelled as a multi-armed bandit problem that additionally regards context. This problem originates from game theory and describes a player faced with several slot machines, colloquially called one-armed bandits. The player's task is to maximise their winnings in the long run, forcing them to try out all the machines, while at the same time playing the ones that yield the most profit in order to learn the expected reward for each slot machine. This is called the exploration-exploitation trade-off, and it is of utmost importance to choose an algorithm that deals with this trade-off in an efficient way.

If the multi-armed bandit problem also regards context, it means that additional information is available to the player before deciding on a slot machine. "Context" in a just-in-time adaptive recommender system is information about the circumstances surrounding a client. Instead of only accessing the knowledge of whether or not an intervention suggestion was accepted, the algorithm solving the problem may also consider supplementary information about the client, which can be medical (i.e., the heart rate), or non-medical (i.e., the client's location). From this perspective, it is clear that the procurement of contextual information as well as the processing of this data by the machine learning algorithm is a central topic in recommender system development.

In order to simulate a response to the intervention suggestions from which the algorithm can learn sequentially, clients must be modelled once an algorithm is selected. This requires in-depth knowledge about the dynamics of the illness that the recommender system is targeting, as well as expertise for interdisciplinary approaches such as behaviour models. Human behaviour depends on a multitude of factors, and it lies in the hands of the developers to decipher which ones are most likely to influence the client's progress concerning their health predicament. The more realistic a client's behaviour is portrayed, the higher the informative value of the overall simulation will be.

This thesis roughly walks through the steps of just-in-time adaptive recommender system development. Its main purpose is to investigate whether a specification of the Thompson Sampling heuristic, the TSRC algorithm, is eligible to deliver supportive interventions within a just-in-time adaptive recommender system.

For this purpose, a general introduction is given on the theoretical background of the just-in-time adaptive intervention design in Chapter 2. It elaborates on the advantages of recommender systems based on these principles, describes and defines key elements of the intervention design, gives an overview of state-of-the-art recommender systems in digital health, and sheds a light on the current challenges of developing functional applications.

Chapter 3 is dedicated to the mathematical formulation of the multi-armed bandit problem and its extension that additionally regards contextual information, the contextual multi-armed bandit problem. Several strategies for solving the bandit problem including Thompson Sampling are presented, and their advantages and disadvantages are discussed. Also, a summary of current applications of multi-armed bandit algorithms in the field of digital health is provided in order to understand the present state of research on this topic.

The Thompson Sampling algorithm is introduced within the framework of machine learning in Chapter 4, and an extension limited to regarding only part of the contextual information available, the TSRC algorithm, is explained in detail. The possible superiority of the extended algorithm over the classical version is analysed, which centres around the complications related to missing contextual data (e.g., due to technical failure).

In order to investigate whether the chosen algorithm is feasible for delivering interventions in a just-in-time adaptive recommender system, several simulations are implemented and run in MATLAB with the help of a model recommender system, and model clients. The implementation of these elements, as well as the TSRC algorithm, is found in Chapter 5, where the intricacies of the model components are described in detail.

Subsequently, Chapter 6 contains the simulation setups, outcomes, and discussions for different simulation scenarios in the hope that they provide viable results towards the research question, including investigations on feature sparsity, and the algorithm's reaction to situations of missing data.

Chapter 7 contains a brief conclusion on the conducted simulations and underlines why the chosen algorithm ought to be considered for use in a recommender system. Furthermore, it provides an outlook on future work and related research.

## 2 Just-in-Time Adaptive Interventions

Just-in-time adaptive interventions (JITAI) combine the principle of just-in-time (JIT) support, which attempts to provide the right type of support at the right time [40], with adaptive intervention, which is a dynamic form of individualization. The result is an intervention design that aims to provide the right type and amount of support at the right time by adapting to an individual's changing state with regard to the individual's current situation and environment [58]. A common approach is to use mobile devices to enable clients to access health interventions whenever and wherever they feel they need help [46]. This chapter gives a description of the JITAI concept and its components, and discusses the application of such intervention designs in health services.

### 2.1 Why Just-In-Time Adaptive Interventions?

A just-in-time adaptive intervention is an intervention design that adapts the provision of support (e.g., the type, timing, or intensity) over time to the changing conditions and circumstances in which individuals may find themselves, henceforth known as *context*, with the goal to deliver support contextually, and in the moment that a person needs it most and is most likely to be receptive to receiving it [75]. The JITAI describes a conceptual foundation which aims to guide researchers in putting its components into practice, in the form of a functional application [31]. In order to provide JIT support flexibly in terms of time and location, the dynamics of an individual's context need to be monitored in real time [47], which is possible due to rapid advances in mobile and sensing technologies [89].

The JITAI design is based on the idea that timing plays an important role in determining whether providing support is beneficial to an individual [58]. Timing is defined as the moment (a static reference point in time) at which a phenomenon or process starts or finishes, or a moment at which data is recorded. The concept of timing in a JITAI is generally event-based, so the question of *when* the right time is, is conditional rather than defined by a time of day. These changes in conditions are expected to occur irregularly [58], and cannot be predicted [6]. Consequently, timing onset and offset separates states that reflect different conditions in which individuals may find themselves [64].

The JITAI concept also implies providing no support when the time is wrong and never

providing the wrong type of support [62]. JITAIs therefore aim to eliminate any action that absorbs resources but adds no value to the desired process, or even disrupts it [81]. Regarding this concept, a JITAI is constructed to identify vulnerable or opportune states in a client. A *state of vulnerability* describes a period of sensitivity to adverse events, and a *state of opportunity* depicts a period of susceptibility to positive behaviour change. The emergence of those states is a dynamic process, influenced by stable and transient, internal and external factors (with regard to the individual). The aim of the JITAI is thus to contain the vulnerable state in an individual and return their condition from vulnerable to latent. JITAIs are also motivated by the importance of capitalising on states of opportunity when an individual is susceptible to positive behaviour change. Also, because JITAIs rely on the capabilities of modern technology and wireless devices for monitoring clients, intervention decisions are made much more rapidly than in standard adaptive interventions [58].

Note that the definition of a JITAI emphasises that the adaption in such an intervention concept is employed by the intervention itself, rather than by the client, so decisions concerning when and how to provide support are based on whether and how the client has interacted with past intervention suggestions (i.e., the intervention protocol) [58]. This approach is grounded on evidence suggesting that clients are often unable to recognise when states of vulnerability and/or opportunity emerge [30]. Therefore, the JITAI approach distinguishes itself from participant-determined approaches that offer an array of available resources, for the client to decide when and what kind of support to initiate [58].

Due to these characteristics, JITAIs find use mainly in health services and the medical domain, see Section 2.3. In these areas, it is especially important that the support provided copes with the rapidly changing states of a client, which is usually impossible to do with an in-person or face-to-face approach, due to lack of time and resources. The development of JITAI applications requires inter- and multidisciplinary efforts: clinicians, behavioural scientists, engineers, statisticians, computer scientists, and human-computer interaction specialists need to be involved in order to design feasible intervention concepts in this field [58].

## 2.2 Design and Concept of a Just-In-Time Adaptive Intervention

The content of this section and its subsections is cited from Nahum-Shani et al. [58], unless otherwise stated.

JITAIIs are multi-component interventions, so it is important to clearly define the components that comprise them, in order to attend to the utility of each component accordingly. Investigating the effectiveness of each component and how well these work together is also critical in the process of optimising a multi-component intervention, and will be discussed

in Section 2.2.6 [22].

A pragmatic framework is provided by Nahum-Shani et al. in [57], which can be used to organise existing and new evidence into a useful model for JITAI construction. The framework aims to help developers think through key factors in order to inform the design of JITAIs and guide further empirical work. A more explicit guide is the template-based intervention design mechanism proposed by Gonul et al. [31], which enables the configuration of evidence-based JITAI components. It incorporates a rule definition language, enabling developers to specify conditions for interventions based on current and former contextual data. This approach allows experts from other domains to create JTIAIs within an expandable software framework, aided by the terminology offered by Nahum-Shani et al. A JITAI consists of 5 key elements:

- *Decision points*: Points in time at which an intervention decision is made, either pre-defined or event-based.
- *Tailoring variables*: Information concerning the individual at a decision point (either static or dynamic), which is used for individualisation (i.e., contextual data) [57].
- *Intervention options*: Array of possible actions which can be employed at a decision point.
- *Decision rules*: Mechanism which inputs information in the form of tailoring variables and outputs a decision [50].
- *Proximal outcome*: Short-term goals the intervention is intended to achieve.

The above variables are interconnected to form an intervention concept, see Figure 2.1.

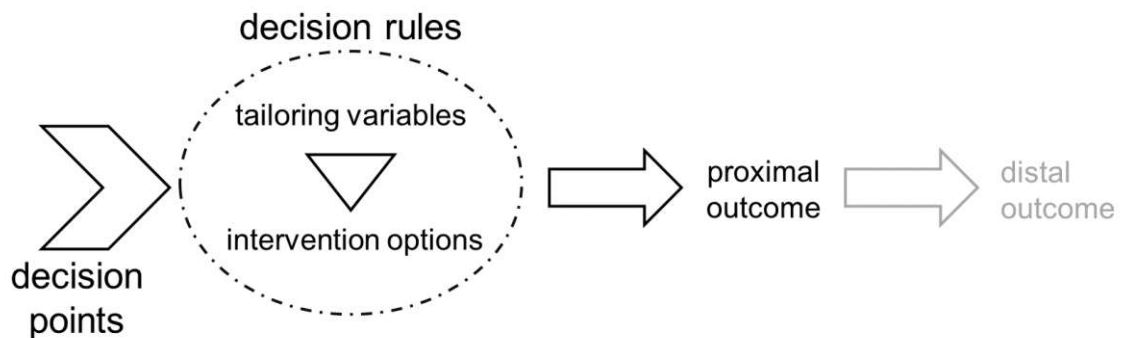


Figure 2.1: Intervention concept for the JITAI design, after [58].

The long-term goals the intervention is trying to achieve are called *distal outcomes*. Caution is needed when both proximal and distal outcome are included in a model. Proximal

outcomes are often mediators (i.e., critical factors in a causal pathway through which the intervention options are designed to impact a distal outcome) [49], but they can also be intermediate measures of the distal outcome. For example, if the goal of the JITAI is to motivate a person to adopt and maintain a more active lifestyle, a proximal outcome can be a certain step count reached per day (i.e., 10,000 steps, as recommended by the U.S. Department of Health and Human Services, 2008), whereas the distal outcome may be quantified by reaching the number of steps each day in a month or year. Thus, proximal and distal outcome refer to the same goal, only on different timescales. In these cases, where the proximal outcome directly corresponds with the distal outcome, the distal outcome can be neglected as a goal since its inclusion merely adds a level of complexity.

Gonul et al. [31] argue that proximal and distal outcome are only additional elements and do not belong to the classical definition of a JITAI, and Tewari and Murphy [79] include the proximal outcome, but not the distal outcome, which is most consistent with other literature on the subject. In this thesis, the distal outcome will be included, but cases will be disregarded where, in practical application, reaching the proximal outcome can lead to a diminishing effect on the distal outcome, for example due to intervention fatigue [67], meaning the client's willingness to follow the suggested interventions is waning, which can be caused by a multitude of factors and is discussed in Sections 2.2.3 and 2.3.2.

The following sections explain the importance and role of the above design components in more detail.

### 2.2.1 Decision Points

A decision point is a point in time at which an intervention decision is made (i.e., when the decision rules decide on an intervention option). A decision point can occur:

- At a pre-specified time interval (e.g., every three minutes) [33]
- At specific times of day (e.g., daily at 9 a.m.) [44]
- Following random prompts [12]

When selecting decision points, the main consideration should be given to how frequently meaningful changes in the tailoring variables are expected to occur. A change is deemed meaningful if it carries implications for choosing an intervention option. Such changes represent entries to, and exits from, vulnerable or opportune states. For example, if the tailoring variable represents the distance  $d$  to a pre-defined risk location for the client, a meaningful change in the tailoring variable occurs if the client's distance  $d$  drops below a certain threshold (i.e., a radius  $r$  around the risk location), or increases above it (either  $d \leq r$  or  $d > r$ ). The first case marks an entry into a vulnerable state, which requires an



intervention. This concept is realised in the mobile phone application A-CHESS, which provides treatment for patients who leave alcohol-use disorder therapy, see Section 2.3.1 [33]. If a change in location is expected to occur every minute, then the frequency of decision points might be one-minute intervals.

The choice of time interval between decision points has an enormous impact on the ability of the JITAI to achieve its goals. States of opportunity might be missed if the timing of the decision points is not synchronised to the frequency of changing conditions, and missed opportunities can result in reduced intervention engagement, or intervention fatigue, see Sections 2.2.3 and 2.3.2.

### 2.2.2 Tailoring Variables

A tailoring variable contains information concerning the individual that is used to decide under what conditions to provide an intervention, and what intervention to provide. In the example in Section 2.2.1, the tailoring variable is the individual's distance  $d$  to a risk location, or their current position (e.g., determined via GPS) [33]. JITAIs can be designed to target more than one proximal outcome, and different tailoring variables may be considered for different proximal outcomes.

JITAIIs rely on tracking devices such as mobile phones, or wireless sensors, to collect patient data, which, due to rapid leaps in technological advancement, are becoming cheaper and more accurate. The values of tailoring variables are obtained by either *active assessment*, or *passive assessment* [86]. Active assessment is also called *ecological momentary assessment* (EMA) and requires the client to self-report, which indicates direct engagement with the JITAI [74]. Passive assessment means monitoring the client passively, with minimal to no engagement on the client's side. For this reason, passive assessment is considered to be less biased than EMA, and developers are encouraged to choose passive assessment over active assessment whenever possible. However, when sensors are used to assess an individual's condition, the measuring needs to be reliable, otherwise the decision rule will perform little better than a random selection of intervention options, and if at least one tailoring variable is invalid, the decision rule may even recommend an adverse option, potentially causing the client harm. Due to the reliance on technology, missing data can occur for various technical reasons (e.g., no power, no GPS signal). It is vital for a functional JITAI to anticipate those situations, which the decision rules need to be equipped to handle.

The selection of tailoring variables should be based on evidence, indicating that a particular variable is useful in aiding intervention decisions, which means that the variable needs to be able to specify conditions marking opportune or vulnerable states, in which

individuals can benefit from one intervention option over another. The proximal outcome of a JITAI should also influence the selection process. In fact, it is often reasonable to use the proximal outcome as a tailoring variable (i.e., to consider earlier success or failure of proximal outcomes to decide on an intervention option). The class of algorithms serving as decision rules, presented below, works with this feedback explicitly, see Chapters 3 and 4.

### 2.2.3 Intervention Options

Intervention options are an array of actions that may be employed at any given decision point. They can include various types, sources, and amounts of support, or media delivering the support, depending on the respective application. The choice of intervention options included in a JITAI should be theoretically as well as empirically driven, and, due to the construction of the class of algorithms working as decision rules, mainly target the proximal outcomes, see Chapters 3 and 4.

As a secondary, but equally important target, intervention options should be designed to manage *intervention engagement*, and cope with *intervention fatigue*. Intervention engagement describes a state of motivational commitment in the client role (i.e., how strongly a client is adhering to the provided support) [45], and intervention fatigue defines the state of emotional weariness associated with intervention engagement (i.e., the client slowly disengaging with the JITAI) [39]. Research in occupational health psychology [10] provides a framework for separating design considerations that primarily concern engagement from those that primarily concern fatigue: results suggest that engagement can be prompted by efforts to fulfil basic psychological needs (e.g., autonomy or competence), while intervention fatigue can be prevented by attending to the demands imposed on the client in terms of time and effort.

Varying the form, presentation, and timing of content delivery is a viable strategy for dealing with both intervention engagement and fatigue [54]. In addition, the inclusion of a *provide nothing* intervention option is recommended, making it possible to not provide any intervention at a decision point. This option should be incorporated to address situations where providing support may lead to adverse effects on the outcome, intervention engagement or fatigue, including situations where the client is unreceptive (e.g., at work, driving, asleep), or when support is not required. Following the example from Section 2.2.1, an array of intervention options when checking whether a client has entered a risk location (i.e., check if  $d \leq r$ ) might be:

$$\text{intervention options} = \left( \begin{array}{c} \text{send message to a trusted third party} \\ \text{provide alert asking whether the client wants to be there [33]} \\ \text{remind the client of their streak for not entering a risk location} \\ \text{provide nothing} \end{array} \right)$$

### 2.2.4 Decision Rules

Decision rules are the crucial component of the intervention design. They map the tailoring variables (i.e., context) to an intervention option (i.e., *action*) [79], while dynamically learning whether and how well the suggested intervention has been accepted. The most prominent approach is to use the contextual multi-armed bandit problem to model this process, which is discussed at length in Chapter 3.

Theoretically, there is a decision rule for each decision point. Following the example from Section 2.2.1, the decision rule at time  $t$  can be:

```

IF  $d \leq r$ 
    intervention option = provide alert
ELSE IF  $d > r$ 
    intervention option = provide nothing
END
    
```

In practice (and especially in this thesis), the mechanism underlying the decision rules is sufficiently adaptable to work at all decision points, and no specification is needed. The decision rule at point  $t$  considers the values of the tailoring variables during the decision process, in order to determine which intervention option to offer. In the example above, the value for  $r$  specifies the condition (or threshold) under which an intervention option should be offered.

Good decision rules are based on an accurate and comprehensive scientific model that highlights experiences and context in which a client is likely to benefit from an intervention, and whether to favour one intervention option compared to the others, in aid of the proximal outcome. Similarly, it is vital to understand what constitutes vulnerable and/or opportune states, the process by which these states emerge, and the possible interventions that can be employed to address and capitalise on them. Additionally, the understanding of how and why intervention engagement and fatigue fluctuate over time, how these fluctuations impact the proximal or distal outcome, and which strategies can enhance engagement and reduce fatigue, must have a prevalent influence on the decision rule design process.

### 2.2.5 Proximal Outcome

Proximal outcomes are the imminent goals the intervention aims to achieve. They are measured after the intervention is provided in order to check whether the intervention is on track in reaching its goal. As mentioned above, identifying and clearly defining the proximal outcomes helps developers select appropriate decision points, tailoring variables, and intervention options, as well as formulating effective decision rules [49].

The relationship between proximal and distal outcomes is already discussed at the beginning of Section 2.2. If distal outcome is taken into account there are multiple pathways through which the intervention can impact the distal outcome [63], and intervention developers might select multiple proximal outcomes to be targeted by the JITAI. The example from Section 2.2.1 can be used to illustrate a possible scenario: if the distal outcome is to prevent alcohol consumption by the client, whose location is monitored regularly and who has also just emerged from alcohol-use disorder therapy, proximal outcomes aiding the distal outcome might be:

- Report regular check-ins with group therapy sessions
- Report regular check-ins with a personal coach
- Not entering any (or some) of the pre-defined risk locations each day (or week)

In this case, a risk location might constitute a place where alcohol consumption has regularly occurred in the past, or is likely to occur.

To prevent poor adherence of JITAIs, developers are encouraged to additionally consider proximal outcomes pertaining to intervention engagement and fatigue [41]. These can be behavioural (i.e., accessing and using the intervention), cognitive (i.e., perceiving the intervention as useful) or affective (i.e., trusting the intervention). When addressing intervention engagement, it is necessary to specify the time horizon of the engagement required in order to achieve the outcome. Similarly, there are a number of proximal outcomes reflecting intervention fatigue.

However, empirical evidence in the field of occupational health psychology suggests that engagement and fatigue are two distinct, yet related concepts that share certain antecedents and consequences [23], and JITAI developers are recommended to attend to both concepts when selecting proximal outcomes, as well as intervention options.

### 2.2.6 Construction and Evaluation of a Just-in-Time Adaptive Intervention

The content of this subsection is cited from Nahum-Shani et al. [57], unless otherwise stated. The framework by Nahum-Shani et al. suggests breaking down the construction of a JITAI into three areas:

- (1) Defining the problem
- (2) Defining the JIT in this context
- (3) Formulating the adaption strategy

### **Defining the Problem**

The problem definition includes specifying whom the intervention is aimed at, in terms of identifying the target population and its key attributes (e.g., employed individuals are generally unavailable for time-intensive interventions during work hours). Furthermore, if included, the distal outcome of the JITAI should be decided upon, and the *temporal progression* of key factors towards the outcome should be determined.

The term “temporal progression” refers to the way in which the intervention process unfolds over time and what role each factor and effect plays. Depending on the problem, this progression is generally not linear or straightforward. JITAI developers are advised to identify different timescales, within which the process leading to either proximal or distal outcome might unfold, and, subsequently, specify dynamics within each timescale, and how factors and effects at different timescales are related.

Another aspect of defining the problem is selecting contenders for proximal outcomes. With or without including a distal outcome in the JITAI concept, proximal outcomes must be chosen depending on the goal, because the health-related target of a JITAI is observed through different factors compared to targets regarding intervention engagement or fatigue.

### **Defining the JIT in This Context**

Defining the JIT aspect of the intervention design means establishing which factors mark states of vulnerability and opportunity, what possible intervention options can affect the proximal outcomes (and can be delivered JIT), and which factors mark unreceptive states to these interventions. For example, if a vulnerable or opportune state occurs at work, the client is likely to be unreceptive, and, even if an intervention is offered JIT, there will be no progress towards the proximal outcome.

### **Formulating the Adaption Strategy**

When formulating the adaption strategy, a set of tailoring variables must be chosen based on the decision of what information about the client is useful in selecting an intervention. The tailoring variables should include factors that mark states of opportunity or vulnerability. Furthermore, it should be clear which intervention option is likely to have the desired effect on the proximal outcome for each level of the tailoring variable. For example, if activity

suggestions are provided by the JITAI, and the weather is a tailoring variable, the client is unlikely to accept a suggestion for an outdoor activity in case the weather is bad, making it necessary for developers to draw a link between outdoor suggestions and good weather.

Finally, developers are required to create appropriate decision rules that link all the information above in a systematic manner in order to operationalise effective adaptation. Figure 2.2 gives a graphical representation of the pragmatic framework by Nahum-Shani et al.

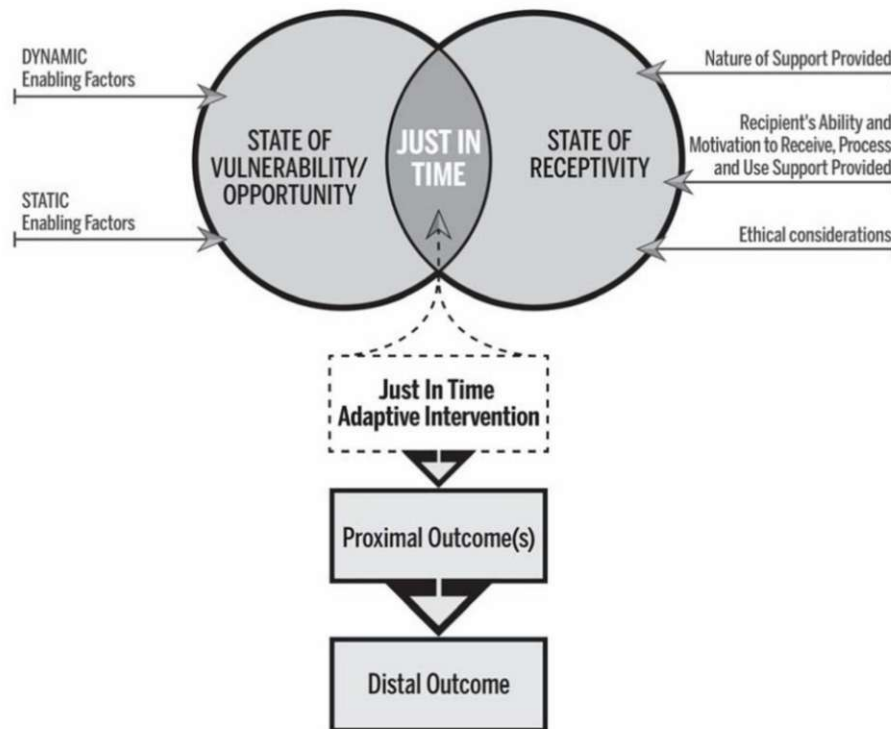


Figure 2.2: Summary of the pragmatic framework for developing JITAIs, from [57].

In order to optimise JITAIs, study designs and data analytic methods are needed to understand causal effects of intervention options [58]. For example, Klasnja et al. [46] present an experimental design, the *micro-randomized trial* (MRT), developed to support the optimisation of JITAIs by enabling modelling of causal effects, and time-varying effect moderation, for intervention components within a JITAI. Its invention is motivated by the claim that researchers currently do not have the appropriate tools to gather evidence for deciding how a JITAI should be adjusted to make it more effective. MRTs provide data on how the effects of different intervention options change over the course of the intervention, and how time-varying contextual and psychological factors moderate the observed changes in intervention-component efficacy. Findings from MRTs can thus help determine decision

rules for when and in what circumstances an intervention option should be delivered.

Micro-randomisation means randomly assigning an intervention option at each decision point. For a multi-component intervention, multiple components can be randomised concurrently, and a study lasting weeks or months may randomise each person hundreds or thousands of times depending on the frequency at which intervention components are delivered. MRTs are well-suited for optimising JITAIs because the repeated randomisation allows developers to assess how causal effects of different intervention components change over the course of the study. Additionally, effect estimations can take advantage of the inter-subject contrasts (i.e., individuals assigned to an intervention option compared to others receiving different interventions), as well as intra-subject comparisons (i.e., an individual assigned to an intervention option compared to the same individual receiving different interventions). These intra-subject comparisons enable MRTs to require far fewer participants than traditional study designs.

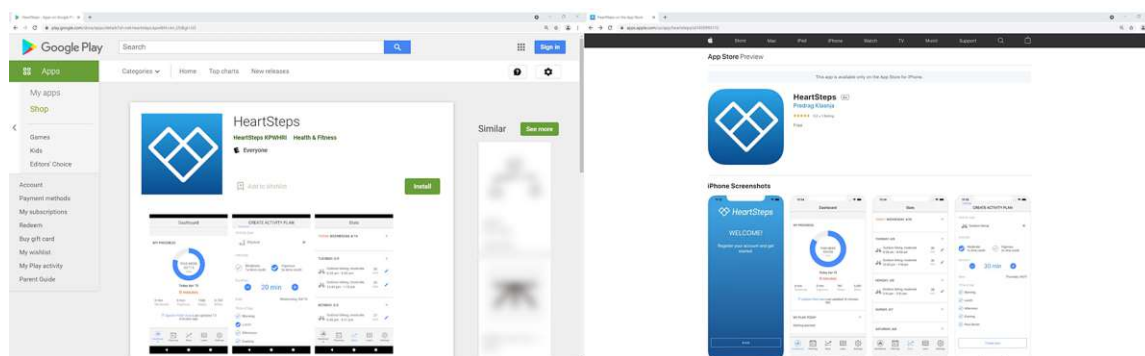


Figure 2.3: Screenshot of the HeartSteps app in Google Play Store and App Store, accessed on 2021 – 08 – 24.

In [46], Klasnja et al. give an example of a six-week MRT for an existing JITAI called HeartSteps, which has been developed into a mobile phone app available for download in the United States [38] and is discussed in more detail in Section 3.5. Originally tested during a trial for improving the physical activity of individuals with blood pressure in the stage 1 hypertension range (120–130 systolic), it delivers activity suggestions to encourage walking while monitoring the client’s daily step count with a Fitbit tracker [52]. There are two types of intervention options available: suggestions to go for a walk and suggestions to stop being sedentary. To optimise the delivery of activity suggestions, an MRT is conducted to evaluate the effects of these two intervention components, giving both types a probability of 50% to be delivered. The proximal effects can then be estimated using standard regression models.

Potential research questions for the MRT about the main effects of suggestions are:

- By how much, on average, does providing an activity suggestion increase the step count over the next 60 minutes relative to no activity suggestion?
- By how much, on average, does providing an activity suggestion change the step count for the remaining part of the same day, or the next day?
- How does the number of availability-appropriate activity suggestions delivered in a day impact the daily self-report of user burden?

MRTs still have several limitations. They are only applicable for the testing of push interventions (i.e., interventions such as reminders or prompts to interact with the JTIAI, that are delivered to clients based on a set of decision rules). Also, MRTs are most appropriate for the testing of intervention components for which proximal outcomes can be defined in a principled way (i.e., components for which theory specifies the outcome directly impacted by the components). Furthermore, MRTs are not suited to testing interventions for very rare events, due to the lack of available data, which diminishes the reliability of the MRT.

The next section introduces the field of mobile health, the methods already in use, as well as challenges for future development.

## 2.3 Applications in Mobile Health

Section 2.1 argues that, due to the nature of their construction, JITAIs find use mainly in health services. Thanks to rapid advancements in technology, and especially since mobile devices are widespread amongst the population, the aim to develop digital support for the self-management of illnesses has gained popularity, and studies have shown the benefits of such applications in the medical domain [36].

The public health practice supported by mobile devices such as mobile phones, patient monitoring devices, and other wireless devices, is defined as *mobile health* (mHealth) by the Global Observatory for eHealth, a service of the World Health Organization (WHO), who first conducted a survey on mHealth to determine its status in the WHO member states in 2011 [89]. In this survey, mHealth is described as involving the use and capitalisation on a mobile phone's core utility of voice and short messaging service, as well as more complex functionalities and applications including general packet radio service, mobile telecommunications (such as 3G and 4G systems), global positioning systems (GPS), and Bluetooth technology.



The report claims that, even in 2011, the penetration of mobile phone networks in many low- and middle-income countries surpasses other infrastructures like paved roads, electricity, and fixed internet deployment, suggesting that, by transforming the way health services are managed and delivered, more people will have access to illness-management, profiting from advanced personalisation and citizen-focused public health and medical care.

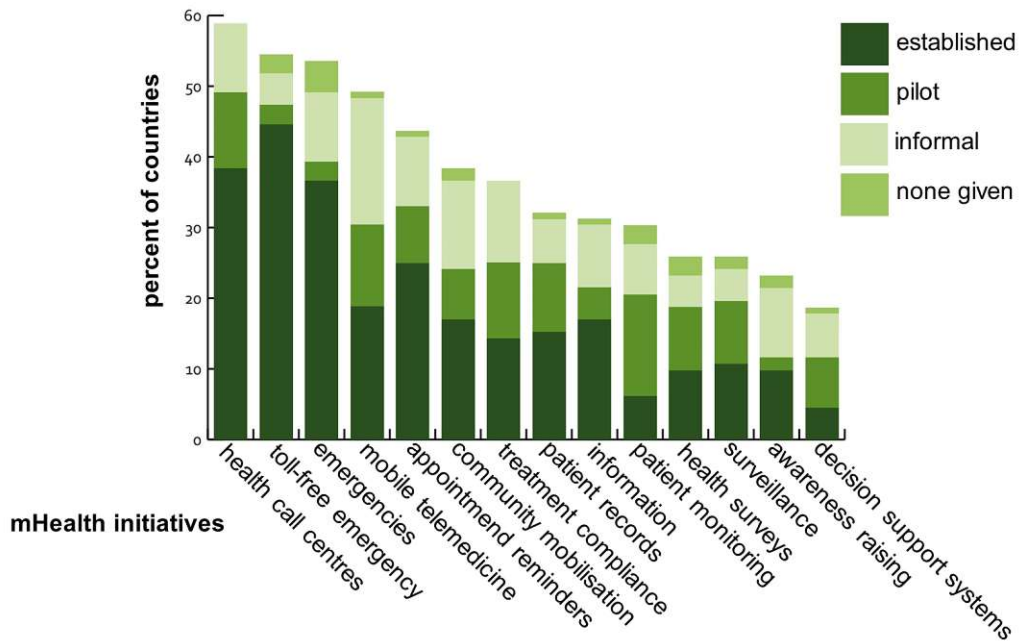


Figure 2.4: Summary of the mHealth initiatives reported to the WHO in its member states in 2011, from [89].

The most frequently reported types of mHealth initiatives were found to be health call centres or health care telephone help lines, and the least frequently reported initiatives were surveillance and decision support systems. Figure 2.4 shows the reported mHealth initiatives in 2011, when mHealth was first recognised by the WHO. Since 2011, much advancement has taken place in these fields, especially in health recommender systems, see Section 2.3.1.

The relevance of mHealth seems evident: health systems worldwide are under increasing pressure to perform under multiple health challenges, chronic staff shortages, and limited budgets [89]. mHealth inventions promise cost-effectiveness, and the reduction of other barriers to treatment, such as the availability of therapists or stigma (especially in the field of mental health) [58]. Also, the current face-to-face approach to medical treatment is collecting data at the time the client visits a clinician’s office, and reviewing self-reported data about the client’s state prior to the appointment through an error-prone mechanism of recalling past events. In mHealth systems, data collection through mobile devices can im-

prove the regularity (and thus quality) of recorded data, and, in case of passive assessment, is also less intrusive for the client [67].

An example follows to illustrate how mHealth can benefit the public health sector: cardiovascular diseases (CVDs) are the leading cause of death globally. In 2019, an estimated 17.9 million people died from CVDs, representing 32% of global deaths, and out of the 17 million premature deaths (under the age of 70) due to chronic diseases, 38% were caused by CVDs. It is important to detect CVDs as early as possible, in order for treatment to yield the best possible results. However, most CVDs can be prevented by addressing behavioural risk factors, such as tobacco use, unhealthy diet, obesity, physical inactivity, and harmful use of alcohol [87]. Section 2.3.1 introduces mHealth interventions working with the JITAI concept. Some target sedentary behaviour (e.g., providing physical activity suggestions), others addiction (e.g., tobacco, alcohol). Instead of, or in addition to, regular appointments with domain experts (i.e., doctors, dieticians, therapists), these applications can help eliminate CVD risk factors while continuously monitoring the client and adjusting the required support according to the client's needs.

The following section gives an overview of current state-of-the-art mHealth applications, designed as JITAIs.

### 2.3.1 State-of-the-Art Mobile Health Applications

Nahum-Shani et al. [58] provide several examples of JITAIs in the field of mHealth, summarising studies in different medical domains. A systematic review of JITAIs to promote physical activity is given by Hardeman et al. [36], who analyse 19 papers reporting 14 unique JITAIs for data about feasibility, acceptability, engagement, effectiveness, and health-economic outcomes. This section examines some of the JITAIs presented in either one or both of those papers, in order to give the reader an idea of current state-of-the-art research of JITAIs in mHealth. Table 2.1 gives an overview of the applications, including short descriptions.

#### A-CHESS

A-CHESS is a mobile phone application designed to improve continuing care for alcohol use disorders (AUDs) by offering emotional and instrumental support at any time and place [34]. A-CHESS is an acronym, standing for Alcohol-Comprehensive Health Enhancement Support System. The theoretical basis of A-CHESS is self-determination theory, which implies that meeting three needs contributes to an individual's functioning: being perceived as competent, feeling related to others, and feeling internally motivated (and not coerced in one's actions) [71]. The app makes it possible for the client to share steps of the recovery

Name of Application	Description
A-CHESS	Mobile phone application to support patients leaving residential alcohol use disorder (AUD) therapy [33].
FOCUS	Mobile phone application offering behavioural intervention by providing illness management support for schizophrenia patients [13].
MyBehavior	Mobile phone application that tracks a client's physical activity and location, detecting walking, running, driving, or being sedentary [67].
SitCoach	Mobile phone application for office workers that delivers messages encouraging activity [82].

Table 2.1: Overview and short description of applications using the JITAI design.

process with their counsellor, including reporting a potential relapse, or sharing the result of a weekly digital assessment. The app also provides discussion groups, web links to external sources helping with recovery, an option to ask an expert, and a panic button which notifies a person of trust if needed. The JIT aspect is realised by accessing the GPS tracker on the smartphone, in order to monitor the client's location regularly, and give alerts in case the client approaches a pre-defined high-risk location where alcohol consumption is likely to take place.

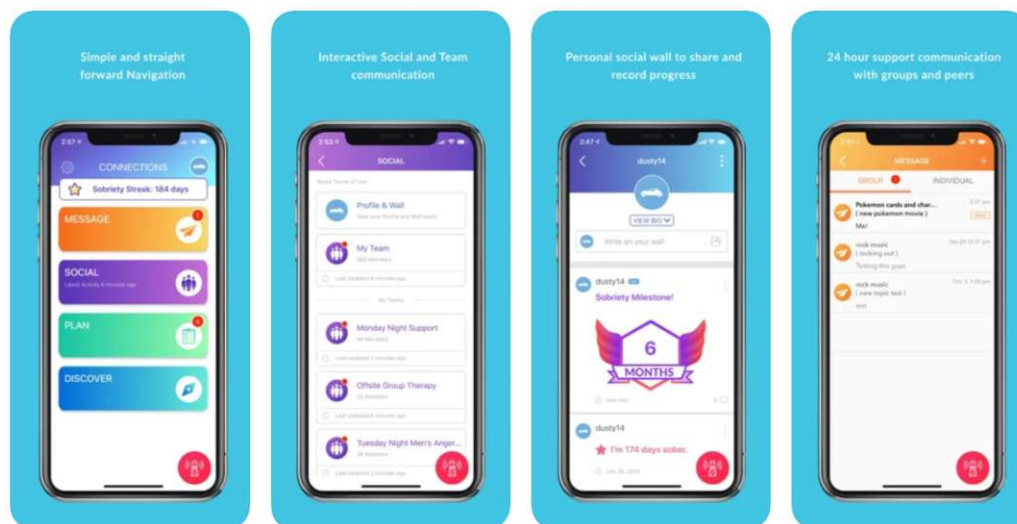


Figure 2.5: Screenshot of the A-CHESS smartphone app in the App Store, accessed on 2021-10-08.

In 2010 and 2011, clients leaving residential care for AUDs received treatment as usual, plus a smartphone with the A-CHESS application for a 12-month randomised trial. Results

showed a significant decrease in risky drinking days compared to clients in the control group, and higher chances of reporting abstinence in the previous 30 days, especially during the last four months of the study, indicating a bright future for mHealth support in addiction recovery [33]. Figure 2.5 shows the preview of the A-CHESS app in the App Store.

## FOCUS

FOCUS is a mobile phone application developed to support individuals with schizophrenia in their illness management [13]. The approach engages patients as active agents in their own treatment; they are encouraged to self-monitor their clinical status, avoid high-risk stressors, stay on track with their medications, and use coping strategies when problems associated with their condition emerge [56]. Due to lack of resources, and low-quality individual treatment, illness management support is rarely available in traditional clinical settings and FOCUS aims to provide an intervention delivery model that increases accessibility, while also providing high-quality illness management strategies for schizophrenia patients.

The FOCUS system consists of several applications that deploy adapted psychosocial intervention techniques, targeting five domains: medication adherence, mood regulation, sleep, social functioning, and coping with persistent hallucinations. The intention behind the intervention framework is that clients may select the areas they would like to focus on from a menu, which could be filled with additional content and treatment targets easily, such as diabetes management or smoking cessation. Once FOCUS is activated on a smartphone, clients are prompted daily to engage with the system via a notification, requesting a check-in. Agreeing launches a brief self-assessment on the status of the client in the target domain. If the system detects difficulties, it encourages the client to engage in self-management strategies directly linked to the problem they endorse.

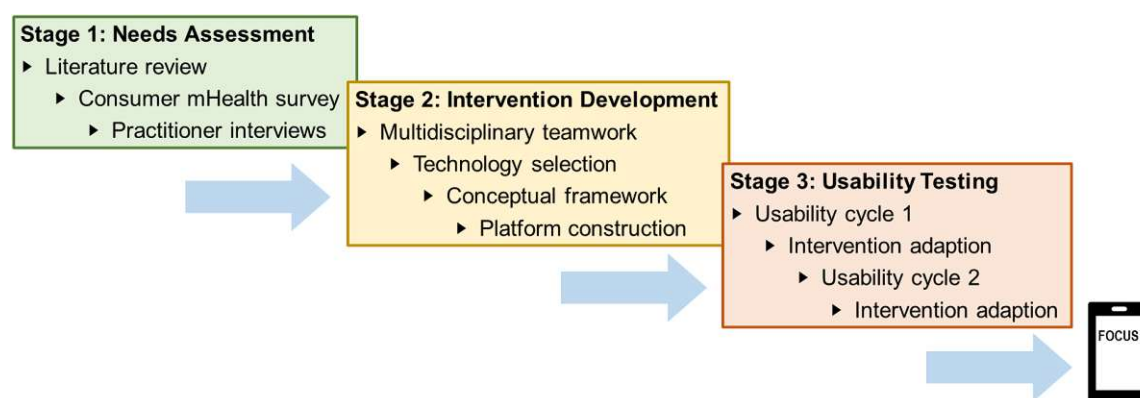


Figure 2.6: Staged development of the FOCUS mobile phone application, after [13].

After thorough research regarding needs assessment, and subsequent intervention development, a study with twelve participants was conducted in 2013, in order to assess whether the intervention design and user interface were easy to use and appealing to the clients. All participants felt confident they would be able to use the system, and the majority was satisfied with how easy it was to use, leading the development team to state that FOCUS is ready for testing in real-world conditions [13]. Figure 2.6 illustrates the stages of development for FOCUS presented by the team of developers.

### MyBehavior

MyBehavior is a mobile phone application that utilises phone sensor data to design unique recommendations for an individual, and subsequently finds activity suggestions that maximise chances of daily calorie burns in clients. It tracks a client's physical activity and location every minute, and issues suggestions once each morning. The detected activities include walking, running, driving, and being stationary. MyBehavior then analyses the location tagged activity data to find patterns that are representative of the client's behaviour.

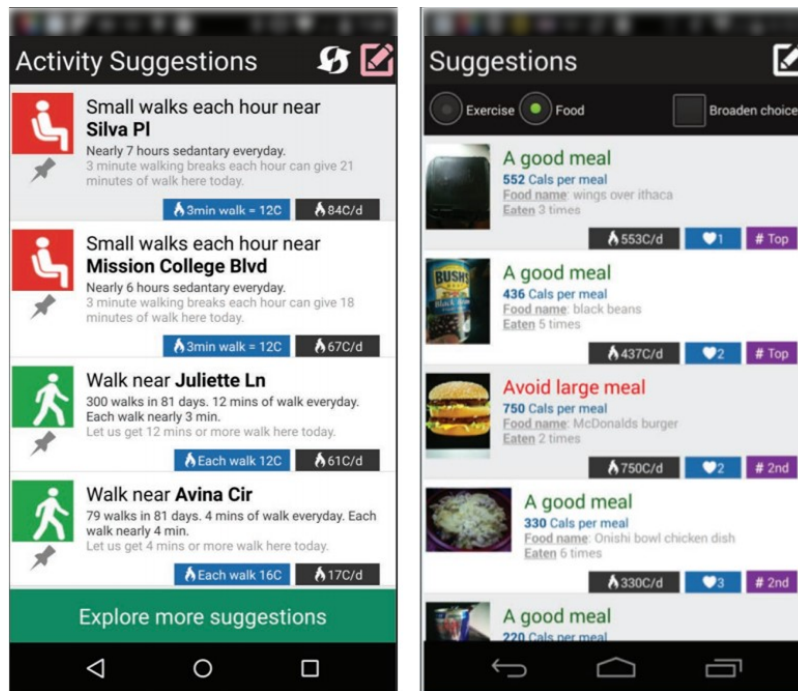


Figure 2.7: Screenshot of activity and food suggestion user interface in MyBehavior, from [66].

Specifically, MyBehavior creates three kinds of individualised suggestions. For stationary behaviour, it pinpoints the location where the client is likely to be stationary and suggests small walking breaks every hour. For walking behaviour, the app locates the different places the client usually walks in and suggests continuing to walk in those locations. For other behaviours (e.g., yoga class, gym exercise), the app encourages the client to keep up the good work [66]. Additionally, MyBehavior allows clients to self-report exercise and food intake. In order to minimise user burden, the food logging works with a crowd-sourcing database. Figure 2.7 shows screenshots of food and activity suggestions in the MyBehavior app.

After two iterations of improvement, a 14-week study with 16 participants was conducted in 2015 to test whether the app had improved in efficacy, and to assess whether the app can enable change in clients beyond the initial novelty period. During a baseline phase of three weeks, data was logged but no suggestions were provided. For two to four subsequent weeks (depending on the client), participants were subjected to control conditions, where suggestions were generated randomly and data on intervention adherence was recorded. After the control phase, participants received MyBehavior suggestions for seven to nine weeks.

Results showed that participants reported higher actionability and relatedness, which researchers believe to be linked to the app's prioritisation of low effort suggestions, also translating into increased walking behaviour, exercise, and decreased calorie intake. In general, a significant improvement of physical activity beyond the initial novelty period could be observed [66].

### **SitCoach**

SitCoach is the result of a quest to create effective persuasive mobile applications aimed at reducing sedentary behaviour, in particular to nudge office workers from their seats. SitCoach monitors physical activity and sedentary behaviour and provides persuasive messages JIT, suggesting active breaks.

After a configured number of inactive minutes, detected by the built-in accelerometer in a smartphone, SitCoach reminds clients to take a break by prompting a persuasive notification. Clients can set their own goals in terms of maximum number of consecutive sitting minutes and number of active minutes per day. The recorded data is stored for each user, and the daily value of active minutes is shared with peer users in order to motivate others through social comparison. Figure 2.8 shows the main screen of SitCoach.

To assess usability and user acceptance, a study was conducted in 8 office workers, who reported the application to be a helpful tool in reducing sedentary behaviour, although it



Figure 2.8: Main screen of the SitCoach application, from [82].

was perceived as being not very appealing. However, most reported to have little awareness of the risks of prolonged sitting, and considered their ability to take breaks to be highly dependent on external factors. A larger 6-week follow-up experiment with 86 participants was administered in 2013, where data collection was moved from the phone to a computer software detecting mouse and keyboard activity. Results showed that JIT notifications can significantly reduce computer activity, and thus sedentary behaviour [82].

The next section highlights current challenges in the development of JITAI applications.

### 2.3.2 Challenges

The JITAI concept promises cost-efficient widespread illness management, and broad access to health services. But in spite of the progress already made, see Section 2.3.1, there are challenges to overcome regarding the development process.

Section 2.2.6 discusses aspects that should be considered during the construction (and evaluation) phases of a JITAI, and several challenges are already addressed there, for example the limitations of micro-randomized trials. The concept of temporal progression [57] (i.e., identifying different timescales within which the intervention process might unfold) in particular can be useful to organise existing evidence of causal links between factors influencing the dynamics of the respective health condition, and identify directions for further research [58].

As discussed in Section 2.2, the development of an efficacious JITAI should be guided by a scientific model that integrates evidence concerning both the dynamics of the health condition, and adherence and retention to JIT interventions. However, most existing theoretical and empirical perspectives are not dynamic. They treat the mechanisms underlying these phenomena as relatively stable, only allowing them to vary as a function of baseline variables (e.g., age, biological sex) [69]. Even when existing theories acknowledge the dynamics behind these mechanisms, they often do not explain how and to what extent they change over time and what support should be offered to address them accordingly.

For example, although existing theories acknowledge that emotional distress changes over time (i.e., is dynamic) and thus needs to be monitored regularly, current theories and models do not specify how rapidly such changes are likely to occur [73]. Even though various interventions targeting distress exist, the implementation in a JIT format would benefit from being informed by dynamic theories of intervention engagement or fatigue to guide the timing, type, and amount of support provided.

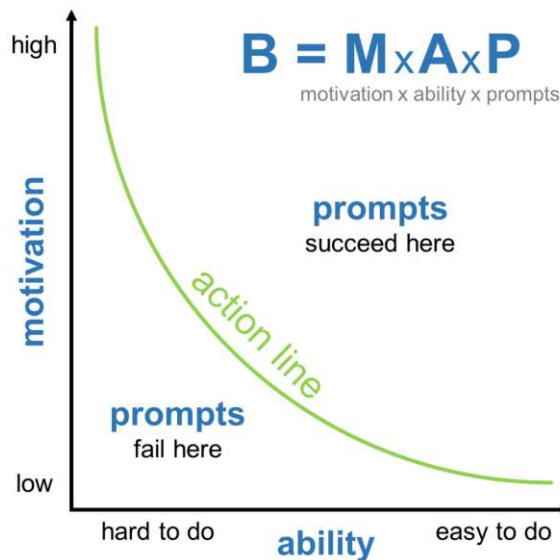


Figure 2.9: A simplified illustration of Fogg's behavior model, after [26].

However, dynamic and comprehensive models of these mechanisms are rare and incomplete [58]. The systematic review of JITAIs to promote physical activity by Hardeman et al. [36] report that only five out of 14 investigated studies of JITAIs claimed a theoretical basis. Still, behaviour theories play a colossal role in the creation of JITAIs. A popular choice is *Fogg's behavior model* [26], which is included in the mobile phone application MyBehavior [66], amongst others [24, 68]. It is based on the idea that three elements must be present at the same time for a behaviour to occur: sufficient motivation, ability, and



prompt (i.e., being triggered to perform the behaviour) [27], and when a certain behaviour does not occur, at least one of these elements is missing. Figure 2.9 illustrates a simplified version of Fogg’s behavior model. After analysing the state-of-the-art methods for creating JITAIs, Nahum-Shani et al. [58] postulate that more attention should be given to interdisciplinary approaches that facilitate the appropriate inclusion of behaviour theory into JITAI development.

Hardeman et al. [36] state that, based on their review of JITAIs to promote physical activity, many facets of JITAI development are in need of improvement, including the demand for incorporating behavioural psychology models. For example, there is no scientifically agreed-upon definition of the JITAI concept, which makes research and comparison with existing intervention designs difficult.

Theory suggests that JITAIs are superior to adaptive intervention designs in situations where JIT support can be beneficial, but the potential of JITAIs can only be fully realised when intervention suggestions are actually delivered JIT. Hardeman et al. only found one study assessing this, during which 43% of interventions were reported not to be JIT [53].

Also, there is mixed evidence in terms of effectiveness and lack of evidence in cost-effectiveness amongst the studies that have been conducted recently, suggesting that JITAIs need to be further evaluated by real-world and clinical representatives concerning uptake, reach, and impact on health inequalities, before deciding whether to adopt them in health settings.

## 3 The Contextual Multi-Armed Bandit Problem

The *contextual multi-armed bandit* (CMAB) *problem* can be found under several names in literature. It is also known as the *bandit problem with side-information* [50], *bandit problem with side observation* [85], *associative reinforcement learning* [42], *reinforcement learning with immediate reward* [1], *associative bandit problem* [76], *bandit problem with covariates* [72], or originally, *bandit problem with a concomitant variable* [91].

The term *contextual bandit problem* was invented in 2007 by Langford and Zhang [48], and it became widely accepted because it is descriptive yet short [79]. It describes a sequential decision making problem where, at each time point, a learning algorithm chooses an action (e.g., treatment), based on the *context* or side information at that point, and receives a reward that reflects the quality of the action under the current context [50]. CMAB problems provide a natural model for developing mHealth interventions: optimising mHealth intervention delivery is the act of learning an intervention option that will result in the best proximal outcome under given circumstances, which is the same as solving the CMAB problem [67].

This chapter discusses the CMAB problem as a subcategory of *multi-armed bandit* (MAB) *problems*, and explains why solving a CMAB problem is equivalent to just-in-time adaptive intervention delivery. Subsequently, different algorithms are introduced that potentially solve the problem. Then, an overview of current applications and future perspectives of MABs and CMABs in mHealth is given.

### 3.1 The Multi-Armed Bandit Problem

The MAB problem is described by Tewari and Murphy [79] as perhaps the simplest model of a sequential decision making problem where one wishes to maximise the cumulative sum of rewards received over some time horizon. Also, MAB algorithms present the simplest way of realising *reinforcement learning* (RL) [15], see Section 4.3. An application working with MABs dynamically learns and influences user behaviour by suggesting actions that maximise the chances of achieving a pre-defined goal [66].

The classical MAB problem has its origins in game theory [83], where a player is faced with  $k$  slot machines (so-called *one-armed bandits*) [8]. Each round, the player chooses one of the  $k$  arms, and receives a reward that is random while the distribution behind it is unknown to the player<sup>1</sup>. The goal of the player is to maximise the total reward received over a period of time. In order to achieve this goal, in each iteration, the player has to balance between trying different bandits to determine the rewards they yield and playing the arms that have previously yielded large rewards, because when an arm is pulled, and the player is rewarded, the potential rewards of other arms are not observed [48]. Thus, it corresponds to balancing the need to acquire more knowledge about the current environment (i.e., the reward distributions of each of the  $k$  possible choices), and the need to optimise rewards based on current knowledge [70].


$t$	1	2	3	4	5	6	...
1							...
2							...
3							...



Figure 3.1: Visualisation of the Bernoulli MAB problem, after [70].

This dilemma is called the *exploration-exploitation trade-off*, and it is this problem that MAB algorithms are addressing, each algorithm in its own way. Easy examples illustrating the trade-off in non-mathematical settings are: restaurant selection (going to a favourite restaurant versus trying an unknown one), oil drilling (drilling at the best known location versus trying a new site), or clinical trials (using the best known treatment versus trying a new, experimental one) [70].

Generally, during exploration, the goal is to form unbiased samples by randomly pulling arms to improve the accuracy of learning. During exploitation, the learning algorithm suggests the best hypothesis learned from the samples formed in the exploration phase, and the arm given by the best hypothesis is pulled, because the goal is to maximise the

<sup>1</sup>This setup describes a *stochastic bandit*. The CMAB framework permits *semi-stochastic bandits* and *fully adversarial bandits*, which describe bandits whose rewards or contexts, or both, are chosen arbitrarily, not adhering to a probability distribution [79], see Section 3.2. However, in this thesis, bandits are assumed to have stochastic rewards.

immediate reward [48]. This means that neither a purely exploring nor a purely exploiting algorithm works best [51], which is illustrated in an example in Section 3.4.1. There are several ways in which MAB algorithms handle the exploration-exploitation trade-off, and examples are discussed in Section 3.4. Figure 3.1 illustrates the classical MAB problem.

If the number of arms  $k$  is known, the MAB problem is called the  $k$ -armed bandit problem. Also, pulling the arm of the  $i$ -th bandit is equal to pulling the  $i$ -th arm of a bandit with  $k$  arms (assuming  $1 \leq i \leq k$  holds). In each iteration, an agent (former player) is faced with the same  $k$  possible choices (i.e.,  $k$  arms on a bandit), each leading to a separate random reward with unknown distribution [70]. Li et al. [51] describe the  $k$ -armed bandit problem as a subcategory of CMAB problems in which the set of arms remains unchanged and contains  $k$  arms for all  $t$ , and the context is assumed to be the same for all  $t$ . Since both the set of arms and contexts are constant at every trial, they make no difference to the bandit algorithm, and this type of bandit is referred to as *context-free*.

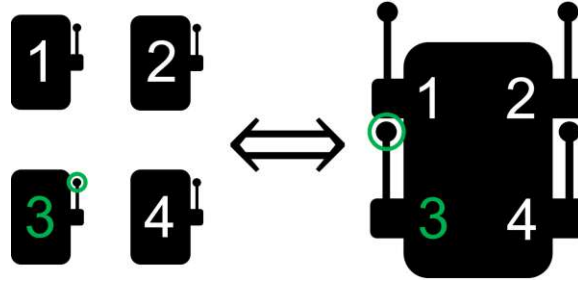


Figure 3.2: Pulling the arm of the  $i$ -th bandit is equal to pulling the  $i$ -th arm of a  $k$ -armed bandit.

This definition will not be adopted. Even though this interpretation is perfectly feasible, CMABs will be seen as a subcategory of MABs, and not vice versa, which seems to be the prevalent view [16]. MAB problems are generally (or for easier handling) defined with a fixed set of  $k$  arms at each iteration, and are thus referred to as  $k$ -armed bandits [70]. Also, when additionally including contextual information into the MAB framework (see Section 3.2), the context is assumed to change at each iteration point.

The following framework is used to mathematically describe MABs, for use in further evaluation. It is cited from Li et al. [51], but repeated in a similar manner in other papers on the subject [2, 5, 14, 48, 50, 79].

Let  $\mathcal{A}$  be the set of  $k$  arms of the bandit available to the agent at each iteration point:

$$\mathcal{A} := \{A_1, \dots, A_k\}$$

The reward for choosing arm  $a_t = A_i \triangleq i$ ,  $i \in \{1, \dots, k\}$ , at point  $t$  is denoted by  $r_{t,a_t} \in \mathbb{R}_0^+$

or, for easier handling, by  $r_t$ , and its (unknown) distribution by  $\mathcal{R}_i$ . The cumulative reward until iteration  $T$ , or *total  $T$ -trial reward*  $RW_T$  is defined as

$$RW_T := \sum_{t=1}^T r_t.$$

MAB algorithms work towards finding the unknown distribution  $\mathcal{R}_i$  for each arm. Usually, a fixed distribution underlying the process is assumed, and the MAB algorithm searches for the unknown parameters describing this distribution [79]. Thus, in practice, MAB algorithms use the *expected reward*  $\mu_t$  for arm  $a_t = i$  at trial  $t$  [2]:

$$\mu_t := \mathbb{E}[r_t]$$

Similarly, the *optimal expected  $T$ -trial reward*  $\mu_T^*$  is defined as

$$\mu_T^* := \mathbb{E} \left[ \sum_{t=1}^T r_{t,a_t^*} \right],$$

where  $a_t^*$  denotes the arm with maximum expected reward at trial  $t$ , and the subscript “ $T$ ” for  $\mu^*$  may be omitted in case the time horizon is not relevant. The goal of a MAB algorithm is to maximise the expected reward  $\mu_t$ . Analogously, the algorithm can choose to minimise the *expected  $T$ -trial regret*

$$\nu_T := \mathbb{E} \left[ \sum_{t=1}^T r_{t,a_t^*} \right] - \mathbb{E} \left[ \sum_{t=1}^T r_{t,a_t} \right] = \sum_{t=1}^T \nu_t,$$

where  $\nu_t$  is the expected regret at trial  $t$  [70]:

$$\nu_t = \mathbb{E}[r_{t,a_t^*}] - \mathbb{E}[r_{t,a_t}]$$

The (value of the)  $T$ -trial regret defined as

$$RG_T = \sum_{t=1}^T r_{t,a_t^*} - \sum_{t=1}^T r_{t,a_t},$$

which describes the (expected) difference in reward between choosing the optimal arm  $a_t^*$  (i.e., the reward that could have accumulated with prior knowledge of the problem) and arm  $a_t$  (i.e., the reward actually accumulated by the MAB algorithm) [79]. MAB algorithms can work with either reward or regret, and the choice is generally linked to the preferred method of analysis to determine the computational effort of the algorithm. After stating which element has been chosen, instead of specifying  $RW_T$ ,  $\mu_T^*$  or  $RG_T$ ,  $\nu_T$  the

regarded quantity will be denoted by  $R_T$ .

---

**Algorithm 1:** Multi-Armed Bandit Algorithm

---

**Input:** arms  $\mathcal{A} = \{1, \dots, k\}$ , reward distributions  $\mathcal{R}_i$

- 1 **for**  $t = 1, 2, \dots$  **do**
- 2     choose arm  $a_t \in \mathcal{A}$
- 3     receive reward  $r_t \sim \mathcal{R}_{i(t)}$
- 4     improve arm-selection strategy with new observation  $(a_t, r_t)$
- 5 **end**

---

Algorithm 1 shows the concept of the MAB framework in an abstract way for an open time horizon [16, 51, 79]. Note that the subscript “ $i(t)$ ” for the reward distribution  $\mathcal{R}_{i(t)}$  implies that at each  $t$ , a different arm may be chosen, and thus the index  $i$  is dependent on the time point. In-depth information on different bandit algorithms is given in Section 3.4.

The most straightforward approach is to consider the *Bernoulli*  $k$ -armed bandit problem. It is a special case of the  $k$ -armed bandit problem, where all  $k$  reward distributions are Bernoulli distributions. The reward  $r_i$  the agent receives when choosing action  $i$  is

$$r_i = \begin{cases} 1 & \text{with probability } p_i \\ 0 & \text{with probability } 1 - p_i \end{cases}, \quad p_i \in (0, 1),$$

and parameters  $p_i, i \in \{1, \dots, k\}$  are unknown to the agent [70]. By pulling an arm, the agent can either succeed ( $r_i = 1$ ) with probability  $p_i$ , or fail ( $r_i = 0$ ) with probability  $1 - p_i$ . In some practical applications, the MAB problem can be simplified by assuming a Bernoulli distribution instead of a more advanced one, which reduces the cumulative reward  $R_T$  from an explicit value to the number of successes after  $T$  iterations [2]:

$$R_T = \sum_{t=1}^T r_t = \sum_{t=1}^T \{\text{success at } t\}$$

Several MAB approaches presented below are illustrated using the Bernoulli setting, see Section 3.4. The following section expands the classical MAB framework to include contextual information, which will be regarded as the standard setting in this thesis.

## 3.2 Regarding Context in a Multi-Armed Bandit

An example for utilising the MAB framework is, as already mentioned in Section 3.1, the exploration-exploitation trade-off problem of choosing a restaurant to go to for dinner, if the choice is between a well-known and liked (maybe favourite) restaurant, and a new one

that has never been visited before. An agent deciding on a restaurant can pick either, based on similar choices made in the past; however, there are factors that can aid the decision, and may in reality substantially influence it, like geographic location, internet reviews from other customers, the weather, the kind of food being served, etc. Considering this side information can support the agent in making an informed decision, and, contrary to having to purely rely on past experiences in choosing a restaurant, may improve the likelihood of selecting the optimal scenario significantly.

Another possible field of application for MABs is matching recommended news articles to a user's personal interests on the internet. A large number of articles is available, to be displayed to visitors on a website, and the goal is to maximise the click-through rate that is generated when users click on news articles. A MAB algorithm employed to pick the articles may go through a substantial trial-and-error phase during exploration trying to match a visitor's news content to their individual preferences, but considering the vast amount of available articles, the enormous number of different visitors, and (possibly) sparse frequency of visits per individual, this strategy does not seem profitable. In order to increase the likelihood of a visitor clicking on an article, the implemented algorithm may take contextual information about the visitor into account (e.g., age group, biological sex, geographical location), which, together with accessing pooled information from visitors with similar context, increases the probability of displaying an article that is relevant to the visitor [51].

These simple examples emphasise how settings without any contextual information rarely occur [48]. Considering additional information will facilitate the decision process for the agent: for example, if the agent has to choose a sporting activity, and is able to take the current weather conditions into account, it is capable of disregarding outdoor activities in case the weather is bad, thus increasing the chance to recommend activities suited not just to the client, but their context.

CMABs are first described by Woodroffe in 1979 [91], according to Rabbi et al. [67], where the context variable is referred to as a concomitant variable. The problem is originally formulated in a clinical trial setting, where two available treatments represent the arms of a bandit: a standard treatment, whose statistical characteristics are known, and a new treatment, whose characteristics are unknown [91]. Contrary to the MAB approach, before making a decision, the agent sees a feature vector representing the context. The agent uses this feature vector along with the feature vectors and arms played in the past, to choose the arm to play in the current round. Over time, the agent's aim is to gather enough information about how the feature vectors and rewards relate to each other, so that it is possible to predict, with some certainty, which arm is likely to give the best reward by regarding the current feature vector [4, 16, 48].

The CMAB problem has many fields of application [15] and is often more suitable than the MAB problem [48], see Section 3.5. Its framework is an extension of the MAB framework to include a context vector.

Let  $\mathcal{A}$  be the set of  $k$  arms available at time  $t$ ,  $r_t$  the reward for choosing arm  $a_t = i$ , and  $\mu_t$  the expected reward obtained at  $t$ . Depending on the implementation,  $R_T$  is either the expected T-trial reward or regret. Any contextual information at time  $t$  is stored in a  $d$ -dimensional vector  $x_t \in \mathbb{R}^d$ . As already mentioned in Section 3.1, the context (or feature) vector is the same for all arms  $A_i \hat{=} i$ ,  $i \in \{1, \dots, k\}$  at time  $t$ , but it will change for different  $t$  [51]. During a trial at  $t$ , before selecting an arm, the agent observes the feature vector  $x_t$ . This vector can be assumed to either be random (i.e., drawn from a distribution, *stochastic bandit*), or adversarial (i.e., assumed to be chosen arbitrarily by an adversary or generated by nature, *semi-stochastic* or *semi-adversarial bandit*). The terms “semi-stochastic” and “semi-adversarial” emphasise that, even though the context is arbitrary, the reward is assumed to be random with an underlying probability distribution [79].

---

**Algorithm 2:** Contextual Multi-Armed Bandit Algorithm

---

**Input:** arms  $\mathcal{A} = \{1, \dots, k\}$ , reward distributions  $\mathcal{R}_i$

- 1 **for**  $t = 1, 2, \dots$  **do**
- 2     observe context  $x_t \in \mathbb{R}^d$
- 3     choose arm  $a_t \in \mathcal{A}$
- 4     receive reward  $r_t \sim \mathcal{R}_{i(t)}$
- 5     improve arm-selection strategy with new observation  $(x_t, a_t, r_t)$
- 6 **end**

---

Algorithm 2 illustrates the concept of the CMAB framework [51]. As a summary of this section, formal definitions for both the CMAB problem and the CMAB algorithm are offered, introduced by Langford and Zhang [48], see Definitions 3.1 and 3.2. These definitions emphasise the difference between the CMAB problem and a CMAB algorithm. However, Langford and Zhang [48] assume a stochastic bandit, whereas this thesis focusses on the semi-stochastic bandit setting, and, as such, the definitions have been adapted accordingly.

**Definition 3.1** (Contextual Bandit Problem). *In a contextual multi-armed bandit problem, there is a set of arms  $\{1, \dots, k\}$ , a context vector  $x_t \in \mathbb{R}^d$ , arbitrarily chosen, and distributions  $\mathcal{R}_i$  from which the reward  $r_i$  for each arm  $i \in \{1, \dots, k\}$  is drawn. The problem is a repeated game: on each round  $t$ , a sample  $(r_1, \dots, r_k)$  is drawn from  $(\mathcal{R}_1, \dots, \mathcal{R}_k)$ , the context  $x_t$  is announced, and then precisely for one arm  $a_t \in \{1, \dots, k\}$  chosen by the player, its reward  $r_t$  is revealed.*



**Definition 3.2** (Contextual Bandit Algorithm). *A contextual bandits algorithm determines an arm  $a_t \in \{1, \dots, k\}$  to pull at each time step  $t$ , based on the previous observation sequences  $(x_1, a_1, r_1) \dots (x_{t-1}, a_{t-1}, r_{t-1})$ , and the current context  $x_t$ .*

From now on, the CMAB framework will be regarded as the standard setting of solving exploration-exploitation trade-off problems, and Section 3.3 explains why it is a feasible approach for implementing decision rules in JITAI applications for mHealth and the clinical setting.

### 3.3 The Contextual Multi-Armed Bandit Approach for Decision Rules in a Clinical Setting

As mentioned in the introduction to Chapter 3, CMAB problems provide a natural model for developing mHealth interventions. This can be attributed to the JITAI design favoured for applications in mHealth introduced in Section 2.2. It is presented as a many-faceted solution for problems in mHealth in Section 2.3, due to it being easily translatable into CMAB problem components [31, 67], as explained below.

Section 3.2 introduces four main elements in a CMAB algorithm:

- Points in time, denoting trials  $t = 1, 2, \dots$
- $k$  arms  $\mathcal{A} = \{A_1, \dots, A_k\} \hat{=} \{1, \dots, k\}$
- $d$ -dimensional contexts (or feature vectors)  $x_t \in \mathbb{R}^d$  for each trial  $t$
- Respective rewards  $r_t \sim \mathcal{R}_{i(t)}$  at each trial  $t$  for chosen arm  $i$

Algorithm 2 shows how these elements interact in a CMAB setting. Depending on the chosen method of implementation, see Section 3.4, exploration and exploitation are balanced in order to maximise the cumulative expected reward (or minimise the cumulative expected regret). In a similar manner, the five components needed in an JITAI design are defined in Section 2.2 [58]:

- Decision points
- Tailoring variables
- Intervention options
- Decision rules
- Proximal outcome

Figure 2.1 illustrates how these components interact, and Sections 2.2.1 - 2.2.5 give detailed explanations as well as providing discussions on their application. At each decision point the decision rules find an intervention option, based on the current values of all, or some, tailoring variables, in order to facilitate the proximal outcome.

There are already similarities in the formulation of both concepts, and the JITAI framework can easily be translated into a CMAB problem: decision points denote the points in time at which a trial takes place, tailoring variables represent any contextual information in the form of a feature vector, and the possible intervention options serve as the arms of a bandit. Furthermore, reaching the proximal outcome is equal to maximising the cumulative reward, or, in case of a Bernoulli bandit, succeeding at a trial [31, 58, 67].



Figure 3.3: Visual representation of the analogy between the elements of the CMAB approach and the components of the JITAI design.

Figure 3.3 illustrates the merging of these concepts graphically. Only the decision rules do not have an immediate analogy. However, Section 2.2.4 states that decision rules are a mechanism that decides on an intervention option based on the tailoring variables, making it immanently clear that decision rules are represented by the CMAB algorithm [58]. Keeping in mind this translation, it follows that solving a CMAB problem is equivalent to optimising intervention delivery, which is the act of learning the intervention option that will result in the best proximal outcome under given circumstances [67]. Figure 3.4 shows the updated JITAI concept from Figure 2.1 with respect to CMAB design.

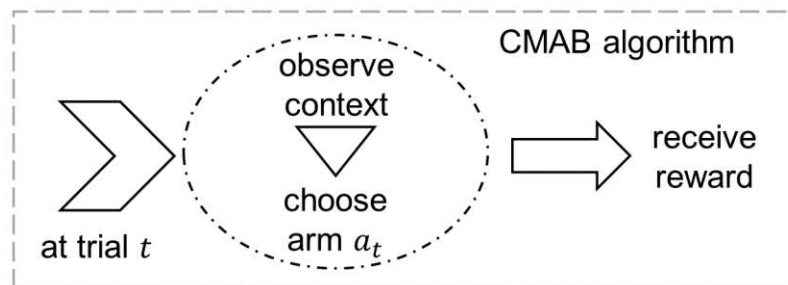


Figure 3.4: JITAI concept from Figure 2.1 adapted for the use of a CMAB algorithm.

Recent interest in contextual bandits has been driven to a large extent by personalisation problems arising on the internet, see the example in Section 3.2. With the emergence of mHealth, Tewari and Murphy [79] state that many ideas, developed to show personalised news articles or advertisements to visitors on websites, will be found useful in personalising mHealth interventions for clients in a particular context.

It is already common practice to use CMAB algorithms for researching personalised adaptive interventions in mHealth, see Section 3.5. The simplest case is presented by only considering two possible intervention options: whether to intervene, or not. Note that this scenario requires a provide nothing option, see Section 2.2.3. For example, in an adaptive intervention targeting physical activity, these options can be whether or not to send a notification, encouraging exercise. Once an intervention option has been chosen, a reward is collected, indicating to what degree the proximal outcome was accomplished, such as, the number of steps walked in a day, or whether the recommended number of steps for that day was achieved (in case of a Bernoulli bandit) [79].

Section 3.4 gives an overview of CMAB algorithms, where their differences, advantages, and disadvantages are being discussed.

## 3.4 Contextual Multi-Armed Bandit Algorithms

This section presents three general strategies for CMAB algorithms. For simplicity, the strategies are illustrated as solutions to the MAB problem in a Bernoulli bandit environment, with the option of being extended into a CMAB framework. Note that a comparison of strategies only holds for a specific problem; strategies are almost impossible to compare in general, and only their advantages and disadvantages can be discussed. The content of this section and any subsequent subsections is cited from Rocca and Rocca [70], unless otherwise stated.

It is a truth universally acknowledged that the selection of an algorithm depends on the specific problem it is supposed to solve, and the CMAB approach is no exception. This section aims to help the reader understand the choice of algorithm presented in Section 4.5, in light of the model problem, with whom the reader gets acquainted in Chapter 5. First, the problem is formulated as a  $k$ -armed Bernoulli bandit problem. Subsequently, the approaches are put forward and illustrated by examples, and their individual characteristics are discussed.

Section 3.1 has briefly introduced the  $k$ -armed CMAB problem with Bernoulli bandits, where the  $k$  reward distributions  $\mathcal{R}_1, \dots, \mathcal{R}_k$  for choosing arms  $\{A_1, \dots, A_k\} \triangleq \{1, \dots, k\}$  are assumed to be Bernoulli distributions  $\mathcal{R}_i = \mathbf{Bernoulli}(p_i)$ . In this case, the reward  $r_i$

the agent receives when choosing action  $i$  is

$$r_i = \begin{cases} 1 & \text{with probability } p_i \\ 0 & \text{with probability } 1 - p_i \end{cases}, \quad p_i \in (0, 1),$$

which can be interpreted as succeeding with probability  $p_i$ , and failing with probability  $1 - p_i$  at a trial. However, the value of the success probability  $p_i$  is not known to the agent. Let  $a_t$  denote the choice of arm at trial  $t$ , then  $r_{t,a_t} = r_t$  denotes the reward received for choosing  $a_t$ . In case of a Bernoulli bandit, the expected reward of chosen action  $a_t$  at time  $t$  is the success probability for the arm:

$$\mu_t = \mathbb{E}[r_t] = p_{a_t}$$

Subsequently, the expected reward for the best arm is  $\max_i(p_i)$ . It follows that the expected regret  $\nu_t$  (i.e., the difference in the expected rewards between the optimal arm, and the chosen arm, see Section 3.1) at time  $t$ , and the expected T-trial regret  $\nu_T$  are

$$\begin{aligned} \nu_t &= \max_i(p_i) - p_{a_t}, \\ \nu_T &= \sum_{t=1}^T \nu_t = T \cdot \max_i(p_i) - \sum_{t=1}^T p_{a_t}. \end{aligned}$$

In practice, two obstacles make the problem difficult to solve: the values of  $p_i$ ,  $i \in \{1, \dots, k\}$ , are unknown, and the expected regret cannot be computed. However, these variables are of interest in a simulation environment, where the exact values of  $p_i$ ,  $i \in \{1, \dots, k\}$ , are known, even though they are assumed not to be, to assess the quality of a strategy. During each iteration, a low regret  $\nu_t$  indicates that the chosen arm is close to the optimum, whereas a high value of  $\nu_t$  indicates that the chosen arm is far from optimal.

Strategy	Concept
$\epsilon$ -Greedy	Pick the arm assumed to be optimal with probability $1 - \epsilon$ , explore a random arm with probability $\epsilon$
Upper Confidence Bound	Pick the action with the highest upper confidence limit (i.e., estimate of the reward)
Thompson Sampling	Pick an action randomly according to their probability to be the best (i.e., yield the highest expected reward)

Table 3.1: Strategies used in CMAB algorithms with short descriptions.

A good strategy distinguishes itself by rapidly decreasing the regret to zero, suggesting that the best option is identified quickly, and subsequently exploited. On the other hand,

in a poor strategy, the values of regret decrease slowly and/or do not reach zero, implying that the best option is either identified after a long time, not identified at all, or not well exploited. Figure 3.5 shows a sketch of possible regret curves for different strategies.

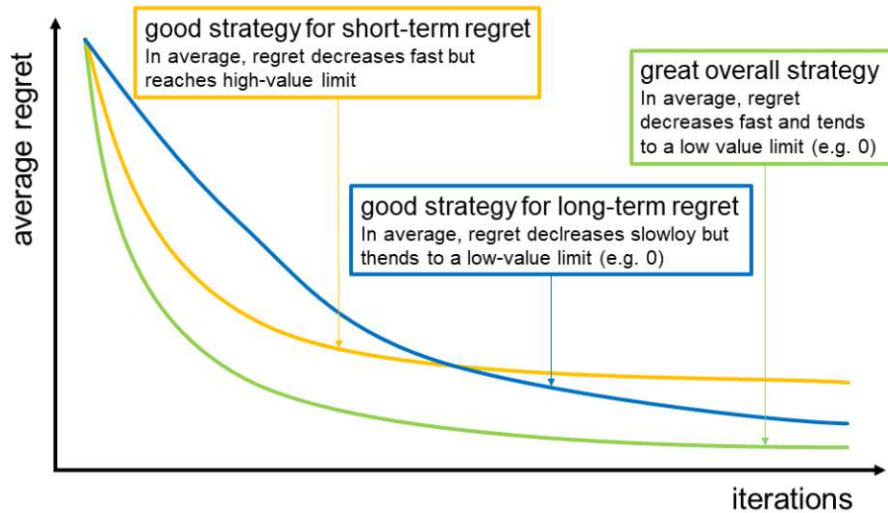


Figure 3.5: Regret curves for different strategies, schematically sketched after [70].

The following sections introduce different strategies to cope with the MAB (and CMAB) problem. A general overview is given in Table 3.1.

### 3.4.1 The $\epsilon$ -Greedy Strategy

The  $\epsilon$ -greedy strategy, also known as the  $\epsilon$ -greedy strategy [51, 79], is an extension of the greedy strategy, which will be discussed first. Following a greedy strategy means making the decision that seems to be the best with regard to the current knowledge in each iteration step, implying that the approach only exploits without exploring. As mentioned in Section 3.1, a purely exploiting algorithm will not deliver the desired output, because if the current knowledge is not accurate enough, the algorithm might get stuck choosing one or several suboptimal arms without any chance to discover better options.

This dilemma can be illustrated using a Bernoulli bandit with three arms  $\mathcal{A} = \{A_1, A_2, A_3\}$  and (known) success probabilities

$$p_1 = 0.3, \quad p_2 = 0.7, \quad p_3 = 0.8.$$

Assume that the prior knowledge of the values of  $p_i$ ,  $i \in \{1, 2, 3\}$  (i.e., the assumed values of  $p_i$  before making any observation)<sup>2</sup> is set to 0.5. Let  $e_i^t$  be the estimated success

<sup>2</sup>This situation corresponds to a *warm start* or *offline learning format*.

probability for each arm at time  $t$ , then it holds that

$$e_1^0 = 0.5, \quad e_2^0 = 0.5, \quad e_3^0 = 0.5.$$

Since a Bernoulli bandit is considered, these estimates also correspond to the estimates of the expected reward for each arm. Assuming a greedy strategy, at  $t = 1$ , the agent chooses the arm with the highest expected reward estimate. Since all estimates have equal value, the agent picks an arm at random, for example  $A_2$ , and, if the trial is successful, the observed reward is 1. Using this knowledge, the estimate of  $A_2$  is updated accordingly<sup>3</sup>:

$$e_1^1 = e_1^0 = 0.5, \quad e_2^1 = \frac{1}{2}(0.5 + 1) = 0.75, \quad e_3^1 = e_3^0 = 0.5$$

The estimates are now no longer equal and, based on the new values, at  $t = 2$ , the agent will choose  $A_2$  again. The problem is immanently clear: if, during successive iterations, the estimate of the expected reward for  $A_2$  never falls below 0.5,  $A_3$  will never be chosen, and the optimal choice will never have been explored. This scenario is likely, since the true value of  $p_2$  is set to 0.7. If all initial estimates  $e_i^0$  are set to 0.1, the same observation can be made when the agent chooses  $A_1$  at  $t = 1$ . Figure 3.6 illustrates the start of three possible simulation outcomes with initial estimates  $e_i^0$  all equally set to 0.5. The agent correctly identifies the best arm quickly in the first simulation run, whereas it can potentially get stuck on a suboptimal arm during the other two simulations, due to lack of exploration.

	Initialisation	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5	Iteration 6
Run 1	1 2 3 0.5 0.5 0.5	1 2 3 0.75 0.5 0.5	1 2 3 0.5 0.5 0.5	1 2 3 0.5 0.5 0.75	1 2 3 0.5 0.5 0.833	1 2 3 0.5 0.5 0.875	1 2 3 0.5 0.5 0.7
Run 2	1 2 3 0.5 0.5 0.5	1 2 3 0.5 0.5 0.25	1 2 3 0.25 0.5 0.25	1 2 3 0.25 0.75 0.25	1 2 3 0.25 0.833 0.25	1 2 3 0.25 0.625 0.25	1 2 3 0.25 0.7 0.25
Run 3	1 2 3 0.5 0.5 0.5	1 2 3 0.75 0.5 0.5	1 2 3 0.5 0.5 0.5	1 2 3 0.375 0.5 0.5	1 2 3 0.375 0.75 0.5	1 2 3 0.375 0.833 0.5	1 2 3 0.375 0.875 0.5

Figure 3.6: Example of greedy strategy simulations in a three-armed bandit with initial estimates set to 0.5 and true values (0.3, 0.7, 0.8), after [70].

Algorithm 3 shows a formal representation of the greedy strategy for a bandit with  $k$  arms, but it does not include any methodology to obtain the initial reward estimates  $e_i^0$ ,

<sup>3</sup>In the context of Bernoulli bandits, using the the average over all observed rewards is sensible due to the nature of the Bernoulli distribution.

which in practice need to be determined either through data, or theory. For example, if the CMAB algorithm is employed to match news articles, or advertisements, on a website to a user's preferences, an initial selection can be curated by considering the content consumption of other users with similar online profiles, capitalising on the assumption that factors like gender, or geographic location, that substantially influence topics of interest [51].

---

**Algorithm 3:** Greedy Strategy

---

**Input:** arms  $\mathcal{A} = \{1, \dots, k\}$ , initial estimates  $e_i$

```

1 for  $t = 1, 2, \dots$  do
2   observe  $S_t := \{l \mid l = \operatorname{argmax}_i e_i\}$ 
3   if  $\|S_t\| = 1$  then
4     choose arm  $a_t = j = \operatorname{argmax}_i e_i$ 
5   else
6     choose random arm  $a_t = j \in S_t$ 
7   end
8   receive reward  $r_t \sim \mathcal{R}(e_j)$ 
9   update  $e_j$ 
10 end

```

---

The  $e$ -greedy algorithm is a simple way of incorporating exploration into a decision making process guided by a greedy strategy. At each iteration point, either the currently optimal arm is chosen with probability  $1 - e$  (exploitation), or a random arm is picked with probability  $e$  (exploration). The parameter  $e \in [0, 1]$  is called the *exploration parameter*, and it balances the exploration-exploitation trade-off: low values of  $e$  indicate a low degree of exploration, and  $e = 0$  describes a purely exploiting (greedy) strategy. Similarly,  $e = 1$  indicates a purely exploring, or random strategy.

The  $e$ -greedy strategy can be explained using the three-armed Bernoulli bandit example above. With the exploration parameter  $e$  set to 0.1, in 10% of iterations, the agent will pick a random arm (any arm, which can also be the currently best one). The issue of potentially missing the optimal arm (in an infinite time horizon) in the greedy strategy is thus addressed. However, a new problem arises: even after the agent has identified the optimal arm, the algorithm continues exploring, and may return to selecting worse choices. The cost of this ever-lasting exploration can be expressed by the expected regret when randomly picking an action, and is computed as follows:

$$\frac{1}{3} (\nu_{p_1} + \nu_{p_2} + \nu_{p_3}) = \frac{1}{3} (\nu_{0.3} + \nu_{0.7} + \nu_{0.8}) = \frac{1}{3} ((0.8 - 0.3) + (0.8 - 0.7) + (0.8 - 0.8)) = 0.2$$

Here, the subscript of the regret at time  $t$  does not denote the iteration point, but the success probability of the respective bandit. After the optimal arm has been identified, the

algorithm produces zero regret with 90% probability, and 0.2 regret with 10% probability. So the average regret can decrease towards  $0.1 \times 0.2 = 0.02$ , but will not go lower. Algorithm 4 formalises the  $\epsilon$ -greedy strategy [79]. Similar to Algorithm 3, the initial exploration phase is missing, and, without loss of generality, the case of identical probability estimates is omitted. Figure 3.7 compares the greedy strategy with the  $\epsilon$ -greedy strategy in terms of average regret along iterations for the three-armed Bernoulli bandit example and different values of  $\epsilon$ .

---

**Algorithm 4:**  $\epsilon$ -Greedy Strategy

---

**Input:** arms  $\mathcal{A} = \{1, \dots, k\}$ , initial estimates  $e_i$ , exploration parameter  $\epsilon$

```

1 for  $t = 1, 2, \dots$  do
2   set  $O_t = \operatorname{argmax}_i e_i$ 
3   set  $E_t =$  randomly selected arm from  $\mathcal{A}$ 
4   with probability  $(1 - \epsilon)$  choose arm  $a_t = j = O_t$ , else choose arm  $a_t = j = E_t$ 
5   receive reward  $r_t \sim \mathcal{R}(e_j)$ 
6   update  $e_j$ 
7 end

```

---

To summarise, the greedy strategy's long term regret is related to the risk of missing the optimal arm forever, and the  $\epsilon$ -greedy strategy's long term regret is related to the inefficiency of exploring forever. Ideally, the algorithm should stop exploring when the agent is certain that the optimal arm has been identified.

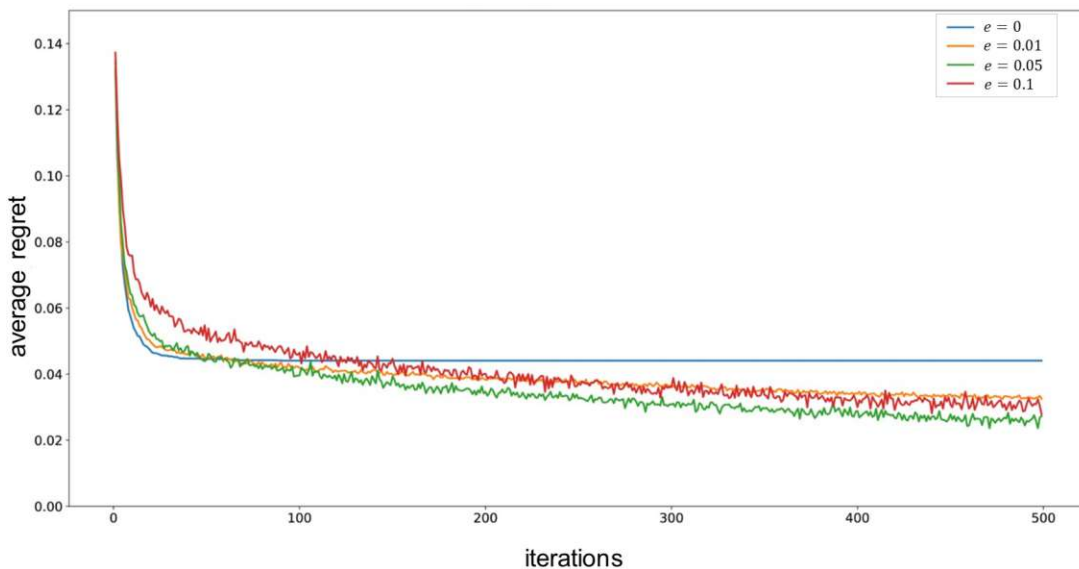


Figure 3.7: Comparison between greedy and  $\epsilon$ -greedy strategy for the three-armed Bernoulli bandit and different values of  $\epsilon$ , from [70].



A possible solution is an  $e$ -greedy strategy with decaying exploration parameter  $e$ . The three-armed Bernoulli bandit example illustrates that a high value for  $e$  is useful at the beginning of the process, because it allows to find the optimal arm quickly, but implies a high long-term average regret, which can be an issue depending on the problem at hand. A low value of  $e$  takes longer to identify the optimal arm, but has lower long-term average regret. Thus, a decaying  $e$ -greedy algorithm starts with a given value for  $e$  and progressively reduces it during iterations at a rate that mirrors the increasing certainty that the agent has found the best option, and exploring less and less as time goes on. However, the optimal decay schedule is difficult to define and varies with the problem. Also, this approach is only relevant for problems that are guaranteed to have their long-term average regret decrease towards zero. Figure 3.8 illustrates the decaying  $e$ -greedy strategy for different decay schedules and compares them to the non-decaying situation for exploration parameter  $e = 0.2$ .

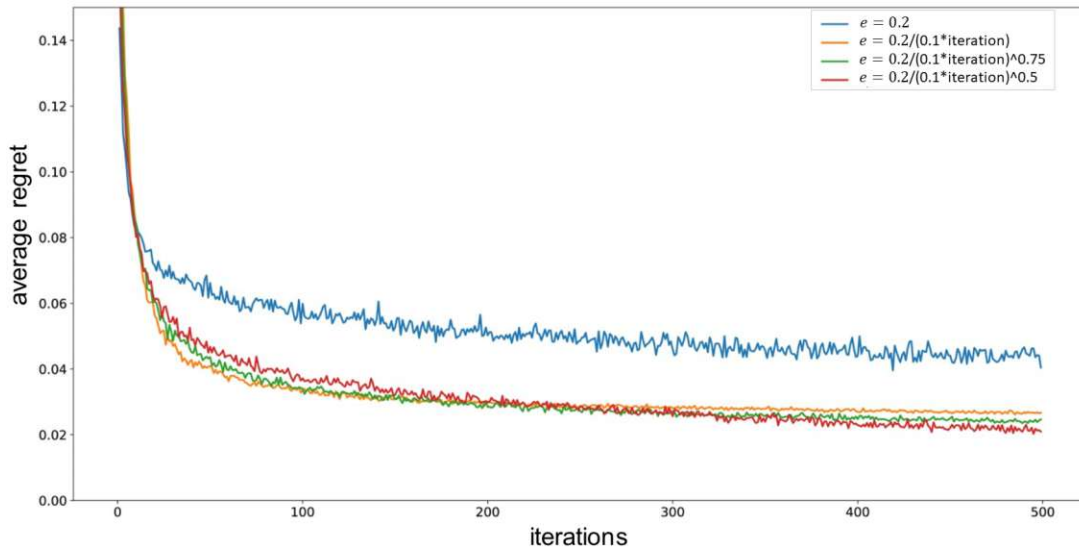


Figure 3.8: Decaying  $e$ -greedy strategy for different decay schedules, from [70].

Another drawback of the  $e$ -greedy strategy is the random nature of the exploration. During an exploration step all arms can be chosen with equal possibility, regardless of any past reward observations, so arms that are already proven to give low rewards with high certainty are just as likely to be chosen as arms with high uncertainty, which could still lead to higher rewards. The strategies introduced in Sections 3.4.2 and 3.4.3 resolve uncertainty in a smarter way, by focusing the exploration on the most relevant arms that could still prove to be optimal. Both of those strategies deal with uncertainty in an explicit way. Optimistic initialisation is one of the implicit approaches. The idea is to start with the most optimistic initial reward estimates, in order to ensure that at least a minimal amount

of exploration will be performed on each arm. In light of the three-armed Bernoulli bandit, this means:

$$e_1^0 = 1, \quad e_2^0 = 1, \quad e_3^0 = 1$$

Here, all arms are initially assumed to be optimal, and exploration is required in order to identify the suboptimal arms.

The advantage of the  $\epsilon$ -greedy strategy lies in its adaptability and simplicity. For example, Li et al. [51] use different versions of the  $\epsilon$ -greedy algorithm to compare the impact of applying various CMAB algorithms on user behaviour for recommending news articles on websites, amongst them are a segmented  $\epsilon$ -greedy algorithm, an  $\epsilon$ -greedy algorithm with a warm start, and a disjoint  $\epsilon$ -greedy algorithm. Another algorithm working with a greedy strategy is the epoch-greedy algorithm, which separates its iterations in epochs, and during each epoch, a single exploration step is performed at the beginning. Once the arm selection strategy has been updated, an exploitation phase follows, and the scenario is repeated in each epoch [48]. The  $\epsilon$ -greedy strategy can also be implemented implicitly: instead of fixing a value for the exploration parameter  $\epsilon$ , certain iteration steps can be fixed as exploration rounds prior to starting the algorithm and, in each iteration, the agent decides whether to exploit or explore by checking if the current  $t$  is an element of a set  $\mathcal{T}_{exp}$  of pre-defined exploration steps. This method is formalised in the linear response bandit algorithm provided by Tewari and Murphy [79], based on the work of Goldenshluger and Zeevi [29].

### 3.4.2 The Upper Confidence Bound Strategy

The *upper confidence bound (UCB) strategy* differs from the  $\epsilon$ -greedy strategy in a significant way. The  $\epsilon$ -greedy strategy introduced in Section 3.4.1 does not require explicit modelling, or quantifying, of knowledge uncertainty. The knowledge at time  $t$  is modelled by a point estimate, which does not reflect the uncertainty about this value. Another approach is to explicitly model the uncertainty, for example with confidence intervals or probability distributions, which encompasses both the current knowledge (i.e., the mean) and the related uncertainty (i.e., the variance of the distribution, or size of the confidence interval) to guide the exploration process. The UCB strategy and the strategy introduced in Section 3.4.3 rely on this concept in different ways.

The idea of the UCB strategy is to substitute the probability estimates  $e_i^t$  of the expected reward  $\mu_t$  (calculated as an average, see Section 3.4.1) by an upper confidence bound for each arm. This behaviour is described as “optimism in front of uncertainty”: the uncertainty about the expected reward of an arm is expressed as a confidence interval, and the optimal arm is chosen “optimistically”, assuming the upper bounds of these intervals to

be the true expected rewards. The width of a confidence interval reflects the uncertainty of the agent’s knowledge about the respective arm [7], and the agent keeps exploring arms that have not yet been proven to yield low rewards instead of arms that produce low rewards with high certainty.

The upper confidence limit of the expected reward can be high for two reasons: either the confidence interval is narrow with high certainty (i.e., the mean is large), thus the upper confidence limit is large and the arm is a good choice, or the confidence interval is wide (i.e., the variance is large), thus having a high upper limit, which indicates much uncertainty about the value of the expected reward [67]. Algorithm 5 formalises this idea for a  $k$ -armed bandit. Note that the superscript “ $t$ ” for the distributions refers to their current parameters in step  $t$ . These parameters do not change during each time step, but are only updated in case arm  $i$  is chosen. For example, if arm  $j$  is chosen for the first time at  $t = 3$ , it holds that

$$\mathcal{R}_j^0 = \mathcal{R}_j^1 = \mathcal{R}_j^2 \neq \mathcal{R}_j^3.$$

---

**Algorithm 5:** Upper Confidence Bound Strategy

---

**Input:** arms  $\mathcal{A} = \{1, \dots, k\}$ , initial expected reward distributions  $\mathcal{R}_i^0$

```

1 for  $t = 1, 2, \dots$  do
2   for  $i = 1, \dots, k$  do
3     | observe upper confidence limit  $u_i$  for  $A_i$ 
4   end
5   choose arm  $a_t = j = \operatorname{argmax}_i u_i$ 
6   receive reward  $r_t \sim \mathcal{R}_{j(t)}^t$ 
7   update expected reward distribution  $\mathcal{R}_{j(t)}^t$ 
8 end

```

---

The UCB strategy implicitly trades off between exploration and exploitation: the agent chooses the arm with the highest upper confidence limit. If the expected reward of that arm has a narrow confidence interval, the observed reward is expected to be sufficiently high and the agent has performed an exploitation step. However, if the expected reward has a wide confidence interval and the observed reward is low, the confidence interval becomes narrower and the upper confidence limit drops, so the agent has performed an exploration step and eliminated a possible optimal choice. Informally speaking: if an arm is chosen whose expected reward has a large mean, it is an exploitation step, and if an arm is chosen whose expected reward has large variance, it is an exploration step [7]. Figure 3.9 gives a graphical representation of exploration and exploitation in this context.

The UCB strategy can be illustrated using the three-armed Bernoulli bandit example from Section 3.4.1. At time  $t$ , assume that the uncertainty about the expected reward of

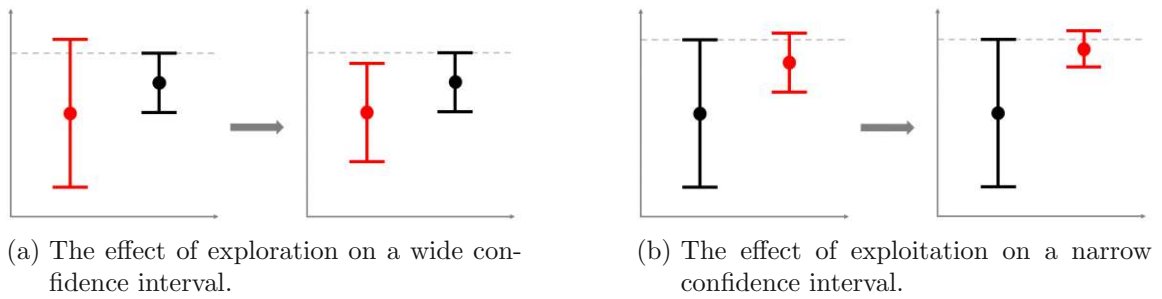


Figure 3.9: Visual representation of exploration and exploitation in a UCB strategy.

each arm  $A_i \in \mathcal{A}$  is modelled by a beta distribution  $\mathbf{Beta}(\alpha_i, \beta_i) \doteq \mathcal{R}_i^t$ . Subsequently, the upper confidence limit for the confidence interval of 80% can be computed for all arms, which stands for the optimistic guess of the true expected reward  $\mu_t$ . The agent then chooses the arm with the highest upper confidence limit, for example  $A_2$ , a reward is observed, and the probability distribution  $\mathcal{R}_2^t$  (i.e., the parameters  $\alpha_i, \beta_i$  of the Bernoulli distribution) describing the uncertainty about the expected reward is updated before starting a new iteration. The concept is graphically illustrated by Figure 3.10.

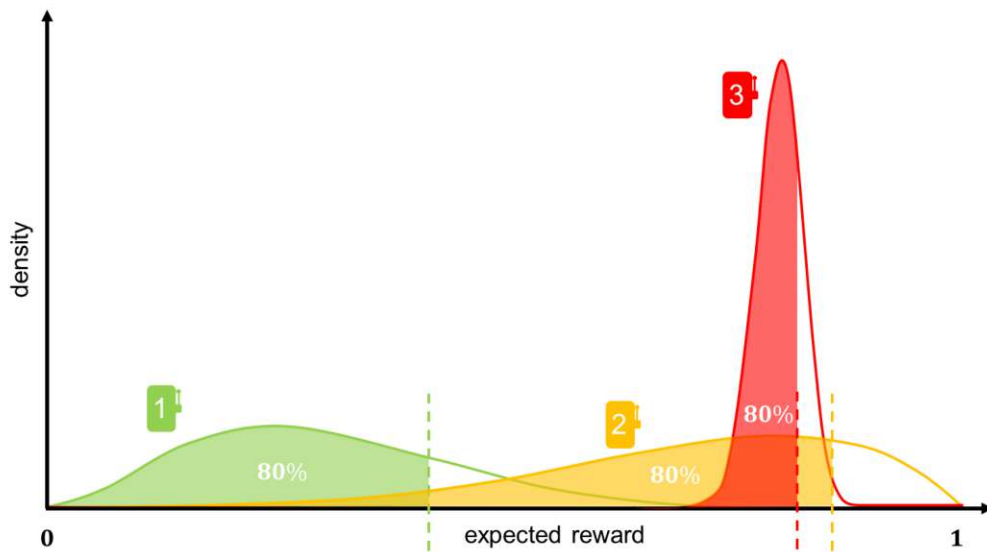


Figure 3.10: Density functions and upper confidence bounds for the three-armed Bernoulli bandit example at 80%, after [70].

There are different ways to derive upper confidence bounds. For example, if the rewards for actions are bounded, the Hoeffding inequality is a viable tool, because it provides a bound on the probability that a sum of bounded independent random variables deviates from its expected value by more than a certain amount. A more general approach equipped

to deal with unbounded variables is to model the uncertainty about the expected reward with a probability distribution that is progressively updated using the Bayes theorem.

The Bayes theorem lies at the heart of Bayesian statistics. Mathematically, it gives an alternate approach to calculate the conditional probability of two events  $A$  and  $B$ : let  $\mathbb{P}(A, B)$  denote the joint probability, that is the probability for  $A$  and  $B$  to occur, and let  $\mathbb{P}(A|B)$  denote the conditional probability for  $A$ , given  $B$ . Note that the conditional probability is generally not symmetrical:  $\mathbb{P}(A|B) \neq \mathbb{P}(B|A)$ . The conditional probability for  $A$ , given  $B$ , can be calculated using the joint probability:

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(A, B)}{\mathbb{P}(B)}$$

However,  $\mathbb{P}(A|B)$  can also be calculated using the reverse conditional probability  $\mathbb{P}(B|A)$ :

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(B|A) \mathbb{P}(A)}{\mathbb{P}(B)} \quad (3.1)$$

Analogously, the formula for the other conditional probability is:

$$\mathbb{P}(B|A) = \frac{\mathbb{P}(A|B) \mathbb{P}(B)}{\mathbb{P}(A)}$$

These formulae are useful in cases when the joint probability is difficult to calculate, or the reverse conditional probability is easily available [17]. Equation 3.1 is called either Bayesian rule or Bayes theorem, named after Thomas Bayes, who is credited with discovering the formula, and whose work was posthumously published in 1763 by a friend [11].

In short, the Bayes theorem gives a formula for calculating the conditional probability without the use of joint probabilities. The terms in Equation 3.1 can be interpreted in a particular way: generally,  $\mathbb{P}(A)$  is referred to as the *prior probability* (short: *prior*). It gives the probability of  $A$  before observing  $B$ .  $\mathbb{P}(A|B)$  is called the *posterior probability* (short: *posterior*) and describes the probability of  $A$  after observing  $B$ . The reverse conditional probability  $\mathbb{P}(B|A)$  is referred to as the *likelihood function*, or *likelihood*. It gives the probability of  $B$  after observing  $A$ , and  $\mathbb{P}(B)$  is called *evidence*. Equation 3.1 can thus be restated:

$$\text{Posterior} = \frac{\text{Likelihood} \times \text{Prior}}{\text{Evidence}}$$

The Bayes theorem finds use in many statistical testing scenarios [17]. A simple, yet topical example illustrates how the Bayesian framework can be applied to such a case, before explaining why it is suitable for the UCB strategy.

During the COVID-19 pandemic, the general population has come to learn that med-

ical tests and diagnostics are not 100% accurate. Depending on the procedure behind a COVID-19 test (PCR, antigen) there is a probability that, for example, a patient is infected with COVID-19, but the test is unable to detect it.<sup>4</sup> The Bayes theorem determines the probability of this scenario: the posterior  $\mathbb{P}(\text{infected}|\text{test negative})$  represents the probability of the investigated outcome, given by the likelihood  $\mathbb{P}(\text{test negative}|\text{infected})$  of the test being negative if the patient is infected, times the prior probability of the patient being infected, over the probability of the COVID-19 test to be negative:

$$\mathbb{P}(\text{infected}|\text{test negative}) = \frac{\mathbb{P}(\text{test negative}|\text{infected}) \mathbb{P}(\text{infected})}{\mathbb{P}(\text{test negative})} \quad (3.2)$$

A similar setup is useful for the UCB strategy. For any chosen arm  $a_t$ , let  $\epsilon$  denote the true expected reward of the arm (in contrast to the estimated reward  $e_t$ ). The independent rewards observed so far are denoted by  $r_1, r_2, \dots, r_{t-1}$ , and the prior at time  $t$  describing the distribution of  $\epsilon$  under these observations is represented by  $\mathbb{P}(\epsilon|r_1, \dots, r_{t-1})$ . The Bayes theorem expresses the probability distribution of  $\epsilon$  with respect to the previous rewards including  $r_t$ :

**Theorem 3.1** (Bayes Theorem).

$$\mathbb{P}(\epsilon|r_1, \dots, r_t) = \frac{\mathbb{P}(r_1, \dots, r_t|\epsilon) \mathbb{P}(\epsilon|r_1, \dots, r_{t-1})}{\mathbb{P}(r_1, \dots, r_t)}$$

Note that, compared to Equation 3.1 (or 3.2), the probability for event  $B$  is a joint probability containing all previously received rewards. Note also that the posterior in step  $t$  becomes the prior for step  $t + 1$ .

A Bayesian framework can be used to initialise and update probability distributions, and the UCB strategy relies on these distributions to guide the exploration. At each iteration  $t$ , the current distributions for all arms (i.e., the posteriors computed from the priors and rewards already observed) are used to compute upper confidence limits. The agent then chooses the arm  $a_t$  with the highest limit and receives a new reward  $r_t$  for this choice. Subsequently, the probability distribution relating to  $a_t$  is updated using the Bayes theorem. Figure 3.11 illustrates a single Bayesian update with a beta distribution prior and a Bernoulli observation in light of the formal definition of the Bayes theorem above.

Similar to the example of the greedy strategy in Figure 3.6, an example of ten steps of Bayesian updates for one arm (Bernoulli observations) and a beta prior can be seen in Figure 3.12 with assumed initial parameters  $\alpha = 1, \beta = 1$ , initial Bernoulli reward estimate  $e^0 = 0.5$ , and true Bernoulli parameter  $p = 0.7$ .

<sup>4</sup>This corresponds to a “false negative” scenario. A “false positive” scenario is also possible.

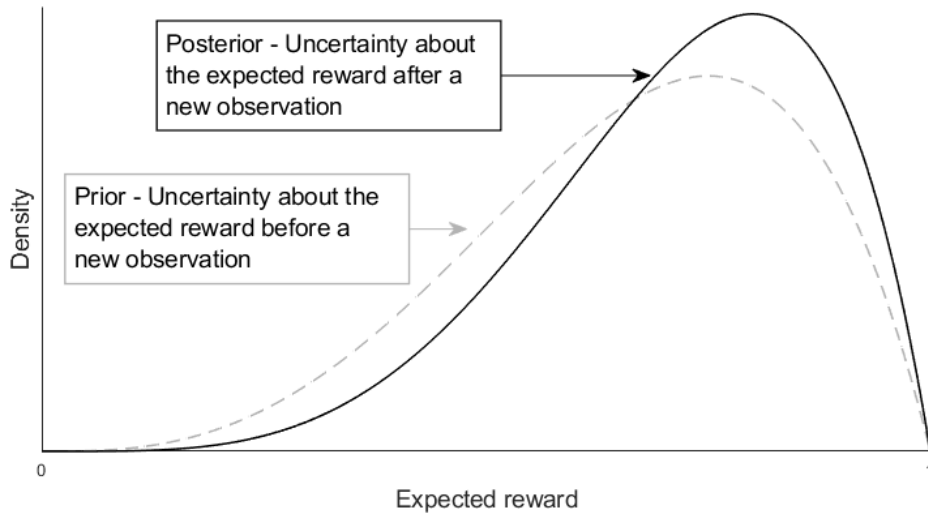


Figure 3.11: Bayesian update rule for beta distribution prior and Bernoulli observation, assuming success, after [70].

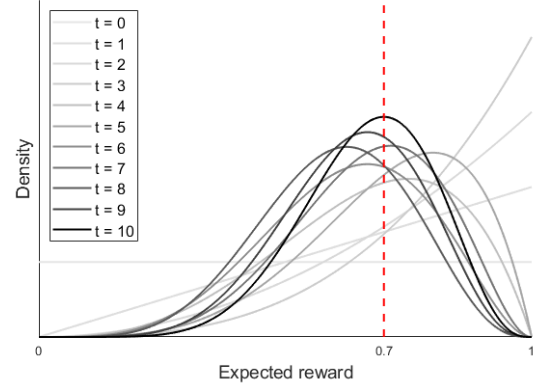
An important distinction must be made regarding the probability distributions manipulated by the algorithm: in Section 3.4.1, the Bernoulli distribution models the randomness of the reward. In the UCB strategy, the probability distributions model the agent’s uncertainty about the expected rewards (i.e., the value of the Bernoulli distribution parameter), and not the randomness of the rewards themselves. These distributions are expected to narrow with time if data is stationary (i.e., no new arms emerge), regardless of the fact that the true reward distributions can have large variance.

Despite the simple theoretical idea, several practical difficulties lie in both the Bayesian update and the upper confidence limit computation. Depending on the problem and the assumptions that are made, the Bayesian update can either be straightforward, for example in case of conjugate prior and likelihood (see below), or almost intractable due to complex computations for the denominator in the Bayesian theorem. Furthermore, once the posterior distribution is obtained it may not be easy to compute the upper confidence limit. An option is to sample from the distribution and estimate the upper bound using the generated samples.

The Bayesian update rule in a UCB strategy for the three-armed Bernoulli bandit looks as follows: for each arm  $\mathcal{A}_i$ , the prior knowledge about its expected reward (i.e., the estimate for  $p_i$ ) is represented by a beta distribution  $\mathbf{Beta}(\alpha_i, \beta_i)$ . In short, the prior is defined over the interval  $[0, 1]$ , depends on two parameters  $\alpha_i, \beta_i$ , and can be seen as the distribution for an unknown success probability, when a given number of successes and failures have been observed [16]. In fact, during each iteration step the distribution is updated as

$t$	Estimated Bernoulli parameter	Prior parameters	Reward observation	Posterior parameters
1	$e^0 = 0.5$	$\alpha = 1, \beta = 1$	$r_1 = 1$	$\alpha = 2, \beta = 1$
2	$e^1 = 0.75$	$\alpha = 2, \beta = 1$	$r_2 = 1$	$\alpha = 3, \beta = 1$
3	$e^2 = 0.8333$	$\alpha = 3, \beta = 1$	$r_3 = 1$	$\alpha = 4, \beta = 1$
4	$e^3 = 0.875$	$\alpha = 4, \beta = 1$	$r_4 = 0$	$\alpha = 4, \beta = 2$
5	$e^4 = 0.7$	$\alpha = 4, \beta = 2$	$r_5 = 1$	$\alpha = 5, \beta = 2$
6	$e^5 = 0.75$	$\alpha = 5, \beta = 2$	$r_6 = 0$	$\alpha = 5, \beta = 3$
7	$e^6 = 0.643$	$\alpha = 5, \beta = 3$	$r_7 = 1$	$\alpha = 6, \beta = 3$
8	$e^7 = 0.6875$	$\alpha = 6, \beta = 3$	$r_8 = 0$	$\alpha = 6, \beta = 4$
9	$e^8 = 0.6111$	$\alpha = 6, \beta = 4$	$r_9 = 1$	$\alpha = 7, \beta = 4$
10	$e^9 = 0.65$	$\alpha = 7, \beta = 4$	$r_{10} = 1$	$\alpha = 8, \beta = 4$
-	$e^{10} = 0.681$	-	-	-

(a) Possible parameter values and observed rewards.



(b) Evolution of the beta distribution considering Bayesian updates.

Figure 3.12: Example of 10 iterations of Bayesian updates assuming beta priors, posteriors, and Bernoulli observations.

$\text{Beta}(S_i(t), F_i(t))$ , with  $S_i(t)$  and  $F_i(t)$  denoting the cumulative successes and failures of arm  $A_i$  up until  $t$ , respectively [16]: the conjugacy property of the beta distribution states that when updating this prior with an observation which follows a Bernoulli distribution, the result is a new beta distribution whose parameters have been updated according to:

$$(\alpha_i, \beta_i) = \begin{cases} (\alpha_i + 1, \beta_i) & \text{for } r_t = 1 \\ (\alpha_i, \beta_i + 1) & \text{for } r_t = 0 \end{cases}$$

By initiating the parameters at  $t = 0$  as  $\alpha_i = 1$  and  $\beta_i = 1$ , the beta distribution turns into the uniform distribution on  $[0, 1]$ , thus making each value for  $p_i$  equally likely in the beginning [16]. In each subsequent iteration the agent chooses an arm based on the UCB strategy (and breaking ties randomly), the reward is observed, and the beta distribution parameters of the chosen arm are updated based on the reward.

In the appropriate setting, the UCB strategy is a powerful tool for implementing MAB and CMAB algorithms. A popular choice is the LinUCB algorithm, first described by Li et al. in [51]. Assuming a linear model for the expected reward (and neither Bernoulli nor beta distributions), a confidence interval can be computed efficiently in closed form. In case of a disjoint model, where parameters are not shared between different arms, the expected reward of arm  $A_i$  is a linear combination of the context vector and some unknown coefficient vector, whose values are determined either by ridge regression or least-squares estimation. The upper confidence limit is then given by a straightforward formula before updating the necessary auxiliary variables [67]. However, there are no known regret bounds for LinUCB yet [79], making any theoretical analysis difficult. The algorithm SupLinUCB



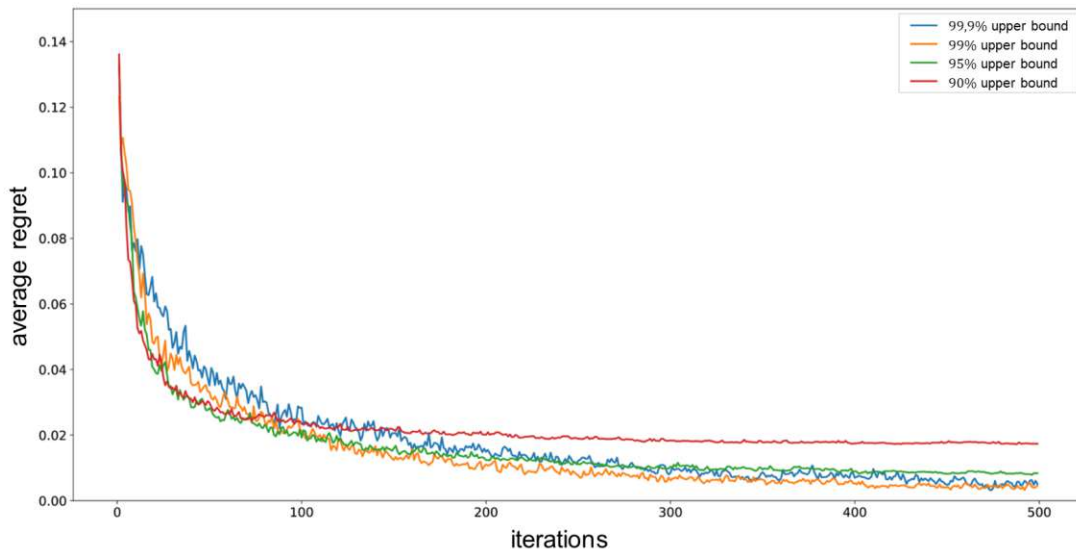


Figure 3.13: UCB strategy for different upper confidence limits, from [70].

is provided by Chu et al. [21], which calls the original BaseLinUCB as a subroutine and has a known regret bound. It is based on the work of Auer [7], who introduces the algorithms LinRel and SupLinRel (including regret bounds), which also use the UCB strategy.

### 3.4.3 The Thompson Sampling Strategy

Contrary to the  $\epsilon$ -greedy strategy in Section 3.4.1, the UCB strategy in Section 3.4.2 relies on a probabilistic representation of knowledge to guide the exploration process, and in each iteration the agent chooses the arm with the highest upper confidence limit. The *Thompson Sampling* (TS) strategy approaches exploration in a similar way: it also works with knowledge uncertainty; however, in the TS strategy, the agent chooses an arm randomly according to its probability to be the best. Thompson Sampling, also known as *Bayesian posterior sampling* [16], dates back to Thompson [80] in 1933, making it one of the oldest heuristics for MAB problems. It is also a family member of randomised probability matching algorithms [4].

In essence, TS works with the Bayesian framework introduced in Section 3.4.2: The TS algorithm maintains priors on the Bernoulli means  $e_i$  for each arm. Due to the conjugacy property (see Section 3.4.2), the beta distribution is the most convenient choice of priors for Bernoulli rewards. The probability density function of the beta distribution  $\mathbf{Beta}(\alpha, \beta)$  is

$$f(x; \alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1} (1-x)^{\beta-1},$$

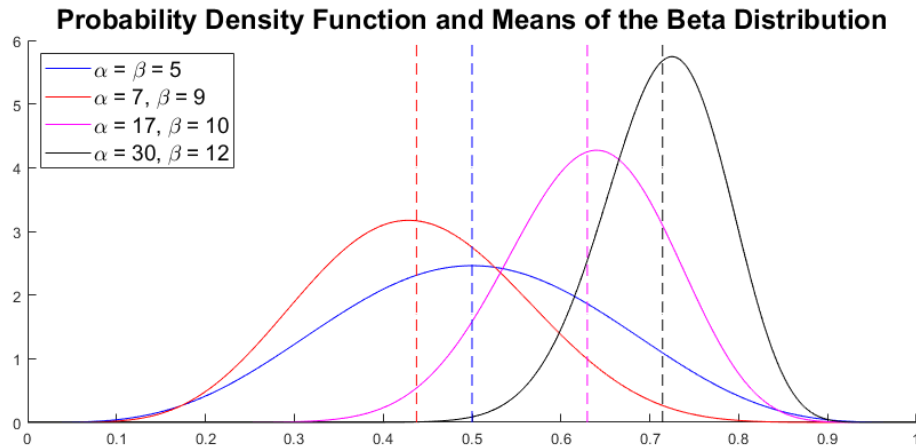


Figure 3.14: Probability density function and mean of the beta distribution, for different values of  $\alpha$ ,  $\beta$ .

and its mean is  $\alpha/(\alpha + \beta)$ . It is apparent from those formulae that the higher the values for  $\alpha$  and  $\beta$  are, the tighter the concentration of the density function around the mean becomes [2]. Figure 3.14 shows the probability density function and the mean of the beta distribution for different values of  $\alpha$ ,  $\beta$ .

Instead of computing a confidence interval, the selection of an arm is defined by each arm's probability to be the best (i.e., to have the highest expected reward). Computing this probability for all arms can be difficult; however, it suffices to draw a random variable from the posterior distribution of each arm, and the arm with the largest sample value is chosen by the agent [16, 18]. This process is strictly equivalent to directly sampling an arm according to its probability to have the highest expected reward, so these probabilities do not have to be computed explicitly, making TS an efficient strategy in terms of exploration (i.e., actions with high probabilities to be optimal are chosen frequently) and exploitation (i.e., actions with little chance to be optimal are rarely chosen).

In case of Bernoulli bandits, the TS algorithm initially assumes prior distributions  $\mathbf{Beta}(\alpha_i, \beta_i)$  for each arm with initial parameters  $\alpha_i, \beta_i$ . A practical approach is to set  $\alpha_i = \beta_i = 1$ , which yields the uniform distribution on the unit interval, so all values of  $[0, 1]$  are equally likely to be the estimated success probability  $e_i$  for  $\mathbf{Bernoulli}(e_i)$ . At time  $t$ , random samples  $e_i$  are drawn from  $\mathbf{Beta}(\alpha_i + S_i, \beta_i + F_i)$ , where  $S_i$  and  $F_i$  denote the cumulative successes and failures until  $t$ , respectively. The agent then chooses the arm with largest sample value, observes the reward, and subsequently updates  $S_i$ , or  $F_i$ , according to the outcome of the trial. Algorithm 6 formalises the TS strategy for the  $k$ -armed Bernoulli bandit with a beta prior [18], thus solving the example problem from Sections 3.4.1 and

**Algorithm 6:** Thompson Sampling Strategy

---

**Input:** arms  $\mathcal{A} = \{1, \dots, k\}$ , initial prior parameters  $\alpha_i, \beta_i$ , auxiliaries  $S_i = F_i = 0$

```

1 for  $t = 1, 2, \dots$  do
2   for  $i = 1, \dots, k$  do
3     | Draw  $e_i$  according to Beta( $\alpha_i + S_i, \beta_i + F_i$ )
4   end
5   choose arm  $a_t = j = \operatorname{argmax}_i e_i$ 
6   receive reward  $r_t \sim \mathbf{Bernoulli}(e_j)$ 
7   if  $r_t = 1$  then
8     |  $S_j = S_j + 1$ 
9   else
10    |  $F_j = F_j + 1$ 
11   end
12 end

```

---

3.4.2: assuming a three-armed Bernoulli bandit, the uncertainty about the unknown values of success probabilities  $p_i$  are modelled with the beta distribution, which is updated in each iteration with Bernoulli observations using the Bayes theorem. In each time step, random samples  $\theta_i$  are drawn from the beta distributions, serving as estimates for the values of  $p_i, i \in \{1, 2, 3\}$  (in the Bernoulli case, the  $p_i$ 's are equal to the expected rewards). The agent then chooses the arm with the highest sample value, observes the reward, and updates the beta distribution parameters of the chosen arm.

Figure 3.15 shows two possible sampling outcomes for fixed beta distribution parameter values  $\alpha_1, \beta_1, \alpha_2, \beta_2, \alpha_3, \beta_3$ . In sampling 1, the highest sampling value is provided by the beta distribution of  $A_3$ , but its probability to be drawn is low, considering its density function. Sampling 2 results in the agent picking  $A_2$  over  $A_3$ , because the sample value is higher, even though its probability to be drawn is lower and the sample value for  $A_3$  is more likely to be the correct estimate for  $p_3$ .

Chapelle and Li [18] show TS to be competitive with, or better than, other methods such as UCB algorithms during an empirical evaluation of the TS strategy, proving that TS is more robust against delayed, or batched, feedback in applications for display advertising and news article recommendation modelled by a CMAB problem. In a thorough comparison of TS with the best known versions of UCB algorithms, Kaufmann et al. [43] show that TS has the lowest regret in the long run. However, the theoretical understanding of TS is still limited. For the basic version of the MAB problem (i.e., omitting context), significant progress was made [2, 3, 43], resulting in optimal regret bounds on the expected regret. However, the CMAB problem does not seem easily amenable to the techniques used so far for analysing TS for the MAB problem. Still, Agrawal and Goyal [4] provide the first

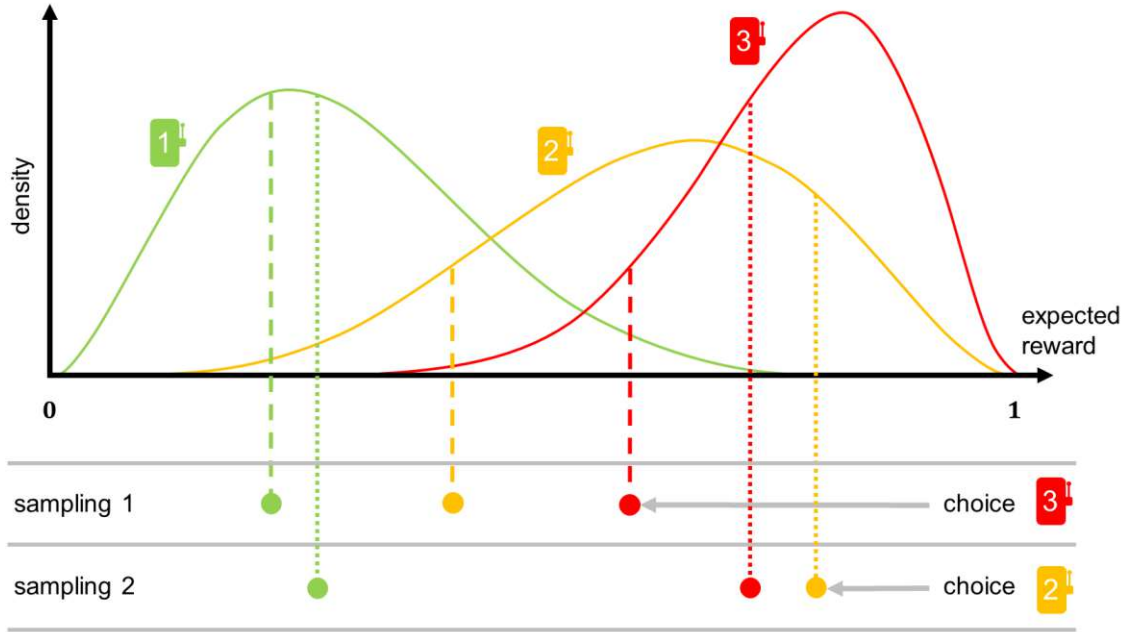


Figure 3.15: Illustration of the Thompson Sampling approach in a three-armed Bernoulli bandit, after [70].

theoretical guarantees for a TS algorithm equipped to solve the CMAB problem, which only hold under assumptions not easily met in a realistic setting.

With regard to incorporating a  $d$ -dimensional context vector  $x(t) \in \mathbb{R}^d$  into the algorithm, the contextual bandit setting with *linear payoff functions* is a commonly used approach [4, 16, 79]. The expected reward is assumed to be a linear function of the context:

$$\mu_t = \mathbb{E}[r_i(t) | x(t)] = \gamma_i^T x(t)$$

The weight vector  $\gamma_i$  associated with  $A_i$  is unknown and to be determined by the TS algorithm. Instead of assuming a Bernoulli distribution, the algorithm draws its  $d$ -dimensional samples  $\tilde{\gamma}_i$  from multivariate Gauss distributions  $\mathbf{N}(\hat{\gamma}_i(t), \sigma_i(t))$  with mean  $\hat{\gamma}_i(t)$  and variance  $\sigma_i(t)$  computed from  $x(t)$ :

$$\begin{aligned} \sigma_i(t) &= v^2 B_i(t) \\ B_i(t) &= I_d + \sum_{\tau=1}^{t-1} x(\tau)x(\tau)^T \\ v &= R \sqrt{\frac{24}{\epsilon} d \ln \frac{1}{\delta}} \end{aligned}$$

with fixed parameters  $R > 0$ ,  $\epsilon, \delta \in (0, 1]$ , and  $\hat{\gamma}_i = B(t)^{-1} \left( \sum_{\tau=1}^{t-1} x(\tau)r(\tau)^T \right)$  [16].

The arm maximising the expected reward  $x(t)^T \tilde{\gamma}_i$  is chosen, its reward  $r(t)$  is observed, and the distribution parameters are updated [16].

Algorithm 7 illustrates TS for the CMAB problem [4, 16].  $\mathbf{I}$  denotes the unit matrix, and  $\mathbf{0}$  the zero vector of dimension  $d$  (the dimension of the context vector  $x(t)$ ). Note that, in application, the reward is not generated by a distribution, but observed based on the activity of the client. However, a distribution behind the reward is assumed, making the bandit problem *semi-adversarial* or *semi-stochastic* due to the hypothesis that the context vector in each time step  $t$  is not generated randomly, but adversarially. Even though the conjugate prior for a likelihood given by the multivariate Gauss distribution is again a multivariate Gauss distribution [4], Bernoulli rewards  $r(t) \in \{0, 1\}$  may be assumed, which translates into the outcome (i.e., success or failure) of a trial.

---

**Algorithm 7:** Thompson Sampling for the CMAB Problem

---

**Input:** arms  $\mathcal{A} = \{1, \dots, k\}$ , initial parameters  $B_i = \mathbf{I}$ ,  $\hat{\gamma}_i = g_i = \mathbf{0}$ , auxiliary  $v$

```

1 for  $t = 1, 2, \dots$  do
2   observe context  $x(t)$ 
3   for  $i = 1, \dots, k$  do
4     sample  $\tilde{\gamma}_i$  from  $\mathbf{N}(\hat{\gamma}_i, v^2 B_i^{-1})$ 
5   end
6   choose arm  $j(t) = \operatorname{argmax}_i x(t)^T \tilde{\gamma}_i$ 
7   receive reward  $r(t)$ 
8   if  $r_t = 1$  then
9      $B_j = B_j + x(t)x(t)^T$ 
10     $g_j = g_j + x(t)r(t)$ 
11     $\hat{\gamma}_j = B_j^{-1}g_j$ 
12  end
13 end

```

---

The TS approach holds potential for modification into more sophisticated algorithms, catering to different environmental circumstances. A situation that can occur easily when working with recorded data is restricted context, for example due to technical failure. A first step has been taken by Bouneffouf et al. [16], introducing the *Thompson Sampling with restricted context* (TSRC) algorithm that exclusively uses restricted context (i.e., a sparse context vector with a fixed number  $n < d$  or percentage of available contextual data to be regarded), see Section 4.5. The same paper also provides a modified TSRC algorithm called Window TSRC (WTSRC) for a non-stationary environment, where the distribution determining the sparse vector can change over time, so instead of converging to a fixed set of restricted context, the agent keeps looking for the optimal set. Another possible extension

is provided by Chapelle and Li [18], called *optimistic Thompson Sampling*, where, similar to the optimistic approach of the UCB strategy, the modified score of the expected reward is never smaller than the mean [55].

TS presents a very effective heuristic for addressing the exploration-exploitation trade-off, which has been proven by extensive experimental evaluation from Chapelle and Li [18]. In its simplest form (i.e., with Bernoulli prior and beta likelihood) it does not have any parameters to tune. TS is easy to implement, and since it is a randomised algorithm, it is robust towards delayed feedback [43]. However, contrary to the UCB strategy, further work has to be done in terms of theoretical analysis, including high probability regret bounds for the MAB problem, and regret bounds for the CMAB problem [4]. Regret bounds make it possible to rudimentarily compare algorithms and strategies independent from the problem, and depend at least on the time horizon  $T$  of the simulation, the dimension  $d$  of the context vector (in case context is regarded), and the probability  $\delta \in (0, 1)$  with which the regret bound holds.

#### In Conclusion

If the knowledge about the reward generating process can be captured by a set of random variables, then the UCB strategy provides a useful tool to deal with the exploration-exploitation trade-off. The estimated means of the random variables reflect the current knowledge of the algorithm in a condensed form, guiding further exploitation, and the widths of the confidence bounds reflect the uncertainty of the algorithm's knowledge, thus guiding further exploration [7]. Since there is no need to explicitly compute probabilities, TS is equipped to deal with a user-interfaced application better than the UCB approach.

The  $\epsilon$ -greedy strategy is similarly ill-equipped to model CMAB problems in a real-life setting. Contrary to both the UCB and the TS approach, the algorithms of the  $\epsilon$ -greedy strategy solve fully stochastic bandit problems, where the context vectors and reward values are sampled from an underlying distribution, which is further from the realistic setting than semi-stochastic algorithms [79]. Also, compared to the uncertainty-based methods, the  $\epsilon$ -greedy strategy is less dynamic in its handling of the exploration-exploitation trade-off.

It is for these reasons that the TS strategy is adopted as the algorithm solving the CMAB problem in this thesis. Chapter 4 classifies the TS approach in light of sequential decision making and reinforcement learning, providing the theoretical background to the implementation of a TS algorithm for a model problem and the subsequent simulation in Chapters 5 and 6. To motivate the model problem and introduce the reader to progress already made, Section 3.5 gives an overview of instances where MAB and CMAB algorithms have already been implemented into mHealth applications.

### 3.5 Applications of Multi-Armed Bandits and Contextual Multi-Armed Bandits in Mobile Health

Section 3.3 illustrates how the CMAB problem setting is a practical tool for implementing AI designs in mHealth interventions. In the past decade significant progress has been made in creating functional applications that work in a MAB or CMAB setting [28, 52, 61, 66, 92, 93], adapting to a client’s intervention preferences in real time. This section provides a short overview of several reinforcement learning (RL) mobile health studies, in which RL methods are applied in an AI setting. RL is formally introduced and discussed at length in Section 4.3. In the meantime, RL methods may be viewed as methods similar to the CMAB setting, where learning is achieved by feedback only. Note that the examples mostly differ from those in Section 2.3, where AI and JITAI concepts in mHealth are discussed, because in spite of the natural translation between the adaptive intervention design and the CMAB approach, AI in mHealth can be achieved in different ways.

Chapter 3 repeatedly mentions the dilemma of the exploration-exploitation trade-off and how it can be solved by MAB (and CMAB) algorithms. Balancing this trade-off is a fundamental aspect of RL, and Liao et al. [52] provide a list of challenges that need to be addressed before RL can be usefully deployed to adapt and optimise mHealth interventions:

(C1) *The RL algorithm must adjust for longer term effects of current actions.*

Due to the construction of the MAB problem, interventions are programmed to have a positive effect on the immediate reward, implying that maximising the reward at each decision point will lead to the maximal cumulative reward. This way, it is likely that the intervention design will produce negative impact on the future rewards due to engagement fatigue on the client’s side (for example, interventions are offered too frequently, making the program a burden). An algorithm working in an mHealth recommender system should thus be equipped to take this process into account.

(C2) *The RL algorithm should learn quickly and accommodate noisy data.*

Depending on the implementation, most RL algorithms require the agent to interact many times with the environment before performing well. This is impractical in mHealth applications, as users can disengage quickly. Also, because mHealth interventions are provided in uncontrolled, in situ, complex environments, both context information as well as reward can be noisy. High noise settings typically require more interactions with the environment to identify optimal choices, implying a trade-off between bias and variance. This needs to be meticulously considered when designing an RL algorithm.

(C3) *The RL algorithm should accommodate some model mis-specifications and non-stationarity.*

Due to the complexity of the context space and unobserved aspects of the current context (like engagement fatigue), the mapping from context to reward is likely to exhibit non-stationarity over long periods of time, which makes it necessary to address this aspect during the design process of the RL algorithm.

(C4) *The RL algorithm should select actions so that after the study is over, secondary data analyses are feasible.*

This is particularly important in case of experimental trials involving clinical populations, allowing multiple stakeholders to analyse the resulting data in a variety of ways.

Four instances of MAB and CMAB approaches in mHealth are presented in the remainder of this section. Table 3.2 gives a short overview before providing a more in-depth discussion. Note that implemented algorithms do not simply follow either of the three strategies in Section 3.4, but combine different aspects and approaches, made necessary by dealing with, amongst others, the challenges listed above.

Application	Description
CalFit	Fitness app on the iOS platform that suggests a daily step goal calculated by the Behaviour Analytics Algorithm, an RL algorithm from a pre-existing weight loss model [93]
HeartSteps	Mobile phone app that tracks the physical activity of clients with stage 1 hypertension blood pressure during a 90-day clinical trial via a CMAB algorithm that uses TS [52]
MyBehavior	Smartphone app that delivers personalised interventions for promoting physical activity and dietary health as a JITAI, via a MAB algorithm [66]
PopTherapy	Mobile phone app that helps clients cope with stress and depression-related symptoms, based on Cognitive Behavioural Theory technology, via a CMAB algorithm [61]

Table 3.2: Short description of MAB- and CMAB-based applications in mHealth.

For PopTherapy, Paredes et al. [61] employ a CMAB algorithm combined with a UCB approach to select an intervention amongst ten types of stress management strategies when the participant requests an intervention in the mobile app, with the goal of maximising stress reduction.

In MyBehavior [66], the MAB algorithm EXP3 is used to select interventions. EXP3 stands for “exponential-weight algorithm for exploration and exploitation” and is first de-



scribed by Auer et al. [9]. It is equipped to deal with fully adversarial bandits because it does not assume any distribution behind the reward-generating scheme whatsoever, making the algorithm a strong contender for the mHealth setting. An extension of the MyBehaviour app is currently being developed, called MyPersonalCoach, which uses the CMAB approach, including context in the previous MyBehaviour setting. Notifications will be sent to a client's smartwatch just-in-time, encouraging habit building [65].

The app CalFit [93] works with an RL system that uses the client's historical daily step count data at the end of each week and estimates a dynamical system for the daily step count, using it to infer the optimal daily step goals for the next seven days, with the goal of maximising the the minimal number of step counts taken in the next week.

These three applications only work with data collected during an initiation phase, but additional data (e.g., from pilot studies) is not regarded when looking for an initial estimation. With regard to challenge C2, the use of pooled data from previous clients can improve the learning process, especially at the beginning of the intervention regime. Furthermore, the RL algorithms of PopTherapy and CalFit require knowledge of the correct model for the reward function, which can be assumed to be fuzzy due to the dimension and complexity of the context space, and potential non-stationarity mentioned in challenge C3. Also, both algorithms base their arm selection process deterministically on history [52].

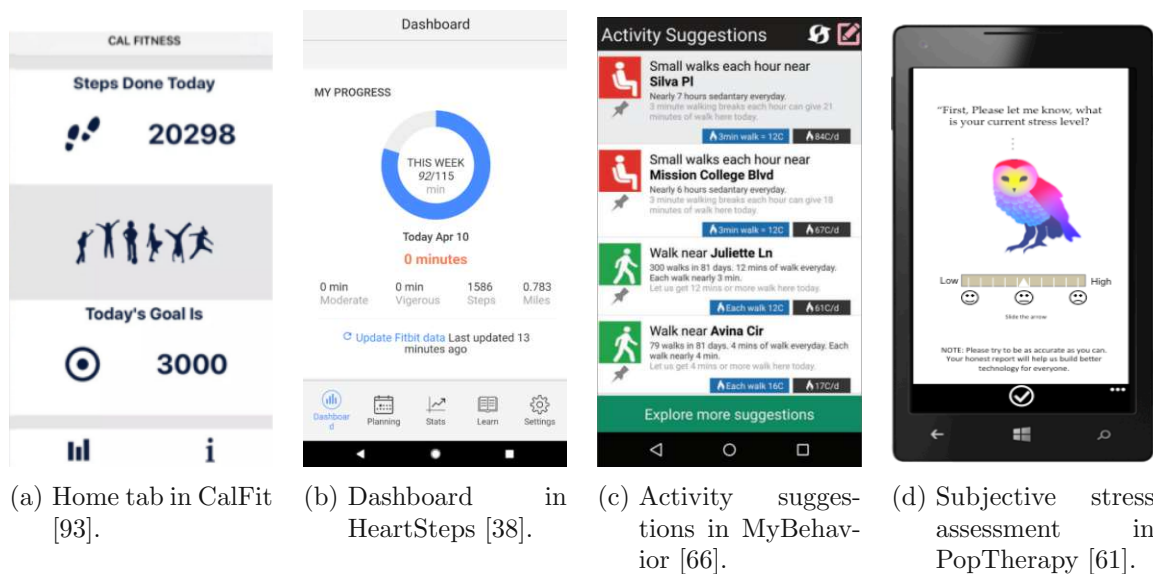


Figure 3.16: Screenshots of different user interfaces in the mobile phone apps discussed in Section 3.5.

Amongst these three apps, only the algorithm used in CalFit attempts to optimise reward over a time period longer than the immediate time step. There is a bias-variance trade-off

when deciding how far into the future the RL algorithm should attempt to maximise its reward. Only focussing on optimising the immediate reward might lead to offering too many interventions (see challenge C1), since treatment tends to have a positive effect on the immediate reward, and a negative effect on future rewards, leading to overall poor performance akin to bias. However, action selection probabilities close to 0 or 1 cause high variance in batch data analysis that uses importance weights, complicating challenge C4 [52].

The design of HeartSteps [52] stands out in comparison to the other algorithms in its elaborateness because the designers of the algorithm address each challenge C1-C4 separately. Having created and tested HeartSteps version 1 during a prior clinical trial, several issues were adjusted, for example a dosage variable was introduced in order to weigh different contexts, thus improving future rewards and meeting challenge C1. It can be concluded that the mobile phone app HeartSteps is currently considered state of the art in adaptive mHealth recommender systems technology. Nevertheless, future work has been proposed: the treatment strategy is not fully personalised (i.e., separately learned for each client), and more sophisticated measures of intervention engagement and fatigue can be used to approximate the effect of delayed reward, while also responding more rapidly in order to prevent premature disengagement from the recommender system.

## 4 A Choice of Thompson Sampling Algorithm

This chapter re-introduces Thompson sampling as a *reinforcement learning* algorithm as a solution to sequential decision making problems. The concept of restricted context in the CMAB problem is discussed, which is part of the model problem in Chapter 5, and subsequently, the TS algorithm used for the implementation and simulation of the model problem is presented.

### 4.1 Markov Decision Processes

The content in this section is cited from Van Otterlo and Wiering [83], if not otherwise stated.

There are several classes of algorithms available for solving the problem of sequential decision making. Generally, such problems are formulated as *Markov decision processes* (MDPs), which describe systems where the environment is represented by a set of states, and actions can be performed to control those states. The goal is to regulate the system in such a way that some pre-defined performance criterion is maximised. Van Otterlo and Wiering state that MDPs have become the de facto standard for learning sequential decision making, like in the CMAB problem.

Formally, the MDP framework consists of four elements:

- States
- Actions
- Transition between states
- A reward function

The states represent the basis of the environment of an MDP, and the number of states (i.e., the cardinality of the set of states  $S$ ) is generally considered to be finite. A state is a unique characterisation of all relevant information concerning the circumstances of the problem to be modelled. For example, the location of an element within a pre-defined

space can be represented by two or three dimensions in a state vector. The set of actions  $A$  is also assumed to be finite, and actions are called upon to control the system in all its possible states (i.e., to move from one location to the other).

By applying action  $a \in A$  in state  $s \in S$ , the system transitions from  $s$  to a new state  $s' \in S$ , based on a probability distribution over the set of possible transitions. The probability of switching to state  $s'$  after performing action  $a$  in state  $s$  is defined by the *transition function*  $T$ :

$$T : S \times A \times S \rightarrow [0, 1]$$

The transition function  $T$  is a probability distribution over all the possible next states: for fixed  $(a, s) \in (A, S)$  it must hold that

$$\sum_{s' \in S} T(s, a, s') = 1,$$

as well as  $T(s, a, s') \geq 0$  and  $T(s, a, s') \leq 1$  for all  $a \in A, s, s' \in S$ . The *reward function* specifies rewards for performing actions, and is generally defined as

$$R : S \times A \rightarrow \mathbb{R}.$$

In an MDP, the reward function implicitly specifies the goal of the learning process and gives directions as to which way the system ought to be controlled. Putting all four elements together results in the formal definition of an MDP, see Definition 4.1.

**Definition 4.1.** *A Markov decision process is a tuple  $(S, A, T, R)$  in which  $S$  is a finite set of states,  $A$  a finite set of actions,  $T$  a transition function defined as  $T : S \times A \times S \rightarrow [0, 1]$ , and  $R$  a reward function defined as  $R : S \times A \rightarrow \mathbb{R}$ .*

Note that this definition can be altered and extended within the MDP framework, which is exceedingly flexible. For example, the reward function can also be defined as  $R : S \rightarrow \mathbb{R}$  or  $R : S \times A \times S \rightarrow \mathbb{R}$ .

In order to solve an MDP problem, key aspects of the solution process need to be defined depending on the chosen method. Solving an MDP implies finding the optimal action selection process, which is roughly described by three components: a *policy*, an *optimality criterion*, and a *value function*.

Given an MDP  $(S, A, T, R)$ , a policy  $\pi$  is a function that outputs an action  $a \in A$  for each state  $s \in S$ :

$$\pi : S \rightarrow A$$

The policy represents the decision-making entity (i.e., the agent): from an initial state  $s_0$ , the policy  $\pi$  suggests an action  $a_0 = \pi(s_0)$  which is subsequently performed. A transition

into state  $s_1$  is made based on the transition function  $T$  (i.e., with probability  $T(s_0, a_0, s_1)$ ), and a reward  $r_0 = R(s_0, a_0)$  is received. In a learning setting, the policy then updates the parameters upon which the action decision is based before choosing another action in  $s_1$ . Figure 4.1 gives a graphical representation of this process. It demonstrates how the policy controls the environment modelled as an MDP.

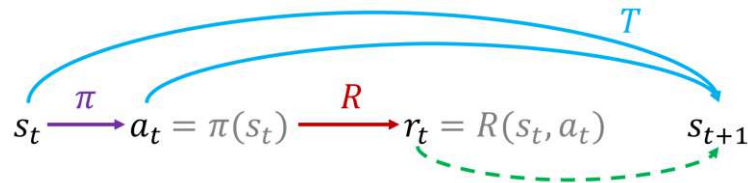


Figure 4.1: Illustration of the direct (solid line) and indirect (dashed line) influences on the transition into a new state via a policy.

The decisions made through the policy are based on the optimality criterion. In general, the quantification of optimality is related to gathering reward, like in the CMAB setting, see Chapter 3. The link between optimality criteria and policies is provided by the value function. It represents an estimation of how good it is to perform a certain action in state  $s \in S$ , and the notion of “how good” is expressed in terms of the optimality criterion.

Despite obvious similarities to the CMAB setting, the next section clarifies why CMABs differ from MDPs, while also justifying the use of typical MDP solution methods for the CMAB problem.

## 4.2 Contextual Multi-Armed Bandits versus Markov Decision Processes

Despite the MDP framework being extensive, it does not directly cater to the CMAB concept, because it relies on deterministic reward functions and transition functions between states. By adjusting formal definitions, the CMAB problem can be interpreted as an MDP: the states in the MDP play the role of context vectors in the CMAB problem, actions correspond to the arms of the bandit, and the performance criterion to be maximised is the immediate reward.

However, some controversy remains. For example, in an MDP setting the algorithm learns values of states, or state-action combinations, while the CMAB algorithm estimates the values of actions<sup>1</sup> [77]. Furthermore, the term “Markov” or “Markovian” defines a distinct property, see Definition 4.2.

<sup>1</sup>The value of an action in the CMAB setting refers to the expected value of the reward for performing an action, i.e. pulling an arm.

**Definition 4.2.** A system being controlled is called Markovian if the result of an action does not depend on the previous actions and visited states (history), but only on the current state:

$$\mathbb{P}(s_{t+1} \mid s_t, a_t, s_{t-1}, a_{t-1}, \dots) = \mathbb{P}(s_{t+1} \mid s_t, a_t)$$

Definition 4.2 implies that the CMAB problem introduced in Chapter 3 is strictly speaking not Markovian, because the probability distribution for each arm is updated once the arm has been chosen, and thus depends implicitly on the history of that arm. Literature on this subject [77, 83] does not elaborate on whether the CMAB framework should be considered Markovian. Still, Sutton and Barto [77] introduce the bandit setting before moving on to the general approach of MDPs. Note that the Bayes theorem (Theorem 3.1), on which the TS algorithm is built, explicitly depends on former values for the reward, and thus on history, calling for a non-Markovian environment. Nevertheless, there are lines of research that employ TS to solve MDP problems [32, 60].

The MDP concept can be adapted into more generalised approaches: Hallak et al. [35] introduce *contextual Markov decision processes*, where stationary context impacts the decision making process. For example, the temporal behaviour of blood sugar levels in diabetes patients is partially influenced by their age and biological sex. These context variables do not change within each measurement, and can be handled accordingly within the contextual MDP framework. Another extension of the Markov concept in MDPs is to consider *j-Markov models*, or *Markov models of the j-th order* [20], where the probability distribution of arm  $a$  at  $t$  depends on the past  $j$  instances of arm choices. However, the bandit framework requires the variable  $j$  to be dynamic and differ between arms. In cases where the context variables may vary from measurement to measurement, Hallak et al. [35] state that the standard approach is to incorporate them into the state space, thus creating a larger MDP, and furthermore that the contextual MDP concept is closely related to the CMAB framework, still implying a distinction between the two.

This thesis adopts the viewpoint that the CMAB framework is closely related to, and slightly overlaps with, the MDP framework, and the class of algorithms chosen to solve the CMAB problem can also solve specific MDP problems. Figure 4.2 visualises this relationship. In any case, the CMAB framework requires learning-based solution methods. The learning paradigm has an important advantage over direct programming (e.g., for robot control) or search and planning methods (e.g., the chess program Deep Blue): it allows for a model-free approach, relieving the designer of the system from the burden of having to make all decisions in the design phase, as the system can cope with uncertainty, dynamic environments, and goals specified in terms of reward measures. The problem is solved for every state, in contrast to merely planning transitions from one state to another, and optimal actions can be determined by interacting with the environment.

### 4.3 Thompson Sampling as a Reinforcement Learning Algorithm

Reinforcement learning refers to a general class of algorithms in the field of machine learning that allows an agent to learn how to behave in an environment where the only feedback is a reward signal, and the goal of the agent is to perform actions that maximise this reward signal in the long run (instead of trying to find hidden structures). Therefore, RL is distinguished from the other two machine learning paradigms, *supervised learning* (i.e., learning from a training set of labelled examples provided by a knowledgeable external supervisor), and *unsupervised learning* (i.e., finding structures hidden in collections of unlabelled data), because it uniquely deals with the exploration-exploitation trade-off, as the issue of balancing exploration and exploitation generally does not arise in supervised or unsupervised learning [77].

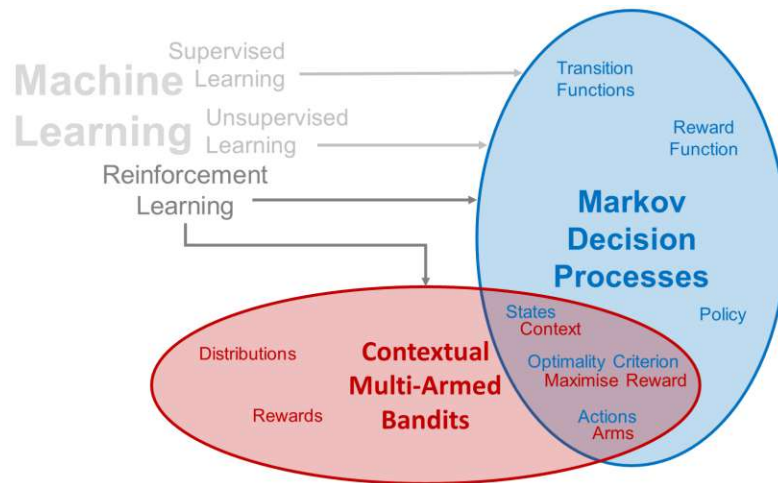


Figure 4.2: Graphical representation of the relationship between CMABs and MDPs, and the different learning paradigms solving them.

In RL, the concept of a *plan*<sup>2</sup> is extended to the notion of a policy, which maps each state to its optimal action based on some measure of optimality. RL algorithms can obtain an optimal policy when models of the environment are not available, adding a focus on approximation and incomplete information to a sequential decision making problem. The absence of a model generates the need to sample from the environment in order to gather statistical knowledge about the unknown model. Historically, RL is part of a decades-long trend within artificial intelligence and machine learning towards greater integration with mathematical domains like statistics and optimisation. Of all forms of machine learning, RL is the closest thing to “natural learning” performed by animals and humans, and many

<sup>2</sup>In terms of artificial intelligence planning, a plan describes a series of actions from a starting state to a goal state.

of the core algorithms of RL were originally inspired by biological learning systems [77].

The main elements of an RL system are the agent and the environment. Figure 4.3 shows how the interaction between these two entities can take place. The distinction between the agent and the environment is generally based on control: everything the agent cannot control is considered part of the environment [77]. At each time instance, the agent chooses an action based on its current state, and the perceptions the agent receives from the environment are a new state and the reward signal.

environment	You are in state 65. You have 4 possible actions.
agent	I take action 2.
environment	You have received a reward of 7 units. You are now in state 15. You have 2 possible actions.
agent	I take action 1.
environment	You have received a reward of -4 units. You are now in state 65. You have 4 possible actions.
agent	I take action 2.
environment	You have received a reward of 5 units. You are now in state 44. You have 5 possible actions.
...	...

Figure 4.3: Example of an interaction between agent and environment from an RL perspective, after [83].

Many of the above aspects, like sampling from the environment and receiving a reward signal, already identify TS as an RL algorithm. For a more detailed classification, observe the four sub-elements of RL algorithms (beyond the agent and the environment), according to Sutton and Barto [77]:

- A policy, which defines the learning agent’s way of behaving at a given time.
- A reward signal, which defines the goal in an RL problem (i.e., what is optimal in an immediate sense).
- A *value function*, which indicates what is optimal in the long run.
- (Optionally) a model for the environment.

Note that TS does not require a model for the environment, and neither does it aim to learn one. It is possible to use RL algorithms in model-based settings, but this approach corresponds to planning and not decision making via a policy. In case of CMABs, learning an environment is nonsensical because the context is adversarial, which means that no pattern, or function, or distribution is assumed behind its generation, thus no parameter



values can be learned. Instead, the chosen actions are rewarded, rather than entries into certain states.

The value function depicts the long-term desirability of actions, contrary to rewards, which determine the immediate intrinsic desirability of actions. In this sense, rewards are primary, whereas values, as predictions of reward, are secondary. Generally, the aim is to find actions that promise high values, not high rewards, because these actions obtain the greatest amount of reward in the long run. It is more difficult to determine values than to determine rewards, since rewards are obtained directly from the environment, and values must be estimated (and repeatedly re-estimated) from the sequence of observations an agent makes over time [77]. In the TS algorithm for the CMAB problem, see Algorithm 7, the value function is the function for the expected reward for arm  $i$ :

$$\mathbb{E}(r_i(t) | x(t)) = \tilde{\gamma}_i^T x(t)$$

Here,  $\tilde{\gamma}_i$  is a coefficient vector drawn from the estimated distribution for  $i$ . The reward signal is the feedback immediately received after performing an action, which TS assumes to be either 0 or 1:

$$r(t) = \begin{cases} 1 & \text{if action is successful} \\ 0 & \text{if action failed} \end{cases}$$

To understand the policy in Algorithm 7, observe an overview of the steps taken during each iteration  $t$ :

1. Observe context  $x(t)$
2. Sample  $\tilde{\gamma}_i$  from the distribution for  $i$ , for all arms
3. Choose arm  $j$  maximising  $\tilde{\gamma}_i^T x(t)$
4. Receive reward  $r(t)$
5. (If necessary) update distribution parameters

The policy is represented by Steps 2, 3 and, if performed, 5. These steps define how to execute the arm selection and how the selection strategy is kept up-to-date. Policies are often described as a mapping from states (i.e., context) to actions. Due to the adversarial nature of context, a more constructive approach is to view the policy as the decision-making rules for the agent, which work based on the feedback from actions after observing a context vector.

Now that the TS algorithm is characterised within the RL paradigm, the description of, or reference to, certain RL aspects can be made in the appropriate terms.

The following section discusses the challenges of restricted context in CMABs, and introduces the mathematical framework to handle the restricted context situation in CMAB algorithms.

## 4.4 Contextual Bandits with Restricted Context

The *contextual bandit with restricted context* (CBRC) describes a formulation of the CMAB model, where only a limited number of features (i.e., context elements) can be observed by the agent at each iteration: instead of accessing  $x(t) \in \mathbb{R}^d$ , the agent sees a sparse vector  $x_n(t) \in \mathbb{R}^d$  of assignments to only  $n \leq d$  features. The CBRC framework is motivated by different problems arising in clinical trials, or JITAI recommender systems, with regard to collecting incomplete data [16]. This section aims to explain why restricted context ought to be a consideration in CMAB-based JITAI applications, and how to incorporate restricted context into the CMAB setting.

### 4.4.1 Handling Restricted Context in a Contextual Multi-Armed Bandit

In sequential decision making, the learning algorithm must choose amongst several actions at each time point. In the CMAB setting, those actions are associated with feature information, or context, and the reward feedback is limited to the chosen option [16]. For example, in clinical trials, the context is represented by a patient’s medical record (e.g., health condition, family history, etc.), the actions correspond to the treatment options that are being compared, and the reward represents the outcome of the proposed treatment (e.g., success or failure) [84], see Figure 4.4.

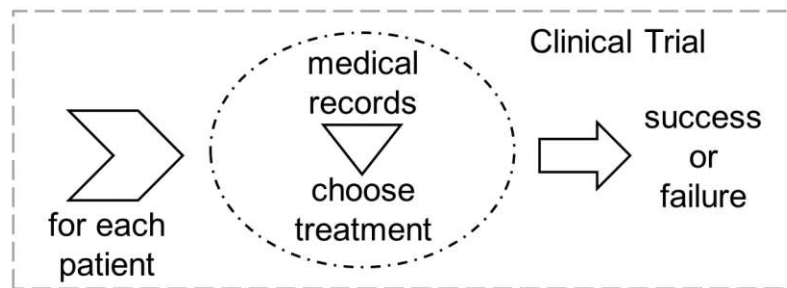


Figure 4.4: A clinical trial conceptualised as a CMAB algorithm, compare Figure 3.4.

Clinical trials are a convenient example for illustrating the CMAB setting, because their purpose is to find balance in the exploration-exploitation trade-off (e.g., between a new treatment method and a known one) [16]. Note that, upon introducing context to the bandit setting, Woodroffe [91] motivates his extension of the classical MAB model by

referring to clinical trials. However, an analysis of clinical trials by Tekin et al. [78] shows that a doctor can only ask a patient a limited number of questions before deciding on a drug prescription, suggesting that the choice of treatment is generally based on the most relevant aspects of a patient’s medical history, and thus relies on restricted contextual information.

Restricted context can also occur undeliberately as a result of faulty measurement. Upon introducing the fundamental principles of tailoring variables as a component in a JITAI, Section 2.2.2 states that the measuring of data needs to be reliable in order to accurately recommend interventions, otherwise the decision rules will perform little better than randomly selecting actions, or worse, an adverse option may be recommended, potentially causing the client harm. Section 2.2.2 further states that an application based on the JITAI concept must anticipate such situations and be equipped to handle them [58].

Nahum-Shani et al. [58] recommend that developers should anticipate and plan the functioning of the decision rules in case measurements on tailoring variables are missing, and note that missing data can occur for various technical reasons, including data corruption (e.g., loss of data due to problems in data storage), device detection failures (e.g., no GPS signal) and human error (e.g., incorrect use of measurement device). Another cause can be poor engagement on the client’s side, in which case indicators of missing data may be employed as tailoring variables that reflect intervention fatigue, see Section 2.2.3.

Nonetheless, in order for a JITAI application to be functional as a recommender system, the decision rules must cover situations of missing data, which means that the CMAB algorithm, acting as decision rules, must include a mechanism to recommend appropriate interventions despite the lack of data.

Generally, due to construction, CMAB algorithms (see Section 3.4) cannot compensate for missing contextual data. The algorithms are designed to choose an action, thereby recommending an intervention, and not to check whether the delivered data is feasible. Furthermore, the problem of missing data is related to the quantification of contextual information. For example, if weather conditions are regarded as context in a CMAB-based JITAI, the simplest way of distinguishing “good” weather from “bad” weather is to assign binary values, whatever the definitions of good and bad may be. The component  $x_{weather}(t) \in \mathbb{R}$  of the context vector  $x(t) \in \mathbb{R}^d$  at time  $t$  is [67]:

$$x_{weather}(t) = \begin{cases} 0 & \text{if weather = bad} \\ 1 & \text{if weather = good} \end{cases}$$

The question arises of what value to assign  $x_{weather}(t)$  in case of missing data, because any CMAB algorithm cannot make a decision if  $x_{weather} = NaN$ . Here, the application must detect the missing data, and decide how to deal with the component outside of the

CMAB algorithm. Another option is to assign

$$x_{weather}(t) = \begin{cases} 0 & \text{if weather = not recorded} \\ 1 & \text{if weather = bad} \\ 2 & \text{if weather = good} \end{cases},$$

which removes the binary character from a binary situation. Furthermore, the numerical value of 0 defines a new setting, in which the contextual component for weather is nondescript.

The CBRC setting introduced by Bouneffouf et al. in 2017 [16] indicates that the research into more applicable CMAB algorithms, and possible extensions, is currently in a promising state. It enables the algorithm to disregard components of the context vector when observing the full vector is either too expensive or impossible. Also, the agent can only request to observe a limited number of features from the vector. However, the upper bound (or *budget*) on the feature subset is fixed for all iterations, but within the budget, the player can choose any feature subset of the given size. The problem is to select the best feature subset so that the overall reward is maximised, which involves exploring the feature space as well as the arm space [16]. For example, if the context vector  $x(t) \in \mathbb{R}^{10}$  is 10-dimensional, and the CBRC budget is set to 4, then only 4 features can be observed at each iteration. The feature vector at  $t$ ,  $x_4(t) \in \mathbb{R}^{10}$ , is a sparse vector with a maximum of four components unequal to zero.

Bouneffouf et al. also propose an algorithm for solving CBRC problems, and the strategy is displayed in Algorithm 8. First, the best subset of features is selected, the values of the chosen elements are observed, and subsequently, the decision-making task is performed based on the selection of the subset of features. The task of selecting a subset of features is modelled as a *combinatorial bandit*, which is described in Section 4.4.2, while the subsequent arm-selection bandit is based on the TS algorithm presented by Agrawal and Goyal [2]. Therefore, the algorithm is called *Thompson sampling with restricted context* (TSRC), and is discussed at length in Section 4.5.

#### 4.4.2 Combinatorial Bandits

The framework for combinatorial bandits is first described by Chen et al. in 2013 [19], motivated by the fact that, in many real-world applications, the bandit problem has a combinatorial nature, where observed rewards correspond to a function of multiple arms instead of only one [25].

An example for the combinatorial bandit setting is an online advertising scenario, where a website contains a set of webpages and has a set of users visiting the website. An advertiser

**Algorithm 8:** Contextual Bandit with Restricted Context Strategy

---

**Input:** arms  $\mathcal{A} = \{1, \dots, k\}$ , set  $C = \{1, \dots, d\}$  of feature indices, budget  $n$

- 1 **for**  $t = 1, 2, \dots$  **do**
- 2     choose subset  $C^n \subseteq C$  of features
- 3     observe values  $x_n(t)$  of features  $C^n$
- 4     choose arm  $a(t) = i \in \mathcal{A}$
- 5     receive reward  $r(t) \sim \mathcal{R}_{i(t)}$
- 6     improve arm-selection strategy with new observation  $(x_n(t), a(t), r(t))$
- 7 **end**

---

places an ad on a subset of selected webpages because due to monetary constraints, only  $n$  webpages can be selected. Each user visits a certain number of webpages, and on each visited webpage has a click-through probability of clicking the ad on that page; however, these probabilities are unknown to the advertiser. The advertiser repeatedly select sets of  $n$  webpages to advertise on, observes the click-through data, and learns the click-through probabilities, thus maximising the number of users clicking on the ad.

In a combinatorial bandit, at each iteration, a set of arms (called a *super arm*) is played together, and the outcomes of all arms in the super arm are revealed. In the advertising example above, the super arm is represented by the selection of  $n$  webpages to advertise on. Note that the framework allows an arbitrary combination of arms into super arms. It is possible to treat every super arm as an arm within the classical MAB framework, and to solve the combinatorial problem with classical MAB methods. However, the number of super arms may be exponential to the problem size due to combinatorial explosion, and after one super arm is played, information regarding the outcomes of underlying arms is observed, which may be shared by other arms [19].

Each arm  $i \in \{1, \dots, k\}$  is associated with a corresponding variable  $y_i(t) \in \mathbb{R}$  that indicates the reward obtained when choosing arm  $i$  at time  $t$ . A constrained set of arm subsets  $S \subseteq P(k)$ , where  $P(k)$  is the power set of  $k$ , is associated with a set of variables  $\{r_M(t)\}_{M \in S}$  for all  $t \geq 1$ . The variable  $r_M(t) \in \mathbb{R}$  indicates the reward associated with selecting a subset of arms (i.e., the super arm)  $M \in S$ , at time  $t$ , where

$$r_M(t) = h(y_i(t), i \in M)$$

for some reward function  $h(\cdot)$ . Thus, in the combinatorial bandit setting, the agent sequentially selects super arms  $M$  from  $S$  and observes rewards  $r_M(t)$  corresponding to the played subsets. The simplest approach for computing  $r_M(t)$  is to consider the sum of the

individual rewards of the arms in  $M$ :

$$r_M(t) = h(y_i(t), i \in M) = \sum_{i \in M} y_i(t)$$

The objective of the combinatorial bandit is to maximise the (joint) reward over time [16]. A TS-based algorithm for combinatorial bandits is provided by Durand and Gagné [25], which is designed to solve combinatorial optimisation problems, like online feature selection, and is depicted in Algorithm 9. Here, each arm  $i$  is associated with two additional parameters,  $c_i(t)$  and  $s_i(t)$ , whose values vary with time. The number of times  $i$  has been selected (i.e.,  $i \in M(t)$ ) until  $T$  is denoted by  $c_i(T)$ . The cumulative reward associated with arm  $i$  until  $T$  is

$$s_i(T) = \sum_{t=1}^T \mathbf{1}_{[i \in M(t)]} r_{M(t)}(t),$$

---

**Algorithm 9:** Thompson Sampling for Combinatorial Bandits

---

**Input:** arms  $\mathcal{A} = \{1, \dots, k\}$ , initial parameter values  $\alpha_i, \beta_i \forall i \in \mathcal{A}$ ,  $S \subseteq P(k)$

**Initialise:**  $c_i(0) = 0$ ,  $s_i(0) = 0 \forall i \in \mathcal{A}$

```

1 for  $t = 1, 2, \dots$  do
2   foreach  $i \in \mathcal{A}$  do
3      $\alpha_i(t) = \alpha_i + s_i(t-1)$ 
4      $\beta_i(t) = \beta_i + c_i(t-1) - s_i(t-1)$ 
5     sample  $\theta_i \sim \mathbf{Beta}(\alpha_i(t), \beta_i(t))$ 
6   end
7   choose  $M(t) = \operatorname{argmax}_{M \in S} \sum_{i \in M} \theta_i$ 
8   observe  $r_{M(t)}$ 
9   update  $c_i(t) = c_i(t-1) + 1$  for  $i \in M$ 
10  if  $r_{M(t)} = 1$  then
11    | update  $s_i(t) = s_i(t-1) + 1$  for  $i \in M$ 
12  end
13 end

```

---

describing the number of successes achieved over all iterations where arm  $i$  was selected. The beta prior thus gives an estimate of the success probability for each arm,

$$\begin{aligned} \theta_i &\sim \mathbf{Beta}(\alpha_i, \beta_i), \\ \alpha_i(t) &= \alpha_i + s_i(t-1), \\ \beta_i(t) &= \beta_i + c_i(t-1) - s_i(t-1), \end{aligned}$$

with  $\alpha_i$  and  $\beta_i$  denoting the initial beta distribution parameters. Due to construction, the prior is updated to a beta posterior at the end of each time step, providing the prior

for the subsequent step. Note that combinatorial feature selection has no need to take into account contextual information, as it is itself the context for the arm-selection bandit, and its aim is to identify the features with the highest probability to result in a successful outcome when the agent decides on an arm.

The next section introduces TSRC, the TS algorithm solving the CBRC problem proposed by Bouneffouf et al. [16], which combines the feature selection via a combinatorial bandit with the arm selection via TS.

## 4.5 Thompson Sampling with Restricted Context

The content of this section is cited from Bouneffouf et al. [16], unless otherwise stated.

In the CBRC setting, like in the CMAB setting, the environment of the bandit is described by a set of features. However, the agent can only choose a limited-size subset of that contextual information to observe, and thus needs to explore the feature space simultaneously to exploring the arms space in order to find the best feature subset.

The TSRC algorithm is unique in its existence as it is currently the first and only approach addressing the problem of restricted context in the CMAB setting within the bandit algorithm itself. As stated in Section 4.4.1, restricted context may be dealt with outside the bandit environment, for example in a JITAI application acting as a recommender system. However, the combination of a combinatorial bandit for feature selection and a contextual bandit for arm selection offers a solution to CBRC problems without the need for additional mechanisms. Note that both bandit settings work with TS, namely the conjugate beta prior and Bernoulli likelihood combination, and the multivariate Gauss prior and posterior respectively.

The TSRC algorithm is displayed in Algorithm 10. Let  $x(t) \in \mathbb{R}^d$  denote the values of the feature vector for features  $(C_1, \dots, C_d)$  at time  $t$ , and let  $C = \{1, \dots, d\}$  be the set of their indices. Additionally, let  $x_n(t)$  denote a vector of only  $n \leq d$  features, representing a projection of  $x(t)$  onto the indices provided by a subset  $C^n$  of  $C$  with  $|C^n| = n$ . Formally, the set of all such vectors is denoted by

$$\mathbb{R}_{C^n} := \{x_n(t) \in \mathbb{R}^n \mid x_n(t) \text{ is a projection onto indices from } C^n\}.$$

Furthermore, consider a set of compound-function policies

$$Q^n = \bigcup_{C^n \in C} \{q : \mathbb{R}^d \rightarrow \mathcal{A} \mid q(x) = \pi_{C^n}(f(x))\},$$

---

**Algorithm 10:** Thompson Sampling with Restricted Context

---

**Input:** arms  $\mathcal{A} = \{1, \dots, k\}$ , feature indices  $C = \{1, \dots, d\}$ , budget  $n$ , constant  $v$   
**Initialise:**  $\forall i \in \mathcal{A}: B_i = I_n, \hat{\gamma}_i = g_i = 0_n, \forall j \in C: S_j^0 = F_j^0 = 1, c_j = s_j = 0$

```

1 for  $t = 1, 2, \dots, T$  do
2   foreach  $j \in C$  do
3      $S_j = S_j^0 + s_j$ 
4      $F_j = F_j^0 + c_j - s_j$ 
5     Sample  $\theta_j \sim \mathbf{Beta}(S_j, F_j)$ 
6   end
7   Select  $C^n(t) = \operatorname{argmax}_{C^n \in C} \sum_{C^n} \theta_j$ 
8   Obtain features  $x_n(t) \in \mathbb{R}^n$ 
9   foreach  $i \in \mathcal{A}$  do
10    | Sample  $\tilde{\gamma}_i \sim \mathbf{N}(\hat{\gamma}_i, v^2 B_i^{-1})$ 
11  end
12  Select  $m(t) = \operatorname{argmax}_{i \in \mathcal{A}} x_n(t)^T \tilde{\gamma}_i$ 
13  Observe  $r_m(t)$ 
14  if  $r_m(t) = 1$  then
15    |  $B_m = B_m + x_n(t)x_n(t)^T$ 
16    |  $g_m = g_m + x_n(t)r_m(t)$ 
17    |  $\hat{\gamma}_m = B_m^{-1}g_m$ 
18    |  $\forall j \in C^n: s_j = s_j + 1$ 
19  end
20   $\forall j \in C^n: c_j = c_j + 1$ 
21 end
```

---

where  $\pi_{C^n}$  describes the projection of  $x(t)$  from  $\mathbb{R}^d$  onto  $\mathbb{R}_{C^n}$ ,

$$\pi_{C^n} : \mathbb{R}^d \rightarrow \mathbb{R}_{C^n},$$

and the function  $f$  maps the restricted context vector to an action:

$$f : \mathbb{R}_{C^n} \rightarrow \mathcal{A}$$

Figure 4.5 shows the policy composition. The TSRC algorithm assumes that the rewards for chosen actions are binary,  $r_i(t) \in \{0, 1\}$ . The objective of the bandit algorithm is to learn a hypothesis  $q$  over  $T$  iterations, maximising the total reward.

The TSRC algorithm requires arms and the feature index set as inputs. The budget  $n$  needs to be fixed, and a constant  $v$  is defined that influences the multivariate Gauss distribution prior, its purpose is discussed later in this section. During the initiation phase, the matrices  $B_i$  and the means  $\hat{\gamma}_i$  are set to unit matrices and 0 respectively. Auxiliary



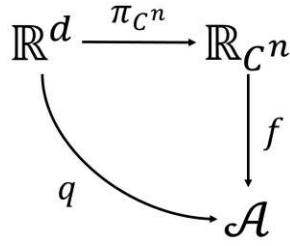


Figure 4.5: Composition of a policy  $q$  in case of restricted context.

variables  $g_i$ , which aid the update of the means  $\hat{\gamma}_i$ , are also set to 0 for each arm. For each feature, the initial beta distribution parameters are set to 1, which leads to a uniform distribution in the initial likelihoods of any feature  $C_j$  to be chosen.

Let  $c_j$  be the number of times the  $j$ -th feature has been selected, and  $s_j$  denote the cumulative reward (i.e. the successes) associated with feature  $C_j$ . At each iteration, the values of the beta distribution parameters  $S_j$  and  $F_j$  are updated to represent the cumulative reward (i.e., the current total number of successes  $s_j$ ) and failures (i.e.,  $c_j - s_j$ ) respectively, see Steps 3 and 4 of Algorithm 10. Subsequently, the success probability  $\theta_j$  is sampled from the corresponding beta distribution for each feature  $C_j$  in Step 5. In Step 7, the subset  $C^n$  is selected, which maximises the sum of those parameters. Note that no combinatorial search is required for this step. Since the individual rewards  $\theta_j$  are non-negative, the set  $C^n$  of arms with the  $n$  highest individual rewards can be chosen.

Now that the best feature subset  $C^n$  is chosen, the algorithm switches to the CMAB setting. First, the restricted feature vector  $x_n(t)$  is observed before sampling  $\tilde{\gamma}_i$  from the multivariate Gauss distribution  $\mathbf{N}(\hat{\gamma}_i, v^2 B_i^{-1})$ , where

$$\hat{\gamma}_i = B_i(t)^{-1} \left( \sum_{\tau=1}^{t-1} x_n(\tau) r_m(t) \right),$$

$$B_i(t) = I_n + \sum_{\tau=1}^{t-1} x_n(\tau) x_n(\tau)^T.$$

Here,  $r_m(t)$  denotes the reward for choosing arm  $m(t)$ , which is either 1 or 0. The expected reward is assumed to be a linear function of the restricted context,

$$\mathbb{E}[r_i(t) | x_n(t)] = x_n(t)^T \tilde{\gamma}_i,$$

and the arm  $m(t)$  that maximises the expected reward is selected, see Step 12. Then, the relevant parameters are updated. Note that  $c_j$  is updated every time feature  $C_j$  is chosen, but  $s_j$  is only updated if the reward  $r_m(t)$  yields success.

The additional factor  $v$  parametrises the algorithm:

$$v := R\sqrt{\frac{24}{\epsilon}n \ln\left(\frac{1}{\delta}\right)},$$

with  $\epsilon \in (0, 1)$ ,  $\delta \in (0, 1)$ , and  $R \geq 0$ . These parameters stem from assumptions made to obtain a regret bound of

$$O\left(n\sqrt{\frac{T^{1+\epsilon} \ln(K)}{\epsilon}} \left(\ln(T) \ln\left(\frac{1}{\delta}\right)\right)\right),$$

given by Agrawal and Goyal [4], who prove a high probability regret bound for their TS algorithm for the CMAB problem with linear pay-off functions and multivariate Gauss prior, thereby providing the first theoretical guarantees for the contextual version of TS. Note that the TSRC algorithm assumes a stationary environment, that is, the probability distributions are considered fixed, and the objective is to identify a subset of features allowing for the optimal context-to-arm mapping.

During empirical evaluation, the performance of the TSRC algorithm is examined and compared to other methods, which are listed in Table 4.1, in terms of accuracy in classifying instances of large datasets. When context is limited, the TSRC algorithm shows superior performance compared to other algorithmic solutions for restricted context (i.e., Random-EI and Random-fix). In fact, the obtained mean error rates suggests that using a fixed randomly selected feature subset (i.e., Random-fix, 49.01%) may be a better strategy than not considering context at all (i.e., MAB, 57.98%), and that disregarding context may be a better approach than randomly changing the choice of the feature at each iteration (i.e., Random-EI, 61.18%) [16].

Name	Description
MAB	The algorithm is the standard TS approach to the non-contextual MAB setting, see Algorithm 6
Fullfeatures	The algorithm depicts the TS algorithm for the CMAB setting with the full set of features, see Algorithm 7
Random-EI	The algorithm selects a random subset of features of specified size $n$ at each iteration (thus “EI”) and invokes TS for CMABs
Random-fix	The algorithm invokes TS for CMABs on a random subset of $n$ features, but the subset is selected once, prior to seeing any data samples, and remains fixed

Table 4.1: List and short descriptions of algorithms tested against TSRC, from [16].

The closer the budget  $n$  is to the full feature setting, the better the performance of the

TSRC algorithm is compared to the full feature TS algorithm. Note that, at 25% sparsity (i.e., when selecting 75% of available features each round), the TSRC algorithm has been found to perform almost as well as the Fullfeatures algorithm, which implies that, at this sparsity level, the TSRC algorithm is able to select an optimal feature subset.

An example illustrates how the feature selection process influences the arm selection bandit algorithm. Let there be three possible features  $\{C_1, C_2, C_3\}$  to select, so  $C = \{1, 2, 3\}$ , and let  $n = 2$  (i.e., two out of three features are chosen each round). Thus, at time  $t$ ,  $C^2(t)$  may be

$$C^2(t) = \{1, 2\}, \text{ or } C^2(t) = \{1, 3\}, \text{ or } C^2(t) = \{2, 3\}.$$

Let  $i$  be a fixed arm, and  $B_i(0) = I_2$ . Step 15 of Algorithm 10 shows the update for  $B_i(t)$ :

$$B_i(t) = B_i(t-1) + x_2(t)x_2(t)^T$$

During each step, the chosen subset of features  $x_2(t)$  influences the matrix update, which in turn influences the parameters for the multivariate Gauss distribution, see Steps 15 to 17. Depending on the choice of features,  $x_2(t)x_2(t)^T$  can take three shapes:

$$x_2(t)x_2(t)^T = \begin{pmatrix} C_i(t)C_i(t) & C_i(t)C_j(t) \\ C_j(t)C_i(t) & C_j(t)C_j(t) \end{pmatrix}, \quad i, j \in \{1, 2\} \vee \{1, 3\} \vee \{2, 3\}.$$

If, for example, at time  $t$ ,  $C^2(t) = \{1, 2\}$  and at time  $t+1$ ,  $C^2(t+1) = \{2, 3\}$ , the update for  $B_i(t+1)$  is:

$$B_i(t+1) = B_i(t-1) + \begin{pmatrix} C_1(t)C_1(t) & C_1(t)C_2(t) \\ C_2(t)C_1(t) & C_2(t)C_2(t) \end{pmatrix} + \begin{pmatrix} C_2(t)C_2(t) & C_2(t)C_3(t) \\ C_3(t)C_2(t) & C_3(t)C_3(t) \end{pmatrix}$$

This shifting of context elements within the update matrix introduces a fuzziness that is not observed for the TS algorithm with the full feature set, see Algorithm 7, where the context elements have their fixed place in  $x(t)x(t)^T$ :

$$x(t)x(t)^T = \begin{pmatrix} C_1(t)C_1(t) & C_1(t)C_2(t) & C_1(t)C_3(t) \\ C_2(t)C_1(t) & C_2(t)C_2(t) & C_2(t)C_3(t) \\ C_3(t)C_1(t) & C_3(t)C_2(t) & C_3(t)C_3(t) \end{pmatrix}$$

In contrast,  $x_2(t)x_2(t)^T$  is more variable, depending on the choice of feature subset:

$$x_2(t)x_2(t)^T = \begin{pmatrix} C_1(t)C_1(t) & \vee & C_2(t)C_2(t) & & * \\ & * & & & \\ & & & C_2(t)C_2(t) & \vee & C_3(t)C_3(t) \end{pmatrix}$$

If the pool of features is extended by one (i.e.,  $C = \{C_1, C_2, C_3, C_4\}$ ), but the budget is kept at  $n = 2$ , the options for elements in matrix  $x_2(t)x_2(t)^T$  also expand:

$$x_2(t)x_2(t)^T = \begin{pmatrix} C_i(t)C_i(t) & C_i(t)C_j(t) \\ C_j(t)C_i(t) & C_j(t)C_j(t) \end{pmatrix}, \quad i, j \in \{1, 2\} \vee \{1, 3\} \vee \{1, 4\} \vee \{2, 3\} \vee \{2, 4\} \vee \{3, 4\}$$

In case of  $d = 3$  and  $n = 2$ , the level of sparsity is at 33%, whereas  $d = 4$  and  $n = 2$  denotes 50% sparsity and indicates more fuzziness in the matrix update, which influences the covariance matrix and the mean for the multivariate Gauss distribution in Step 10. It can be deduced that this fuzziness is responsible for Random-EI performing generally worse in the empirical study by Bouneffouf et al. [16] compared to both Random-fix and TSRC in cases of restricted context.

## 5 Methods of Implementation

This chapter describes in detail the structure of a simple model JITAI, the strategy for modelling different client responses, and the implementation of the TSRC algorithm (Algorithm 10) as a decision rule for the model JITAI.

First, the model JITAI is introduced, and its components are identified according to the definitions in Chapter 2. After explaining the construction of the model clients, the implementation of the TSRC algorithm and its peripheral files in MATLAB is outlined, which includes implementation trees as a graphical representation of the code's architecture.

The implementation of the code is executed in MATLAB, version R2020b, with a license for academic use issued by Technische Universität Wien, on a PC running 64-bit Windows 10 Enterprise. MATLAB has proven to be a reliable program that offers practical solutions for constructing easy-to-use scripts, and has many plot options available in order to present a variety of simulation results in a comprehensive manner.

### 5.1 The Model Just-In-Time Adaptive Intervention

To investigate whether the TSRC algorithm is feasible as a decision rule within the JITAI setting, a model JITAI must be constructed with which to perform the simulation experiments. The design of such a model ought to be affected by the limitations of the consecutive simulation and its results. This means that the JITAI components must be chosen with regard to what the simulation results can be expected to display clearly. For example, if an intervention option is feasible in a real-life setting, but the simulation is not assumed to portray explicitly either merits or disadvantages of choosing that option, it ought to be omitted during the design process. Thus, each component of the model JITAI has been chosen with the aim to obtain distinct simulation results, which are expected to be interpretable in a straight-forward way.

The general aim of the model JITAI is to support habit building in the activity domain, sharing a resemblance with the existing JITAI applications HeartSteps and MyBehavior, see Section 3.5. However, neither of those applications are equipped to deal with missing data within their decision rules, even though HeartSteps is designed to include the delayed effect of treatment, which is modelled as an MDP [52]. Note that this thesis does not claim

any validity of the JITAI design towards potential benefits to a health issue. It merely aims to aid the understanding of the underlying processes when employing the TSRC algorithm as a decision rule, and presents possible solutions to general issues concerning JITAI construction and policy implementation.

The assumed setup is as follows. The client is equipped with a mobile fitness tracker (ideally, a smart watch) and the JITAI app on a smartphone. The model JITAI provides an activity suggestion once per day, and the fitness tracker reports whether or not the client has adhered to the suggestion to the JITAI. A possible reason for employing the model JITAI can be habit building towards a more active lifestyle for clients who are at risk of, or recovering from, cardiovascular disease, which might include overweight, obesity, or diseases related to these issues [90].

The model JITAI considers four tailoring variables, or features, for decision making: the weather, the client's availability, their motivation, and their fitness level. Each of these are quantifiable in a real-life setting: weather data, paired with the client's location, is available to the JITAI over the internet. The client's availability may be measured by registering the screen time of their smartphone, and their fitness level can be assessed in different ways, either via the *body-mass index* (BMI), or by surveying the resting heart rate with the mobile fitness tracker. The client's motivation requires active assessment, for example, via a notification on the smartphone that asks the client to indicate their willingness for physical activity on a sliding scale.

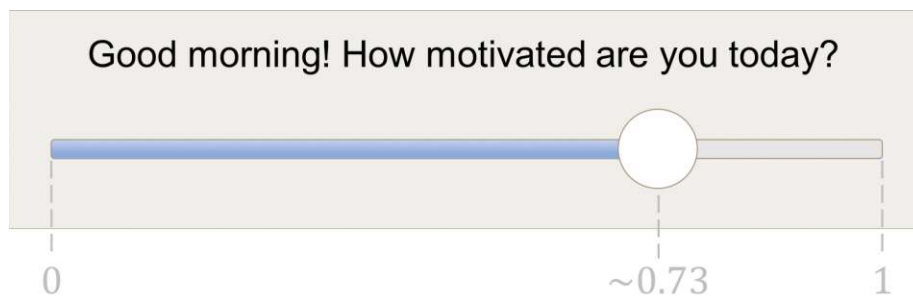


Figure 5.1: Sliding scale for assessing a client's motivation, and normalisation of the value.

As a result of the peculiarity of disregarding a certain number of features in the TSRC algorithm, see Section 4.5, the tailoring variables must be normalised, and without loss of generality their values lie in the unit interval  $[0, 1]$ . Due to the different natures of the features, normalising presents an obstacle in a real-life setting, but solutions can be found. For example, the weather can be normalised within the frame of the past 365 days with regard to the temperature and precipitation forecast of the day, see Section 5.3, and the motivation value, to which the client fixes the slider each morning, can be assumed

to be a number between 0 and 1, see Figure 5.1. Also, within each feature, 1 indicates “good”, whereas 0 indicates “bad”. For example, the greater the value for availability, the more free time the client has to spend on physical activity. Note that the weather feature does not simply equal the measured maximum temperature, but a combination of what makes weather data interpretable as “good” for outdoor activities. Rain or snow, as well as extreme heat or cold, are seen as unpractical, and a combination of different kinds of weather data needs to be considered in order to pass judgement. Also note that the fitness level of a client is not assumed to be static, but to vary slightly within a certain interval corresponding to the daily physical constitution of the client, for example due to fatigue or exhaustion.

There are six intervention options available to the JITAI, which have the potential to respond uniquely to the values of the tailoring variables:

- Low intensity indoor training
- Low intensity outdoor training
- Medium intensity indoor training
- Medium intensity outdoor training
- High intensity indoor training
- High intensity outdoor training

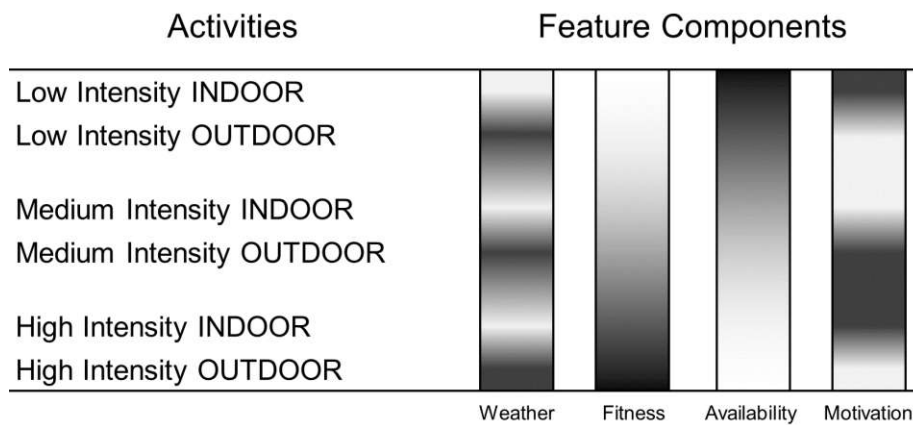


Figure 5.2: Possible influence pattern on the intervention options. Darker areas represent high probabilities of activity acceptance when feature values are high.

The differentiation between indoor and outdoor training appeals to the weather feature. The levels of training intensity directly relate to a client’s fitness and their availability, but

in reverse proportionality, which is preceded by the assumption that low intensity training takes longer (i.e., taking a walk), compared to high intensity training, which requires a higher level of fitness, but is more concise regarding time. The motivation feature may indicate greater motivation for certain activities compared to others depending on the client, or can be modelled in a more complex way, for example, by depending on the weather. Figure 5.2 illustrates these general assumptions.

At each decision point, a fixed number  $n < 4$  of tailoring variables is considered by the TSRC algorithm acting as decision rules. In the JITAI context, successfully performing the suggested activity is the proximal outcome, whereas a distal outcome may be to improve the fitness level, or lower the resting heart rate. Table 5.1 provides an overview of the model JITAI's elements, and Figure 5.3 illustrates how the model JITAI is working along a timeline.

JITAI Elements	Description
Decision points	Once per day, in the morning (e.g., at 7 a.m.)
Tailoring variables	Weather, fitness, availability, motivation
Intervention options	Low intensity indoor/outdoor, medium intensity indoor/outdoor, high intensity indoor/outdoor
Decision rules	TSRC algorithm (Algorithm 10)
Proximal outcome	Achieving the daily activity goal (i.e., performing the suggested activity)
Distal outcome	Improving the individual fitness, or lowering the BMI, or lowering the resting heart rate

Table 5.1: Summary of the JITAI elements of the model JITAI.

Note that there are many other aspects in a working JITAI that have been omitted for simplicity's sake that require consideration in a real-life setting. For example, the individual working schedules of clients ought to be taken into consideration, so the period of time when they are unreceptive to the chosen intervention option varies from client to client. A working JITAI must address this issue, possibly by letting the client choose the point in time at which the daily suggestion is presented in order to maximise convenience, and thus effectiveness of suggestion delivery. Furthermore, intervention engagement and fatigue are not considered by the model JITAI, and are subsequently disregarded when modelling the client.

The next section explains how clients are modelled, and how their responses are varied in different simulations.



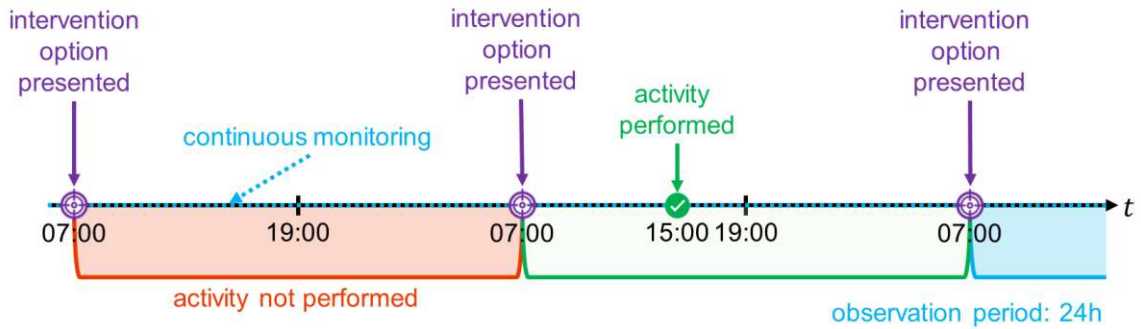


Figure 5.3: The model JITAI working along a timeline of 24h observation periods, showing both cases of detecting and not detecting the suggested activity.

## 5.2 Modelling the Client

To verify whether the TSRC algorithm delivers the expected results, each of the four features is regarded separately, because in a real-life setting each client is disposed individually towards the features and their connections to the intervention options. By isolating single features, the response of the TSRC algorithm can be analysed with the least amount of noise. This means modelling a client whose attitude towards the activity suggestions depends only on the value of that feature. For example, when isolating the weather feature, a client is modelled who only considers the weather data in terms of deciding whether or not to accept the intervention option.

Different clients are created by implementing them as particular classes in MATLAB, because the class setting, which consists of properties and methods that can be called upon in functions and plots, has proven to be most convenient. All client classes share the same properties, which are listed Table 5.2. The classes also share the same methods, that is, each class has three methods whose names are identical between classes, and on which the TSRC algorithm calls during calculations. However, the definitions of these methods differ, resulting in different client responses.

The first method `client.initiation(T)` is identical between clients and initiates a client's empty properties  $\mathbf{f}, \mathbf{m}, \mathbf{a}, \mathbf{r}$ , as zero vectors, and the default weather value for all  $T$  days as 0.5. Note that  $T$  gives the number of days the simulation will go through (i.e., the simulation horizon). The initiation method is necessary because the properties in the `classdef` file cannot access variables from the workspace, namely  $T$ , so the simulation horizon cannot be altered dynamically. By using an initiation method,  $T$  can be given to the method as an input, and the size of the relevant properties can change for different simulation horizons.

During each iteration  $t \leq T$  of the TSRC algorithm, the client's feature data is generated

Properties	Description
$f$	Vector of length $T$ , which holds the fitness level of the client on each day. Default value: empty
$m$	Vector of length $T$ , which holds the client's motivation on each day. Default value: empty
$a$	Vector of length $T$ , which holds the client's availability on each day. Default value: empty
$w$	Vector of length $T$ , which holds the normalised weather data on each day. Default value: empty
$r$	Vector of length $T$ , which holds the client's response to the chosen action suggestion of the day, either 0 or 1. Default value: empty
$av_{WE}$	Base value for availability during the weekend. Default value: 0.5
$var_{WE}$	Maximum interval for varying availability during the weekend. Default value: 0.5
$av_{WD}$	Base value for availability on weekdays. Default value: 0.5
$var_{WD}$	Maximum interval for varying availability on weekdays. Default value: 0.5
$av_F$	Base value for fitness. Default value: 0.5
$var_F$	Maximum interval for varying fitness. Default value: 0.5
$av_M$	Base value for motivation. Default value: 0.5
$var_M$	Maximum interval for varying motivation. Default value: 0.5
$f^*$	Threshold that must be exceeded for fitness value to improve. Default value: 0.5

Table 5.2: Properties and their descriptions of the MATLAB classes for all clients.

for that day, filling (or overwriting)  $f_t$ ,  $m_t$ ,  $a_t$ , and  $w_t$ , and saving the client's response to the suggested activity in  $r_t$ . For this, the TSRC algorithm calls on the class method `client.singleStepClientDataGenerator(t)`, which generates the feature values and delivers them to the algorithm. In the case of the motivation feature, a number within the interval  $(av_M - var_M, av_M + var_M)$  is created via the following method:

$$m_t = av_M \pm z \cdot var_M,$$

where  $z$  is a random number between 0 and 1, generated by the MATLAB function `rand()` which samples from the uniform distribution between 0 and 1, and the sign is determined randomly. If the generated value for  $m_t$  exits the unit interval it is corrected to the nearest interval border:

$$m_t > 1 \Rightarrow m_t = 1,$$

$$m_t < 0 \Rightarrow m_t = 0$$

Note that, when investigating different motivation patterns, this formula may be varied between clients. The same calculation and, if necessary, correction is done for the client's availability  $\mathbf{a}_t$ . However, the client data generator distinguishes between weekdays and weekends. If  $\text{mod}(t, 7) = 6$  or  $\text{mod}(t, 7) = 0$  (i.e., if it is assumed to be Saturday or Sunday),  $\mathbf{a}_t$  is computed using the weekend base value and interval border:

$$\mathbf{a}_t = av_{WE} \pm z \cdot var_{WE}$$

On weekdays,  $\mathbf{a}_t$  is calculated using the weekday availability parameters:

$$\mathbf{a}_t = av_{WD} \pm z \cdot var_{WD}$$

Again,  $z$  is given by the function `rand()` and the sign is chosen randomly.

The client's fitness  $\mathbf{f}_t$  is generated using the same pattern as the motivation feature, including an additional aspect. Under the assumption that regular exercise improves the client's fitness level, a fitness threshold value  $f^*$  is used to depict the client's personal fitness improvement: if the client has exceeded their personal fitness threshold over the past week (i.e., if the client has accepted the offered interventions, thus exercising more than  $f^*$  times during the previous week), the baseline fitness  $av_F$  improves by 5% on the following Monday, and the maximum interval for varying the fitness value declines by 5%. Note that a decline in the client's fitness is not considered in the model, but the implementation can be made analogously by considering a "laziness threshold", where the base fitness level declines and the variation interval increases by a certain percentage if the client has not exercised often enough during the previous week.

Also note that feasible weather data is not generated by a class method, but via a separate function `weatherGenerator.m`, see Section 5.3, because the weather depends on the client implicitly, through their location. The weather generation function is run by the script after the client has been initialised, and the resulting vector of length  $T$  overwrites the default weather data from the initiation method before running the TSRC algorithm.

The third class method `client.clientResponseGenerator(t,k)` is also accessed by the TSRC algorithm, and it generates the client's response to the intervention options in step  $t$ . Here, the feature data is combined into a formula for each intervention option, resulting again in a value between 0 and 1. These formulae differ between clients, and reflect their predisposition towards the activity suggestions. For example, when isolating the weather feature, the response of a client who prefers outdoor activities when the weather is good, is  $1 - \mathbf{w}_t$  for indoor activities (i.e., the worse the weather, the greater the willingness to train indoors), and  $\mathbf{w}_t$  for outdoor activities (i.e., the better the weather, the greater the willingness to train outdoors).

The resulting values, depicted by  $l_I(t)$ ,  $l_O(t)$ , for low intensity training indoors and outdoors, respectively,  $m_I(t)$ ,  $m_O(t)$ , for medium intensity training, and  $h_I(t)$ ,  $h_O(t)$ , for high intensity training, provide the success probabilities for six separate Bernoulli distributions, one for each intervention option. The client response generator then draws from these distributions and returns a six-dimensional vector  $x(t) = (x_1(t), \dots, x_6(t))$ ,

$$x_i(t) = 0 \quad \vee \quad x_i(t) = 1 \quad \forall i \in \{1, \dots, 6\},$$

whose entries correspond to the success or failure of the random draw for each activity. For the isolated weather feature, this equals a response vector  $x(t)$ ,

$$\begin{aligned} x_1(t) &\sim \text{Bernoulli}(l_I(t)), & l_I(t) &= 1 - w_t, \\ x_2(t) &\sim \text{Bernoulli}(l_O(t)), & l_O(t) &= w_t, \\ x_3(t) &\sim \text{Bernoulli}(m_I(t)), & m_I(t) &= 1 - w_t, \\ x_4(t) &\sim \text{Bernoulli}(m_O(t)), & m_O(t) &= w_t, \\ x_5(t) &\sim \text{Bernoulli}(h_I(t)), & h_I(t) &= 1 - w_t, \\ x_6(t) &\sim \text{Bernoulli}(h_O(t)), & h_O(t) &= w_t. \end{aligned}$$

Drawing from a Bernoulli distribution naturally imitates the behaviour of the client concerning daily activity suggestions. The formulae corresponding to the intervention options are designed in such a way that higher values correspond to greater eagerness for performing the action, but even if the values are sufficiently high, the client might still not exercise because of their daily constitution (e.g., having a headache), or short-term plan changes, or simply forgetting.

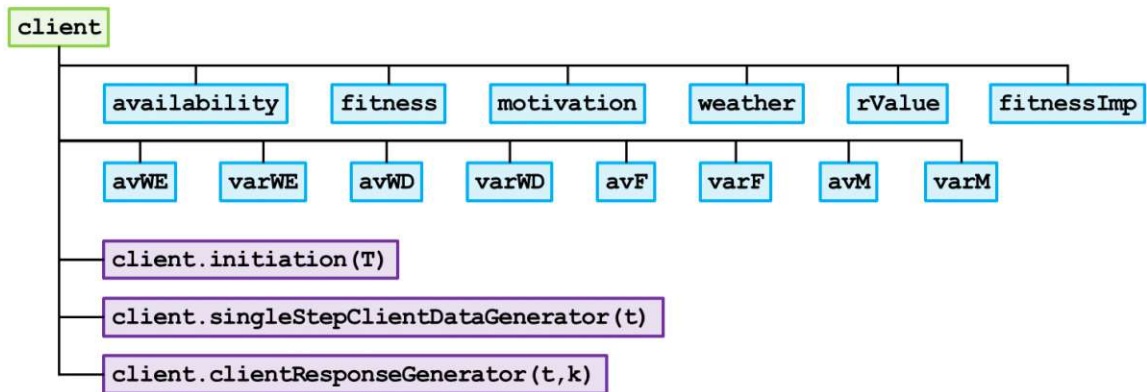


Figure 5.4: Implementation tree illustrating the MATLAB class properties (blue) and methods (purple) for clients.

Class Names	Description
<code>clientAvailability1</code>	Availability-sensitive client who decides based on the values of $\mathbf{a}_t$ , preferring low and medium intensity training when availability is high, and high intensity training when availability is low
<code>clientAvailability2</code>	Availability-insensitive client who decides based on $\mathbf{f}_t, \mathbf{m}_t, \mathbf{w}_t$
<code>clientFitness1</code>	Fitness-sensitive client who decides based on $\mathbf{f}_t$ , preferring low intensity training when fitness is low, and medium and high intensity training when fitness is high
<code>clientFitness2</code>	Fitness-insensitive client who decides based on $\mathbf{a}_t, \mathbf{m}_t, \mathbf{w}_t$
<code>clientMotivation1</code>	Motivation-sensitive client who decides based on $\mathbf{m}_t$ , and whose motivation is higher for three of the interventions
<code>clientMotivation2</code>	Motivation-sensitive client who decides based on $\mathbf{m}_t$ , and whose motivation is higher for the three other interventions, complementary to <code>clientMotivation1</code>
<code>clientMotivation3</code>	Motivation-sensitive client who is motivated for the same three interventions as <code>clientMotivation1</code> , but whose motivation depends linearly, but non-continuously on the weather, i.e., $\mathbf{m}_t$ is higher when $\mathbf{w}_t$ is high, and lower when $\mathbf{w}_t$ is low
<code>clientMotivation4</code>	Motivation-sensitive client who is motivated for the same three interventions as <code>clientMotivation3</code> , but whose motivation depends linearly, but non-continuously on the weather
<code>clientWeather1</code>	Weather-sensitive client who decides based on $\mathbf{w}_t$ , preferring outdoor training when the weather is good, and indoor training when the weather is bad
<code>clientWeather2</code>	Weather-insensitive client who decides based on $\mathbf{a}_t, \mathbf{f}_t, \mathbf{m}_t$

Table 5.3: List of all client MATLAB classes used in simulations.

The composition of specific clients is explained with different simulation setups in Chapter 6. Table 5.3 displays the names of the client classes used in simulation, and a short description of their specification, and Figure 5.4 gives the implementation tree of the client classes. The next section explains the architecture of the MATLAB code, and elaborates on the secondary files necessary for simulating the use of the model JITAI by a client.

### 5.3 The Code Architecture

The MATLAB code is targeted at providing viable simulation results. In terms of isolating singular features, each simulation setup is depicted in its own MATLAB script, which is divided into sections that can be run with different setups in order to progress through the

simulation. All scripts share several functions, such as the implementation of the TSRC algorithm, as well as the weather generator and plotting functions; they follow the same structure which is depicted below.

The first section is comprised of the initiation phase. Here, the simulation horizon  $T$ , the number of intervention options  $k$ , the number of features  $d$ , and the budget for the restricted context  $n$  are defined. Since performing Monte Carlo simulations is necessary due to the simulation setup, the number of Monte Carlo simulation runs  $s$  is fixed, as well as  $v$ ,

$$v = R\sqrt{\frac{24}{\epsilon}n\ln\left(\frac{1}{\delta}\right)},$$

for  $\epsilon, \delta \in (0, 1)$ , and  $R \geq 0$ . Note that  $v$  is part of the covariance matrix in the multivariate Gauss distribution in Algorithm 10, see Section 4.5. The parameter  $\alpha$  is also defined, which determines the size of the confidence interval for the resulting averaged values of the Monte Carlo simulation.

The second section contains the setup for a single simulation run with a generated client, to which the TSRC algorithm is then applied. The client is initialised by assigning it a client class, and the initiation method is used to preallocate the empty properties as vectors sized according to the simulation horizon. If needed, the weather generator function `weatherGenerator.m` subsequently generates a  $T$ -dimensional array of weather data, which overwrites the initiated default data for `client.weather`. The client's properties are then displayed in the command window.

### 5.3.1 The Weather Generator Function `weatherGenerator.m`

The weather generator function generates different weather scenarios for the simulation scripts, and requires the simulation horizon  $T$  and a weather scenario specification as inputs. There are four available options, one of which must be specified as a string in the function header: `good` sets  $w_t$  to 0.9 for all  $t \leq T$ , indicating  $T$  days of excellent weather, `bad` sets  $w_t$  to 0.1 for  $T$  days of dreadful weather, `yearSine` simulates changing weather conditions over the period of a year with the help of a sine function, and `year2009` provides normalised weather data from 2009 for a maximum of  $T = 365$  days.

The weather data given by the sine function follows the idea that weather (for outdoor sporting activities) is best in spring and autumn, and, due to excessive heat or cold, worst in summer and winter. Therefore, the absolute value of a sine function that stretches its period length over 365 days imitates these conditions in an idealised way, see Figure 5.5.

The normalised weather data from 2009 is sourced from Nemeč et al. [59], who homogenised weather data from Austrian weather stations per diem between 1948 and 2009,

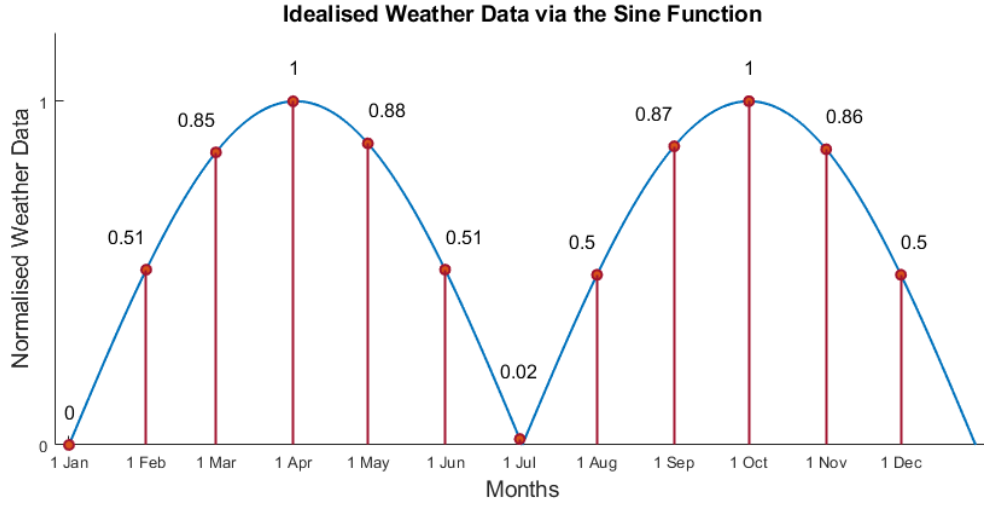


Figure 5.5: Illustration of the idealised weather data for one year via the sine function.

in order to analyse the resulting data sets in the light of climate change signals, and who have made their homogenised data freely available for academic research purposes.

The daily values for maximum temperature, minimum temperature, and precipitation, recorded by the weather station at Salzburg airport throughout 2009, are read into the workspace via the function `weatherNormalisation2009.m` from text files, and three arrays of length  $T$  are created,  $t_{max}$ ,  $t_{min}$ , and  $p$ , which contain the data values. Each vector is then normalised within itself to obtain  $t_{max}^N$ ,  $t_{min}^N$ , and  $p^N$ , for example:

$$p_t^N = \frac{p_t - \min\{p_t \mid t \in \tau\}}{\max\{p_t \mid t \in \tau\} - \min\{p_t \mid t \in \tau\}} \quad \forall t \in \tau = \{1, \dots, T\}$$

A comprehensive normalised weather vector  $\mathbf{w}$  is then obtained by taking the average of  $t_{max}^N$ ,  $t_{min}^N$ , and  $p^N$ , each day,

$$\mathbf{w}_t = \frac{t_{max_t}^N + t_{min_t}^N + p_t^N}{3} \quad \forall t \in \tau = \{1, \dots, T\},$$

since empirical research has proven that the average of the normalised vectors best represents the different weather conditions throughout the year.

Note that taking the additive inverse of the precipitation values (i.e.,  $1 - p_t^N$ ) is more coherent from a model perspective, because less precipitation generally implies better weather conditions. However, due to the nature of the normalisation formula, the shape of the data curve is barely altered when changing  $p_t^N$  to  $1 - p_t^N$  in the equation above. The data points are mainly pushed higher, which is expected to yield less explicit simulation outcomes. Note also that the data extraction can be altered to extract data from any other weather

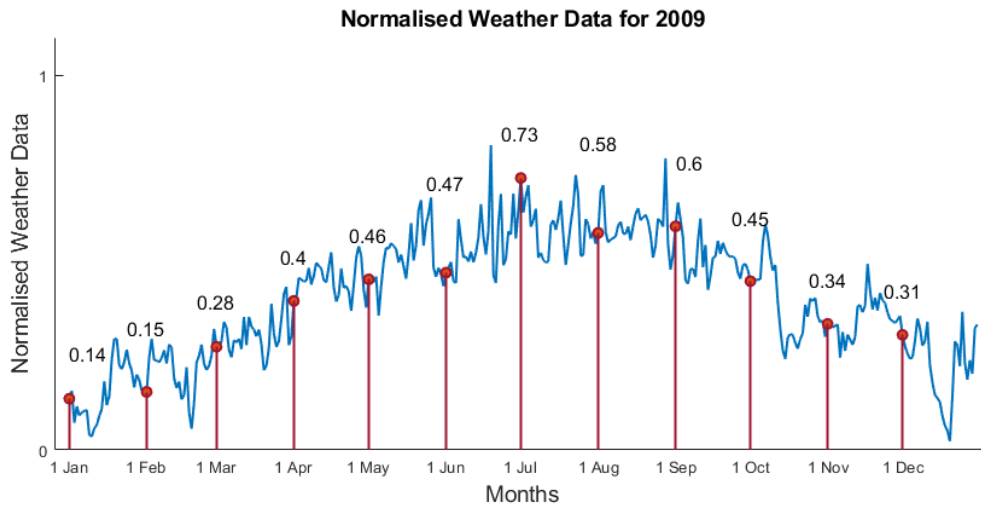


Figure 5.6: Illustration of the normalised weather data for 2009.

station, or any other year, by adjusting the detection markers and text files in the weather normalisation function. Figure 5.6 shows the normalised weather data for 2009. The resulting data does not mirror the general form of the idealised year, but shows that the weather conditions for outdoor activities are better in summer, and worse in winter.

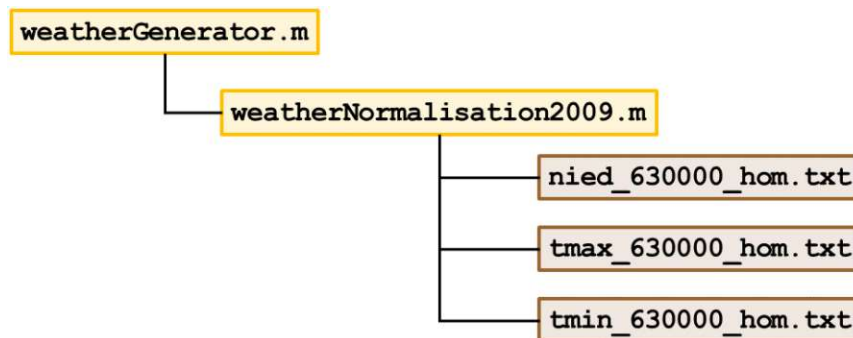


Figure 5.7: Implementation tree of `weatherGenerator.m`, showing the functions (yellow) and text files (brown) involved.

Figure 5.7 shows the implementation tree of the weather generator function, including the source files from which the normalised weather data from 2009 is taken.

### 5.3.2 The Function `TSRC.m`

After the weather data is generated and saved to the weather feature in the client's properties, the `TSRC` algorithm is run once with the provided data in order to test whether the simulation setup is functional.



The algorithm is implemented as function `TSRC.m` and follows the steps specified in Algorithm 10. The client object is needed as input, as well as  $T$ , the number of intervention options  $k$ , the number of features  $d$ , the feature budget  $n$ , and the value  $v$ . The option of passing a seed value to the function is available in order to obtain reproducible simulation results, but if no seed is needed, an empty array should be given as input parameter instead.

The function is also equipped for simulations with missing data (i.e., imitating the case when feature data cannot be accessed by the JITAI, for example due to technical failure). In that case, the attribute `missing` needs to be specified in the function handle, and the probability of missing data  $q$  must be passed to the function as a value between 0 and 1. For simulations during which data is assumed to be available for all features and all days, the attribute `default` is used, and the probability must be an empty array,  $q = []$ .

Note that the TSRC algorithm not only accesses a client's properties, but the methods for generating the client's feature data, `client.singleStepClientDataGenerator(t)`, and their response to activity suggestions, `client.clientResponseGenerator(t,k)`. The interaction between the JITAI policy and the environment (i.e., the modelled client) occurs through these methods. In a real-life setting, the feature data is obtained by measurements, and the client response is reported instead of generated.

The function `TSRC.m` outputs an array of length  $T$ , which holds the chosen activity recommended by the JITAI for each day. Furthermore, it returns an array of size  $T \times 2$  that records the features whose data is missing. The array has non-zero entries only if the attribute `missing` is chosen, and thus can be neglected in the default setting. The algorithm also gives an array of size  $T \times d$  back to the workspace, which holds the cumulative count of all features in terms of how often they have been selected during the simulation, as well as the client whose properties may have been altered by the algorithm.

After the TSRC algorithm has finished calculating, four plots are generated, including a double plot, which are implemented as functions and called upon in the script in order to portray the simulation results. Table 5.4 shows their function names, and a short description of what they display.

### 5.3.3 The Monte Carlo Simulation Function `monteCarloTSRC.m`

Since the TSRC algorithm is dependent on random numbers drawn from probability distributions, a single run of the algorithm function is not guaranteed to yield viable results due to the uncertainties involved in the calculation process. Thus, it is necessary to perform a *Monte Carlo* (MC) *simulation*, during which the TSRC algorithm is run several times under the same conditions (i.e., with the same client), and then to consider the average results in order to deduce valid statements. The third section of the script contains the

Plot Function	Description
<code>clientResponsePlot.m</code>	Illustration of which activity is chosen at each decision point, and whether the client has accepted, or declined
<code>armFrequencyPlot.m</code>	Display of the number of times each activity is chosen after $T$ simulation days, and how often it has been accepted, or declined
<code>cumulativeArmFrequencyPlot.m</code>	Double plot to give a cumulative representation of when each activity was selected, and whether the selected activity was accepted
<code>cumulativeFeatureSelectionPlot.m</code>	Cumulative representation of the feature selection in temporal progression

Table 5.4: List of plots generated after a single run of `TSRC.m`.

function `monteCarloTSRC.m`, which executes the MC simulation of the TSRC algorithm over a pre-defined number of runs  $s$ .

Determining the number of runs in a MC simulation is directly related to the accuracy of the result in terms of a confidence interval. Following the central limit theorem which states that, for a large number  $s$  of random variables, assumed to be independent and identically distributed with finite variance  $\sigma^2$  and mean  $\mu$ ,

$$\lim_{N \rightarrow \infty} \frac{\sqrt{N}(\bar{X}_s - \mu)}{\sigma} \sim \mathbf{N}(0, 1),$$

where  $\bar{X}_s$  is the sample mean of those random variables. This means that the percentiles of the Gauss distribution can be used to construct a confidence interval, which can be expected to be accurate enough if the sample size is sufficiently large.

Let  $\alpha$  be in  $[0, 1]$ , and  $\Phi_{1-\frac{\alpha}{2}}$  the respective  $(1 - \frac{\alpha}{2})$ -percentile of the Gauss distribution. The difference between the sample mean  $\bar{X}_s$  and the actual mean  $\mu$  of the sample population can be estimated via  $\alpha$ :

$$\begin{aligned} P\left(-\Phi_{1-\frac{\alpha}{2}} \leq \frac{\sqrt{s}(\bar{X}_s - \mu)}{\sigma} \leq \Phi_{1-\frac{\alpha}{2}}\right) &\approx 1 - \alpha \\ \Leftrightarrow P\left(\bar{X}_s - \Phi_{1-\frac{\alpha}{2}} \frac{\sigma}{\sqrt{s}} \leq \mu \leq \bar{X}_s + \Phi_{1-\frac{\alpha}{2}} \frac{\sigma}{\sqrt{s}}\right) &\approx 1 - \alpha \end{aligned}$$

Furthermore, the unknown variance  $\sigma^2$  can be approximated by the sample variance  $S_s$ , and the  $100(1 - \alpha)\%$  confidence interval for the value of  $\mu$  is thus given by [37]

$$\left[ \bar{X}_s - \Phi_{1-\frac{\alpha}{2}} \frac{\sqrt{S_s}}{\sqrt{s}}, \bar{X}_s + \Phi_{1-\frac{\alpha}{2}} \frac{\sqrt{S_s}}{\sqrt{s}} \right].$$

In terms of the MC simulation, the confidence interval is displayed in the plots which depict both the average activity choices and feature selection numbers, making it possible to deduce a grade of accuracy for the displayed results.

Since the MC simulation function calls on `TSRC.m` it requires the same input parameters, and, additionally, the number of simulation runs  $s$ . Again, a seed value can be passed to the function for reproducibility, and, in case of missing data, the correct attribute needs to be set and the probability  $q$  must again be passed to the function. Note that the seed value is appointed within the MC function, but outside of the `TSRC` function, which is run without a seed. This way, the  $s$  simulation runs yield varying results, but the variety of results, and thus their average is reproduced when calling `monteCarloTSRC.m` again with an identical seed value. An array of size  $s \times T$  that holds the daily activity choices per run, and a three-dimensional array of size  $T \times d \times s$  holding the cumulative feature choice matrices are returned to the script as output. Figure 5.8 displays the implementation tree of the MC function, including all functions, classes, properties, and methods that are called during a MC simulation.

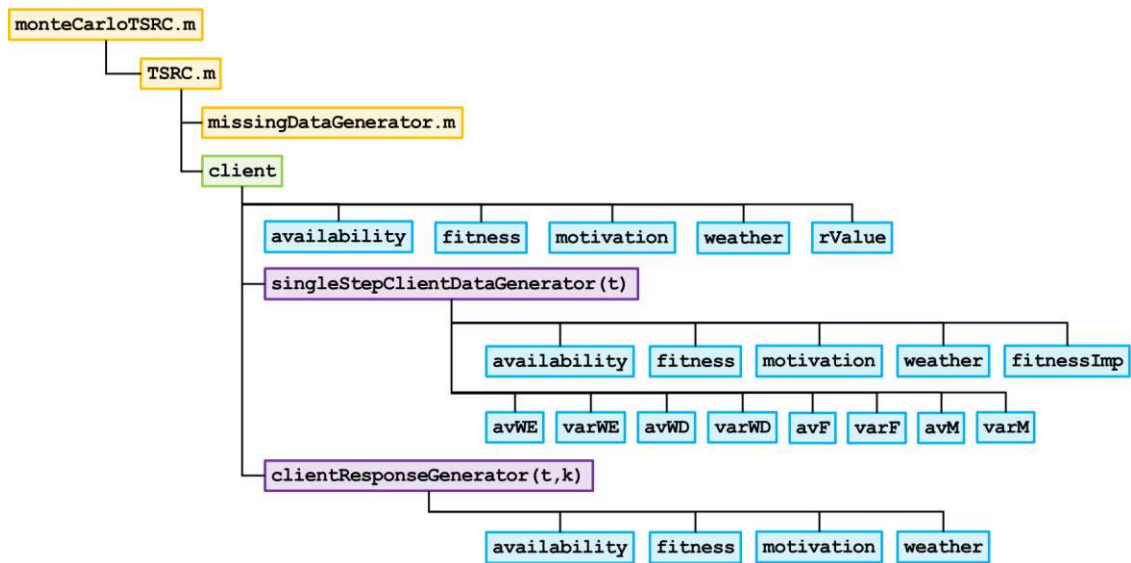


Figure 5.8: Implementation tree of `monteCarloTSRC.m`, showing the functions, client properties, and methods called during a MC simulation.

Note that there is another distinction in terms of conclusiveness between the MC simulation results and the results of one `TSRC` simulation run. For example, whether some activity suggestions are chosen more frequently than others may be displayed conclusively by either setting, but a representation of which arm was offered at  $t \leq T$ , and how the client responded, cannot be averaged comprehensively. Under these circumstances, a singular simulation run may yield informative results that are not covered by the MC simulation.

Plot Function	Description
<code>armMonteCarloPlot.m</code>	Average number of activity choices after $T$ trials, and the borders of the $100(1 - \alpha)\%$ confidence intervals, as well as the average number of activity acceptances after $T$ trials, and the borders of the $100(1 - \alpha)\%$ confidence intervals
<code>featureMonteCarloPlot.m</code>	Average of the number of times each feature was selected after $T$ trials, and the borders of the $100(1 - \alpha)\%$ confidence intervals

Table 5.5: List of plots generated after a MC simulation with `TSRC.m`.

After the MC simulation is completed, two plots are generated via functions within the script, see Table 5.5. The MC simulation setting can be used to compare results between various client setups, sparsity levels, and instances of missing data.

In terms of isolating a feature the initial approach is to compare a feature-sensitive client to a feature-insensitive client. The feature-sensitive client is designed in such a way that it bases the decision about accepting an activity suggestion solely on the value of the isolated feature, see the example of the weather-sensitive client in Section 5.2. The corresponding feature-insensitive client balances the decision equally between the other three features. For example, the response vector  $x(t) = (x_1(t), \dots, x_6(t))$  generated by `client.clientResponseGenerator(t,k)` for the weather-insensitive client is

$$\begin{aligned}
 x_1(t) &\sim \text{Bernoulli}(l_I(t)), & l_I(t) &= \frac{\mathbf{f}_t + \mathbf{m}_t + (1 - \mathbf{a}_t)}{3}, \\
 x_2(t) &\sim \text{Bernoulli}(l_O(t)), & l_O(t) &= \frac{\mathbf{f}_t + \mathbf{m}_t + (1 - \mathbf{a}_t)}{3}, \\
 x_3(t) &\sim \text{Bernoulli}(m_I(t)), & m_I(t) &= \frac{\mathbf{f}_t + \mathbf{m}_t + (1 - \mathbf{a}_t)}{3}, \\
 x_4(t) &\sim \text{Bernoulli}(m_O(t)), & m_O(t) &= \frac{\mathbf{f}_t + \mathbf{m}_t + (1 - \mathbf{a}_t)}{3}, \\
 x_5(t) &\sim \text{Bernoulli}(h_I(t)), & h_I(t) &= \frac{\mathbf{f}_t + \mathbf{m}_t + (1 - \mathbf{a}_t)}{3}, \\
 x_6(t) &\sim \text{Bernoulli}(h_O(t)), & h_O(t) &= \frac{\mathbf{f}_t + \mathbf{m}_t + (1 - \mathbf{a}_t)}{3}.
 \end{aligned}$$

Note that the general assumption is that  $\mathbf{a}$  is inversely proportional to the fitness level, see Section 5.1.

When considering different levels of sparsity, there are five options for  $d = 4$  depicted in Table 5.6. Note that  $n = 3$  is considered the standard setting for all simulation runs unrelated to sparsity investigation. Furthermore, due to the preallocation of variables that

require the value of  $n$ , `TSRC.m` cannot work with a sparsity of 100%. However, results can be compared for  $0 < n \leq d$ .

n	Sparsity	Comments
0	100%	Simulation with <code>TSRC.m</code> not possible due to allocation problem, equals the case of a multi-armed bandit without context
1	75%	Simulation with <code>TSRC.m</code> possible
2	50%	Simulation with <code>TSRC.m</code> possible
3	25%	Simulation with <code>TSRC.m</code> possible
4	0%	Simulation with <code>TSRC.m</code> possible, equals the classical TS algorithm with full features, see Algorithm 7

Table 5.6: Sparsity levels for the TSRC algorithm.

In the light of restricted context, or missing data, an analysis of how much sparsity still yields results similar to the full-featured case is necessary in order to argue for the restricted context setting as a decision rule in JITAI recommender systems.

The MC simulation also allows the analysis of simulation results of feature-sensitive clients in the missing data setting, compared to the same client in the default setting. In this case, the attribute `missing` must be specified in the function handle of `monteCarloTSRC.m`, which calls upon the TSRC algorithm with the same attribute, and  $q \in [0, 1]$  must be specified as well. If missing data is regarded, the algorithm performs a side step after sampling

$$\theta_j \sim \mathbf{Beta}(S_j, F_j), \quad j \in \{1, \dots, d\}$$

in the combinatorial bandit setting (that determines which  $n$  feature values are observed by the contextual bandit), see Algorithm 10. In each iteration, missing data may occur with probability  $q$ , which is modelled by the function `missingDataGenerator.m` that generates a random number via `rand()`, and compares it to  $q$ :

$$y = \begin{cases} 1 & \text{if } \mathbf{rand}() < q \\ 0 & \text{else} \end{cases}$$

If  $y = 1$ , one of the  $d$  features is randomly chosen to exhibit the lack of data, for example feature  $i \in \{1, \dots, d\}$ , and the drawn value of  $\theta_i$  is overwritten:

$$\theta_i = 0$$

This way, feature  $i$  is guaranteed not to be selected for observation in that iteration, because the TSRC algorithm picks the features with the  $n$  largest values. However, in this implementation, the respective feature value  $i_t$  is neither deleted nor altered in any other way, but remains intact.

## 6 Simulation

This chapter describes the setup and results of various simulation runs with the model JITAI from Section 5.1, where the applicability of the TSRC algorithm as a decision rule within the JITAI framework is investigated with the use of model clients from Table 5.3.

First, simulations for the isolated features are examined, ordered in terms of increasing complexity in implementation efforts, see Sections 6.1, 6.2, 6.3, and 6.4. Then, comprehensive simulation results are observed across all features in order to investigate varying levels of sparsity, see Scenario 6.5, and the case of missing data, see Scenario 6.6. The different simulation scenarios are enclosed in separate sections, and, within each section, the setup for the simulation, the simulation results, and the discussion on the outcome, are regarded individually.

### 6.1 The Weather Feature

The weather feature is chosen as the starting point, because the client classes used for the weather feature simulations are simplest regarding construction, compared to any other client class from Table 5.3.

Two classes exist for the weather feature, `clientWeather1` which defines a weather-sensitive client (i.e., a client that is susceptible only to the weather feature, and none of the others), and `clientWeather2` which is the complementary, or antagonistic, weather-insensitive client (i.e., a client that is susceptible to all features but the weather). In both cases, the default values for all but one client property are kept after employing the initiation method: only the weather property is set to one of the four weather scenarios given by `weatherGenerator.m`, see Section 5.3.1.

Note that the weather is the only feature not explicitly dependent on the client, because the client's behaviour, or interaction with the model JITAI, is assumed to have no influence over the weather, so the values for  $w_t$  are known for all  $t \leq T$  before the simulation begins. This implies that the client does not change location for the duration of the simulation, or that the change in the client's location is already factored into the simulation. For easier handling, the former is assumed.

### 6.1.1 Comparing Different Weather Scenarios for a Weather-Sensitive Client

In order to compare different weather scenarios, `clientWeather1` is used in a single simulation run, and subsequently for a MC simulation for each weather type.

#### Simulation Setup

The general parameters for the simulation, as explained in Section 5.3, are listed in Table 6.1. Note that, after fixing the values for  $R$ ,  $\epsilon$ , and  $\delta$ ,  $v$  is calculated by the formula given in Section 4.5. Also note that these parameter values are unchanged during all simulations of the isolated features, namely Scenarios 6.1.2, 6.2.1, 6.2.2, 6.3.1, 6.3.2, 6.4.2, and 6.4.1.

$T = 365$	$n = 3$	$R = 0.1, \delta = 0.8, \epsilon = 0.9$
$k = 6$	$s = 200$	$v \approx 0.4225$
$d = 4$	$\alpha = 0.05$	

Table 6.1: Simulation parameters for Scenario 6.1.1.

`clientWeather1` is initiated with standard parameters according to the initiation method `client.initiation(T)`, and only the weather data varies between simulation runs. Table 6.2 shows the default values of the client properties from Table 5.3 after the initiation method, but before overwriting the weather data with the output of the weather generator function.

$\mathbf{f} = \mathbf{0} \in \mathbb{R}^T$	$\mathbf{r} = \mathbf{0} \in \mathbb{R}^T$	$var_{WD} = 0.5$	$av_M = 0.5$
$\mathbf{m} = \mathbf{0} \in \mathbb{R}^T$	$av_{WE} = 0.5$	$av_F = 0.5$	$var_M = 0.5$
$\mathbf{a} = \mathbf{0} \in \mathbb{R}^T$	$var_{WE} = 0.5$	$var_F = 0.5$	$f^* = 4$
$\mathbf{w} = (0.5, \dots, 0.5) \in \mathbb{R}^T$	$av_{WD} = 0.5$		

Table 6.2: Class property values for `clientWeather1` in Scenario 6.1.1.

The weather sensitivity of the client manifests in their response to the activity suggestions. Thus, it is defined within the method `client.clientResponseGenerator(T, k)`, where the expected probability for accepting the activity suggestion for indoor training is given by

$$l_I(t) = m_I(t) = h_I(t) = 1 - \mathbf{w}_t,$$

and the expected probability for accepting outdoor training activities is given by

$$l_O(t) = m_O(t) = h_O(t) = \mathbf{w}_t.$$



Weather Scenario	Description	Function Attribute	Feature Values Seed Values
Good weather	$T$ days of great weather conditions in terms of outdoor activities	good	$\mathbf{w} = (0.9, \dots, 0.9) \in \mathbb{R}^T$ seeds: (157, 8)
Bad weather	$T$ days of awful weather conditions in terms of outdoor activities	bad	$\mathbf{w} = (0.1, \dots, 0.1) \in \mathbb{R}^T$ seeds: (616, 51054)
Ideal weather	$T$ days of slowly changing weather conditions simulated by $ \sin(x) $ over a period of 365 days, see Figure 5.5	yearSine	$w_t =  \sin(\frac{2\pi t}{365}) $ seeds: (72253, 45689)
2009 weather	Normalised weather data from 2009, provided by Nemeč et al. [59]	year2009	see Figure 5.6 seeds: (1169, 42)

Table 6.3: Weather feature values and seeds (single run, MC) for `clientWeather1` in Scenario 6.1.1.

Table 6.3 provides a description of the different weather scenarios being investigated, and includes the seed values for each simulation, so that the results can be reproduced.

## Results

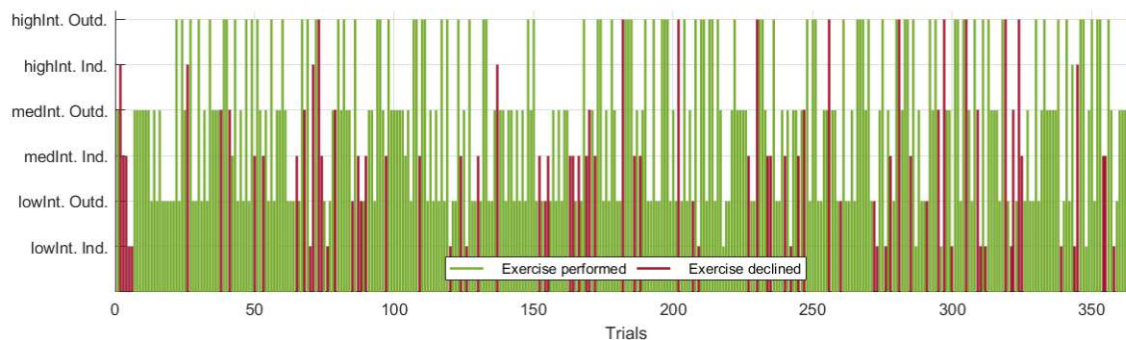
Single runs of the TSRC algorithm with the weather-sensitive client show a clear affinity towards agreeing to outdoor activities for the good weather setting, and also display a similar inclination for indoor activities in the bad weather setting, see Figure 6.1. For example, in Figure 6.1b, the client’s preference for indoor activities manifests distinctly: during the first 50 days, outdoor activities are repeatedly suggested to the client and subsequently declined, whereas indoor activities are mostly accepted.

good	lowInt. ind.	lowInt. outd.	medInt. ind.	medInt. outd.	highInt. ind.	highInt. outd.
activity suggestion	$17 \pm 2$	$100 \pm 3$	$20 \pm 2$	$105 \pm 3$	$18 \pm 2$	$104 \pm 3$
activity acceptance	$2 \pm 2$	$91 \pm 3$	$2 \pm 2$	$95 \pm 3$	$2 \pm 2$	$94 \pm 3$

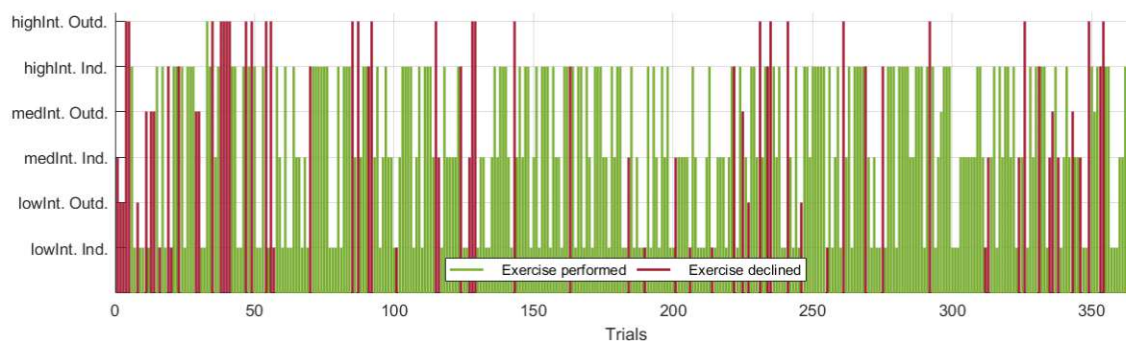
Table 6.4: MC simulation results for good weather and `clientWeather1`.

The same tendency can be seen in the MC simulation for both scenarios, where, during the year of good weather, the client prefers outdoor activities on average, and, as a consequence, outdoor activities are suggested more regularly. For 200 runs, the rounded mean

values and 95% confidence intervals (CIs) of how often an activity has been chosen and accepted are displayed in Table 6.4. Note that the variables from the client class definition are used to represent the activities.



(a) Activity selection and acceptance, good weather scenario.



(b) Activity selection and acceptance, bad weather scenario.

Figure 6.1: Client response for single runs of `TSRC.m` and `clientWeather1`.

During the year of bad weather, the algorithm adjusts accordingly throughout the MC simulation and suggests indoor activities more frequently since these are favoured on average. The rounded means and CIs of the MC simulation are shown in Table 6.5.

<b>bad</b>	lowInt. ind.	lowInt. outd.	medInt. ind.	medInt. outd.	highInt. ind.	highInt. outd.
activity suggestion	107 ± 4	16 ± 2	104 ± 4	14 ± 2	110 ± 4	15 ± 2
activity acceptance	96 ± 4	2 ± 2	94 ± 4	1 ± 2	99 ± 4	1 ± 2

Table 6.5: MC simulation results for bad weather and `clientWeather1`.

The more realistic seasonal weather scenarios, namely the ideal weather and the weather from 2009, yield less specific results about activity suggestions when considering only one simulation run. Generally, the MC simulation is more significant. However, the regularity of the seasonal change can be seen when looking at the cumulative number of activity

suggestions and acceptances of a single simulation run in the ideal weather setting, see Figure 6.2.

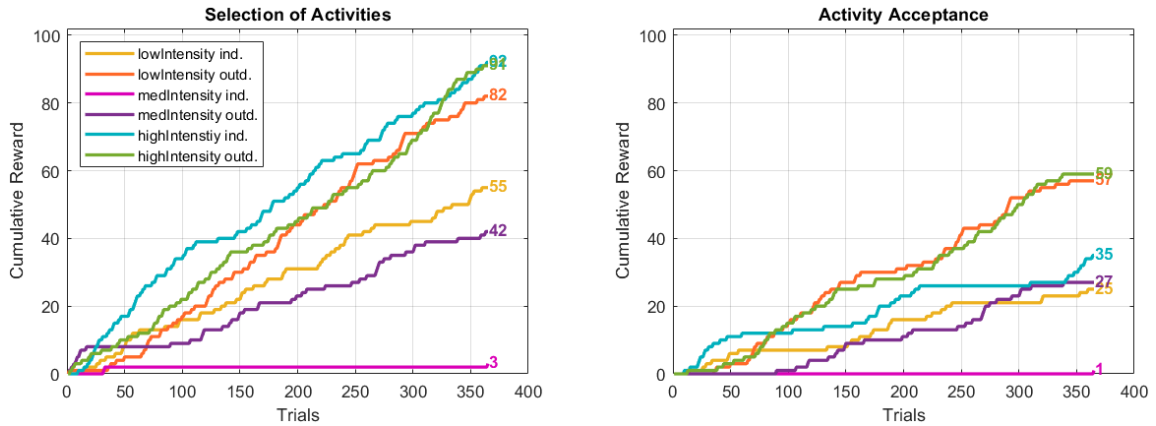
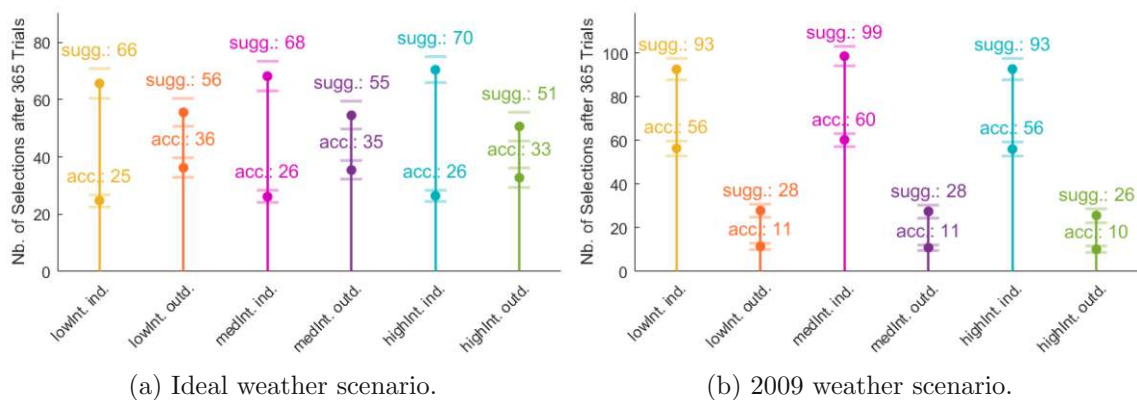


Figure 6.2: Client response to activity suggestions over time, ideal weather setting.

The left plot shows the cumulative activity selection (i.e., how often the TSRC algorithm chooses the respective activities) over time, with high intensity indoor and outdoor training being the most recommended activities. The client’s response to the activity suggestions, which is depicted on the right, shows that the client reacts in waves to different suggestions. For example, the high intensity indoor activity is accepted frequently during the first 60 days, then it is declined between days 60 and 160, before acceptance surges again until around day 210. It is further rejected until around day 339, and, before the year finishes, it is being accepted again by the client. Note that, by comparing both plots, it can be seen that the TSRC algorithm recommends the activity during the stagnant phases of the acceptance plot, but the client mostly declines.



(a) Ideal weather scenario.

(b) 2009 weather scenario.

Figure 6.3: MC simulation results for clientWeather1 after 365 days for ideal and 2009 weather.

The shape of this curve is mirrored loosely by the low intensity indoor training, and the outdoor activities exhibit the same behaviour, only they climb in acceptance when the indoor activities are stagnant, showing that the changing seasonality is registered by the TSRC algorithm which in turn reacts to it promptly.

<code>year2009</code>	lowInt. ind.	lowInt. outd.	medInt. ind.	medInt. outd.	highInt. ind.	highInt. outd.
activity suggestion	93 ± 5	28 ± 3	99 ± 5	28 ± 3	93 ± 5	26 ± 3
activity acceptance	56 ± 5	11 ± 3	60 ± 5	11 ± 3	56 ± 5	10 ± 3

Table 6.6: MC simulation results for 2009 weather and `clientWeather1`.

Such explicit comparisons between activity suggestions cannot be derived from a MC simulation. In fact, the MC simulation outcome, as depicted in Figure 6.3a, for the ideal year is slightly biased towards outdoor activities when viewing the respective numbers of activity acceptances. This is because the number of times an activity is suggested, or accepted, is counted after the simulation is complete, and not cumulatively over time, so the seasonality cannot be displayed explicitly.

<code>yearSine</code>	lowInt. ind.	lowInt. outd.	medInt. ind.	medInt. outd.	highInt. ind.	highInt. outd.
activity suggestion	67 ± 5	52 ± 5	67 ± 5	51 ± 5	68 ± 5	60 ± 5
activity acceptance	26 ± 5	34 ± 5	25 ± 5	34 ± 5	26 ± 5	40 ± 5

Table 6.7: MC simulation results for ideal weather and `clientWeather1`.

The rounded means and CIs of the ideal weather scenario, and the 2009 weather scenario, are portrayed in Tables 6.6, and 6.7. Furthermore, there is no obvious bias concerning the preference of certain features during the contextual bandit search in any weather scenario. Table 6.8 shows the rounded means and CIs of the feature selection after 200 runs, for each weather scenario specifically.

Features	<code>good</code>	<code>bad</code>	<code>yearSine</code>	<code>year2009</code>
Weather	252 ± 15	280 ± 12	279 ± 10	275 ± 10
Fitness	273 ± 13	279 ± 23	273 ± 10	274 ± 10
Availability	285 ± 12	272 ± 13	273 ± 9	267 ± 11
Motivation	286 ± 11	264 ± 14	270 ± 10	279 ± 9

Table 6.8: Rounded means and CIs of feature selection for all weather scenarios and `clientWeather1`.

## Discussion

The simulation results indicate that the TSRC algorithm correctly interprets the weather-sensitive client's preferences during simulation runs with consistent weather conditions (i.e., for good and bad weather). With changing seasonality, the client can be observed to behave according to their pre-defined inclinations, see Figure 6.2, preferring indoor activities when the weather is bad, and outdoor activities when the weather is good. However, the TSRC algorithm does not react fast enough to learn these preferences in a seasonal setting, which is illustrated by the difference in numbers between activity selections and acceptances in both seasonal weather scenarios, see Tables 6.6 and 6.7.

In the ideal weather scenario, the TSRC algorithm seems to grossly misjudge the client's attitude towards indoor activities. The ideal weather data is symmetrical and its mean value over 365 days is approximately 0.63, which correctly results in a slight favour for outdoor activities on the client's side, see Figure 6.3a. An explanation for the overly frequent recommendations of indoor activities may be that the TSRC algorithm builds up a bias towards phases of bad weather, which is represented in the weather data at the beginning of the simulation, see Figure 5.5.

This phenomenon is not as prominent in the 2009 weather scenario, which also starts with weather data indicating bad weather, see Figure 5.6. Figure 6.3b shows a clear preference for indoor activities in activity suggestions and acceptances, which draws attention to the weather data values obtained by the weather generator function. The minimum of  $w$  is given by

$$w_{\min} = 0.0272$$

on day 354 of the year, which is the 20th of December in a non-leap year, so it almost coincides with the day of the winter solstice. The maximum of  $w$  is

$$w_{\max} = 0.8186$$

on the 170th day of the year, namely the 19th of June, around the time of the summer solstice. However, the average of  $w$  is approximately 0.39, indicating that the weather conditions for outdoor training are generally not agreeable throughout the year, so even if the bias towards bad weather is affecting the activity selection, it does not cause unrealistic or openly skewed results.

Future work may investigate whether the phenomenon observed for the ideal weather scenario occurs due to a bias formed by the TSRC algorithm, and if so, how much it influences the activity selection process. For example, it is possible to generate a weather scenario with good weather in the beginning (i.e., for the first three months) and bad

weather throughout the rest of the year, in order to examine these simulation results.

The response of the TSRC algorithm to seasonality requires further examination in general. Since Scenario 6.1.1 shows a good performance of the TSRC algorithm as a decision rule throughout consistent weather conditions, its behaviour during seasonal changes may be investigated by generating weather data that either progressively improves, or declines, and by considering the MC simulation results.

### 6.1.2 Comparing a Weather-Sensitive Client to a Weather-Insensitive Client

The results of the single simulation run and the MC simulation for the weather-sensitive client from Scenario 6.1.1 are compared to a client from the complementary, weather-insensitive client class `clientWeather2`.

#### Simulation Setup

The simulation setup for client `clientWeather2` is mostly congruent with the simulation setup for the weather-sensitive client in Scenario 6.1.1. The general simulation parameters are identical, as are the class property values for `weatherClient2`, so they can be found in Tables 6.1, and 6.2.

The weather data for the weather-insensitive client varies between simulation runs as well. Each weather scenario will be run with `clientWeather2`. The only difference to the weather-sensitive client lies in method `client.clientResponseGenerator(t,k)`. The weather-sensitive client relies on all features but the weather, so the probability for accepting any activity is given by:

$$l_I(t) = l_O(t) = m_I(t) = m_O(t) = h_I(t) = h_O(t) = \frac{f_t + m_t + (1 - a_t)}{3}$$

Note that these three features are weighted equally. The seeds for the simulation runs are also identical to those of Scenario 6.1.1 in order to make the results as comparable as possible.

#### Results

Contrary to the weather-sensitive client, the weather-insensitive client does not show preference for any particular activity. Since the daily weather conditions are not regarded by the client when making a decision about the activity suggestion, and all suggestions are weighted equally, this outcome is expected.

Figure 6.4 shows the results of the MC simulations for good and bad weather, respectively. Contrary to the weather-sensitive case, it is not apparent which scenario is depicted, based

## 6 Simulation

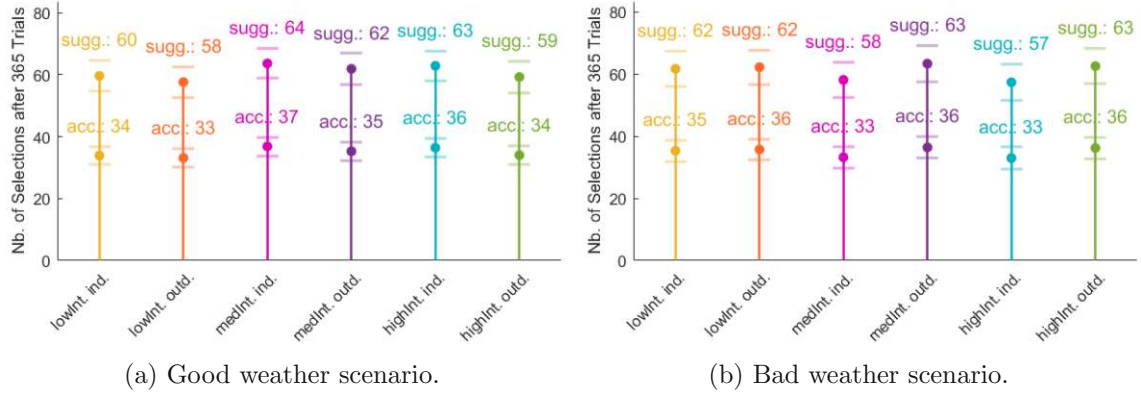


Figure 6.4: MC simulation results for `clientWeather2` after 365 days for good and bad weather.

on the graphs alone. Table 6.9 displays the rounded means and CIs for the good weather scenario, and Table 6.10 shows the same for the bad weather scenario. The rounded means are almost equally distributed between different activities, and the CIs are slightly larger compared to the weather-sensitive cases, see Tables 6.4, and 6.5.

Almost identical results are obtained when running the seasonal weather scenarios, see Tables 6.11 and 6.12, confirming again that, as long as the weather is not regarded, there will be no weather-related bias formed by the TSRC algorithm.

good	lowInt. ind.	lowInt. outd.	medInt. ind.	medInt. outd.	highInt. ind.	highInt. outd.
activity suggestion	60 ± 5	58 ± 5	64 ± 5	62 ± 5	63 ± 5	59 ± 5
activity acceptance	34 ± 5	33 ± 5	37 ± 5	35 ± 5	36 ± 5	34 ± 5

Table 6.9: MC simulation results for good weather and `clientWeather2`.

Note that, in all scenarios but 2009, the number of days on which the activity suggestion is accepted is lower compared to the weather-sensitive client. The scenario for 2009 yields only 204 acceptances in the weather-sensitive case, and 206 in the weather-insensitive case.

bad	lowInt. ind.	lowInt. outd.	medInt. ind.	medInt. outd.	highInt. ind.	highInt. outd.
activity suggestion	62 ± 6	62 ± 6	58 ± 6	63 ± 6	57 ± 6	63 ± 6
activity acceptance	35 ± 6	36 ± 6	33 ± 6	36 ± 6	33 ± 6	36 ± 6

Table 6.10: MC simulation results for bad weather and `clientWeather2`.

In the bad weather scenario, out of 365 days, the client has only accepted the intervention

209 times on average, see Table 6.10, compared to 293 times for the weather-sensitive client, see Table 6.5. In fact, the average number of acceptances is between 206 and 210 for all four weather scenarios, so 56 – 57% of suggestions are accepted by the client.

<code>year2009</code>	lowInt. ind.	lowInt. outd.	medInt. ind.	medInt. outd.	highInt. ind.	highInt. outd.
activity suggestion	63 ± 5	64 ± 6	59 ± 5	61 ± 6	63 ± 6	55 ± 5
activity acceptance	36 ± 5	36 ± 6	33 ± 5	35 ± 6	35 ± 6	31 ± 5

Table 6.11: MC simulation results for 2009 weather and `clientWeather2`.

<code>yearSine</code>	lowInt. ind.	lowInt. outd.	medInt. ind.	medInt. outd.	highInt. ind.	highInt. outd.
activity suggestion	62 ± 5	61 ± 5	59 ± 5	66 ± 5	60 ± 5	57 ± 5
activity acceptance	36 ± 5	35 ± 5	34 ± 5	38 ± 5	34 ± 5	33 ± 5

Table 6.12: MC simulation results for ideal weather and `clientWeather2`.

The number of times each feature is chosen during the combinatorial bandit is displayed as rounded means and CIs in Table 6.13. Similar to the weather-sensitive case, there is no clear preference obvious in the choosing of features, and the widths of the CIs are also comparable to the results from Scenario 6.1.1.

Features	<code>good</code>	<code>bad</code>	<code>yearSine</code>	<code>year2009</code>
Weather	285 ± 9	278 ± 10	267 ± 11	274 ± 11
Fitness	275 ± 11	271 ± 11	279 ± 10	274 ± 11
Availability	270 ± 11	274 ± 11	278 ± 10	274 ± 11
Motivation	265 ± 11	272 ± 11	270 ± 10	273 ± 11

Table 6.13: Rounded means and CIs of feature selection for all weather scenarios and `clientWeather2`.

## Discussion

The complementary weather-insensitive client is used to create counter-evidence for the dynamics of the TSRC algorithm regarding the weather feature. Scenario 6.1.1 shows that, when isolating the weather feature, the TSRC algorithm correctly identifies a client’s activity preferences. Scenario 6.1.2 proves that the TSRC algorithm does not prioritise activities associated with the weather when the weather feature is not regarded, see Figure



6.4. Since the other three features are of equal importance to the client, and the client response is identical between activities, no preference for any activity is expected.

The simulation results deliver exactly this; regardless of the chosen weather scenario, all activities are recommended almost equally often, and the weather-insensitive client accepts them in the same manner, without singling out any distinct favourites.

The percentage of activity acceptances throughout all weather scenarios is comprehensible when looking at the methods of the client class `clientWeather2`. The default values for the simulation parameters that are responsible for client data generation in method `client.singleStepClientDataGenerator(t)` are set to 0.5, and the success probability in `client.clientResponseGenerator(t,k)` is the average of all features values except the weather. This means that the probability of the client accepting a suggestion is, on average, around 50%. However, during simulation, the client's fitness may improve, slightly raising the average fitness value  $av_F$  and diminishing the varying interval  $var_F$ , which accounts for raising the success probability of accepting a suggestion to 56 – 57% for all activities, just slightly above the 50% mark.

It can be concluded that the the weather should be considered in a JITAI recommender system that targets exercise, because it is conceivable that the weather is an important factor for clients when deciding whether or not to perform the suggested activity. Furthermore, the TSRC algorithm has been proven to correctly distinguish between different client preferences regarding the weather feature in isolation.

## 6.2 The Fitness Feature

When isolating the fitness feature, the aim is to investigate the TSRC algorithm's response to clients who only consider their own fitness level when deciding whether or not to accept an exercise suggestion. Subsequently, such clients are compared to antagonistic clients, who consider all features but the fitness feature when making a decision.

Again, two classes exist for simulation. The fitness-sensitive class `clientFitness1` is identical to the weather-sensitive client class in all but one regard: instead of depending only on the weather feature, the probability for accepting an activity suggestion depends only on the client's fitness level. The complementary fitness-independent client class `clientFitness2` is similar to the weather-insensitive client class, because its success probability for accepting a suggestion depends on all features but the fitness level.

Note that, in order to avoid weather-related bias, the default weather value of 0.5 for each day is not overwritten with a weather scenario from the weather generator function, resulting in a year of mediocre weather.

### 6.2.1 Comparing Different Fitness Levels

The client class `clientFitness1` is used for single simulation runs and MC simulations for clients with different fitness levels.

#### Simulation Setup

The general simulation parameter values are unchanged, and can be found in Table 6.1. Scenario 6.2.1 examines the response of the TSRC algorithm to a fit client, compared to an unfit client. In both cases, the client is initiated with standard parameters according to method `client.initiation(T)`. Then, the base fitness value  $av_F$ , and the varying fitness interval width  $var_F$ , are overwritten in the workspace. Table 6.14 shows the property values for the fit client before a simulation run.

$\mathbf{f} = 0 \in \mathbb{R}^T$	$\mathbf{r} = 0 \in \mathbb{R}^T$	$var_{WD} = 0.5$	$av_M = 0.5$
$\mathbf{m} = 0 \in \mathbb{R}^T$	$av_{WE} = 0.5$	$av_F = 0.9$	$var_M = 0.5$
$\mathbf{a} = 0 \in \mathbb{R}^T$	$var_{WE} = 0.5$	$var_F = 0.8$	$f^* = 3$
$\mathbf{w} = (0.5, \dots, 0.5) \in \mathbb{R}^T$	$av_{WD} = 0.5$		

Table 6.14: Class property values for fit `clientFitness1` in Scenario 6.2.1.

Note that the fitness level of a client is influenced by two factors: the values  $av_F$  and  $var_F$ , which are responsible for generating the fitness value  $\mathbf{f}_t$ , and the weekly fitness improvement threshold  $f^*$ , which indicates how easily the client's fitness improves. When creating a fit client,  $av_F$  is assumed to be close to 1, and the threshold  $f^*$  is assumed to be low in order to increase  $av_F$  quicker.

$\mathbf{f} = 0 \in \mathbb{R}^T$	$\mathbf{r} = 0 \in \mathbb{R}^T$	$var_{WD} = 0.5$	$av_M = 0.5$
$\mathbf{m} = 0 \in \mathbb{R}^T$	$av_{WE} = 0.5$	$av_F = 0.3$	$var_M = 0.5$
$\mathbf{a} = 0 \in \mathbb{R}^T$	$var_{WE} = 0.5$	$var_F = 0.3$	$f^* = 6$
$\mathbf{w} = (0.5, \dots, 0.5) \in \mathbb{R}^T$	$av_{WD} = 0.5$		

Table 6.15: Class property values for unfit `clientFitness1` in Scenario 6.2.1.

Table 6.15 shows the property values of the unfit client. Note that the fitness improvement threshold is raised as well. Note further that  $f^* > 7$  cause the values for  $av_F$  and  $var_F$  to be stagnant, since, due to the frequency of the decision point, exercising more than seven times per week is not possible.

The fitness of a client is assumed to be directly proportional to the intensity of the exercise. Since there are three intensity levels in the pool of activity suggestions, the

method `client.clientResponseGenerator(t,k)` is constructed in a way that low fitness values encourage low intensity exercise,

$$l_I(t) = l_O(t) = 1 - \mathbf{f}_t,$$

and high fitness values animate medium and high intensity training:

$$m_I(t) = m_O(t) = h_I(t) = h_O(t) = \mathbf{f}_t$$

These formulae are fixed regardless of the client's fitness. Table 6.16 gives the seed values for the single simulation runs and the MC simulations. Note that the same seed is used for both the fit, and the unfit client.

fit & unfit	Single run	MC simulation
Seed	6030	89

Table 6.16: Seed values for `clientFitness1` in Scenario 6.2.1.

## Results

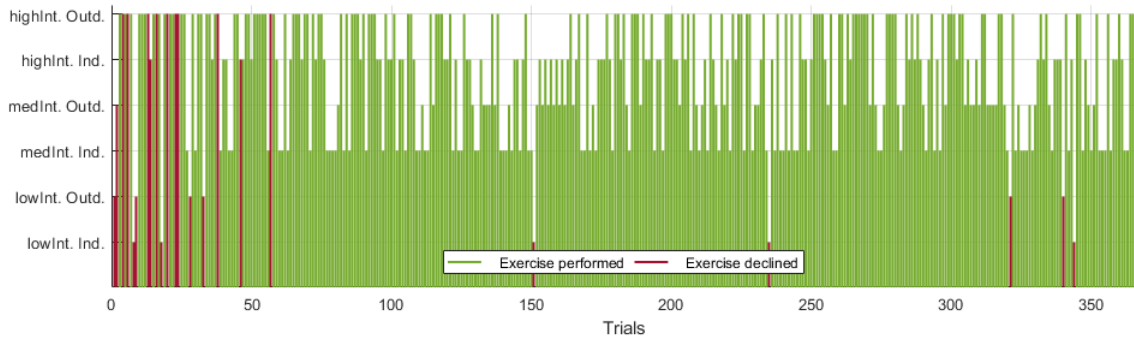
Single runs of the TSRC algorithm show a clear tendency towards fitness-appropriate activities in both the fit and the unfit client, see Figure 6.5

Within the first 50 days, the fit client refuses low and high intensity training, see Figure 6.5a, the first because of their higher fitness level, and the second because their fitness level is not yet high enough. However, after day 57, the medium and high intensity suggestions are always accepted, and the low intensity training is refused if it is at all suggested.

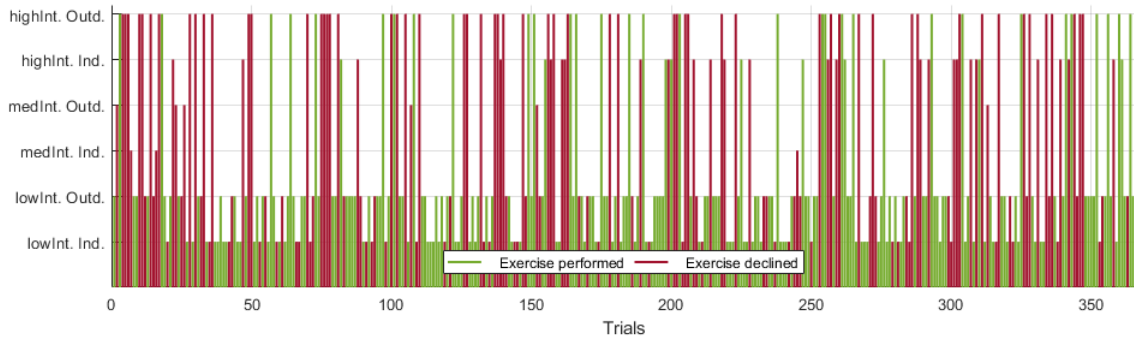
The unfit client is not able to improve their fitness level enough in order to be comfortable with medium or high intensity training, see Figure 6.5b. In fact, during this single run, the client's base fitness value  $av_F$  is still at 0.3, along with its varying interval width  $var_F$ . Note that the medium intensity activities are recommended to the unfit client within the first 50 days, but rarely after this point, and they are always rejected.

fit	lowInt. ind.	lowInt. outd.	medInt. ind.	medInt. outd.	highInt. ind.	highInt. outd.
activity suggestion	14 ± 2	15 ± 2	84 ± 4	83 ± 4	86 ± 4	83 ± 4
activity acceptance	1 ± 2	1 ± 2	82 ± 4	81 ± 4	84 ± 4	82 ± 4

Table 6.17: MC simulation results for fit `clientFitness1`.



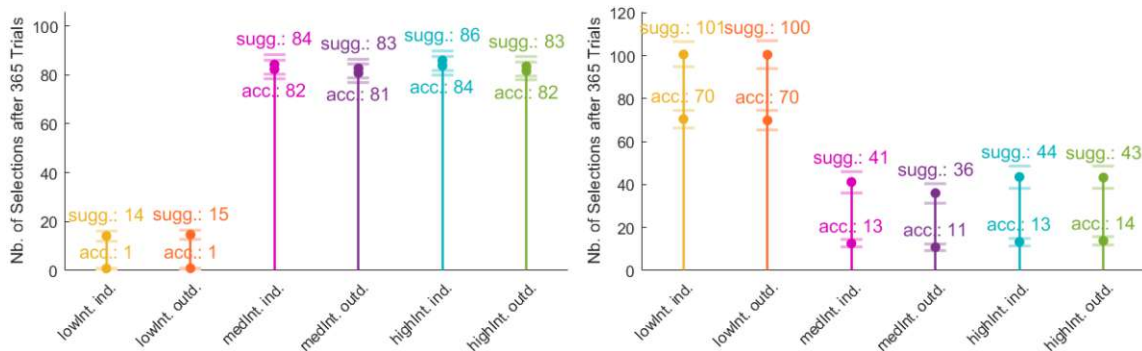
(a) Activity selection and acceptance, fit client.



(b) Activity selection and acceptance, unfit client.

Figure 6.5: Client response for single runs of `TSRC.m` and `clientFitness1`.

The same trend can be seen in the MC simulation results. Table 6.17 shows the rounded means and CIs for the fit client, and Figure 6.6a gives the respective graphical representation. Due to the high fitness level of the client, the medium and high intensity activities are on average almost always accepted, whereas the low intensity training is suggested a few times, but barely accepted.



(a) Fit client.

(b) Unfit client.

Figure 6.6: MC simulation results for fit and unfit `clientFitness1` after 365 days.

The TSRC algorithm identifies the opposite behaviour for the unfit client. The low intensity training options are accepted most often. However, since the unfit client is only prone to accept two options, the amount of acceptances for unsuitable exercise suggestions is higher compared to the fit client. Table 6.18 displays the rounded means and CIs for the unfit client, and Figure 6.6b shows the MC simulation results.

unfit	lowInt. ind.	lowInt. outd.	medInt. ind.	medInt. outd.	highInt. ind.	highInt. outd.
activity suggestion	101 ± 6	100 ± 6	41 ± 5	36 ± 5	44 ± 5	43 ± 5
activity acceptance	70 ± 6	70 ± 6	13 ± 5	11 ± 5	13 ± 5	14 ± 5

Table 6.18: MC simulation results for unfit `clientFitness1`.

There is no clear preference for any feature during the feature selection process throughout the MC simulations for either client setting. However, a slight decrease in popularity can be noted in the motivation feature for the fit client, and both motivation and availability for the unfit client.

Features	fit	unfit
Weather	276 ± 15	282 ± 11
Fitness	277 ± 16	280 ± 11
Availability	274 ± 16	267 ± 13
Motivation	267 ± 16	266 ± 12

Table 6.19: Rounded means and CIs of feature selection for fit and unfit `clientFitness1`.

## Discussion

As expected, the TSRC algorithm correctly identifies the preferred activities in terms of a client’s fitness level. Different investigations into the dynamics of the TSRC algorithm’s response can follow, for example by considering a client who starts out with a low fitness level and improves continuously, or a client who starts with a high fitness level and whose fitness declines by not exercising enough. In that case, a “laziness threshold” needs to be included in the client class, and if the client exercises less than the threshold requires, their fitness level drops by a certain percentage.

Note that the 50-day mark observed in Figure 6.5a is apparent in a single simulation run for the second time. When isolating the weather feature, the single run for the bad weather scenario exhibits a similar “breaking-in period”, see Figure 6.1b.

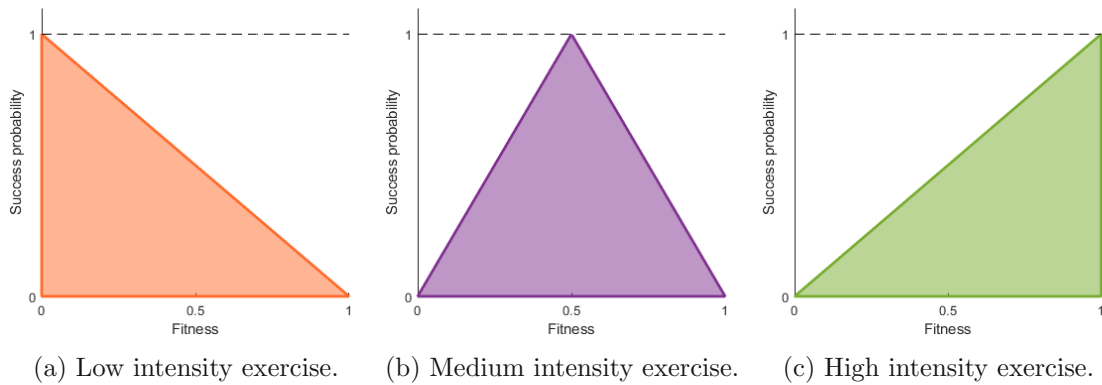


Figure 6.7: Success probabilities for three exercise intensity levels, for modelling the response of `clientFitness1`.

The method `client.clientResponseGenerator(t,k)` could be enhanced so that the medium intensity training is preferred by clients with a medium fitness level: instead of just using  $f_t$ , the formulae for  $m_I(t)$  and  $m_O(t)$  change to:

$$m_I(t) = m_O(t) = \begin{cases} 2f_t & \text{if } f_t \leq 0.5 \\ 2(1 - f_t) & \text{if } f_t > 0.5 \end{cases}$$

The success probabilities for the different training intensities are displayed in Figure 6.7. Some test runs with this modified version of `clientFitness1` promise similarly accurate, if not less explicit, simulation results, indicating that the TSRC algorithm can correctly interpret the three-level model, but the lines between the activity levels become more blurred. The formulae in `client.clientResponseGenerator(t,k)` do not have to be linear, either. Non-linear formulae that determine the success probabilities of activity suggestions also ought to be investigated.

### 6.2.2 Comparing a Fitness-Sensitive Client to a Fitness-Insensitive Client

The fitness levels of the fit and unfit clients from Scenario 6.2.1 are applied to the fitness-insensitive client class `clientFitness2` in order to compare the results to the outcome of Scenario 6.2.1.

#### Simulation Setup

The simulation parameters and class property values are depicted in Tables 6.1, 6.14, and 6.15. The only difference to Scenario 6.2.1 is that the formulae for the success probabilities in the client response method `client.clientResponseGenerator(t,k)` are independent

of the fitness feature:

$$l_I(t) = m_I(t) = h_I(t) = \frac{(1 - w_t) + m_t + (1 - a_t)}{3}$$

$$l_O(t) = m_O(t) = h_O(t) = \frac{w_t + m_t + (1 - a_t)}{3}$$

The seed values for the simulations are unchanged as well, and can be seen in Table 6.16.

## Results

The results of the MC simulation for the fit client are depicted in Table 6.20 and Figure 6.8a. There is no obvious preference for any of the exercise suggestions. Instead, the number of times each activity is suggested is divided evenly between the intervention options, with each activity being on average suggested every six trials.

fit	lowInt. ind.	lowInt. outd.	medInt. ind.	medInt. outd.	highInt. ind.	highInt. outd.
activity suggestion	63 ± 5	62 ± 5	59 ± 5	60 ± 5	57 ± 5	63 ± 4
activity acceptance	32 ± 5	31 ± 5	29 ± 5	30 ± 5	29 ± 5	32 ± 5

Table 6.20: MC simulation results for fit `clientFitness2`.

The number of acceptances is also divided equally between activities, and each activity is accepted approximately every second time it is suggested. On average, the acceptance rate is 183 out of 365 trials, so just over 50%.

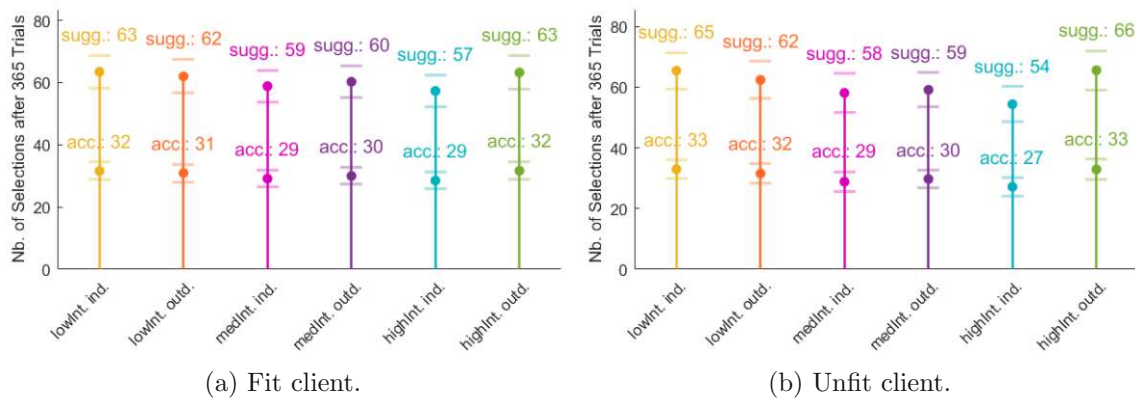


Figure 6.8: MC simulation results for fit and unfit `clientFitness2` after 365 days.

The MC simulation results for the unfit client are displayed in Table 6.21 and Figure 6.8b. Similar to the fitness-insensitive client with higher fitness level, no bias in the activity

suggestions and acceptances towards the fitness feature is apparent. However, there is a dip in activity suggestions and acceptances for the high intensity indoor training.

unfit	lowInt. ind.	lowInt. outd.	medInt. ind.	medInt. outd.	highInt. ind.	highInt. outd.
activity suggestion	65 ± 6	62 ± 6	58 ± 6	59 ± 6	54 ± 6	66 ± 6
activity acceptance	33 ± 6	32 ± 6	29 ± 6	30 ± 6	27 ± 6	33 ± 6

Table 6.21: MC simulation results for unfit `clientFitness2`.

The number of activity selections resides around one sixth of all trials for each activity, and every second suggestion is accepted on average. This gives a mean acceptance rate of just over 50%, which equals 184 acceptances in 365 trials.

Features	fit	unfit
Weather	274 ± 9	279 ± 10
Fitness	275 ± 10	280 ± 10
Availability	268 ± 11	268 ± 11
Motivation	278 ± 9	268 ± 11

Table 6.22: Rounded means and CIs of feature selection for fit and unfit `clientFitness2`.

Again, there is no obvious feature preference in both MC simulations. For the fit client, availability is chosen least often, and for the unfit client, motivation and availability are chosen 11 and 12 times less than the weather and fitness features, which is around half the width of the respective CIs.

## Discussion

In direct comparison, the TSRC algorithm performs as expected, not forming any bias towards the fitness feature when it is not regarded by the client, and this observation is made regardless of the pre-defined fitness level, underlining the TSRC algorithm's validity as decision rules in a JITAI in a similar way to Scenario 6.1.2. The outcome of equally distributed suggestions and acceptances throughout all activities is expected and shows complementary behaviour to the explicit preferences of the fitness-sensitive client exhibited in Scenario 6.2.1.

Also, as discussed in Scenario 6.1.2, the average percentage of overall acceptances resides around 50%, indicating that the TSRC algorithm correctly interprets the (evenly distributed) preferences of the fitness-insensitive client, whose property values are at 0.5



for all properties except the fitness-related base value and varying interval,  $av_F$  and  $var_F$ , which are not regarded.

Note that, even though the weather influences the success probability for outdoor activities via  $w_t$  and the success probability for indoor activities via  $1 - w_t$ , there is no weather-related bias because the values  $w_t$  are initialised at 0.5 for all  $t$  during the initiation method.

### 6.3 The Availability Feature

When isolating the availability feature, the interest lies in seeing whether the TSRC algorithm can correctly identify the preferences of an individual who is more available compared to one who is rarely available. The assumption is that lower intensity training is more time consuming (i.e., taking a walk) in contrast to higher intensity training (i.e., 15-minute cardio workout). When disregarding all other features, these circumstances should be evident in the simulation outcome.

Again, two client classes exist for simulation purposes. The availability-sensitive class `clientAvailability1` only regards a client's availability when making a decision, while the availability-insensitive class `clientAvailability2` considers all features except the client's availability. First, different levels of availability are compared for the availability-sensitive client class in Scenario 6.3.2, and the results are examined in contrast to the availability-insensitive client in Scenario 6.3.1.

#### 6.3.1 Comparing Different Levels of Availability

The availability-sensitive client class `clientAvailability1` is used for simulation in order to compare a client with high availability to a client with low availability.

##### Simulation Setup

The general simulation parameters are unchanged, see Table 6.1. The properties for an available client of the availability-sensitive client class `clientAvailability1` are displayed in Table 6.23. Note that, after the property initiation method `client.initiation(T)`, the properties responsible for weekend and weekday availability,  $av_{WE}$ ,  $var_{WE}$ ,  $av_{WD}$ ,  $var_{WD}$ , are overwritten with values representing high availability.

The property values for the unavailable client are shown in Table 6.26. Note that, for both the available and unavailable client, the weather is kept at the default value of 0.5 for all  $t$ . The method `client.clientResponseGenerator(t,k)` defines the client's focus on the availability values. Here, the idea of considering three levels of availability, diametrically

$\mathbf{f} = \mathbf{0} \in \mathbb{R}^T$	$\mathbf{r} = \mathbf{0} \in \mathbb{R}^T$	$var_{WD} = 0.6$	$av_M = 0.5$
$\mathbf{m} = \mathbf{0} \in \mathbb{R}^T$	$av_{WE} = 0.9$	$av_F = 0.5$	$var_M = 0.5$
$\mathbf{a} = \mathbf{0} \in \mathbb{R}^T$	$var_{WE} = 1$	$var_F = 0.5$	$f^* = 4$
$\mathbf{w} = (0.5, \dots, 0.5) \in \mathbb{R}^T$	$av_{WD} = 0.8$		

Table 6.23: Class property values for available `clientAvailability1` in Scenario 6.3.1.

$\mathbf{f} = \mathbf{0} \in \mathbb{R}^T$	$\mathbf{r} = \mathbf{0} \in \mathbb{R}^T$	$var_{WD} = 0.6$	$av_M = 0.5$
$\mathbf{m} = \mathbf{0} \in \mathbb{R}^T$	$av_{WE} = 0.3$	$av_F = 0.5$	$var_M = 0.5$
$\mathbf{a} = \mathbf{0} \in \mathbb{R}^T$	$var_{WE} = 0.4$	$var_F = 0.5$	$f^* = 4$
$\mathbf{w} = (0.5, \dots, 0.5) \in \mathbb{R}^T$	$av_{WD} = 0.1$		

Table 6.24: Class property values for unavailable `clientAvailability1` in Scenario 6.3.1.

opposed to the levels of exercise intensity, is implemented, as suggested in the discussion of Scenario 6.2.2. In practice, this means:

$$\begin{aligned}
 l_I(t) &= l_O(t) = \mathbf{a}_t, \\
 h_I(t) &= h_O(t) = (1 - \mathbf{a}_t), \\
 m_I(t) &= m_O(t) = \begin{cases} 2\mathbf{a}_t & \text{if } \mathbf{a}_t \leq 0.5 \\ 2(1 - \mathbf{a}_t) & \text{if } \mathbf{a}_t > 0.5 \end{cases}
 \end{aligned}$$

Similar to Figure 6.7, Figure 6.9 gives the success probabilities of the client response generator method for `clientAvailability1`. The higher the client's availability is, the more likely they are to accept low intensity exercise. The less available a client is, the more likely they are to accept high intensity activity suggestions. The probability for accepting medium intensity exercises is highest when being moderately available.

The seed values for this scenario, as well as the complementary Scenario 6.3.2, are given in Table 6.25.

available & unavailable	Single run	MC simulation
Seed	34216	2542

Table 6.25: Seed values for `clientAvailability1` in Scenario 6.3.1.

## Results

Single runs for client `clientAvailability1` already show that the available client clearly prefers to accept low intensity training, see Figure 6.10a. Even though high intensity

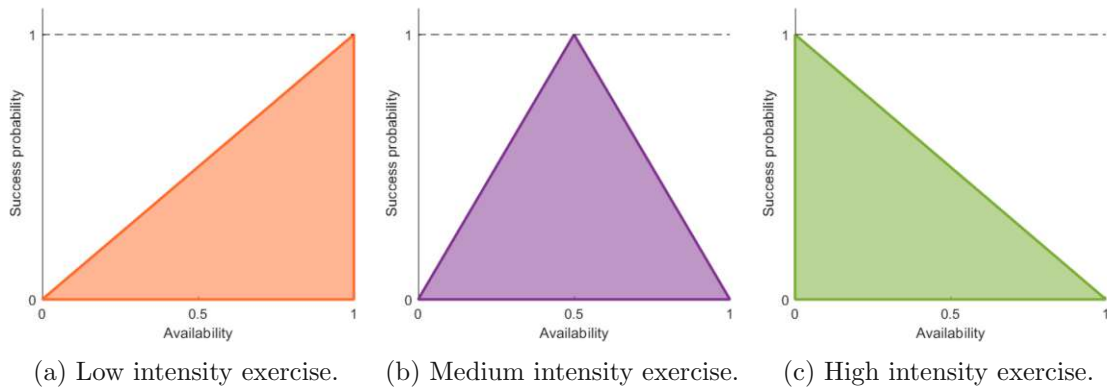


Figure 6.9: Success probabilities for three exercise intensity levels, for modelling the response of `clientAvailability1`.

training is recommended more than medium intensity training, it is accepted infrequently.

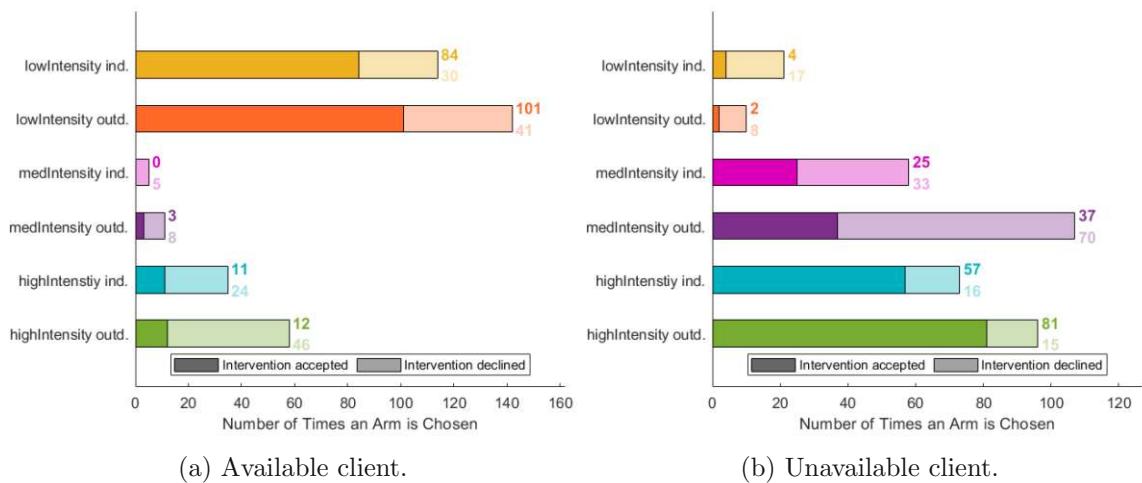


Figure 6.10: Total number of suggestions and acceptances for available and unavailable `clientAvailability1` after 365 days.

A single run in the unavailable client setting shows a stepwise increase in the number of activity acceptances from low intensity training to high intensity training, see Figure 6.10b. Even though medium intensity outdoor training is recommended most often at 107 suggestions, it is only accepted 37 times.

The same stepwise increase in exercise acceptances can be seen as a result of the MC simulation for the unavailable client, depicted in Table 6.26 and Figure 6.11b. On average, high intensity activities are accepted most frequently, followed by medium intensity activities, and both low intensity exercises are suggested and accepted least often.

In the MC simulation, the available client exhibits the same stepwise pattern as the

## 6 Simulation

unavailable	lowInt. ind.	lowInt. outd.	medInt. ind.	medInt. outd.	highInt. ind.	highInt. outd.
activity suggestion	26 ± 4	28 ± 4	52 ± 5	49 ± 5	105 ± 5	105 ± 5
activity acceptance	7 ± 4	6 ± 4	21 ± 5	20 ± 5	81 ± 5	82 ± 5

Table 6.26: MC simulation results for unavailable `clientAvailability1`.

available	lowInt. ind.	lowInt. outd.	medInt. ind.	medInt. outd.	highInt. ind.	highInt. outd.
activity suggestion	102 ± 4	100 ± 5	47 ± 5	44 ± 5	37 ± 4	34 ± 4
activity acceptance	75 ± 4	73 ± 5	16 ± 5	15 ± 5	10 ± 4	10 ± 4

Table 6.27: MC simulation results for available `clientAvailability1`.

unavailable client, see Table 6.27 and Figure 6.11a. Contrary to the unavailable client, the available client receives mainly low intensity training suggestions, which are frequently accepted. Medium and high intensity exercises are suggested less often, and are even less frequently accepted.

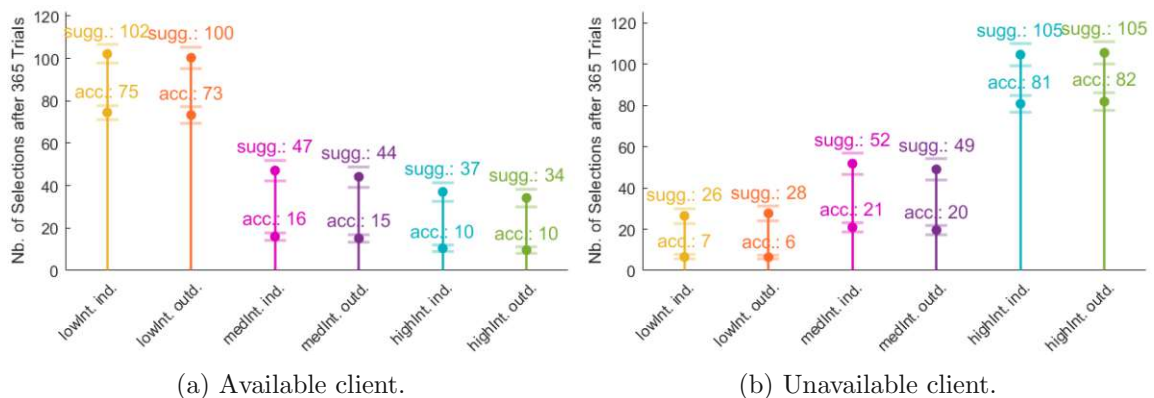


Figure 6.11: MC simulation results for available and unavailable `clientAvailability1` after 365 days.

Concerning feature selection during the combinatorial bandit, there is no clear favourite in either availability scenario during the MC simulation. However, the weather feature is slightly less popular during the simulation with the available client, and both weather and fitness are regarded less often for the unavailable client.

Features	available	unavailable
Weather	258 ± 13	267 ± 13
Fitness	278 ± 11	259 ± 14
Availability	281 ± 11	292 ± 10
Motivation	277 ± 11	277 ± 12

Table 6.28: Rounded means and CIs of feature selection for available and unavailable `clientAvailability1`.

### Discussion

The TSRC algorithm is seen to correctly and distinctly identify the preferences of an availability-sensitive client, even when considering three levels of availability, which is graphically illustrated in Figure 6.11. There is an apparent favouritism for the respective suitable training intensity in both MC simulations. The acceptance rate is around 74% and 73% for the available client and both low indoor and outdoor intensity training, and around 77% and 78% for the unavailable client and both high intensity indoor and outdoor training. In contrast, the acceptance rates for medium and low intensity training are below 41% for the unavailable client. For the available client, the acceptance rates for medium and high intensity training even lie below 34%.

Further investigation can be conducted into a more dynamic availability model for the client, which may include more sophisticated, potentially non-linear formulae in the client response generator. For example, the availability of a client could vary within a year cycle, or include random phases of non-availability to imitate vacation time, in order to explore the TSRC algorithm's response to such rapid changes.

### 6.3.2 Comparing an Availability-Sensitive Client to an Availability-Insensitive Client

The availability-insensitive client class `clientAvailability2` is used to compare clients of high and low availability to the simulation outcome of Scenario 6.3.1.

#### Simulation Setup

The parameter values of the simulation are unchanged and can be found in Table 6.1. Furthermore, the property values of the available and unavailable client are unchanged, see Tables 6.23 and 6.24.

The difference to class `clientAvailability1` lies in the definition of the client response method, where the formulae for determining the success probability of accepting an activity

suggestion are:

$$l_I(t) = m_I(t) = h_I(t) = \frac{\mathbf{f}_t + \mathbf{m}_t + (1 - \mathbf{w}_t)}{3}$$

$$l_O(t) = m_O(t) = h_O(t) = \frac{\mathbf{f}_t + \mathbf{m}_t + \mathbf{w}_t}{3}$$

The seed values for the simulation runs are displayed in Table 6.25, as they are unaltered.

## Results

The MC simulation results for the availability-insensitive client of high availability are displayed in Table 6.29 and Figure 6.12a, which show almost equal numbers in both activity suggestions and acceptances throughout all activities.

available	lowInt. ind.	lowInt. outd.	medInt. ind.	medInt. outd.	highInt. ind.	highInt. outd.
activity suggestion	60 ± 5	61 ± 5	61 ± 5	62 ± 5	60 ± 5	61 ± 5
activity acceptance	34 ± 5	35 ± 5	34 ± 5	36 ± 5	35 ± 5	35 ± 5

Table 6.29: MC simulation results for available `clientAvailability2`.

The amount of exercise suggestions range between 60 and 62 on average, whereas activities are accepted between 34 and 36 times out of 365 trials, which is slightly less than 10%.

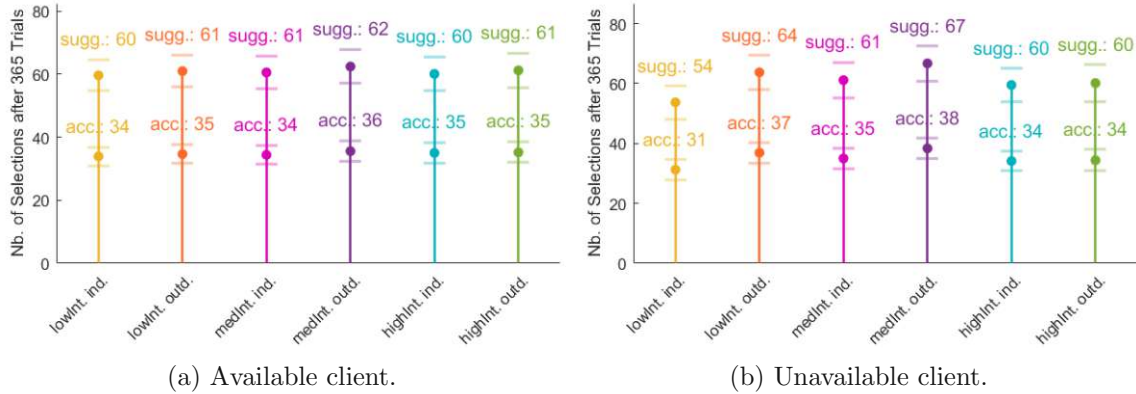


Figure 6.12: MC simulation results for available and unavailable `clientAvailability2` after 365 days.

The MC simulation for the availability-insensitive client of low availability shows a slight bias towards outdoor activities, with indoor activities being recommended only around 48%

of 365 simulation days on average. The average number of acceptances for each activity lies between 31 and 35 for indoor exercise, and between 34 and 37 for outdoor exercise.

unavailable	lowInt. ind.	lowInt. outd.	medInt. ind.	medInt. outd.	highInt. ind.	highInt. outd.
activity suggestion	54 ± 6	64 ± 6	61 ± 6	67 ± 6	60 ± 5	60 ± 6
activity acceptance	31 ± 6	37 ± 6	35 ± 6	38 ± 6	34 ± 5	34 ± 6

Table 6.30: MC simulation results for unavailable `clientAvailability2`.

There is no distinct preference in the feature selection for both MC simulations. The motivation feature is least often selected during simulations with the available client, and both availability and motivation are least frequently chosen in the unavailable client setting.

Features	available	unavailable
Weather	273 ± 11	276 ± 11
Fitness	278 ± 10	284 ± 9
Availability	275 ± 11	268 ± 12
Motivation	269 ± 11	267 ± 11

Table 6.31: Rounded means and CIs of feature selection for available and unavailable `clientAvailability2`.

## Discussion

As anticipated, the TSRC algorithm does not form a bias towards exercise intensity, which is linked in reverse proportionality to a client's availability. However, a bias is found in the number of activity suggestions towards indoor activities for the unavailable client. Its origin may be explained during further investigation into the dynamics of the TSRC algorithm.

Since the weather feature is set to  $w_t = 0.5$  for all  $t$ , a preference for indoor activities is not expected, because the weather component for the success probability is

$$1 - w_t = 0.5,$$

compared to the weather component for outdoor activities:

$$w_t = 0.5$$

When changing the indoor weather component to  $w_t$ , the bias still exists, which is expected, since the bias is not found in the available client setting, which uses the inverse

proportionality of weather data for success probability calculation of indoor activities.

The bias thus cannot be explained by investigating the weather feature alone. The most likely cause is coincidence: the raised numbers for outdoor activities, especially in the low and medium intensity range, may only correlate, instead of being determined by a common cause. Simulations with different seed values show similar preference patterns for the unavailable client setting, where two or three activities are marginally more popular in recommendations and acceptances without having an exercise intensity or weather component in common.

## 6.4 The Motivation Feature

Isolating the motivation feature results in the most complex client class definitions out of all features. It is not modelled as a rigid factor that strictly adheres to a consistent formula throughout all activity suggestions, but should be seen as a dynamic influence on the success probability formulae. Now that the TSRC algorithm has proven to perform sufficiently well regarding simple relationships between features and activities, it is worth investigating more intricate feature models.

In the modelling context, the motivation feature leaves the most room for different models. A client's motivation may be assumed to be consistently high or low only for certain activities, and inconsistent for others, or the activity suggestions may depend linearly or non-linearly on the client's motivation. All of these circumstances represent realistic scenarios which are worth examining.

One of many possible models is adopted to explore the motivation feature. In Scenario 6.4.1, two client classes, `clientMotivation1` and `clientMotivation2`, are used to investigate the TSRC algorithm's response to clients who are highly motivated for three out of six activities implemented in a conservative way. Scenario 6.4.2 explores a model extension with the help of two additional clients, `clientMotivation3` and `clientMotivation4`, where the feature value  $m_t$  depends on the weather value  $w_t$ .

### 6.4.1 Comparing Clients Motivated for Different Activities

The client classes `clientMotivation1` and `clientMotivation2` are used in single simulation runs and MC simulations, to see how the TSRC algorithm reacts when the client response generator pushes for certain activities.



### Simulation Setup

The general simulation parameters are found in Table 6.1. The difference between both clients lies in the method `client.clientResponseGenerator(t,k)`, where the success probabilities of all activities are weighted, indicating the clients' preferences regardless of pre-defined activity attributes.

$\mathbf{f} = \mathbf{0} \in \mathbb{R}^T$	$\mathbf{r} = \mathbf{0} \in \mathbb{R}^T$	$var_{WD} = 0.5$	$av_M = 0.7$
$\mathbf{m} = \mathbf{0} \in \mathbb{R}^T$	$av_{WE} = 0.5$	$av_F = 0.5$	$var_M = 0.8$
$\mathbf{a} = \mathbf{0} \in \mathbb{R}^T$	$var_{WE} = 0.5$	$var_F = 0.5$	$f^* = 4$
$\mathbf{w} = (0.5, \dots, 0.5) \in \mathbb{R}^T$	$av_{WD} = 0.5$		

Table 6.32: Class property values for `clientMotivation1` and `clientMotivation2` in Scenario 6.4.1.

The class property values for both clients are thus identical, and are found in Table 6.32. They only deviate from the default setup by setting the baseline motivation value  $av_M$  to 0.7, and the varying interval  $var_M$  to 0.8. This way, the effect of heightened motivation ought to be magnified. Client `clientMotivation1` is most motivated for high intensity indoor training, followed by medium intensity outdoor training, and low intensity indoor training. After these, the other activities are seen as equally desirable, but the higher the motivation, the less likely it is that these are accepted. The success probabilities for activity acceptance are given by:

$$h_I(t) = \frac{\mathbf{m}_t + 3}{4}, \quad m_O(t) = \frac{\mathbf{m}_t + 2}{3}, \quad l_I(t) = \frac{\mathbf{m}_t + 1}{2},$$

$$l_O(t) = m_I(t) = h_O(t) = 1 - \mathbf{m}_t$$

Client `clientMotivation2` is modelled complementarily: low intensity outdoor training is most popular, followed by medium intensity indoor training, and high intensity indoor training. The client response generator method calculates the success probabilities for accepting the exercise suggestion thusly:

$$l_O(t) = \frac{\mathbf{m}_t + 3}{4}, \quad m_I(t) = \frac{\mathbf{m}_t + 2}{3}, \quad h_O(t) = \frac{\mathbf{m}_t + 1}{2},$$

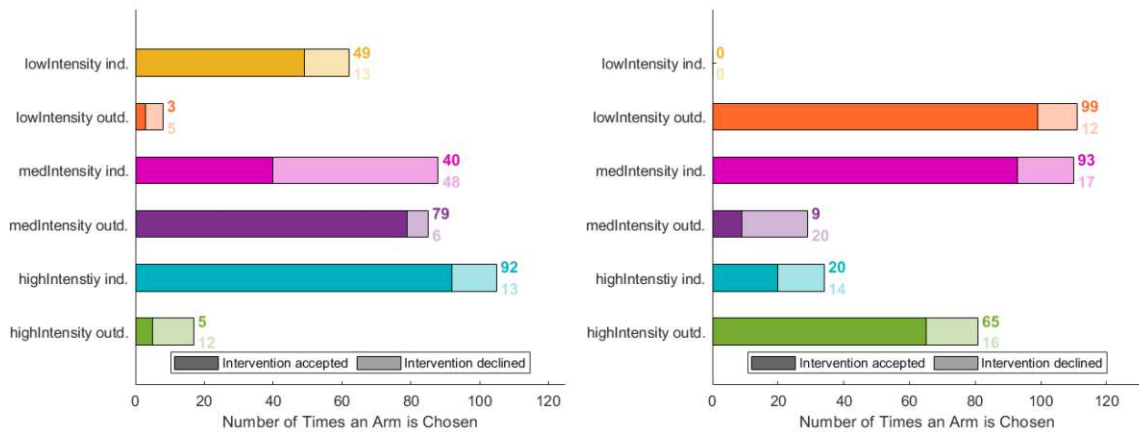
$$l_I(t) = m_O(t) = h_I(t) = 1 - \mathbf{m}_t$$

Note that “high motivation” should not be interpreted as an absolute. That is, in Scenarios 6.4.1 and 6.4.2,  $\mathbf{m}_t \rightarrow 1$  does not indicate excessive motivation for any activity, but instead being more motivated for favoured activities. Conversely,  $\mathbf{m}_t \rightarrow 0$  yields greater motivation for the less desirable training suggestions.

The seed for the MC simulation for both client classes is 23.

## Results

Single runs of the TSRC algorithm show a tendency to display the respective favourites of both client classes, see Figure 6.13. For client `clientMotivation1`, Figure 6.13a shows the contrast between the most popular activity, medium intensity outdoor training, and one of the least favourite activities, medium intensity indoor training. Even though both exercises are recommended 85 and 88 times respectively, the more desired activity is accepted in around 93% of cases, whereas medium intensity indoor training is only taken up around 45% of the time. Although no such discrepancy is shown in the results for the single run and `clientMotivation2`, the client's degree of preference is replicated exactly in both suggestions and acceptances. Note that, in this instance, low intensity indoor training is never recommended to the client.



(a) Client `clientMotivation1`, seed: 69.

(b) Client `clientMotivation2`, seed: 81132.

Figure 6.13: Total number of suggestions and acceptances for `clientMotivation1` and `clientMotivation2` after 365 days.

The MC simulation results for `clientMotivation1` mirror the client's preferences perfectly. As seen in Table 6.33 and Figure 6.14a, the acceptance numbers for all three preferred activities are high, whereas the discrepancy between suggestions and acceptances is larger.

<code>clientMotivation1</code>	lowInt. ind.	lowInt. outd.	medInt. ind.	medInt. outd.	highInt. ind.	highInt. outd.
activity suggestion	81 ± 5	36 ± 4	34 ± 4	88 ± 5	92 ± 5	34 ± 4
activity acceptance	66 ± 5	14 ± 4	14 ± 4	78 ± 5	83 ± 5	13 ± 4

Table 6.33: MC simulation results for `clientMotivation1`.

For example, high intensity outdoor training is accepted in only 38% of cases on average, whereas the uptake of high intensity indoor training averages around 90%.

clientMotivation2	lowInt. ind.	lowInt. outd.	medInt. ind.	medInt. outd.	highInt. ind.	highInt. outd.
activity suggestion	31 ± 4	90 ± 5	87 ± 5	34 ± 4	35 ± 4	89 ± 5
activity acceptance	12 ± 4	81 ± 5	76 ± 5	13 ± 4	14 ± 4	73 ± 5

Table 6.34: MC simulation results for clientMotivation2.

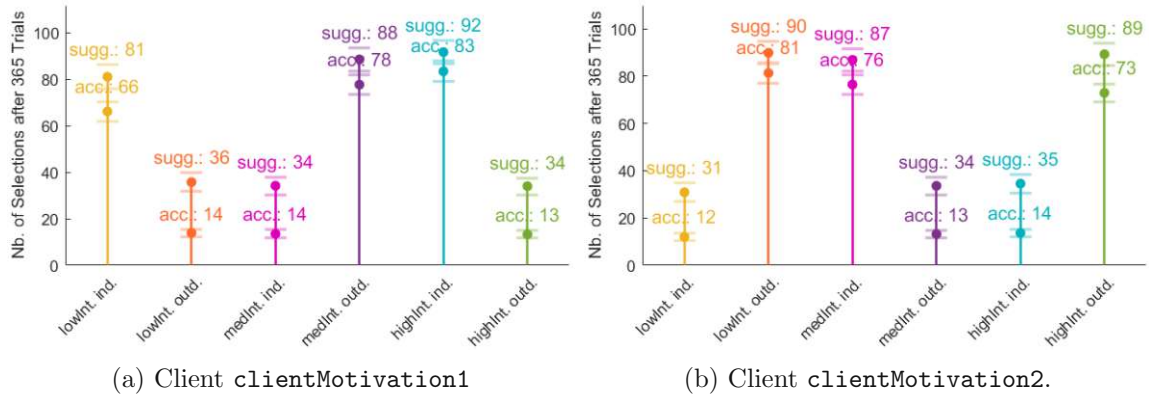


Figure 6.14: MC simulation results for clientMotivation1 and clientMotivation2 after 365 days.

The response of the TSRC algorithm to clientMotivation2 is depicted in Table 6.34 and Figure 6.14b, and shows the same result for the complementary setup. Low intensity outdoor training has an average acceptance rate of 90%. In contrast, medium intensity outdoor training is accepted only in 38% of cases.

Features	clientMotivation1	clientMotivation2
Weather	274 ± 11	260 ± 12
Fitness	270 ± 12	280 ± 10
Availability	271 ± 12	269 ± 12
Motivation	280 ± 12	287 ± 10

Table 6.35: Rounded means and CIs of feature selection for clientMotivation1 and clientMotivation2.

There is no bias towards any particular feature during the combinatorial bandit search for clientMotivation1. A preference is obvious for clientMotivation2, where fitness and motivation are, on average, chosen more often than availability and the weather feature.

## Discussion

The TSRC algorithm performs equally well for the more dynamic motivation model, correctly identifying the favourite activities for both client classes. Further investigation into more complex models is possible, especially with regard to more realistic setups, in which behavioural models could be implemented.

The complexity of the motivational model depends on the choice of features, including the amount that can be regarded  $d$ , and on the feature budget  $n$ , with which the TSRC algorithm works. If the motivation feature expresses the client's disposition towards performing the suggested activity, then other features should be chosen in a way that accommodates a motivational model (i.e., if the model for motivation requires a value representing the daily weather conditions, the weather should be included as a tailoring variable in the JITAI).

### 6.4.2 Comparing Different Models for Motivation

The client classes `clientMotivation3` and `clientMotivation4`, for which the daily values for motivation depend on the daily weather values, are used for simulation in the different weather scenarios depicted in Scenario 6.1.1, and the results are compared to the simulation outcome in Scenario 6.4.1.

#### Simulation Setup

The general simulation parameters are displayed in Table 6.1. Furthermore, the values for the properties of the client classes are identical to those of Scenario 6.4.1, which are given in Table 6.32.

Client `clientMotivation3` is motivated for the same three activities as client `clientMotivation1`. Thus, the class method `client.clientResponseGenerator(t,k)` in `clientMotivation3` is equal to the one in `clientMotivation1`:

$$h_I(t) = \frac{m_t + 3}{4}, \quad m_O(t) = \frac{m_t + 2}{3}, \quad l_I(t) = \frac{m_t + 1}{2},$$

$$l_O(t) = m_I(t) = h_O(t) = 1 - m_t$$

The same holds for `clientMotivation4` and its antetype `clientMotivation2`, so the formulae in the client response generator are:

$$l_O(t) = \frac{m_t + 3}{4}, \quad m_I(t) = \frac{m_t + 2}{3}, \quad h_O(t) = \frac{m_t + 1}{2},$$

$$l_I(t) = m_O(t) = h_I(t) = 1 - m_t$$

The difference to the client classes of Scenario 6.4.1 lies in the data generator method

`client.singleStepClientDataGenerator(t)`. Since the motivation is assumed to be dependent on the weather, the daily values for motivation are not generated via the formula that uses  $av_M$  and  $var_M$ , see Section 5.2. Instead,  $m_t$  is the output of a function of  $w_t$ :

```

IF  $w_t \leq 0.3$ 
     $m_t = \frac{w_t}{2}$ 
ELSE IF  $w_t \in (0.3, 0.7)$ 
     $m_t = w_t$ 
ELSE IF  $w_t \geq 0.7$ 
     $m_t = \frac{w_t + 1}{2}$ 
END

```

Figure 6.15 shows the resulting values of  $m_t$  for the 2009 weather scenario. The dampening of low values and the amplification of high values are displayed distinctly. Note that the generation of  $m_t$  is identical between `clientMotivation3` and `clientMotivation4`, as these classes only differ in their activity preferences.

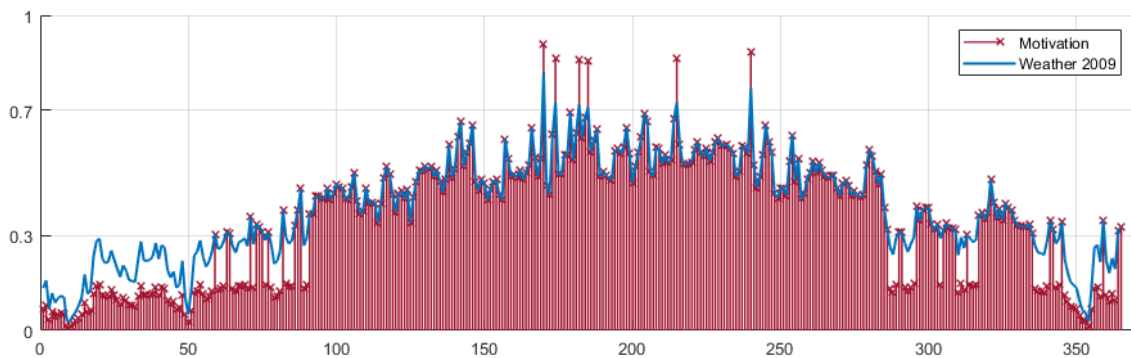


Figure 6.15: Motivation values as a function of the weather, 2009 weather scenario.

The seed values for the simulation results are given in Table 6.36.

	good	bad	yearSine	year2009
Seeds	9667	50037	4	564

Table 6.36: Seed values for the MC simulations for different weather scenarios and both `clientMotivation3` and `clientMotivation4`.

## Results

The MC simulation results for the good weather scenario show large numbers in activity selection as well as acceptances of the preferred activities for both clients, see Tables 6.37, 6.38, and Figure 6.16. The less desired activities are on average selected ten times less often, and barely accepted.

clientMotivation3 & good	lowInt. ind.	lowInt. outd.	medInt. ind.	medInt. outd.	highInt. ind.	highInt. outd.
activity suggestion	103 ± 3	18 ± 2	16 ± 1	103 ± 3	110 ± 3	16 ± 1
activity acceptance	100 ± 3	1 ± 2	1 ± 1	101 ± 3	108 ± 3	1 ± 1

Table 6.37: MC simulation results for clientMotivation3 and good weather.

The high acceptance rate for favoured activities is expected, since the good weather scenario gives  $w_t = 0.9$  constantly, so  $m_t = 0.95$  for all  $t$ . As a consequence, the top three activities have success probabilities of 0.9875, 0.9833, and 0.975 for all  $t$  and both client classes. On the other hand, the less desired activities are accepted with a success probability of 0.05 for all trials.

clientMotivation4 & good	lowInt. ind.	lowInt. outd.	medInt. ind.	medInt. outd.	highInt. ind.	highInt. outd.
activity suggestion	17 ± 2	105 ± 3	105 ± 3	16 ± 1	16 ± 1	106 ± 3
activity acceptance	1 ± 2	104 ± 3	103 ± 3	1 ± 1	1 ± 1	103 ± 3

Table 6.38: MC simulation results for clientMotivation4 and good weather.

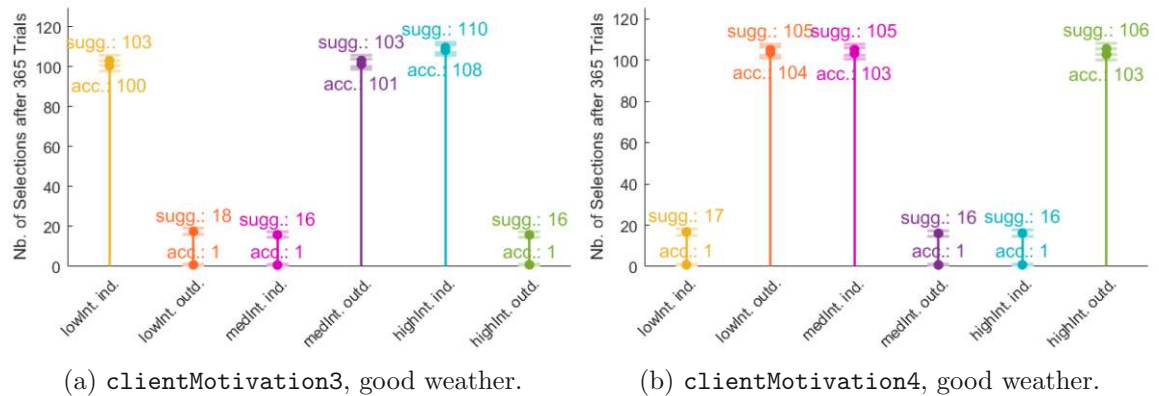


Figure 6.16: MC simulation results for clientMotivation3 and clientMotivation4, and good weather after 365 days.

There are noticeable variations in the number of times each feature is chosen during the

combinatorial bandit search. For `clientMotivation3`, the availability feature is selected least often on average, and the CI cannot compensate for this deficiency, see Table 6.39. The most frequently selected feature is even more distinct for `clientMotivation4`: the weather feature is chosen almost 300 times out of 365 trials, whereas fitness, availability, and motivation are chosen less than 270 times on average.

Features	<code>clientMotivation3</code>	<code>clientMotivation4</code>
Weather	278 ± 13	296 ± 10
Fitness	283 ± 12	263 ± 14
Availability	261 ± 14	269 ± 13
Motivation	274 ± 13	268 ± 14

Table 6.39: Rounded means and CIs of feature selection for `clientMotivation3` and `clientMotivation4`, good weather scenario.

The preferred activities are not identified for the bad weather scenario in either client, see Tables 6.40 and 6.41. Instead, less desirable activities are chosen the most often: the number of recommendations for less popular activities is on average increased by a factor of 4, and the amount of activity selections for the preferred activities is diminished by around 60%.

<code>clientMotivation3</code> & <code>bad</code>	lowInt. ind.	lowInt. outd.	medInt. ind.	medInt. outd.	highInt. ind.	highInt. outd.
activity suggestion	32 ± 4	73 ± 6	76 ± 6	54 ± 6	52 ± 6	78 ± 6
activity acceptance	17 ± 4	70 ± 6	72 ± 6	37 ± 6	40 ± 6	74 ± 6

Table 6.40: MC simulation results for `clientMotivation3` and bad weather.

<code>clientMotivation4</code> & <code>bad</code>	lowInt. ind.	lowInt. outd.	medInt. ind.	medInt. outd.	highInt. ind.	highInt. outd.
activity suggestion	77 ± 6	53 ± 5	46 ± 5	76 ± 6	78 ± 6	34 ± 4
activity acceptance	73 ± 6	41 ± 5	32 ± 5	72 ± 6	74 ± 6	18 ± 4

Table 6.41: MC simulation results for `clientMotivation4` and bad weather.

This is due to the weather value  $w_t$  staying constantly at 0.1 throughout the simulation year, which yields a constant motivation value of  $m_t = 0.05$ . The success probability of any non-favoured activity is thus fixed at 0.95 for all  $t$ , and the top three preferred activities have success probabilities of 0.7625, 0.6833, and 0.525. This deterioration in activity acceptance is reflected in the MC simulation outcomes depicted in Figure 6.17.

## 6 Simulation

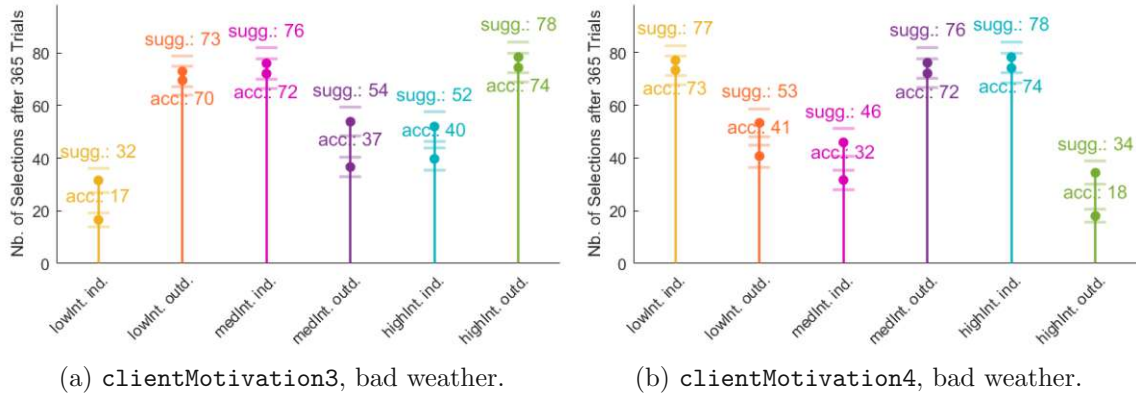


Figure 6.17: MC simulation results for `clientMotivation3` and `clientMotivation4`, and bad weather after 365 days.

For both clients, the third most popular activity is accepted the least often on average (i.e., low intensity indoor training for `clientMotivation3` and high intensity outdoor training for `clientMotivation4`). These activities are suggested in only 10% of cases, and the acceptance rate of slightly more than 50% mirrors the success probability of 0.525.

The average numbers for feature selection in the combinatorial bandit are divided somewhat evenly for `clientMotivation3`, ranging within 13 instances. A similar outcome is seen for `clientMotivation4`, where the average numbers for each feature range within 12 trials, see Table 6.42.

Features	clientMotivation3	clientMotivation4
Weather	279 ± 10	269 ± 12
Fitness	266 ± 12	272 ± 11
Availability	273 ± 12	274 ± 11
Motivation	276 ± 10	281 ± 10

Table 6.42: Rounded means and CIs of feature selection for `clientMotivation3` and `clientMotivation4`, bad weather scenario.

The MC simulation results for the ideal year setting are displayed in Tables 6.43, 6.44, and Figure 6.18. For both clients, the preferred activities are identified in correct order of favouritism. This can be attributed to the fact that even though the weather data exhibits non-stationary seasonal behaviour, when generating  $\mathbf{m}_t$ , only weather values below 0.3 are decreased according to the formula. For all  $w_t \geq 0.3$ , it holds that  $\mathbf{m}_t \geq w_t$ , which in turn boosts the success probabilities of the favoured activities.

The less desirable activities are still suggested a considerable amount of times, but the average acceptance rate lies between 37% and 40% for each activity and either client. Note



clientMotivation3 & yearSine	lowInt. ind.	lowInt. outd.	medInt. ind.	medInt. outd.	highInt. ind.	highInt. outd.
activity suggestion	72 ± 5	46 ± 5	48 ± 5	75 ± 5	80 ± 5	43 ± 5
activity acceptance	60 ± 5	17 ± 5	18 ± 5	66 ± 5	74 ± 5	17 ± 5

Table 6.43: MC simulation results for clientMotivation3 and ideal weather.

that only 71 out of 365 weather values  $w_t$  range below the 0.3 mark in the ideal weather scenario, implying that the success probability for a less desirable arm exceeds 0.85 in around 20% of trials.

clientMotivation4 & yearSine	lowInt. ind.	lowInt. outd.	medInt. ind.	medInt. outd.	highInt. ind.	highInt. outd.
activity suggestion	46 ± 5	85 ± 5	77 ± 5	46 ± 4	45 ± 5	67 ± 5
activity acceptance	17 ± 5	77 ± 5	68 ± 5	17 ± 4	17 ± 5	55 ± 5

Table 6.44: MC simulation results for clientMotivation4 and ideal weather.

In contrast,  $w_t \geq 0.7$  holds in 186 trials, which raises the success probability of the most desirable arm for either client over 92% in around 51% of cases. Subsequently, the average acceptance rates for the top three activities are 93%, 88%, and 83% for clientMotivation3 and 91%, 88%, and 82% for clientMotivation4.

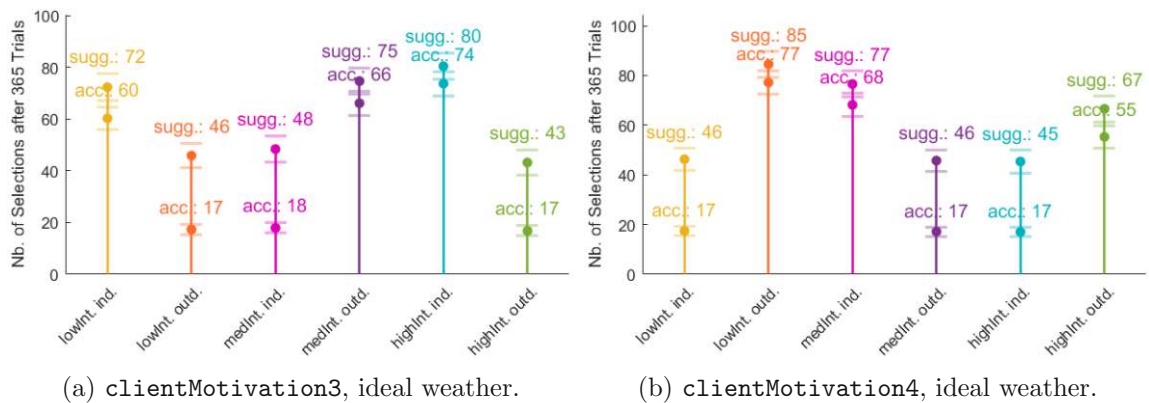


Figure 6.18: MC simulation results for clientMotivation3 and clientMotivation4, and ideal weather after 365 days.

Concerning the average numbers of feature selection for the ideal weather scenario, there is a clear outlier for clientMotivation3. Availability is chosen on average 15 times less often than the next frequent feature. On the other hand, the feature selection for clientMotivation4 is evenly distributed between features, where the average amount of

times any feature is chosen ranges within 10 trials.

Features	clientMotivation3	clientMotivation4
Weather	277 ± 10	279 ± 9
Fitness	276 ± 9	269 ± 10
Availability	261 ± 11	276 ± 9
Motivation	281 ± 9	271 ± 10

Table 6.45: Rounded means and CIs of feature selection for `clientMotivation3` and `clientMotivation4`, ideal weather scenario.

The MC simulation results for the 2009 weather setting are displayed in Tables 6.46, 6.47, and Figure 6.19. Since the 2009 weather data is taken from recorded measurements, the outcome of this MC simulation can be assumed to deliver the most accurate results in terms of identifying preferred activities.

clientMotivation3 & year2009	lowInt. ind.	lowInt. outd.	medInt. ind.	medInt. outd.	highInt. ind.	highInt. outd.
activity suggestion	43 ± 5	69 ± 6	68 ± 6	54 ± 5	66 ± 5	64 ± 6
activity acceptance	30 ± 5	45 ± 6	44 ± 6	42 ± 5	56 ± 5	42 ± 6

Table 6.46: MC simulation results for `clientMotivation3` and 2009 weather.

For `clientMotivation3`, the top activity (i.e., high intensity indoor training) is accepted the most often on average, at 56 trials. However, the second and third most accepted activities are low intensity outdoor and medium intensity indoor training, both of which are not preferred by the client. Also, the top activity is only the third most recommended suggestion (66 times), after both low (69 times), and medium (68 times) intensity training. However, these averages of activity suggestions still range within the CIs of one another, so that a definite outcome cannot be obtained with 200 MC simulation runs. Investigations into MC simulations with more runs indicate that the top activity is still not the most recommended suggestion.

clientMotivation4 & year2009	lowInt. ind.	lowInt. outd.	medInt. ind.	medInt. outd.	highInt. ind.	highInt. outd.
activity suggestion	68 ± 6	62 ± 6	57 ± 5	65 ± 6	68 ± 5	45 ± 5
activity acceptance	44 ± 6	52 ± 6	45 ± 5	42 ± 6	44 ± 5	31 ± 5

Table 6.47: MC simulation results for `clientMotivation4` and 2009 weather.

Also note that the activity suggestion with the least number of acceptances is the third

most favoured activity of both `clientMotivation3` and `clientMotivation4`.

The behaviour of `clientMotivation4` is similar to that of `clientMotivation3`. The top activity, low intensity outdoor training, is accepted most often, at 52 trials on average. It is also only the third most suggested activity (62 times) after the less desirable choices low and high intensity indoor training, holding both at 68 suggestions on average, and medium outdoor training, which is suggested 65 times.

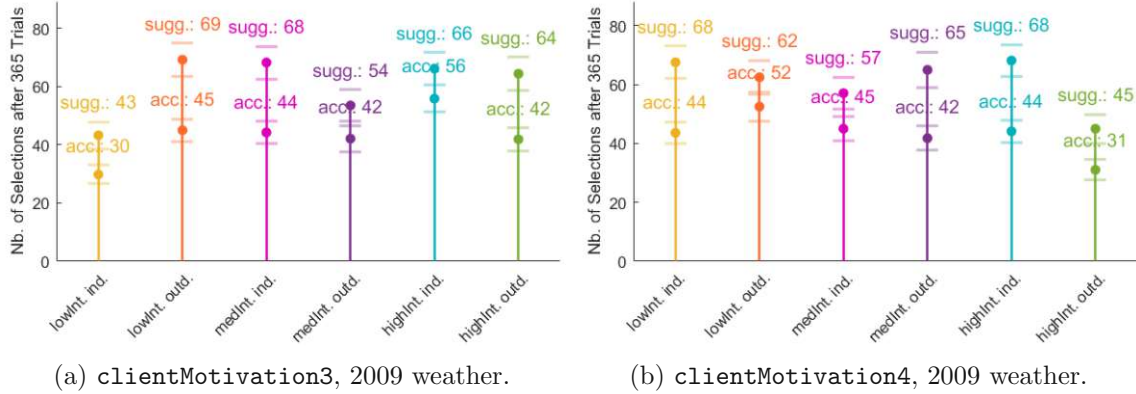


Figure 6.19: MC simulation results for `clientMotivation3` and `clientMotivation4`, and 2009 weather after 365 days.

Figure 6.15 gives a possible explanation as to why the preferred activities are not distinctly identified. Out of 365 data points,  $w_t < 0.3$  holds in 116 cases. Furthermore,  $w_t > 0.7$  only holds in six instances. This means that on 243 days, which is around 67% of trials, the value for motivation is directly taken from the weather data,  $m_t = w_t$ . Due to the large amount of low values for  $m_t$ , the success probability is high for the all desirable activities and either client. In fact, on 267 days the success probability for less popular activities lies above 50%.

Features	<code>clientMotivation3</code>	<code>clientMotivation4</code>
Weather	274 ± 9	266 ± 11
Fitness	268 ± 10	276 ± 9
Availability	275 ± 9	285 ± 8
Motivation	278 ± 9	268 ± 11

Table 6.48: Rounded means and CIs of feature selection for `clientMotivation3` and `clientMotivation4`, 2009 weather scenario.

Table 6.48 shows the average feature selection numbers for the ideal weather scenario. There is no apparent bias towards the motivation feature, and the feature selection is divided fairly evenly between the features for both clients.

## Discussion

During MC simulations with two differently motivated clients, whose motivation is modelled as a function of the weather, it can be observed that the bias towards the favoured activities is apparent, regardless of the weather scenario. However, by looking at a variety of weather conditions, it is clear that a client's favourite activities are not identified in the same way as in Section 6.4.1.

The weather data plays an important part in assessing which activities a client is most likely to accept (i.e., which activities a client is most motivated for). In the bad weather setting, this means that the client actually prefers the less desirable activities over the top three ones, due to the formulae calculating the success probabilities within the client response generator method. On the other hand, the good weather setting amplifies the results from Scenario 6.4.1.

It can be concluded that this model for motivation is a feasible approach towards more complex relationships between different features. However, additional research into more intricate formulae for any feature should be conducted. Yet the issue of modelling a more realistic client is strictly removed from the implementation of the TSRC algorithm, which is proven over and over again to be sufficiently adaptive to different client classes, underlining its applicability as a possible algorithm for decision rules in a JITAI recommender system.

## 6.5 Investigating Feature Sparsity

Investigations into feature sparsity levels are necessary due to the aspect of restricted context in the TSRC algorithm. It means answering the question of how many features need to be regarded so that the simulation outcome yields results sufficiently close to the full-featured TS algorithm.

In all previous simulation scenarios 25% sparsity is considered, so three out of four features are selected to be regarded by the contextual bandit at each trial, see Sections 6.1, 6.2, 6.3, and 6.4. Fixing a level of sparsity for simulations with the TSRC algorithm is a prerequisite for testing the functionality of the algorithm, and the choice of  $n = 3$  is influenced by the results presented by Bouneffouf et al. [16], who show that the performance of the TSRC algorithm at 25% sparsity is almost equal to the full-featured TS algorithm when classifying data sets.

This scenario looks into the sparsity levels not yet investigated in previous simulations, namely  $n = 1$  (75% sparsity),  $n = 2$  (50% sparsity), and  $n = 4$  (0% sparsity, full-featured). One representative setting per isolated feature is selected and the MC simulation results are compared to the earlier outcomes at 25% sparsity.

Note that the implementation of the algorithm does not permit  $n = 0$ , or 100% sparsity (i.e., a context-free bandit algorithm), see Section 5.3.3. Note further that feature sparsity is also relevant when considering cases of missing data. Ideally, the error introduced by restricted context (instead of using all features) should compensate for the problems caused by a potential lack of feature data, see Scenario 6.6.

### Simulation Setup

The general simulation parameters are depicted in Table 6.49. For feasible comparisons with simulation results from previous scenarios, the general simulation parameters remain mostly unchanged. Only the feature budget  $n$  and the algorithm parameter  $v$  vary, since their values adjust according to the sparsity level.

$T = 365$	$s = 200$
$k = 6$	$\alpha = 0.05$
$d = 4$	$R = 0.1, \delta = 0.8, \epsilon = 0.9$

Table 6.49: Simulation parameters for Scenario 6.5.

In order to investigate different levels of sparsity, the feature budget  $n$  must vary between runs. Subsequently, the value for  $v$ , which depends explicitly on  $n$ , also changes. Table 6.50 shows the values for  $v$  in each sparsity setting, with parameters  $R, \delta, \epsilon$ , given in Table 6.49.

Sparsity Level	Feature Budget	Value for $v$
0% Sparsity	$n = 4$	$v \approx 0.4879$
25% Sparsity	$n = 3$	$v \approx 0.4225$
50% Sparsity	$n = 2$	$v \approx 0.3450$
75% Sparsity	$n = 1$	$v \approx 0.2439$

Table 6.50: Different sparsity levels and values for  $v$  in all client settings.

Table 6.51 lists the representative settings for each isolated feature as well as the MC simulation seeds, and provides links to the scenarios for details about the client classes, the exact property values of the respective clients, and the implementation of the client response generator method.

Note that any of the other client settings from previous scenarios may be used for analogous comparisons of feature sparsity levels.

Feature	Representative Setting	Seed	Simulation Setup
Weather	clientWeather1, good weather	8	Scenario 6.1.1
Fitness	fit clientFitness1	89	Scenario 6.2.1
Availability	available clientAvailability1	2542	Scenario 6.3.1
Motivation	clientMotivation1	23	Scenario 6.4.1

Table 6.51: Representative settings for sparsity investigation.

## Results

The results of the MC simulations for the full-featured TS algorithm and all simulation settings are displayed in Figure 6.20. In all four settings, the client’s preferred activities are identified distinctly, and the CIs are visibly narrower for both activity suggestions and acceptances compared to the 25% sparsity setting.

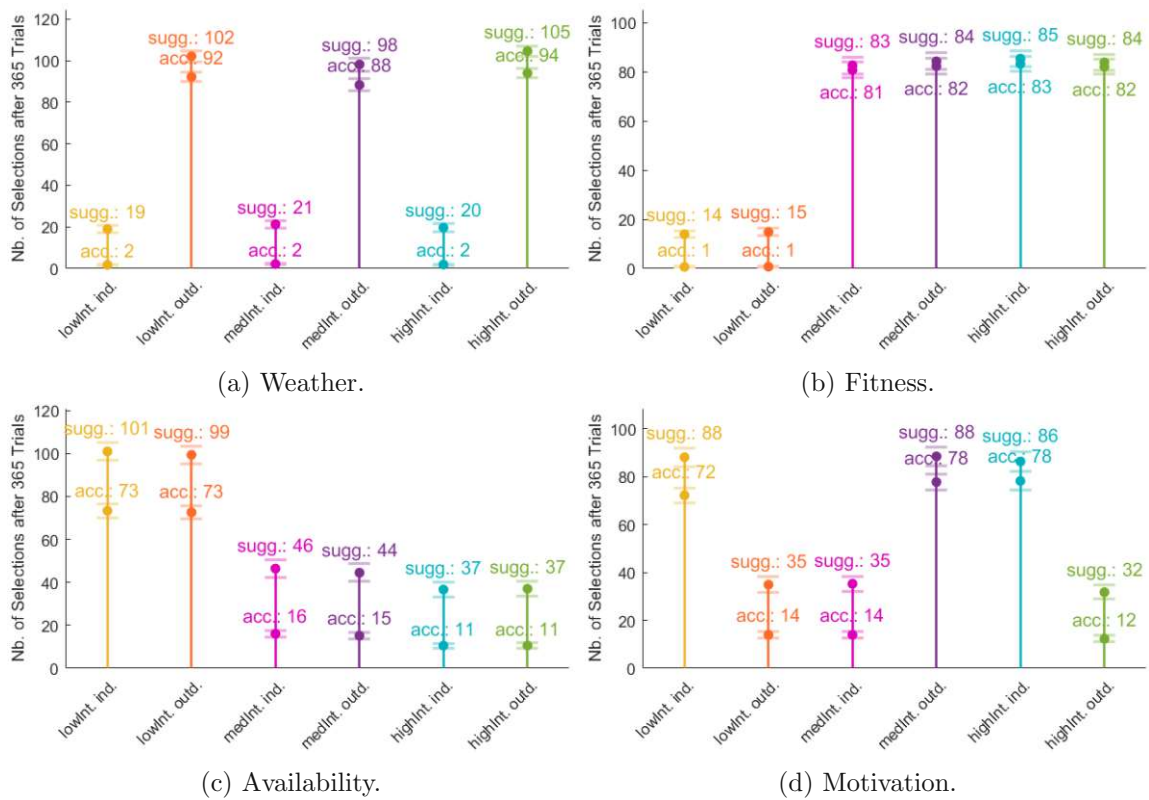


Figure 6.20: MC simulation results for 0% sparsity (i.e., full-featured TS), and all client settings.

When regarding the weather feature, the widths of the CIs for the favoured activities range between 5 – 6% of the average number of suggestions, and between 6 – 7% of the

average number of acceptances, which is identical to those of the 25% sparsity simulations in Scenario 6.1.1.

A slightly larger margin is observed for the fitness feature: the CI widths of the preferred activities lie within 7 – 8% of the average number of suggestions and acceptances in the full-featured setting, but within 9 – 10% at 25% sparsity, see Scenario 6.2.1.

The small increase in CI width is also apparent for the availability feature, where the full-featured TS algorithm produces CIs in the range of around 8% of the average number of suggestions, and around 11% of the average number of acceptances. In contrast, the simulation results of Scenario 6.3.1 show CI widths of 8 – 10% of the average number of activity suggestions, and 11 – 14% of the average number of activity acceptances.

The motivation feature simulation depicts a similar outcome. The CI widths for activity acceptances of the preferred activities are around 9% of the average number of suggestions, whereas for acceptances, the CI widths lie within 10–11% of the average. However, the 25% sparsity setting yields CI widths of 11 – 12% of the average for suggestions, and 11 – 15% of the average for acceptances.

clientWeather1 & good	lowInt. ind.	lowInt. outd.	medInt. ind.	medInt. outd.	highInt. ind.	highInt. outd.
activity suggestion, $n = 4$	19 ± 2	102 ± 3	21 ± 2	98 ± 3	20 ± 2	105 ± 3
activity acceptance, $n = 4$	2 ± 2	92 ± 3	2 ± 2	88 ± 3	2 ± 2	94 ± 3
activity suggestion, $n = 3$	17 ± 2	100 ± 3	20 ± 2	105 ± 3	18 ± 2	104 ± 3
activity acceptance, $n = 3$	2 ± 2	91 ± 3	2 ± 2	95 ± 3	2 ± 2	94 ± 3
activity suggestion, $n = 2$	14 ± 2	110 ± 4	13 ± 2	110 ± 4	17 ± 2	102 ± 5
activity acceptance, $n = 2$	1 ± 2	99 ± 4	1 ± 2	99 ± 4	2 ± 2	91 ± 5
activity suggestion, $n = 1$	12 ± 4	108 ± 9	11 ± 3	113 ± 8	8 ± 3	113 ± 8
activity acceptance, $n = 1$	1 ± 4	97 ± 9	1 ± 3	102 ± 8	1 ± 3	101 ± 8

Table 6.52: MC simulation results for clientWeather1, good weather, and different sparsity levels.

The outcome of the MC simulations for the weather feature in all sparsities is displayed in Table 6.52. On all sparsity levels, the preferred activities (i.e., outdoor training) are correctly identified. However, at 75% sparsity, the width of the CIs for the preferred activities almost triples in size compared to the full-featured version.

A similar observation is made when looking at Figure 6.21, which displays the feature selection for the weather feature at 50%, and 75%, sparsity. The widths of the CIs for features in the 50% sparsity setting range between 15 – 17% of the average number of feature selections. In the case of 75% sparsity, the widths range from 24 – 27% of the

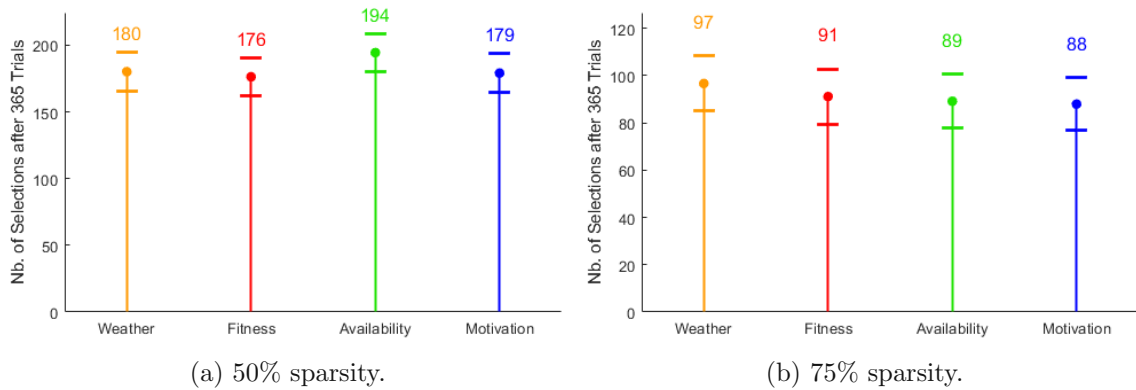


Figure 6.21: Overall feature selection for different sparsity levels, `clientWeather1`, and good weather.

average number of selections. In comparison, the fluctuations for the selection numbers in Scenario 6.1.1 lie between 7 – 12% of the average values.

Table 6.53 shows the rounded means and CIs for the feature selection on all sparsity levels for client `clientWeather1` and the good weather setting.

<code>clientWeather1</code> & good	Weather	Fitness	Availability	Motivation
0% Sparsity ( $n = 4$ )	$365 \pm 0$	$365 \pm 0$	$365 \pm 0$	$365 \pm 0$
25% Sparsity ( $n = 3$ )	$252 \pm 15$	$273 \pm 13$	$285 \pm 12$	$286 \pm 11$
50% Sparsity ( $n = 2$ )	$180 \pm 15$	$176 \pm 14$	$194 \pm 15$	$179 \pm 15$
75% Sparsity ( $n = 1$ )	$97 \pm 12$	$91 \pm 12$	$89 \pm 12$	$88 \pm 11$

Table 6.53: Rounded means and CIs of feature selection for `clientWeather1`, good weather, and different sparsity levels.

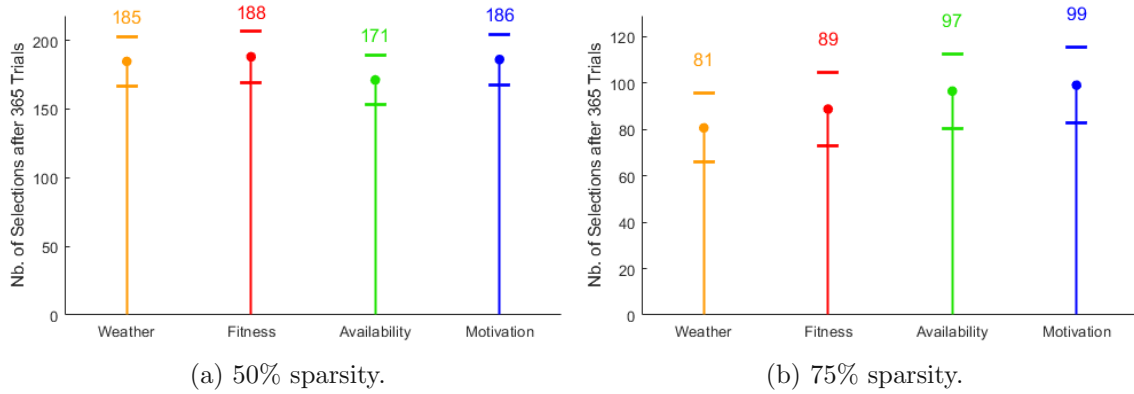
The MC simulation results for the fitness feature are displayed in Table 6.54. In general, the outcome for the preferred activities (i.e., medium and high intensity training) is the same as for the weather feature. The correct features are identified even at low sparsity levels, and the widths of the CIs for the preferred activities grow in proportion to increased sparsity: at 75% sparsity, the width of the CIs are over three times the size of the CIs obtained by the full-featured TS algorithm.

Figure 6.22 shows the means and CIs for the feature selection of `clientFitness1` in the fit property setting. At 75% sparsity, there is a distinct drop in the number of choices for the weather feature previously unseen in lower sparsity levels for the same setup. Also, the widths of the CIs more than triple compared to the “default” case of 25% sparsity: from widths within 11 – 12% of the average number of feature selections, they rise to 32 – 37%.

The rounded means and CIs of the feature selection on all sparsity levels for fit client



clientFitness1	lowInt. ind.	lowInt. outd.	medInt. ind.	medInt. outd.	highInt. ind.	highInt. outd.
activity suggestion, $n = 4$	$14 \pm 1$	$15 \pm 2$	$83 \pm 3$	$84 \pm 3$	$85 \pm 3$	$84 \pm 3$
activity acceptance, $n = 4$	$1 \pm 1$	$1 \pm 2$	$81 \pm 3$	$82 \pm 3$	$83 \pm 3$	$84 \pm 3$
activity suggestion, $n = 3$	$14 \pm 2$	$15 \pm 2$	$84 \pm 4$	$83 \pm 4$	$86 \pm 4$	$83 \pm 4$
activity acceptance, $n = 3$	$1 \pm 2$	$1 \pm 2$	$82 \pm 4$	$81 \pm 4$	$84 \pm 4$	$82 \pm 4$
activity suggestion, $n = 2$	$13 \pm 3$	$11 \pm 2$	$82 \pm 5$	$85 \pm 5$	$84 \pm 6$	$90 \pm 6$
activity acceptance, $n = 2$	$1 \pm 3$	$1 \pm 2$	$80 \pm 5$	$83 \pm 5$	$82 \pm 6$	$88 \pm 6$
activity suggestion, $n = 1$	$14 \pm 6$	$12 \pm 5$	$86 \pm 10$	$80 \pm 10$	$86 \pm 10$	$86 \pm 10$
activity acceptance, $n = 1$	$1 \pm 6$	$1 \pm 5$	$84 \pm 10$	$78 \pm 10$	$84 \pm 10$	$84 \pm 10$

Table 6.54: MC simulation results for fit `clientFitness1` and different sparsity levels.Figure 6.22: Overall feature selection for fit `clientFitness1` and different sparsity levels.

`clientFitness1` are given in Table 6.55.

Table 6.56 displays the rounded means and CIs for the activity suggestions and acceptances of `clientAvailability1` in the available property setting. Again, the triple increase in CI width is observed compared to the results of the full-featured TS algorithm for the client's preferred activities (i.e., low intensity indoor and outdoor training), similar to the simulations for both the weather and fitness feature.

However, the same trend does not wholly apply to the feature selection. Table 6.57 gives the rounded means and CIs of the average feature selection for `clientAvailability1` in the available property setting for different sparsity levels.

Whereas the width of the CI for the weather feature increases from around 10% of the average number of feature selections for  $n = 3$  to only around 19% for  $n = 1$ , the CI width of the availability feature covers around 8% of the average number of feature selections for  $n = 3$ , but almost triples to around 21% for  $n = 1$ .

## 6 Simulation

clientFitness1	Weather	Fitness	Availability	Motivation
0% Sparsity ( $n = 4$ )	365 ± 0	365 ± 0	365 ± 0	365 ± 0
25% Sparsity ( $n = 3$ )	276 ± 15	277 ± 16	274 ± 16	267 ± 16
50% Sparsity ( $n = 2$ )	185 ± 18	188 ± 18	171 ± 18	186 ± 19
75% Sparsity ( $n = 1$ )	81 ± 15	89 ± 16	97 ± 16	99 ± 16

Table 6.55: Rounded means and CIs of feature selection for different sparsity levels and fit `clientFitness1`.

clientAvailability1	lowInt. ind.	lowInt. outd.	medInt. ind.	medInt. outd.	highInt. ind.	highInt. outd.
activity suggestion, $n = 4$	101 ± 4	99 ± 4	46 ± 4	44 ± 4	37 ± 3	37 ± 4
activity acceptance, $n = 4$	73 ± 4	73 ± 4	16 ± 4	15 ± 4	11 ± 3	11 ± 4
activity suggestion, $n = 3$	102 ± 4	100 ± 5	47 ± 5	44 ± 5	37 ± 4	34 ± 4
activity acceptance, $n = 3$	75 ± 4	73 ± 5	16 ± 5	15 ± 5	10 ± 4	10 ± 4
activity suggestion, $n = 2$	92 ± 7	106 ± 7	47 ± 7	47 ± 6	39 ± 6	35 ± 6
activity acceptance, $n = 2$	66 ± 7	76 ± 7	16 ± 7	16 ± 6	11 ± 6	10 ± 6
activity suggestion, $n = 1$	103 ± 12	93 ± 11	44 ± 10	51 ± 11	42 ± 10	32 ± 8
activity acceptance, $n = 1$	74 ± 12	68 ± 11	15 ± 10	17 ± 11	12 ± 10	9 ± 8

Table 6.56: MC simulation results for available `clientAvailability1` and different sparsity levels.

clientAvailability1	Weather	Fitness	Availability	Motivation
0% Sparsity ( $n = 4$ )	365 ± 0	365 ± 0	365 ± 0	365 ± 0
25% Sparsity ( $n = 3$ )	258 ± 13	278 ± 11	281 ± 11	277 ± 11
50% Sparsity ( $n = 2$ )	177 ± 13	191 ± 13	179 ± 13	183 ± 13
75% Sparsity ( $n = 1$ )	96 ± 9	93 ± 9	90 ± 9	86 ± 9

Table 6.57: Rounded means and CIs of feature selection for different sparsity levels and available `clientAvailability1`.

Figure 6.23 shows the MC simulation results for available `clientAvailability1` at 50%, and 75%, sparsity. There is a distinct trend towards the weather feature as the most frequently selected feature at 75% sparsity, compared to 50% sparsity where the most prominent feature is the client's fitness.

The MC simulation results for the isolated motivation feature are displayed in Table 6.58, which shows the rounded means and CIs for activity suggestions and acceptances for

## 6 Simulation

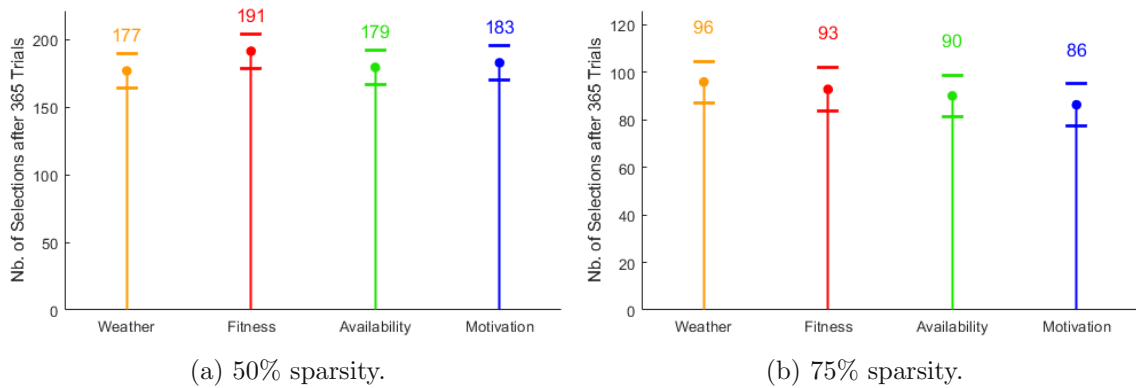


Figure 6.23: Overall feature selection for available `clientAvailability1` and different sparsity levels.

`clientMotivation1`. In contrast to the previous three cases, the CI width increases only by a factor of around 2.4 in the client’s preferred activity suggestions and acceptances (i.e., low intensity indoor training, medium intensity outdoor training, and high intensity indoor training) at 75% sparsity compared to the full-featured setting.

<code>clientMotivation1</code>	lowInt. ind.	lowInt. outd.	medInt. ind.	medInt. outd.	highInt. ind.	highInt. outd.
activity suggestion, $n = 4$	$88 \pm 4$	$35 \pm 3$	$35 \pm 3$	$88 \pm 4$	$86 \pm 4$	$32 \pm 3$
activity acceptance, $n = 4$	$72 \pm 4$	$14 \pm 3$	$14 \pm 3$	$78 \pm 4$	$78 \pm 4$	$12 \pm 3$
activity suggestion, $n = 3$	$81 \pm 5$	$36 \pm 4$	$34 \pm 4$	$88 \pm 5$	$92 \pm 5$	$34 \pm 4$
activity acceptance, $n = 3$	$66 \pm 5$	$14 \pm 4$	$14 \pm 4$	$78 \pm 5$	$78 \pm 5$	$13 \pm 4$
activity suggestion, $n = 2$	$82 \pm 6$	$30 \pm 5$	$34 \pm 5$	$93 \pm 7$	$90 \pm 6$	$37 \pm 5$
activity acceptance, $n = 2$	$66 \pm 6$	$11 \pm 5$	$13 \pm 5$	$82 \pm 7$	$82 \pm 6$	$14 \pm 5$
activity suggestion, $n = 1$	$85 \pm 10$	$31 \pm 9$	$32 \pm 8$	$90 \pm 10$	$95 \pm 10$	$32 \pm 8$
activity acceptance, $n = 1$	$68 \pm 10$	$12 \pm 9$	$12 \pm 8$	$79 \pm 10$	$86 \pm 10$	$12 \pm 8$

Table 6.58: MC simulation results for `clientMotivation1` and different sparsity levels.

Table 6.59 depicts the MC simulation results in terms of feature selection for client `clientMotivation1`. At CI widths between 20 – 25% for  $n = 1$ , the isolated motivation feature setting displays the second largest increase in CI width (after the weather setting) for the feature selection in the combinatorial bandit.

The feature selection numbers for `clientMotivation1` at 50%, and 75%, sparsity are displayed in Figure 6.24a. Note that for 75% sparsity, the motivation feature is chosen significantly less often than the other three features, on average only 80 times. In contrast,

clientMotivation1	Weather	Fitness	Availability	Motivation
0% Sparsity ( $n = 4$ )	$365 \pm 0$	$365 \pm 0$	$365 \pm 0$	$365 \pm 0$
25% Sparsity ( $n = 3$ )	$274 \pm 11$	$270 \pm 12$	$271 \pm 12$	$280 \pm 12$
50% Sparsity ( $n = 2$ )	$176 \pm 13$	$179 \pm 13$	$180 \pm 13$	$194 \pm 14$
75% Sparsity ( $n = 1$ )	$95 \pm 10$	$90 \pm 10$	$99 \pm 10$	$80 \pm 10$

Table 6.59: Rounded means and CIs of feature selection for different sparsity levels and `clientMotivation1`.

it is the most regarded feature for 50% sparsity at 194 selections out of 365 trials.

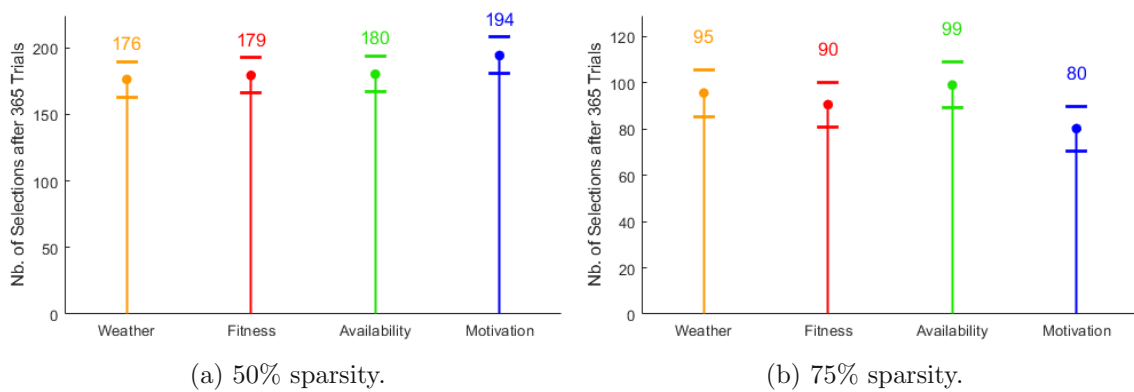


Figure 6.24: Overall feature selection for `clientMotivation1` and different sparsity levels.

## Discussion

The MC simulations at  $n = 4$  prove that the results of Bouneffouf et al. [16] concerning the high accuracy of the 25% sparsity setting are accurate, even when applying the TSRC algorithm as decision rules in a JITAI recommender system. Throughout all simulation settings, there is a noticeable difference between simulation outcomes of the full-featured TS algorithm and the restricted context setting at 25% sparsity; however, the margin is small enough to consider the restricted context setting at  $n = 3$  to be equal to the full-featured TS algorithm. Such small deviations from the full-featured simulation results displayed in Figure 6.20 may be worth accepting in exchange for a mechanism that can potentially solve the problem of missing data.

It can be concluded that all levels of feature sparsity produce roughly the same result in all four simulation settings, correctly identifying the different clients' preferred activities when only regarding one feature. However, the CIs for both activities and features grow wider as the sparsity level increases. Still, in the case of the isolated weather feature, a

sparsity level of 50% yields results sufficiently close to the full-featured algorithm setting in terms of average activity suggestion and acceptance. It is irrelevant to the outcome that the average feature selection numbers show a wider variety between single simulation runs of the MC simulation. In light of increased sparsity, this development is expected: the less features the algorithm can observe at a trial, the longer it takes to identify which ones are useful, and which ones are not. If anything, it illustrates how adaptive the TSRC algorithm is to changing situations, being able to identify the correct activities without having to overuse the observed weather data.

A short investigation into the trade-off between the number of MC simulation runs  $s$ , and the level of sparsity  $d$ , almost confirms that, even at lower sparsity levels, results of similar or higher quality (comparable results, narrower CIs) to the full-featured setting can be achieved when increasing  $s$  for fixed CI parameter  $\alpha$ , underlining that the TSRC algorithm can still correctly identify a client's preferred activities at high sparsity.

However, the TSRC algorithm may not yield results this explicit for more complicated client models. Modelling accurate client responses is vital because a JITAI recommender system must be proven to be effective in order to move on to a clinical trial, where its usefulness is determined by running tests with real-life participants. Since the TSRC algorithm is proven to be accurate for different sparsity levels and different features, research into more elaborate client models ought to be the next step in investigating the algorithm's usefulness as decision rules.

## 6.6 The Case of Missing Data

The implications of faulty measurements, or missing data, on both the JITAI and the TSRC algorithm are discussed in Sections 2.2.2 and 4.4.1. This simulation scenario examines how the TSRC algorithm deals with cases of missing data, and investigates its potential to solve any of the problems that occur within a JITAI recommender system when it is sporadically lacking feature data. In accordance with Scenario 6.5, the same four simulation setups are employed, each representing an isolated feature setting from previous sections.

### Simulation Setup

The general simulation parameters are depicted in Table 6.60. Note that the sparsity level is fixed at 25%, and varying sparsity levels are not regarded.

Two different values for the missing data probability  $q$  are considered:  $q = 0.2$  and  $q = 0.7$ . As explained in Section 5.3.3, at each trial  $t$ , one of the features  $w_t$ ,  $f_t$ ,  $a_t$ ,  $m_t$ , is missing its data with probability  $q$ . The function `missingDataGenerator.m` is implemented in

such a way that the case of missing data is established before, and the respective feature exhibiting lack of data is selected thereafter.

$T = 365$	$n = 3$	$R = 0.1, \delta = 0.8, \epsilon = 0.9$
$k = 6$	$s = 200$	$v \approx 0.4225$
$d = 4$	$\alpha = 0.05$	$q = 0.2 \vee q = 0.7$

Table 6.60: Simulation parameters for Scenario 6.6.

The client classes as well as the seeds used for simulation are depicted in Table 6.51.

## Results

The MC simulation outcomes for `clientWeather1` and the good weather setting show almost identical results between both missing data probabilities, in terms of the rounded means and CIs for activity suggestions and acceptances. Furthermore, they are exceptionally close to the simulation results of the default case in Scenario 6.1.1, see Tables 6.4 and 6.61. The difference for all activities is so minimal that it can be neglected.

<code>clientWeather1</code> & <b>good</b>	lowInt. ind.	lowInt. outd.	medInt. ind.	medInt. outd.	highInt. ind.	highInt. outd.
activity suggestion, $q = 0.2$	$18 \pm 2$	$103 \pm 4$	$18 \pm 2$	$104 \pm 4$	$17 \pm 2$	$104 \pm 3$
activity acceptance, $q = 0.2$	$2 \pm 2$	$92 \pm 4$	$2 \pm 2$	$94 \pm 4$	$2 \pm 2$	$94 \pm 3$
activity suggestion, $q = 0.7$	$19 \pm 2$	$103 \pm 3$	$20 \pm 2$	$102 \pm 3$	$19 \pm 2$	$102 \pm 3$
activity acceptance, $q = 0.7$	$2 \pm 2$	$93 \pm 3$	$2 \pm 2$	$92 \pm 3$	$2 \pm 2$	$92 \pm 3$

Table 6.61: MC simulation results for different missing data probabilities, `clientWeather1`, and good weather.

The same outcome is observed for the other three isolated features. Table 6.62 gives the rounded means and CIs for activity suggestions and acceptances in both missing data probabilities for the isolated fitness feature, and Table 6.17 in Scenario 6.2.1 provides the MC simulation results for the default case.

The (rounded) widths of the CIs are identical for each activity. Furthermore, the less desired activities (i.e., low intensity training) show average suggestion rates of around 14 – 15, and the preferred activities for the fit client (i.e., medium and high intensity training) exhibit average suggestion rates of just over 80 in all three cases.

Table 6.63 displays the MC simulation results for activity selection and acceptance in the isolated availability feature setting for both missing data cases. The widths of the CIs for the preferred activities (i.e., low intensity training) remain at around 10% of both the average numbers for activity suggestions and acceptances.

clientFitness1	lowInt. ind.	lowInt. outd.	medInt. ind.	medInt. outd.	highInt. ind.	highInt. outd.
activity suggestion, $q = 0.2$	14 ± 2	14 ± 2	85 ± 4	81 ± 4	84 ± 4	86 ± 4
activity acceptance, $q = 0.2$	1 ± 2	1 ± 2	83 ± 4	80 ± 4	82 ± 4	84 ± 4
activity suggestion, $q = 0.7$	14 ± 2	14 ± 2	83 ± 4	84 ± 4	88 ± 4	82 ± 4
activity acceptance, $q = 0.7$	1 ± 2	1 ± 2	81 ± 4	83 ± 4	85 ± 4	80 ± 4

Table 6.62: MC simulation results for `clientFitness1` and different missing data probabilities.

clientAvailability1	lowInt. ind.	lowInt. outd.	medInt. ind.	medInt. outd.	highInt. ind.	highInt. outd.
activity suggestion, $q = 0.2$	99 ± 5	103 ± 5	41 ± 5	47 ± 5	38 ± 4	36 ± 4
activity acceptance, $q = 0.2$	72 ± 5	75 ± 5	14 ± 5	16 ± 5	11 ± 4	10 ± 4
activity suggestion, $q = 0.7$	100 ± 5	102 ± 5	48 ± 5	50 ± 5	32 ± 4	33 ± 4
activity acceptance, $q = 0.7$	73 ± 5	74 ± 5	16 ± 5	18 ± 5	9 ± 4	9 ± 4

Table 6.63: MC simulation results for `clientAvailability1` and different missing data probabilities.

clientMotivation1	lowInt. ind.	lowInt. outd.	medInt. ind.	medInt. outd.	highInt. ind.	highInt. outd.
activity suggestion, $q = 0.2$	83 ± 5	37 ± 4	33 ± 4	88 ± 5	91 ± 4	34 ± 4
activity acceptance, $q = 0.2$	68 ± 5	15 ± 4	13 ± 4	77 ± 5	82 ± 4	13 ± 4
activity suggestion, $q = 0.7$	84 ± 5	33 ± 4	31 ± 4	92 ± 5	92 ± 5	34 ± 4
activity acceptance, $q = 0.7$	68 ± 5	13 ± 4	12 ± 4	80 ± 5	84 ± 5	13 ± 4

Table 6.64: MC simulation results for `clientMotivation1` and different missing data probabilities.

The rounded means and CIs for activity suggestions and acceptances for the isolated motivation feature are shown in Table 6.64. Compared to the default case depicted in Table 6.33, the CI widths are the same for both missing data probabilities. In the default setting, around 90% of the suggestions for the client’s favourite activity (i.e., high intensity indoor training) are accepted, followed by around 89% for their second favourite activity (i.e., medium intensity indoor training), and 82% for their third most popular activity (i.e., low intensity indoor training). In the missing data setting, these acceptance rates are found to be 90%, 88%, and 82% for  $q = 0.2$ , and 91%, 87%, and 81% for  $q = 0.7$ .

The same trend of minimal differences to the default case is exhibited in all MC simulation

Simulation Setup	Weather	Fitness	Availability	Motivation
clientWeather1, good, $q = 0.2$	$274 \pm 8$	$274 \pm 8$	$274 \pm 8$	$273 \pm 8$
clientWeather1, good, $q = 0.7$	$275 \pm 2$	$272 \pm 3$	$274 \pm 3$	$275 \pm 5$
clientFitness1, $q = 0.2$	$275 \pm 10$	$271 \pm 10$	$275 \pm 10$	$274 \pm 10$
clientFitness1, $q = 0.7$	$270 \pm 3$	$274 \pm 3$	$274 \pm 3$	$277 \pm 2$
clientAvailability1, $q = 0.2$	$275 \pm 7$	$272 \pm 7$	$276 \pm 7$	$271 \pm 8$
clientAvailability1, $q = 0.7$	$275 \pm 2$	$273 \pm 3$	$275 \pm 2$	$273 \pm 3$
clientMotivation1, $q = 0.2$	$269 \pm 8$	$267 \pm 8$	$271 \pm 8$	$288 \pm 6$
clientMotivation1, $q = 0.7$	$274 \pm 2$	$272 \pm 2$	$273 \pm 2$	$276 \pm 2$

Table 6.65: Rounded means and CIs of feature selection for all four client classes and different missing data probabilities.

results regarding feature selection, see Table 6.65. Three out of four isolated feature settings show little discrepancy in the average number of feature selections between both missing data probabilities. The isolated motivation feature displays a different dynamic: the default setting as well as the missing data setting for  $q = 0.2$  distinctly select the motivation feature the most often compared to the other three. However, for  $q = 0.7$ , this favouritism is equalled out, suggesting that the missing data generator may have affected the motivation feature disproportionately more than the other three.

Note that the rounded means and CIs for feature selection in the default case are found in the left column of Tables 6.4, 6.17, 6.27, and 6.33.

## Discussion

The proximity of the results for the missing data setting to the default case is not unexpected. Depending on how many features are assumed to be missing at the same time and the budget  $n$ , the TSRC algorithm possesses a natural way of dealing with missing data, without compromising the learning procedure with erroneous information: as long as the number of disregarded features  $d - n$  is equal to the amount of features that are expected to have unreliable data values in the same iteration, the TSRC algorithm is expected to show similar performance to the default case, because it will omit  $d - n$  features regardless of whether the neglected information is viable. Due to the implementation of the missing data generator function, see Section 5.3.3, no more than one feature may be missing per trial. Since,  $d - n = 1$ , the TSRC algorithm disregards one feature anyway.

However, problems may arise when more than  $d - n$  feature values are invalid. The fixed budget  $n$  ought to be set according to the amount of features that are expected to be faulty



in the same iteration. In situations where this number is exceeded, the TSRC algorithm should not recommend adversary suggestions to a client, and additional mechanisms must be considered to compensate for these cases.

For example, it is possible to introduce a “dummy value” that is assigned to feature  $j$  in case its measurements are erroneous, in addition to setting its combinatorial bandit draw,  $\theta_j$ , to zero. Most times, when the feature budget  $n$  covers the lack of data, the data value of  $j$  is not regarded. However, if  $\theta_j = 0$  for more than  $d - n$  features, at least one of the erroneous feature data is chosen, in which case the feature value should not distort the results too much. For the model JITAI, this could mean setting the respective feature value to 0.5, since it is the default value for all class properties.

In order to investigate the TSRC algorithm’s response to this (more realistic) portrayal of the case of missing data, the codes for the algorithm `TSRC.m` and the missing data generator function `missingDataGenerator.m` can be extended to include this functionality.

## 7 Conclusion and Outlook

The outcomes of the simulation scenarios in Chapter 6 confirm that the TSRC algorithm ought to be considered as the decision rules component in a JITAI recommender system. The algorithm correctly identifies the activity preferences of various model clients that base the decision of whether or not to accept an activity suggestion on a singular contextual feature, see Scenarios 6.1.1, 6.2.1, and 6.3.1. Furthermore, the results also reveal that no bias is formed towards activities linked to the isolated feature when the client considers all feature values except for the isolated feature, see Scenarios 6.1.2, 6.2.2, and 6.3.2. The linear relationship between feature values and success probabilities for activity suggestions is thus easily recognised by the TSRC algorithm. Even when raising the complexity level of the connection between the isolated feature and the client's reaction, see Scenarios 6.4.1 and 6.4.2, the algorithm correctly identifies the client's favoured activities in order of preference.

There is no apparent reason why the TSRC algorithm should not yield equally authentic results when interacting with a real person. However, the stationarity of the model clients must be acknowledged. In a real-life setting, a client is likely to show inconsistent behaviour, or to go through periods of consistency, and thus the learning rate of the TSRC algorithm is most likely slowed down. An attempt has been made by Bouneffouf et al. [16] to tackle non-stationary problems using TS: as mentioned in Section 3.4.3, the WTSRC algorithm is an extension of the TSRC algorithm that assumes periods of stationarity. It allows the number of successes  $S_j$  and failures  $F_j$ ,  $j \in \{1, \dots, d\}$ , which influence the combinatorial bandit (i.e., the feature selection) directly and the contextual bandit (i.e., the activity selection) indirectly, to reset within each period window. This way, adaptivity can be achieved within a pre-defined window length.

In general, the next step in the analysis of the TSRC algorithm as decision rules should be to model more complex (i.e., more realistic) client behaviour. Many aspects of client behaviour have been simplified, or omitted, when creating clients for the model JITAI, for the purpose of obtaining clearly interpretable simulation results. For example, intervention engagement and fatigue have not been regarded in model clients or the model JITAI. However, as mentioned in Chapter 2, these mechanisms are shown to significantly influence a client's attitude towards the JITAI intervention design.

A possible way to expand existing client classes, and include these processes, is to consider different delivery systems for the same activity as separate suggestions. For example, if there are two different notification messages that deliver the same activity suggestion, they may be regarded as two separate suggestions. In the case of the model JITAI from Section 5.1, the number of possible activities increases from 6 to 12. The response generator within the client class then calculates different success probabilities for all 12 activity suggestions. In turn, the model JITAI may count the number of times the suggestions have been accepted in the past, and if the adherence to one of them is low, the program may switch to the alternative notification.

The performance of the TSRC algorithm in terms of feature sparsity exceeds expectations. On average, the simulation results at 75% sparsity still mirror the full-featured case throughout all isolated feature setups. This indicates that feature sparsity offers sufficient levels of adaptivity to compensate for erroneous, or missing, feature values. However, the performance of the TSRC algorithm is expected to decline with an increasing number of features  $d$ . Furthermore, the natural trade-off between feature sparsity and missing data cannot work if more than  $d - n$  feature values are unavailable. The discussion of Scenario 6.6 elaborates on the problems arising from this issue and offers dummy values as one possible solution. Another approach is to look into machine-learning based data compensation methods that can generate feature values that are most likely to occur at a specific decision point. In any case, research should be conducted into comparing the TSRC algorithm to the full-featured TS algorithm working with data compensation in terms of performance and efficiency, and a combination of both concepts is likely to be similarly competitive.

An effort should also be made to further research theoretical performance evaluation. As mentioned in Chapter 3, TS is only recently being analysed in terms of computational effort (i.e., the expected regret) [2], namely regret bounds [79]. A theoretical background like this does not yet exist for the TSRC algorithm, but it might shed further light on the applicability of the algorithm as JITAI decision rules.

Even with all these boxes ticked, the road to a functional JITAI is still long: Chapter 2 elaborates on the design and implementation facets of working JITAIs, for example deciding what functionality the JITAI will have regarding illness management, identifying intervention options, or conducting MRTs. The former requires an already programmed mobile phone application, which can only be created after the RL algorithm behind the intervention recommendation is proven to be functional.

It is apparent that further research into the subject of RL algorithms as decision rules in JITAI recommender system touches on many disciplines: data science, statistics, behavioural psychology, medicine, mathematics, engineering, and computer science must naturally play a part in both the planning and execution of the final product. In terms of

mHealth, much progress has already been made, but equal amounts of work, or even more, still lie ahead. Making health services easily available to all global citizens is a target set by the WHO [89], and mHealth applications can help pave the way. After all, it can be concluded that the topic of JITAIs in mHealth, and the underlying algorithms that make them possible, is progressing at a remarkable speed, helping both patients and caretakers enter into a new era in healthcare and medicine.

## List of Algorithms

1	Multi-Armed Bandit Algorithm . . . . .	29
2	Contextual Multi-Armed Bandit Algorithm . . . . .	31
3	Greedy Strategy . . . . .	38
4	$\epsilon$ -Greedy Strategy . . . . .	39
5	Upper Confidence Bound Strategy . . . . .	42
6	Thompson Sampling Strategy . . . . .	50
7	Thompson Sampling for the CMAB Problem . . . . .	52
8	Contextual Bandit with Restricted Context Strategy . . . . .	68
9	Thompson Sampling for Combinatorial Bandits . . . . .	69
10	Thompson Sampling with Restricted Context . . . . .	71

## List of Figures

2.1	Intervention concept for the JITAI design, after [58]. . . . .	6
2.2	Summary of the pragmatic framework for developing JITAI, from [57]. . . . .	13
2.3	Screenshot of the HeartSteps app in Google Play Store and App Store, accessed on 2021-08-24. . . . .	14
2.4	Summary of the mHealth initiatives reported to the WHO in its member states in 2011, from [89]. . . . .	16
2.5	Screenshot of the A-CHESS smartphone app in the App Store, accessed on 2021-10-08. . . . .	18
2.6	Staged development of the FOCUS mobile phone application, after [13]. . . . .	19
2.7	Screenshot of activity and food suggestion user interface in MyBehavior, from [66]. . . . .	20
2.8	Main screen of the SitCoach application, from [82]. . . . .	22
2.9	A simplified illustration of Fogg’s behavior model, after [26]. . . . .	23
3.1	Visualisation of the Bernoulli MAB problem, after [70]. . . . .	26
3.2	Pulling the arm of the $i$ -th bandit is equal to pulling the $i$ -th arm of a $k$ -armed bandit. . . . .	27
3.3	Visual representation of the analogy between the elements of the CMAB approach and the components of the JITAI design. . . . .	33
3.4	JITAI concept from Figure 2.1 adapted for the use of a CMAB algorithm. . . . .	33
3.5	Regret curves for different strategies, schematically sketched after [70]. . . . .	36
3.6	Example of greedy strategy simulations in a three-armed bandit with initial estimates set to 0.5 and true values (0.3, 0.7, 0.8), after [70]. . . . .	37
3.7	Comparison between greedy and $e$ -greedy strategy for the three-armed Bernoulli bandit and different values of $e$ , from [70]. . . . .	39
3.8	Decaying $e$ -greedy strategy for different decay schedules, from [70]. . . . .	40
3.9	Visual representation of exploration and exploitation in a UCB strategy. . . . .	43
3.10	Density functions and upper confidence bounds for the three-armed Bernoulli bandit example at 80%, after [70]. . . . .	43
3.11	Bayesian update rule for beta distribution prior and Bernoulli observation, assuming success, after [70]. . . . .	46

3.12 Example of 10 iterations of Bayesian updates assuming beta priors, posteriors, and Bernoulli observations. . . . .	47
3.13 UCB strategy for different upper confidence limits, from [70]. . . . .	48
3.14 Probability density function and mean of the beta distribution, for different values of $\alpha$ , $\beta$ . . . . .	49
3.15 Illustration of the Thompson Sampling approach in a three-armed Bernoulli bandit, after [70]. . . . .	51
3.16 Screenshots of different user interfaces in the mobile phone apps discussed in Section 3.5. . . . .	56
4.1 Illustration of the direct (solid line) and indirect (dashed line) influences on the transition into a new state via a policy. . . . .	60
4.2 Graphical representation of the relationship between CMABs and MDPs, and the different learning paradigms solving them. . . . .	62
4.3 Example of an interaction between agent and environment from an RL perspective, after [83]. . . . .	63
4.4 A clinical trial conceptualised as a CMAB algorithm, compare Figure 3.4. . . . .	65
4.5 Composition of a policy $q$ in case of restricted context. . . . .	72
5.1 Sliding scale for assessing a client’s motivation, and normalisation of the value. . . . .	77
5.2 Possible influence pattern on the intervention options. Darker areas represent high probabilities of activity acceptance when feature values are high. . . . .	78
5.3 The model JITAI working along a timeline of 24h observation periods, showing both cases of detecting and not detecting the suggested activity. . . . .	80
5.4 Implementation tree illustrating the MATLAB class properties (blue) and methods (purple) for clients. . . . .	83
5.5 Illustration of the idealised weather data for one year via the sine function. . . . .	86
5.6 Illustration of the normalised weather data for 2009. . . . .	87
5.7 Implementation tree of <code>weatherGenerator.m</code> , showing the functions (yellow) and text files (brown) involved. . . . .	87
5.8 Implementation tree of <code>monteCarloTSRC.m</code> , showing the functions, client properties, and methods called during a MC simulation. . . . .	90
6.1 Client response for single runs of <code>TSRC.m</code> and <code>clientWeather1</code> . . . . .	97
6.2 Client response to activity suggestions over time, ideal weather setting. . . . .	98
6.3 MC simulation results for <code>clientWeather1</code> after 365 days for ideal and 2009 weather. . . . .	98
6.4 MC simulation results for <code>clientWeather2</code> after 365 days for good and bad weather. . . . .	102
6.5 Client response for single runs of <code>TSRC.m</code> and <code>clientFitness1</code> . . . . .	107

6.6	MC simulation results for fit and unfit <code>clientFitness1</code> after 365 days. . . .	107
6.7	Success probabilities for three exercise intensity levels, for modelling the response of <code>clientFitness1</code> . . . . .	109
6.8	MC simulation results for fit and unfit <code>clientFitness2</code> after 365 days. . . .	110
6.9	Success probabilities for three exercise intensity levels, for modelling the response of <code>clientAvailability1</code> . . . . .	114
6.10	Total number of suggestions and acceptances for available and unavailable <code>clientAvailability1</code> after 365 days. . . . .	114
6.11	MC simulation results for available and unavailable <code>clientAvailability1</code> after 365 days. . . . .	115
6.12	MC simulation results for available and unavailable <code>clientAvailability2</code> after 365 days. . . . .	117
6.13	Total number of suggestions and acceptances for <code>clientMotivation1</code> and <code>clientMotivation2</code> after 365 days. . . . .	121
6.14	MC simulation results for <code>clientMotivation1</code> and <code>clientMotivation2</code> after 365 days. . . . .	122
6.15	Motivation values as a function of the weather, 2009 weather scenario. . . .	124
6.16	MC simulation results for <code>clientMotivation3</code> and <code>clientMotivation4</code> , and good weather after 365 days. . . . .	125
6.17	MC simulation results for <code>clientMotivation3</code> and <code>clientMotivation4</code> , and bad weather after 365 days. . . . .	127
6.18	MC simulation results for <code>clientMotivation3</code> and <code>clientMotivation4</code> , and ideal weather after 365 days. . . . .	128
6.19	MC simulation results for <code>clientMotivation3</code> and <code>clientMotivation4</code> , and 2009 weather after 365 days. . . . .	130
6.20	MC simulation results for 0% sparsity (i.e., full-featured TS), and all client settings. . . . .	133
6.21	Overall feature selection for different sparsity levels, <code>clientWeather1</code> , and good weather. . . . .	135
6.22	Overall feature selection for fit <code>clientFitness1</code> and different sparsity levels. . . . .	136
6.23	Overall feature selection for available <code>clientAvailability1</code> and different sparsity levels. . . . .	138
6.24	Overall feature selection for <code>clientMotivation1</code> and different sparsity levels. . . . .	139



## List of Tables

2.1	Overview and short description of applications using the JITAI design. . . .	18
3.1	Strategies used in CMAB algorithms with short descriptions. . . . .	35
3.2	Short description of MAB- and CMAB-based applications in mHealth. . . .	55
4.1	List and short descriptions of algorithms tested against TSRC, from [16]. . .	73
5.1	Summary of the JITAI elements of the model JITAI. . . . .	79
5.2	Properties and their descriptions of the MATLAB classes for all clients. . . .	81
5.3	List of all client MATLAB classes used in simulations. . . . .	84
5.4	List of plots generated after a single run of TSRC.m. . . . .	89
5.5	List of plots generated after a MC simulation with TSRC.m. . . . .	91
5.6	Sparsity levels for the TSRC algorithm. . . . .	92
6.1	Simulation parameters for Scenario 6.1.1. . . . .	95
6.2	Class property values for <code>clientWeather1</code> in Scenario 6.1.1. . . . .	95
6.3	Weather feature values and seeds (single run, MC) for <code>clientWeather1</code> in Scenario 6.1.1. . . . .	96
6.4	MC simulation results for good weather and <code>clientWeather1</code> . . . . .	96
6.5	MC simulation results for bad weather and <code>clientWeather1</code> . . . . .	97
6.6	MC simulation results for 2009 weather and <code>clientWeather1</code> . . . . .	99
6.7	MC simulation results for ideal weather and <code>clientWeather1</code> . . . . .	99
6.8	Rounded means and CIs of feature selection for all weather scenarios and <code>clientWeather1</code> . . . . .	99
6.9	MC simulation results for good weather and <code>clientWeather2</code> . . . . .	102
6.10	MC simulation results for bad weather and <code>clientWeather2</code> . . . . .	102
6.11	MC simulation results for 2009 weather and <code>clientWeather2</code> . . . . .	103
6.12	MC simulation results for ideal weather and <code>clientWeather2</code> . . . . .	103
6.13	Rounded means and CIs of feature selection for all weather scenarios and <code>clientWeather2</code> . . . . .	103
6.14	Class property values for fit <code>clientFitness1</code> in Scenario 6.2.1. . . . .	105
6.15	Class property values for unfit <code>clientFitness1</code> in Scenario 6.2.1. . . . .	105
6.16	Seed values for <code>clientFitness1</code> in Scenario 6.2.1. . . . .	106
6.17	MC simulation results for fit <code>clientFitness1</code> . . . . .	106

6.18	MC simulation results for unfit <code>clientFitness1</code> . . . . .	108
6.19	Rounded means and CIs of feature selection for fit and unfit <code>clientFitness1</code> . . . . .	108
6.20	MC simulation results for fit <code>clientFitness2</code> . . . . .	110
6.21	MC simulation results for unfit <code>clientFitness2</code> . . . . .	111
6.22	Rounded means and CIs of feature selection for fit and unfit <code>clientFitness2</code> . . . . .	111
6.23	Class property values for available <code>clientAvailability1</code> in Scenario 6.3.1. . . . .	113
6.24	Class property values for unavailable <code>clientAvailability1</code> in Scenario 6.3.1. . . . .	113
6.25	Seed values for <code>clientAvailability1</code> in Scenario 6.3.1. . . . .	113
6.26	MC simulation results for unavailable <code>clientAvailability1</code> . . . . .	115
6.27	MC simulation results for available <code>clientAvailability1</code> . . . . .	115
6.28	Rounded means and CIs of feature selection for available and unavailable <code>clientAvailability1</code> . . . . .	116
6.29	MC simulation results for available <code>clientAvailability2</code> . . . . .	117
6.30	MC simulation results for unavailable <code>clientAvailability2</code> . . . . .	118
6.31	Rounded means and CIs of feature selection for available and unavailable <code>clientAvailability2</code> . . . . .	118
6.32	Class property values for <code>clientMotivation1</code> and <code>clientMotivation2</code> in Scenario 6.4.1. . . . .	120
6.33	MC simulation results for <code>clientMotivation1</code> . . . . .	121
6.34	MC simulation results for <code>clientMotivation2</code> . . . . .	122
6.35	Rounded means and CIs of feature selection for <code>clientMotivation1</code> and <code>clientMotivation2</code> . . . . .	122
6.36	Seed values for the MC simulations for different weather scenarios and both <code>clientMotivation3</code> and <code>clientMotivation4</code> . . . . .	124
6.37	MC simulation results for <code>clientMotivation3</code> and good weather. . . . .	125
6.38	MC simulation results for <code>clientMotivation4</code> and good weather. . . . .	125
6.39	Rounded means and CIs of feature selection for <code>clientMotivation3</code> and <code>clientMotivation4</code> , good weather scenario. . . . .	126
6.40	MC simulation results for <code>clientMotivation3</code> and bad weather. . . . .	126
6.41	MC simulation results for <code>clientMotivation4</code> and bad weather. . . . .	126
6.42	Rounded means and CIs of feature selection for <code>clientMotivation3</code> and <code>clientMotivation4</code> , bad weather scenario. . . . .	127
6.43	MC simulation results for <code>clientMotivation3</code> and ideal weather. . . . .	128
6.44	MC simulation results for <code>clientMotivation4</code> and ideal weather. . . . .	128
6.45	Rounded means and CIs of feature selection for <code>clientMotivation3</code> and <code>clientMotivation4</code> , ideal weather scenario. . . . .	129
6.46	MC simulation results for <code>clientMotivation3</code> and 2009 weather. . . . .	129

6.47 MC simulation results for <code>clientMotivation4</code> and 2009 weather. . . . .	129
6.48 Rounded means and CIs of feature selection for <code>clientMotivation3</code> and <code>clientMotivation4</code> , 2009 weather scenario. . . . .	130
6.49 Simulation parameters for Scenario 6.5. . . . .	132
6.50 Different sparsity levels and values for $v$ in all client settings. . . . .	132
6.51 Representative settings for sparsity investigation. . . . .	133
6.52 MC simulation results for <code>clientWeather1</code> , good weather, and different sparsity levels. . . . .	134
6.53 Rounded means and CIs of feature selection for <code>clientWeather1</code> , good weather, and different sparsity levels. . . . .	135
6.54 MC simulation results for fit <code>clientFitness1</code> and different sparsity levels. . .	136
6.55 Rounded means and CIs of feature selection for different sparsity levels and fit <code>clientFitness1</code> . . . . .	137
6.56 MC simulation results for available <code>clientAvailability1</code> and different spar- sity levels. . . . .	137
6.57 Rounded means and CIs of feature selection for different sparsity levels and available <code>clientAvailability1</code> . . . . .	137
6.58 MC simulation results for <code>clientMotivation1</code> and different sparsity levels. .	138
6.59 Rounded means and CIs of feature selection for different sparsity levels and <code>clientMotivation1</code> . . . . .	139
6.60 Simulation parameters for Scenario 6.6. . . . .	141
6.61 MC simulation results for different missing data probabilities, <code>clientWeather1</code> , and good weather. . . . .	141
6.62 MC simulation results for <code>clientFitness1</code> and different missing data prob- abilities. . . . .	142
6.63 MC simulation results for <code>clientAvailability1</code> and different missing data probabilities. . . . .	142
6.64 MC simulation results for <code>clientMotivation1</code> and different missing data probabilities. . . . .	142
6.65 Rounded means and CIs of feature selection for all four client classes and different missing data probabilities. . . . .	143

# Index

- A-CHESS, 8, 17
- Active assessment, 8
- Adaptive Intervention (AI), 4
  
- Bayes theorem, 44, 45
- Bernoulli bandit, 29, 35
- Bernoulli distribution, 34
- Beta distribution, 48
  
- CalFit, 55
- Central limit theorem, 89
- Combinatorial bandit, 67
- Context-free bandit, 27
- Contextual bandit algorithm, 31
- Contextual bandit problem, 31
- Contextual bandit with restricted context (CBRC), 65
- Contextual multi-armed bandit (CMAB), 25
- Contextual multi-armed bandits, 30
  
- Decision points, 6, 7
- Decision rules, 6, 10
- Distal Outcome, 6
  
- e-Greedy strategy, 35, 36
- Ecological momentary assessment, 8
- Epoch-greedy strategy, 41
- Evidence, 44
- Expected reward, 28
- Expected T-trial regret, 28
  
- Exploration parameter, 38
- Exploration-exploitation trade-off, 26
  
- FOCUS, 19
- Fogg's behavior model, 23
  
- HeartSteps, 14, 55
  
- Intervention concept, 6
- Intervention engagement, 9
- Intervention fatigue, 7, 9
- Intervention options, 6, 9
- Intervention protocol, 5
  
- JTIAI, 4
- Just-in-time (JIT), 4
  
- Likelihood function, 44
  
- Markov decision process, 59
- Markov property, 61
- Micro-randomized trial, 146
- Micro-randomized trial (MRT), 13
- Mobile health (mHealth), 15
- Monte Carlo simulation, 88
- Multi-armed bandit, 25
- MyBehavior, 20, 55
  
- Optimal expected T-trial reward, 28
- Optimistic initialisation, 40
  
- Passive assessment, 8
- Policy, 59, 62, 63

- PopTherapy, 55
- Posterior probability, 44
- Prior probability, 44
- Proximal outcome, 6, 11
- Reinforcement learning (RL), 54, 62
- Reward function, 59
- Semi-adversarial bandit, 31
- Semi-stochastic bandit, 31
- SitCoach, 21
- State of opportunity, 5
- State of vulnerability, 5
- Stochastic bandit, 31
- Supervised learning, 62
- Tailoring variables, 6, 8
- Temporal Progression, 22
- Temporal progression, 12
- Thompson Sampling (TS), 35, 48
- Thompson Sampling with restricted context (TSRC), 52
- Timing, 4
- Total T-trial reward, 28
- Transition function, 59
- Unsupervised learning, 62
- Upper confidence bound strategy, 35, 41
- Value function, 59, 63
- Warm start, 36
- Window TSRC, 52, 145

## Bibliography

- [1] N. Abe, A. W. Biermann, and P. M. Long. Reinforcement learning with immediate rewards and linear hypotheses. *Algorithmica*, 37:263–293, 2003.
- [2] S. Agrawal and N. Goyal. Analysis of thompson sampling for the multi-armed bandit problem. In S. Mannor, N. Srebro, and R. C. Williamson, editors, *Proceedings of the 25th Annual Conference on Learning Theory*, volume 23, pages 39.1–39.26, 2012.
- [3] S. Agrawal and N. Goyal. Further optimal regret bounds for thompson sampling. In C. M. Carvalho and P. Ravikumar, editors, *Proceedings of the 16th International Conference on Artificial Intelligence and Statistics*, volume 31, pages 99–107, 2013.
- [4] S. Agrawal and N. Goyal. Thompson sampling for contextual bandits with linear pay-offs. In S. Dasgupta and D. Mcallester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28, pages 127–135, 2013.
- [5] M. K. Ameko, M. L. Beltzer, L. Cai, M. Boukhechba, B. A. Teachman, and L. E. Barnes. Offline contextual multi-armed bandits for mobile health interventions: A case study on emotion regulation. In *Fourteenth ACM Conference on Recommender Systems*, page 249–258, 2020.
- [6] D. G. Ancona, G. A. Okhuysen, and L. A. Perlow. Taking time to integrate temporal research. *Academy of Management Review*, 26(4):512–529, 2001.
- [7] P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3:397–422, 2002.
- [8] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47:235–256, 2002.
- [9] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32:48–77, 2002.
- [10] A. B. Bakker, E. Demerouti, and A. I. Sanz-Vergel. Burnout and work engagement: The jd-r approach. *Annual Review of Organizational Psychology and Organizational Behavior*, 1(1):389–411, 2014.

- [11] T. Bayes and R. Price. An essay towards solving a problem in the doctrine of chances. by the late rev. mr. bayes, f. r. s. communicated by mr. price, in a letter to john canton, a. m. f. r. s. *Philosophical Transactions (1683-1775)*, 53:370–418, 1763.
- [12] D. Ben-Zeev, C. Brenner, M. Begale, J. Duffecy, D. Mohr, and K. Mueser. Feasibility, acceptability, and preliminary efficacy of a smartphone intervention for schizophrenia. *Schizophrenia Bulletin*, 40(6):1244–1253, 2014.
- [13] D. Ben-Zeev, S. M. Kaiser, C. J. Brenner, M. Begale, J. Duffecy, and D. C. Mohr. Development and usability testing of focus: a smartphone system for self-management of schizophrenia. *Psychiatric rehabilitation journal*, 36(4):289–296, 2013.
- [14] D. Bouneffouf, A. Bouzeghoub, and A. L. Gançarski. A contextual-bandit algorithm for mobile context-aware recommender system. In T. Huang, Z. Zeng, C. Li, and C. S. Leung, editors, *Neural Information Processing*, volume 19, pages 324–331, 2012.
- [15] D. Bouneffouf, I. Rish, and C. Aggarwal. Survey on applications of multi-armed and contextual bandits. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, 2020.
- [16] D. Bouneffouf, I. Rish, G. Cecchi, and R. Féraud. Context attentive bandits: Contextual bandit with restricted context. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 1468–1475, 2017.
- [17] J. Brownlee. A gentle introduction to bayes theorem for machine learning. <https://machinelearningmastery.com/bayes-theorem-for-machine-learning/>. accessed on 20201-08-30.
- [18] O. Chapelle and L. Li. An empirical evaluation of thompson sampling. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, page 2249–2257, 2011.
- [19] W. Chen, Y. Wang, and Y. Yuan. Combinatorial multi-armed bandit: General framework and applications. In S. Dasgupta and D. McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28, pages 151–159, 2013.
- [20] S. Choudhuri. Additional bioinformatic analyses involving nucleic-acid sequences. In S. Choudhuri, editor, *Bioinformatics for Beginners*, pages 157–181. 2014.
- [21] W. Chu, L. Li, L. Reyzin, and R. Schapire. Contextual bandits with linear payoff functions. In G. Gordon, D. Dunson, and M. Dudík, editors, *Proceedings of the 14th*

- International Conference on Artificial Intelligence and Statistics*, volume 15, pages 208–214, 2011.
- [22] L. Collins, I. Nahum-Shani, and D. Almirall. Optimization of behavioral dynamic treatment regimens based on the sequential, multiple assignment, randomized trial (smart). *Clinical Trials*, 11(4):426–434, 2014.
- [23] E. Crawford, J. Lepine, and B. Rich. Linking job demands and resources to employee engagement and burnout: A theoretical extension and meta-analytic test. *The Journal of Applied Psychology*, 95:834–848, 2010.
- [24] K. Ding, J. Li, and H. Liu. Interactive anomaly detection on attributed networks. In *Proceedings of the 12th ACM International Conference on Web Search and Data Mining*, pages 357–365, 2019.
- [25] A. Durand and C. Gagné. Thompson sampling for combinatorial bandits and its application to online feature selection. In *Proceedings of the 28th AAAI Conference Workshop on Sequential Decision-Making with Big Data*, 2014.
- [26] B. Fogg. Website of the fogg behavior model, by dr. bj fogg from stanford university. <https://behaviormodel.org/>. accessed on 2021-08-26.
- [27] B. Fogg. A behavior model for persuasive design. In *Proceedings of the 4th International Conference on Persuasive Technology*, number 40, 2009.
- [28] E. M. Forman, S. G. Kerrigan, M. L. Butryn, A. S. Juarascio, S. M. Manasse, S. Ontañón, D. H. Dallal, R. J. Crochiere, and D. Moskow. Can the artificial intelligence technique of reinforcement learning use continuously-monitored digital data to optimize treatment for weight loss? *Journal of Behavioral Medicine*, 42(2):276–290, 2019.
- [29] A. Goldenshluger and A. Zeevi. A linear response bandit problem. *Stochastic Systems*, 3(1):230–261, 2013.
- [30] R. Z. Goldstein, A. Craig, A. Bechara, H. Garavan, A. R. Childress, M. P. Paulus, and N. D. Volkow. The neurocircuitry of impaired insight in drug addiction. *Trends in Cognitive Sciences*, 13(9):372–380, 2009.
- [31] S. Gonul, T. Namli, S. Huisman, G. B. Laleci Erturkmen, I. H. Toroslu, and A. Cosar. An expandable approach for design and personalization of digital, just-in-time adaptive interventions. *Journal of the American Medical Informatics Association*, 26(3):198–210, 2018.



- [32] A. Gopalan and S. Mannor. Thompson sampling for learning parameterized markov decision processes. In P. Grünwald, E. Hazan, and S. Kale, editors, *Proceedings of The 28th Conference on Learning Theory*, volume 40, pages 861–898, 2015.
- [33] D. H. Gustafson, F. M. McTavish, M.-Y. Chih, A. K. Atwood, R. A. Johnson, M. G. Boyle, M. S. Levy, H. Driscoll, S. M. Chisholm, L. Dillenburg, A. Isham, and D. Shah. A smartphone application to support recovery from alcoholism: A randomized clinical trial. *JAMA Psychiatry*, 71(5):566–572, 2014.
- [34] D. H. Gustafson, B. R. Shaw, A. Isham, T. Baker, M. G. Boyle, and M. Levy. Explicating an evidence-based, theoretically informed, mobile technology-based system to improve outcomes for people in recovery for alcohol dependence. *Substance Use & Misuse*, 46(1):96–111, 2011.
- [35] A. Hallak, D. Di Castro, and S. Mannor. Contextual markov decision processes. *ArXiv*, 2015. ArXiv ID: 1502.02259v1.
- [36] W. Hardeman, J. Houghton, K. Lane, A. Jones, and F. Naughton. A systematic review of just-in-time adaptive interventions (jitais) to promote physical activity. *International Journal of Behavioral Nutrition and Physical Activity*, 16:31, 2019.
- [37] M. Haugh. Monte-carlo simulation - output analysis for monte-carlo. [http://www.columbia.edu/~mh2078/MonteCarlo/MCS\\_Output\\_Analysis\\_MasterSlides.pdf](http://www.columbia.edu/~mh2078/MonteCarlo/MCS_Output_Analysis_MasterSlides.pdf). Presentation slides, accessed on 2021-10-21.
- [38] Google play store - heartsteps. [https://play.google.com/store/apps/details?id=net.heartsteps.kpw&hl=de\\_AT&gl=US](https://play.google.com/store/apps/details?id=net.heartsteps.kpw&hl=de_AT&gl=US). accessed on 2021-08-20.
- [39] B. W. Heckman, A. R. Mathew, and M. J. Carpenter. Treatment burden and treatment fatigue as barriers to health. *Current Opinion in Psychology*, 5:31–36, 2015.
- [40] B. Hoffmann and D. Ritchie. Using multimedia to overcome the problems with problem based learning. *Instructional Science*, 25:97–115, 1997.
- [41] E. Jochems, C. Mulder, A. van Dam, H. Duivenvoorden, S. Scheffer, W. Spek, and C. Van der Feltz-Cornelis. Motivation and treatment engagement intervention trial (motivate-it): The effects of motivation feedback to clinicians on treatment engagement in patients with severe mental illness. *BMC Psychiatry*, 12:209, 2012.
- [42] L. P. Kaelbling. Associative reinforcement learning: A generate and test algorithm. *Machine Learning*, 15(3):299–319, 1994.

- [43] E. Kaufmann, O. Cappe, and A. Garivier. On bayesian upper confidence bounds for bandit problems. In N. D. Lawrence and M. Girolami, editors, *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics*, volume 22, pages 592–600, 2012.
- [44] A. C. King, D. K. Ahn, B. M. Oliveira, A. A. Atienza, C. M. Castro, and C. D. Gardner. Promoting physical activity through hand-held computer technology. *American journal of preventive medicine*, 34(2):138–142, 2008.
- [45] G. King, M. Currie, and P. Petersen. Child and parent engagement in the mental health intervention process: A motivational framework. *Child and Adolescent Mental Health*, 19(1):2–8, 2014.
- [46] P. Klasnja, E. B. Helker, S. Shiffman, A. Boruvka, D. Almirall, A. Tewari, and S. A. Murphy. Micro-randomized trials: An experimental design for developing just-in-time adaptive interventions. *Health Psychology*, 34(0):1220–1228, 2015.
- [47] S. Kumar, W. J. Nilsen, A. Abernethy, A. Atienza, K. Patrick, M. Pavel, W. T. Riley, A. Shar, B. Spring, D. Spruijt-Metz, D. Hedeker, V. Honavar, R. Kravitz, R. C. Lefebvre, D. C. Mohr, S. A. Murphy, C. Quinn, V. Shusterman, and D. Swendeman. Mobile health technology evaluation: The mhealth evidence workshop. *American Journal of Preventive Medicine*, 45(2):228–236, 2013.
- [48] J. Langford and T. Zhang. The epoch-greedy algorithm for contextual multi-armed bandits. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, page 817–824, 2007.
- [49] P. W. Lavori and R. Dawson. Introduction to dynamic treatment strategies and sequential multiple assignment randomization. *Clinical Trials*, 11(4):393–399, 2014.
- [50] H. Lei, A. Tewari, and S. A. Murphy. An actor-critic contextual bandit algorithm for personalized mobile health interventions. e-Print arXiv:1706.09090, 2017. published on arXiv.org, statML.
- [51] L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, page 661–670, 2010.
- [52] P. Liao, K. Greenewald, P. Klasnja, and S. Murphy. Personalized heartsteps: A reinforcement learning algorithm for optimizing physical activity. In *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, volume 4, pages 1–22, 2020.

- [53] Y. Lin, J. Jessurun, B. de Vries, and H. Timmermans. Motivate: towards context-aware recommendation mobile system for healthy living. In *5th International ICST Conference on Pervasive Computing Technologies for Healthcare*, 2012.
- [54] F. Mael and S. Jex. Workplace boredom: An integrative model of traditional and contemporary approaches. *Group & Organization Management*, 40(2):131–159, 2015.
- [55] B. C. May, N. Korda, A. Lee, and D. S. Leslie. Optimistic bayesian sampling in contextual-bandit problems. *Journal of Machine Learning Research*, 13(67):2069–2106, 2012.
- [56] K. T. Mueser, P. S. Meyer, D. L. Penn, R. Clancy, D. M. Clancy, and M. P. Salyers. The illness management and recovery program: Rationale, development, and preliminary findings. *Schizophrenia Bulletin*, 32(Suppl 1):S32–S43, 2006.
- [57] I. Nahum-Shani, E. B. Hekler, and D. Sprujitz-Metz. Building health behavior models to guide the development of just-in-time adaptive interventions: A pragmatic framework. *Health Psychology*, 34(0):1209–1219, 2015.
- [58] I. Nahum-Shani, S. Smith, B. Spring, L. Collins, K. Witkiewitz, A. Tewari, and S. Murphy. Just-in-time adaptive interventions (jitais) in mobile health: Key components and design principles for ongoing health behavior support. *Annals of Behavioral Medicine*, 52(6):446–462, 2018.
- [59] J. Nemeč, C. Gruber, B. Chimani, and I. Auer. Trends in extreme temperature indices in austria based on a new homogenised dataset. *International Journal of Climatology*, 33(6):1538–1550, 2013.
- [60] Y. Ouyang, M. Gagrani, A. Nayyar, and R. Jain. Learning unknown markov decision processes: A thompson sampling approach. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, page 1333–1342, 2017.
- [61] P. Paredes, R. Gilad-Bachrach, M. Czerwinski, A. Roseway, K. Rowan, and J. Hernandez. Poptherapy: Coping with stress through pop-culture. In *Proceedings of the 8th International Conference on Pervasive Computing Technologies for Healthcare*, page 109–117, 2014.
- [62] S. Pauker, E. Zane, and D. Salem. Creating a safer health care system: finding the constraint. *JAMA*, 294:2906–2908, 2005.
- [63] K. J. Preacher and A. F. Hayes. Asymptotic and resampling strategies for assessing and comparing indirect effects in multiple mediator models. *Behavior Research Methods*, 40(3):879–891, 2008.

- [64] L. Quintens and P. Matthyssens. Involving the process dimensions of time in case-based research. *Industrial Marketing Management*, 39(1):91–99, 2010. Case Study Research in Industrial Marketing.
- [65] M. Rabbi. Personal website of mashfiqui rabbi, phd student at cornell university. <https://www.cs.cornell.edu/~ms2749/#>. accessed on 2021-08-20.
- [66] M. Rabbi, M. H. Aung, M. Zhang, and T. Choudhury. Mybehavior: automatic personalized health feedback from user behaviors and preferences using smartphones. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, page 707–718, 2015.
- [67] M. Rabbi, P. Klasnja, T. Choudhury, A. Tewari, and S. Murphy. Optimizing mhealth interventions with a bandit. In *Digital Phenotyping and Mobile Sensing: New Developments in Psychoinformatics*, pages 277–291, 2019.
- [68] V. Rajanna, R. Lara-Garduno, D. J. Behera, K. Madanagopal, D. Goldberg, and T. Hammond. Step up life: A context aware health assistant. In *Proceedings of the Third ACM SIGSPATIAL International Workshop on the Use of GIS in Public Health*, page 21–30, 2014.
- [69] W. T. Riley. Theoretical models to inform technology-based health behavior interventions. In L. Marsch, S. Lord, and J. Dallery, editors, *Behavioral Health Care and Technology Using Science-Based Innovations to Transform Practice*, pages 13–23, 2014.
- [70] J. Rocca and B. Rocca. The exploration-exploitation trade-off: intuitions and strategies. <https://towardsdatascience.com/the-exploration-exploitation-dilemma-f5622fbc1e82>. accessed on 2021-07-16.
- [71] R. M. Ryan and E. L. Deci. Self-regulation and the problem of human autonomy: does psychology need choice, self-determination, and will? *Journal of Personality*, 74(6):1557–1585, 2006.
- [72] J. Sarkar. One-armed bandit problems with covariates. *The Annals of Statistics*, 19(4):1978–2002, 1991.
- [73] S. C. Segerstrom and D. B. O’Connor. Stress, health and illness: four challenges for the future. *Psychology & Health*, 27(2):128–140, 2012.
- [74] S. Shiffman and A. A. Stone. Ecological momentary assessment: A new tool for behavioral medicine research. In *Technology and Methods in Behavioral Medicine*, pages 117–131, 1998.

- [75] D. Spruijt-Metz, C. K. F. Wen, B. M. Bell, S. Intille, J. S. Huang, and T. Baranowski. Advances and controversies in diet and physical activity measurement in youth. *American Journal of Preventive Medicine*, 55(4):e81–e91, 2018.
- [76] A. L. Strehl, C. Mesterharm, M. L. Littman, and H. Hirsh. Experience-efficient learning in associative bandit problems. In *Proceedings of the 23rd International Conference on Machine Learning*, page 889–896, 2006.
- [77] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. 2017. Second edition.
- [78] C. Tekin, O. Atan, and M. Van Der Schaar. Discover the expert: Context-adaptive expert selection for medical diagnosis. *IEEE Transactions on Emerging Topics in Computing*, 3(2):220–234, 2015.
- [79] A. Tewari and S. A. Murphy. From ads to interventions: Contextual bandits in mobile health. In *Mobile Health: Sensors, Analytic Methods, and Applications*, pages 495–517, 2017.
- [80] W. R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3-4):285–294, 1933.
- [81] P. Turnbull, N. Oliver, and B. Wilkinson. Buyer-supplier relations in the uk - automotive industry: Strategic implications of the japanese manufacturing model. *Southern Medical Journal*, 13:159–168, 1992.
- [82] S. Van Dantzig, G. Geleijnse, and A. Halteren. Towards a persuasive mobile application to reduce sedentary behavior. *Personal and Ubiquitous Computing*, 17(6):1237 – 1246, 2011.
- [83] M. van Otterlo and M. Wiering. Reinforcement learning and markov decision processes. In *Reinforcement Learning: State-of-the-Art*, pages 3–42, 2012.
- [84] S. S. Villar, J. Bowden, and J. Wason. Multi-armed bandit models for the optimal design of clinical trials: Benefits and challenges. *Statistical Science*, 30(2):199–215, 2015.
- [85] C.-C. Wang, S. R. Kulkarni, and H. V. Poor. Bandit problems with side observations. *IEEE Transactions on Automatic Control*, 50(3):338–355, 2005.
- [86] S. J. Wenze and I. W. Miller. Use of ecological momentary assessment in mood disorders research. *Clinical Psychology Review*, 30(6):794–804, 2010.

- [87] WHO. Cardiovascular diseases fact sheet. [https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-\(cvds\)](https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds)). accessed on 2021-07-19.
- [88] WHO. Global strategy on digital health 2020-2025. Licence: CC BY-NC-SA 3.0 IGO.
- [89] WHO. mhealth: New horizons for health through mobile technologies: second global survey on ehealth. [https://www.who.int/goe/publications/goe\\_mhealth\\_web.pdf](https://www.who.int/goe/publications/goe_mhealth_web.pdf). accessed on 2021-07-19.
- [90] WHO. Obesity and overweight fact sheet. <https://www.who.int/news-room/fact-sheets/detail/obesity-and-overweight>. accessed on 2021-07-19.
- [91] M. Woodroffe. A one-armed bandit problem with a concomitant variable. *Journal of the American Statistical Association*, 74(368):799–806, 1979.
- [92] E. Yom-Tov, G. Feraru, M. Kozdoba, S. Mannor, M. Tennenholtz, and I. Hochberg. Encouraging physical activity in patients with diabetes: Intervention using a reinforcement learning system. *Journal of Medical Internet Research*, 19(10):e338, 2017.
- [93] M. Zhou, Y. Mintz, Y. Fukuoka, K. Goldberg, E. Flowers, P. Kaminsky, A. Castillejo, and A. Aswani. Personalizing mobile fitness apps using reinforcement learning. In A. Said and T. Komatsu, editors, *Joint Proceedings of the ACM IUI 2018 Workshops*, volume 2068, 2018.