IT UNIVERSITY OF COPENHAGEN

IT University of Copenhagen
Digital Design

Tactile Games

Operationalising Difficulty in Puzzle Games

Jeppe Theiss Kristensen

July 30th, 2022

A Dissertation submitted in partial fulfilment
of the requirements for the degree of
Doctor of Philosophy

Declaration

IT UNIVERSITY OF COPENHAGEN

Jeppe Theiss Kristensen

Digital Design
Direct phone: +45 6018 3057
E-mail: jetk@itu.dk

July 30, 2022

Declaration for submission of PhD thesis

I, Jeppe Theiss Kristensen, declare that this thesis - submitted in partial fulfilment of the requirements for the conferral of PhD, from the IT University of Copenhagen - is solely my own work unless otherwise referenced or attributed. Neither the thesis nor its content have been submitted (or published) for qualifications at another academic institution.

Yours sincerely

Jeppe Theiss Kristensen [illegible]

Abstract (Dansk)

Denne erhvervs-PhD har hovedsageligt beskæftiget sig med at bestemme og modellere sværhedsgraden i spil, og specifikt puzzle-spil til mobil (*mobile puzzle games*). Sværhedsgraden spiller en afgørende rolle for spillerens engagementet i sådanne spil, så det at opnå en dybere forståelse og være i stand til at forudsige den opfattede sværhedsgrad på spiller-niveau er vigtige mål ikke kun for forskere men også for den overordnede industri.

I begyndelsen fokuserede projektet på at skabe en *playtesting agent* der automatisk kan spille nyt indhold og nye baner igennem i et kommercielt mobile puzzle games. Til det formål udviklede vi et *reinforcement learning* system der kunne fungere inden for bestemte tekniske begrænsninger såsom ingen mulighed for at bruge optagne spillespor eller tree-search metoder. Selv om agenten ikke var i stand til at opnå menneskelignende egenskaber på alle baner, så var et forskningsbidrag at de bedste 10% af agentens forsøg var stærkt korreleret med spillerdata, samt hvordan agenten kan trænes på en hurtig og robust måde.

Idet agenten ikke var tilstrækkelig i sig selv til at forudsige sværhedsgraden af nye baner, startede vi med at addressere spørgsmål vedrørende hvordan det er muligt at forbinde spiller- og agent-adfærd. Det første forskningsområde var mere fokuseret på at svare på spørgsmålet om, hvad sværhedsgraden egentlig er i mobile puzzle games og hvad det indebærer at forudsige sværhedsgraden af baner en spiller endnu ikke er stødt på. Fra dette projekt var der to forskningsbidrag: for det første foreslog vi en parametrisk fordeling til at modellere antallet af skridt spillerne har brug for til at klare en bane, hvilket kan bruges til at automatisk justere sværhedsgraden. For det andet foreslog vi, hvordan *Factorization Machines* (FM) kan bruges til at forudsige antallet af forsøg hver enkelt spiller forventes at bruge på banerne, samt hvordan spillerevner og grundlæggende sværhedsgrad af baner kan beskrives via de latente faktorer i modellen.

Det sidste forskningsfokus var at binde det hele sammen – hvordan kan data fra playtesting agenten bruges sammen med personaliserede forudsigelser til at forudsige sværhedsgraden på ikke bare eksisterende men også nyt, originalt indhold? Resultaterne viste at brug af data fra agenten er nyttigt til at forudsige adfærden på nyt indhold og forbedre forudsigelserne. Selv om FM metoden er nyttig til personaliserede forudsigelser på eksisterende indhold med spillerdata, så fungerer ikke-personaliserede forudsigelser via neurale network bedre på nyt indhold. Der er derfor ikke én tilgang som virker bedst til alle use cases, men i alle use cases er præcisionen god nok til at bruge i en kommerciel kontekst.

Abstract

The main line of investigation of this industrial PhD has been determining and modelling difficulty in games, specifically in mobile puzzle games. Difficulty plays a crucial role in player engagement in such games, so gaining a deeper understanding and being able to predict the perceived difficulty on a player level are important goals not only for researchers but for the industry at large.

The initial work focused on creating a playtesting agent that would be able to automatically play through new content in a commercial puzzle game. For this purpose, we developed a reinforcement learning setup that could operate within a number of technical constraints, such as no possibility of using player play traces or tree-search. While the agent did not reach human-level performance on the full-scale problem, a key finding was that the top $\sim$10% performances of the agent on a level were strongly correlated with player data. Additionally, we proposed ways to train the playtesting agent in a quick and robust way.

With the agent not being enough by itself to predict the difficulty of new levels, we started to address the question of how to link agent behaviour to player behaviour. The first line of research was more focused on answering the question of what difficulty actually is in puzzle games and what it entails to predict difficulty for *any* content a player has not yet encountered. There were two main findings from this work: first, we proposed a parametric distribution for modelling the number of actions players spend for completing a level, paving the way for dynamic difficulty adjustment, and second, how individual difficulty predictions for the players are possible using factorization machines by capturing player skill and intrinsic level difficulty with latent factors.

In the last line of research, the objective was to tie it all together – how can agent behaviour data be used together with personalised predictions for estimating the difficulty of not just old content but also new, novel content? The results showed that agent data does indeed have high predictive power on new content and can improve personalised predictions. While the factorization machine approach is useful for personalised predictions on old content, for predictions on new content, non-personalised predictions using a standard artificial neural network worked better. There is therefore not one approach that works the best in all use-cases, but in each of the use-cases, the accuracy of the methods is high enough for being used in a commercial context.

Acknowledgements

First of all, I would like to thank my university supervisor, Paolo Burelli. You have helped me navigate both academia and industry, and the trust you have put in me, whether it has been regarding deciding on research directions, organising conferences or other pursuits, has been a large motivation for me and a huge factor in where I am today. I hope we get to collaborate in the future.

I would also like to thank my company supervisor, Arturo Valdivia. I thank you for the interesting discussions we have had, and your ability to transform the data science department at Tactile Games has given me a great perspective on the industry.

I would also like to thank the two previous company supervisors, Christophe Carvenius and Morten Nielsen. Although Christophe left early on in the project, I want to thank you for your role in getting this project up and running. Even if Morten was a temporary stand-in, I want to thank you for the professionalism, efficiency and patience that you have had regarding this project. You really helped (re)kick-start the research, especially with the Game AI trip to New York.

A special thanks goes to the Innovation Fund Denmark and Tactile Games for funding the PhD. I feel that the collaboration has been mutually beneficial for all parties and helps bridge the gap between academia and industry. I appreciate the last-month assistance from Tactile Games.

A benefit of being at both a university and a company is that you have double the amount of colleagues. I therefore want to thank all the colleagues at Tactile Games, from the other data scientists and the core team to the level designers and game developers. In particular, I want to thank Vince for always being up for pushing our understanding, Celia for assistance with DS tasks, and Lasse, Anibal, and LP for technical assistance with Unity. From ITU, I want to thank Rasmus for your quick proofs-of-concept on my problems that I hope to imitate one day, González-Duque for your Bayesian motivation, Sebastian for your assistance early on, and the rest of the REAL lab for the memorable experiences. I also want to thank the rest of Digital Design for the great discussions and retreats to Kildegaard.

Despite COVID-19, I managed to go for a short research stay abroad at Aalto University in Finland. I want to give a big thanks to Perttu Hämäläinen and Christian Guckelsberger for not just your hospitality but also your intellectual contributions to my work. My PhD would not have been the same without our collaboration. I also want to thank the rest of the people at Aalto for the social events that have been sorely missed during these last few years. And the Thursday pancakes.

I want to thank my family and friends for all the support and offers of help. I especially appreciate the hospitality of my partner's family in this last sprint of the PhD in Rhodes. Your help and concern have been really helpful. ευχαριστώ πολύ!

Lastly, I want to thank my partner, Arxodia. Your patience and support are what have gotten me through this project, and I cannot express my appreciation with words.

Contents

List of Figures

List of Tables

Nomenclature

A2C	Advantage Actor-Critic
A3C	Asynchronous Advantage Actor-Critic
AFM	Additive Factors model
ALE	Arcade Learning Environment
CMA-ES	Covariance Matrix Adaptation Evolution Strategy
CoG	Conference on Games
ConvNN	Convolutional neural network
DDA	Dynamic difficulty adjustment
DP	Dynamic programming
DQN	Deep Q-Network
ES	Evolution strategy
FM	Factorisation Machines
GAIL	Generative Adversarial Imitation Learning
HCI	Human-computer interaction
HMM	Hidden Markov model
HyperNEAT	Hyper NeuroEvolution of Augmenting Topologies
IRL	Inverse reinforcement learning
KC	Knowledge Component
KL	Kullback-Leiber
MC	Monte Carlo
MCTS	Monte Carlo Tree Search
MDP	Markov decision process
ML	Machine learning
NEAT	NeuroEvolution of Augmenting Topologies
NN	Neural network
PCG	Procedural content generation
PG	Policy Gradient
POMDP	Partially observable Markov decision process
PPO	Proximal Policy Optimization
RF	Random Forest
RL	Reinforcement learning
RNN	Recurrent neural network
RPG	Role-playing game
SAC	Soft Actor-Critic
TD	Temporal difference
ToG	Transactions on Games
TRPO	Trust Region Policy Optimization
UCB	Upper confidence bounds
UCT	Upper confidence bounds to trees
UIST	User Interface Software and Technology

a	Action
A	Advantage
D_{KL}	Kullback-Leiber divergence
Q	State-action value function
R	Discounted return
S	State
V	Value function
π	Policy

Part I

Prologue

1 Introduction

When somebody calls something *difficult* – whether it is a game or something else – most of us will intuitively understand what the person means. However, for researchers and game developers alike, this is vague. In order to be able to work with the concept, it is necessary to *operationalise* difficulty. That is, it is necessary to define what difficulty is so it can be measured and put into use. In this dissertation, the specific focus is on *puzzles games*. Together with an industrial partner, we explore some of the challenges of operationalising difficulty in their games and propose ways to better understand how to model and utilise difficulty and automatise parts of the game design process.

1.1 Motivation

A question that game developers frequently ask themselves is whether their game has the optimal difficulty or not – and with good reason. How difficult a player finds a game is strongly linked with the player experience and how enjoyable they find it (Vorderer et al., 2003), which is, by and large, the main concern for game developers. Therefore, operationalising difficulty is a key focus for not just game developers but also researchers to ensure an optimal player experience.

The player experience can be understood through the concept of *flow*, which is described by Csikszentmihalyi (1990). It describes a state of concentration and deep enjoyment where the person feels immersed in what they are doing and loses track of time. This optimal experience is valid for other domains than games, but common to each field is the requirement that the

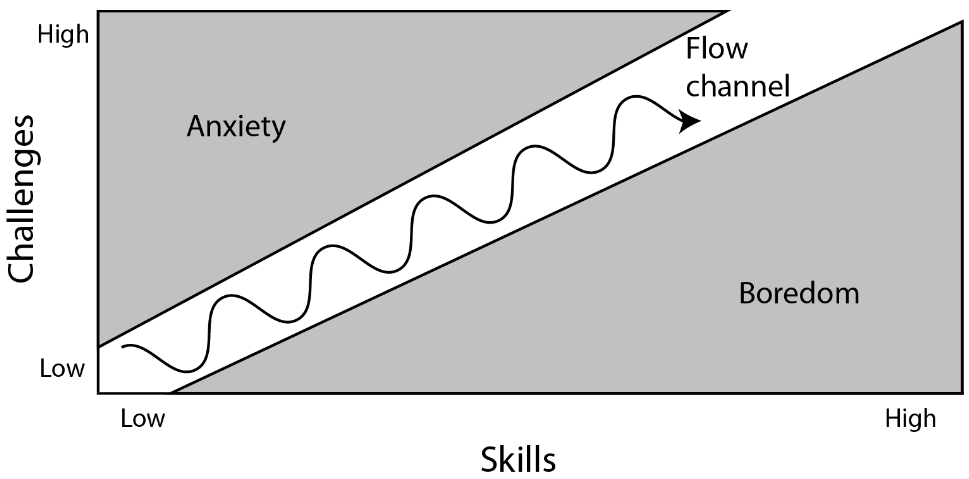


Figure 1.1: *Flow* model. Adapted from Schell (2019).

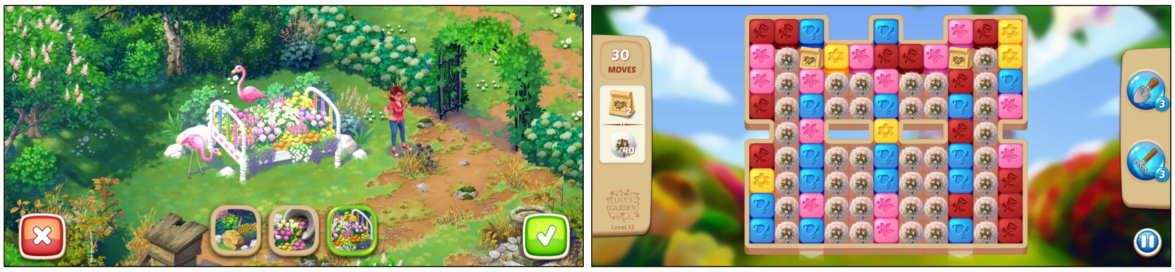


Figure 1.2: Screenshots from the game Lily's Garden by Tactile Games – a typical game in the mobile puzzle genre.

presented challenge matches the person's competence. It is often illustrated as a 2-dimensional plot, as shown in Fig. 1.1 where, in order to reach the *flow* state, the player experience should be in the flow channel. If the game balance is off by either being too easy or hard, players can become bored or frustrated and ultimately quit the game.

From this viewpoint, there is not a simple way to operationalise difficulty since it very much depends on both the player but also what kind of challenge. As Denisova et al. (2020) outline, there are multiple facets to *challenge* and what affects the players' perception of challenge. For any difficulty-modelling framework, the ability to account for differences between the individual players is, therefore, a highly desirable but also challenging goal.

These considerations about difficulty also have clear industrial applications. In the fiercely competitive mobile game industry, which makes up more than half the global \$175B video game industry[1], a particularly important area of focus is user retention. A widely adopted business model in this market is a free-to-play model which relies on users not just installing the game but also spending time and becoming invested in the game in order for the in-game monetisation schemes to be effective. For game developers, this means constantly balancing the currently available content and introducing new content to keep the game experience optimal and exciting.

This is also the case for the industrial partner of this dissertation, Tactile Games. The typical release schedule of content for one of their games (e.g. Lily's Garden, Fig. 1.2) consists of around 50 new puzzle levels on a bi-weekly basis, each requiring multiple manual playthroughs and iterations by the level designers. This is just for one out of several live games and not even counting the continuous quality check process of old content that is a large part of the day-to-day work of the level designers. Being able to measure, estimate, predict and optimise the difficulty of all content is, therefore, a crucial tool in the toolbox for Tactile Games and other game companies for enabling informed decisions that can engage new players and *keep* them engaged.

While most gaming companies operationalise difficulty one way or another, the process is often characterised by being reactive, where the decision-makers await the result from analyses of previously collected data. However, a more proactive approach where issues can be identified early on has the potential to improve the player experience significantly and also improve the workflow of game designers in the long run. This is the starting point of this dissertation. The

[1]https://newzoo.com/insights/articles/global-games-market-to-generate-175-8-billion-in-2021-despite-a-slight-decline-the-market-is-on-track-to-surpass-200-billion-in-2023/

overarching theme is how difficulty can be operationalised in realistic scenarios so that game designers can proactively adjust the game content for the optimal player experience. The results of this dissertation are validated through experiments using data from the industrial partner, which ultimately allows the results to generalise to other games and bridge the gap between research and industry.

1.2 Problem definition and applications

An essential focus of the dissertation is to consider plausible scenarios in a typical game company in which operationalising difficulty is useful and what tools that are available for that. From interviews with game and level designers at Tactile Games, we have identified several scenarios in the game design process where being able to operationalise and model difficulty has the potential to improve their workflow and gain a deeper understanding of player behaviour.

One such scenario is when the level designers need to maintain existing content. An essential commercial focus for games is to grow their player base, and as new players are acquired and start playing the game, the existing content may be outdated in several ways. Since the players typically need to complete the levels linearly in order to progress to the next level, the game designers may want to optimise and change the difficulty to let new players catch up. Another possibility is that the new players come from similar games and tend to be more skilled than the average player and thus need harder content to feel challenged. These considerations also extend to players who have been playing the game for some time but still have plenty of content to consume. In addition, players may be served the previous content out of the typical order, such as through tournaments or in-game challenges. Identifying how difficult players perceive any content is therefore crucial for a more fair and balanced experience.

Being able to serve these personalised experiences to the players ultimately relies on being model difficulty on the existing content and estimating how modifying the content also changes the individual experiences. Therefore, the research related to this scenario focuses on leveraging the extensive amount of historical data available about the previous players and their performance on said content for modelling and operationalising difficulty.

Another workflow that can be improved is when level designers create new content and the difficulty needs to be evaluated. As mentioned previously, the level designers typically create new levels continuously that need to be playtested to determine both the difficulty and whether it is an engaging level. Similarly, other games sometimes employ procedural content generation methods where the newly created content also needs to be evaluated but in an even more automated fashion. The challenge in this scenario is that there is very little or no player data available. While this is commonly overcome by having the designers make use of playtesting scripts or evaluate the content themselves, it does not scale very easily. The time it takes to evaluate the content is proportional to the amount of content, and with multiple supported games, this can quickly add up. Additionally, new mechanics (or different mechanics between games) can quickly render the previous playtesting scripts unusable and require non-trivial changes to capture the additional complexity, which can just as easily lead to unexpected behaviours.

The research related to this scenario of being able to evaluate new content aims toward creating a way to learn and playtest content automatically. A key challenge of this research is that it should not only be able to automatically adjust to any game or new content but also allow the level designers to infer the perceived difficulty of the players that encounter the content.

1.3 Research questions

The discussion above outlines two scenarios where operationalising difficulty is helpful: Modelling difficulty on existing content so that level designers can more easily adjust the difficulty depending on the level and players, and estimating the difficulty on new content before it is released to the players. The main research questions of this dissertation relate to how we can resolve the challenges in the two scenarios and are as follows:

1. How can data from players be used to operationalise and model difficulty?
2. How can difficulty be measured and predicted on content with little to no player data?

In order to answer these questions, we conducted the research by utilising both experimental and model methodologies and focusing on defining and modelling difficulty using data from the industrial partner for validating the methods. This line of research is rooted in the player modelling and human-computer interaction (HCI) domains and contributes with methods and knowledge for player modelling in games. A brief overview of this research will be given in Section 1.4.

Another component of the research in this thesis consists of building automated playtesting methods that are possible to apply in an industrial context. This research contributes with practical knowledge to the reinforcement learning field and demonstrates through experiments possible avenues for bridging the gap between research and industry. How this relates to this PhD will be introduced in Section 1.5.

Through these research topics, the PhD project contributes to increasing the knowledge about automated playtesting and provides a complete framework for evaluating game content in a live game, from the initial conception of playtesting to more practical implementations of state-of-the-art methods. All the results are validated against data from a live commercial game and demonstrate proof-of-concept methods that can scale to both large and small games. The dissertation also highlights limitations in current approaches and offers solutions to said problems, providing a comprehensive framework for playtesting in puzzle games.

The thesis is article-based and includes five papers authored during the PhD. A list of the included papers is shown in Table 1.1. These papers comprise the main body of papers that are concerned with estimating difficulty on puzzle levels, with two relating to building a playtesting agent and the remaining three relating to modelling players and difficulty.

We exclude two other papers that have been a part of the PhD, with one as the main author and the other one as a co-author. The first paper (Kristensen and Burelli, 2019) focuses on churn in games, and while churn and difficulty are linked (e.g., Roohi et al. (2021)), the study mainly focuses on methods for churn prediction and does not contain any direct link with difficulty.

Article	#	Section	**Citation**	**Status**
Statistical Modelling of Level Difficulty in Puzzle Games	1	3.3	Kristensen et al. (2021)	Conference paper, published in CoG 2021
Personalised Game Difficulty Prediction Using Factorisation Machines	2	3.4	Kristensen et al. (2022)	Conference paper, published in UIST 2022
Strategies for Using Proximal Policy Optimization in Mobile Puzzle Games	3	4.2	Kristensen and Burelli (2020)	Conference paper, published in FDG 2020
Estimating Player Completion Rate in Mobile Puzzle Games Using Reinforcement Learning	4	4.3	Kristensen et al. (2020)	Short paper, published in CoG 2020
Difficulty Modelling in Puzzle Games	5	5.2	Kristensen and Burelli (2022)	Journal paper, under review in ToG

Table 1.1: Included papers in the main body of the dissertation. Listed in the order of appearance in the dissertation. CoG: *Conference on Games.* FDG: *Foundations of Digital Games.* ToG: *Transactions on Games.* UIST: *User Interface Software and Technology.*

The second paper (Hald et al., 2020) focuses on methods for procedural content generation (PCG) in puzzle games. While that also has its uses for developing playtesting agents (e.g., Justesen et al. (2018); Risi and Togelius (2020)), it has not been used in that context in this dissertation.

1.4 Difficulty in puzzle games

How difficulty is defined and operationalised in games depends on the context. For example, in adversarial games such as chess or first-person shooter games, the fairest game balance would be if there was an equal 50% of winning, all things being equal. In many single-player games, regardless of genre, the player often has the option of adjusting the difficulty themselves, affecting the speed or likelihood of completing the tasks.

In puzzle games, particularly commercial mobile puzzle games, the game developers often pre-define the difficulty to allow for better monetisation through in-game purchases or ads. The difficulty is most often defined by the number of attempts, or inversely the pass rate, that players spend to complete a level (e.g. in games by Tactile Games, Candy Crush + variations (Gudmundsson et al., 2018; Poromaa, 2017), Angry Birds + variations (Roohi et al., 2021) and multiple Electronic Arts games (Xue et al., 2017)).

In order to get a better understanding of how the difficulty is affected by specific gameplay parameters, Paper 1 (Kristensen et al., 2021) examines how the number of actions, or moves, taken by the players for completing a puzzle level can be modelled. It is found that using a negative binomial distribution to describe the move distribution is possible for 85% of the levels. From this statistical description of the levels, it is possible to estimate how changing the move limit affects the probability of completing the level. This opens the door to dynamically adjusting the difficulty, which has been proposed as a way to increase player engagement (Hunicke, 2005). A modified approach has also been tested out in practice in one of Tactile Games' games, which will be discussed more in detail in Section 3.1.

A limitation of the previous work was that it does not account for individual differences. The next step is, therefore, to investigate the feasibility of predicting how an individual player perceives the difficulty, which is examined in Paper 2 (Kristensen et al., 2022). By using a matrix factorisation method called *Factorization Machines* (FM) (Rendle, 2012), which has been used in recommendation settings and other sparse user-item data contexts, we demonstrate the how these personalised predictions not only allow better differentiation between players but also provide an interpretable model that level designers can use for other projects than difficulty adjustment. An extension to this work where we included playtesting data was tested out in Paper 5 (Kristensen and Burelli, 2022) and will be discussed in the next section.

1.5 Automated playtesting

Playtesting game content is an important step in the design process when creating new content. It is during this time the designers are able to tweak game parameters to ensure the content is not just playable but also enjoyable for the players. With the complexity of games today, both in terms of technical implementations and mechanics, it can be a very time-consuming process to playtest every nook and cranny, so the industry has started to automate this process to a larger and larger extent.

Developing methods for automated playtesting faces two questions: *what* needs to be playtested, and *how* can this be achieved. For example, evolutionary methods have been used to test out the viability of different card combinations in the collectable card game *Hearthstone* (García-Sánchez et al., 2018). Various approaches often also employ AI methods, such as reinforcement learning (RL) agents for finding inaccessible areas on a map (Bergdahl et al., 2020; Gordillo et al., 2021) or Monte-Carlo Tree Search (MCTS) or deep learning methods for determining the difficulty (as measured by the pass rate) of puzzle game levels (Mugrai et al., 2019; Gudmundsson et al., 2018).

The field of AI for playing games has received much attention over recent years, and it is the track at Conference on Games with the most paper submission[2]. While this increased interest in the field pushes state-of-the-art methods, employing such methods in practice has many pitfalls and difficulties: Not all games allow for tree-search-based methods like MCTS; they require a lot of hyperparameter tuning and implementation choices (Engstrom et al., 2019); or some may struggle to work for specific games (Kamaldinov and Makarov, 2019).

In this dissertation, the focus has been on understanding what affects the efficiency of RL methods in puzzle games. Paper 3 (Kristensen and Burelli, 2020) first explores how some of these common limitations can be overcome in the specific case of *Lily's Garden*, one of Tactile Games' top games. Three specific strategies for more robust learning are proposed: shuffling the colours of the board pieces, including an action mask in the observations, and resetting the game once a certain number of steps to prevent catastrophic learning.

Paper 4 (Kristensen et al., 2020) extends this work by considering ten times the number

[2]In 2020 `https://ieee-cog.org/2020/program`

of levels and correlates the agent performance with player data. A stronger correlation with player behaviour is observed by only considering the best 10% of the agent playthroughs, with similar findings also reported for other games (Roohi et al., 2021). As noted by Gudmundsson et al. (2018), the agent learns how to play very differently than human players. Rather than attempting to develop an agent that plays exactly like a human, Paper 5 (Kristensen and Burelli, 2022) examines how different methods and data types can be used together with agent data to predict the difficulty on levels with no observations of how players have played the level.

1.6 Structure of the thesis

The thesis is split into three major parts.

The first part contains the introduction and background chapters.
Chapter 1 provides the motivation for the different studies conducted during the PhD as well as an overview of the overall dissertation.
Chapter 2 describes related work in these research fields to which this dissertation contributes.

The second part is the main body of the dissertation and includes a more in-depth discussion of the results from the included articles and how they answer the research questions laid out in Chapter 1.
Chapter 3 discusses how difficulty is modelled when there is rich player data available.
Chapter 4 discusses the development of a playtesting agent that can be used to play puzzle levels.
Chapter 5 focuses on the results from using a playtesting agent and how it is used to address the cold start issue when new content needs to be evaluated while having limited historical data available to model it.

The third part is a discussion and conclusion about the results and impact of the study and possible channels for future work.
Chapter 6 contains a critical discussion of the results and limitations presented in the main body of the work, as well as possible future work. In the end, there is a summary of all the work.

2 Background

The two major lines of research in this thesis regarding modelling players and difficulty and building AI agents for playtesting have their roots in different research fields. While these two areas often overlap in research and application areas, the focus is often on one or the other. An important viewpoint in this PhD study is that they must be treated with equal importance.

This chapter first introduces difficulty as a general concept and how modelling and predicting difficulty in various games have been done. The second section focuses on reinforcement learning and introduces relevant technical terms and concepts for developing automated playtesting methods. The last section provides an overview of methods and approaches that have been used for difficulty prediction and playtesting other aspects of games such as quality assurance and general testing.

2.1 Operationalising difficulty

Difficulty is the core concept that is used throughout this dissertation. However, what constitutes difficulty is often ill-defined and changes across games (Denisova et al., 2020), making it hard to operationalise and leading to unclear research objectives. In this section, we delineate how the term *difficulty* is used in literature and applied in different games as well as explaining how we define it in this dissertation. Lastly, we provide an overview of how it has been modelled to allow for its operationalisation.

2.1.1 Characterising difficulty

Before discussing how difficulty is typically operationalised in games, we must characterise what we mean by difficulty. Difficulty is often used to describe how easily a player can overcome obstacles and complete the tasks (Denisova et al., 2017), but the experience is highly individual, depending on the player's skill. We, therefore, often differentiate between the difficulty as a whole and the *perceived* difficulty, which throughout this dissertation will refer to the experienced difficulty of a singular player.

We can also differentiate between describing difficulty as a *relational* or an *intrinsic* metric (Denisova et al., 2020). When players are asked how difficult they find a task, it describes a relational attribute between the player and the task. On the other hand, in other works that attempt to describe the difficulty of a task (e.g. the win rate of an AI agent on game levels (González-Duque et al., 2020)), it is often used as an intrinsic property of the task. This subtle distinction between describing a task by the relational or its intrinsic difficulty is especially relevant in this dissertation: While we may be able to use playtesting agents to determine ground truths about puzzle game levels – e.g., the optimal strategies, the minimum number of moves required to complete a level or the pass rate depending on random seeds – the experienced difficulty of the players is not necessarily the same.

Another consideration is that difficulty is often used synonymously with *challenge*, but as Denisova et al. (2020) notes, they are not the same. The word *difficult* commonly carries a negative connotation, and a *difficult* task puts forth an expectation of struggle and frustration. While we use difficulty with a neutral valence in this dissertation, a *challenging* task carries a different connotation and suggests the task is mentally stimulating and engaging. This differentiation is relevant since feeling challenged is one of the key elements for players to find games enjoyable (Adams, 2014; Alexander et al., 2013; Brockmyer et al., 2009), but it depends on other things than just the game difficulty. It also depends on the recent history and skill of the player as well as on game elements that introduce uncertainty or affect the chance of winning (Denisova et al., 2020).

Challenge can be broken down into different types of challenges, the two main ones being *cognitive* and *performative* (Ermi and Mäyrä, 2007; Cox et al., 2012). A cognitive challenge tests the player's ability to reason and solve problems, which is typical in puzzle games like *Lily's Garden*. Performative challenges depend on the players' reaction times, speed, and physical interactions with the game. Action and platform games like the *Dark Souls*[1] series or *Super Meat Boy*[2] are typical examples of such challenges where timing is crucial, and the physical limitations of the player play a key role. Cole et al. (2015) extend this by referring to those types of challenges as *functional*, which is contrasted with *emotional* challenges where the focus is not what the players can do but rather how they feel. In games with narrative elements (e.g., *Papers, Please*[3]) or characters that player feels connected with (e.g., *Life is Strange*[4]), the player can not overcome emotional challenges through functional skills but instead require them to resolve ambiguity and tension in the narrative. Many games consist of multiple types of challenges, such as *The Elder Scrolls V: Skyrim*[5], that require the player to employ performative skills in combat, cognitive skills for solving puzzles and emotional choices for various quests throughout the game.

In the scope of this dissertation, we focus on specific aspects of difficulty in puzzle games. While difficulty is related to challenge as discussed above, we do not explore all aspects of challenge. One of the central games in this dissertation, *Lily's Garden*, does contain narrative aspects that can pose emotional challenges, but it is outside the scope of this dissertation to investigate its impact on the player experience. Similarly, puzzle games do not generally require any performative skills, so the main focus revolves around cognitive challenges and how difficulty can affect this. In order to define what we mean by difficulty in puzzle games so it can be operationalised, we outline how it has been introduced and defined in other games in the next section.

[1]`https://en.bandainamcoent.eu/dark-souls`
[2]`http://supermeatboy.com/`
[3]`https://papersplea.se/`
[4]`https://lifeisstrange.square-enix-games.com/en-gb/`
[5]`https://elderscrolls.bethesda.net/en/skyrim`

2.1.2 Difficulty in games

Difficulty is one of the key aspects game designers want to be able to measure to ensure a good player experience at all points in the game (Alexander et al., 2013). Early on, many games employ different tutorials at the beginning of the game to slowly familiarise the players with core concepts of the game to allow the players to gradually acquire the necessary skills to progress (Juul, 2011). Later on, it is also essential to ensure that players do not feel stuck on challenges that exceed their skill, which can negatively affect their enjoyment (Drey et al., 2021). It is, therefore, crucial for game designers to be able to judge whether there is a good correlation between acquired skills and difficulty, which can allow them to make the necessary adjustments to the content.

For this purpose, we need to operationalise difficulty. The way difficulty is operationalised should reflect the player experience and enable the level designers to measure the impact of any changes easily, but this can vary from game to game. Lomas et al. (2017) give a precise definition of difficulty as *the probability of task failure*, which provides an interpretable metric for measuring and comparing difficulty. For example, winning a coin toss is less difficult than guessing the correct number on a die ($p(\text{heads}) = 0.50 > p(6) = 0.17$), but while this idea of probability of success is intuitive, the complexity of many games do not allow putting the difficulty into a simple formula (e.g., Aponte et al. (2011b)). Difficulty is therefore typically described as a relational attribute between the players and the games, and any measurement is derived from the player's performance within the game. Such measurements generally consider one or more metrics that describe the *chance of success*, *progression/score* and *time/actions taken*.

One important distinction is that in some literature, difficulty is a loose concept (e.g. easy/medium/hard). Changing the difficulty in these cases is accomplished by adjusting game parameters that are assumed to capture difficulty (e.g. the number of enemies (Allison and Polich, 2008)), where any changes in the player performance metrics are considered a consequence of the change in difficulty. However, we argue that for operationalising difficulty, these metrics *are* the difficulty, and changes in game parameters should be evaluated against these metrics to determine whether these changes make the game more difficult or not. In the following, we will highlight a few illustrative examples of how different games use these metrics for difficulty.

Allison and Polich (2008) examine the physiological reaction of players in a first-person shooter game where they play through 4 different difficulty levels with an increasing number of enemies. To evaluate the players' performance, they measure the kills, wounds, shots fired, and mission success/failure. Similarly, Vicencio-Moreira et al. (2015) also use kills and mission success along with questionnaires to evaluate how different balancing methods affect these metrics when two players of different skill levels are matched up against each other. Common in these two examples and similar works (Kneer et al., 2016; Mora et al., 2015; Hristova, 2017; Aponte et al., 2011a) are the metrics of kills, which can be considered a score metric, and the probability of success as measured by the win rate.

In some genres, it is not easy to directly measure the player's performance. In the case of role-playing games (RPG), the goals are often more open in nature and often let the players choose a pre-defined difficulty themselves. These types of games often rely on emotional challenges, where a questionnaire like CORGIS (Denisova et al., 2020) is warranted. However, as discussed by Bostan and Öğüt (2009), if the difficulty is adjusted automatically, the difficulty should reflect the likelihood of the player reaching their goal. In other instances where there is no clear goal or victory condition, metrics capturing the progression are often used instead. For example, Aponte et al. (2011b) consider the number of pellets eaten by *Pac-man* before losing all three lives as the difficulty evaluation function. While such score measurements depend on the game, they provide a way to compare the relative difficulty in the games.

Some games often consist of distinct tasks, but it is not always easy to measure player performance directly. For example, Pusey et al. (2021) investigate how difficulty can be measured in the puzzle games *The Witness*[6], *Untitled Goose Game*[7] and *Baba is You*[8]. They consider metrics related to the time it takes for the players to complete the puzzles and the number of incorrectly submitted solutions. While they find a correlation between these two metrics, in games like *Untitled Goose Game*, the tasks can be completed in different orders or are optional, and there is not an easy way to measure the number of actions or attempts. They, therefore, note that not all games lend themselves to having their difficulty being described by one metric. In a similar setting, Linehan et al. (2014) use the minimum number of actions taken to solve puzzles in four commercial games (*Portal 1*[9], *Portal 2*[10], *Braid*[11] and *Lemmings*[12]), but this analysis relies on the researchers denoting playthroughs.

In genres with distinct, limited tasks, both the probability of success and time taken are commonly used. While not explicitly related to games, similar concepts are also used for estimating the difficulty of programming tasks (Effenberger et al., 2019), where the time taken to learn new things or error rate on tasks are considered. For platform games, Wehbe et al. (2017) consider, in addition to the time taken and success rate, the reported difficulty of the players. For more traditional puzzle games such as Sudoku, in the work by Pelánek (2011) they consider the mean solution time required to complete a Sudoku puzzle for evaluating the difficulty.

For the specific type of games considered in this dissertation, mobile puzzle games, similar ways of operationalising difficulty have been adopted. For example, for the *Angry Birds* games, both the pass rate of the players and the number of birds used are commonly considered (Roohi et al., 2021; Stephenson and Renz, 2019; Kaidan et al., 2016). In a similar vein, difficulty in Candy Crush games is described by the player success rate (Gudmundsson et al., 2018; Poromaa, 2017). In this dissertation, the game we focus on is the puzzle game *Lily's Garden*, which shares not only similar mechanics with these kinds of puzzle games but also commercial context. Operationalising difficulty as the probability of success has been established in this section as a common measurement across many genres, and for commercial puzzle games, it is the norm. We therefore also adopt this operationalising of difficulty as the probability of

[6]`http://the-witness.net/`
[7]`https://goose.game/`
[8]`https://hempuli.com/baba/`
[9]`https://store.steampowered.com/sub/469/`
[10]`https://www.thinkwithportals.com/`
[11]`http://braid-game.com/`
[12]DMA Design, Lemmings, Amiga, Psygnosis, 1991.

success, or pass rate, of a given level. However, we note that in some instances, it may be more natural to consider the inverse pass rate, which instead describes the average number of attempts players spend on the levels.

2.1.3 Modelling difficulty and players

Measuring difficulty is one thing. Being able to predict the expected difficulty is another. Whether it is for *dynamic difficulty adjustment* (DDA) (Hunicke, 2005) or measuring the player experience for *procedural content generation* (PCG) (Yannakakis and Togelius, 2011), predicting the difficulty and player experience is necessary for taking proactive steps regarding the design of the content. Depending on the use case, three components can be considered for these modelling purposes: a formal model of gameplay, the content itself and/or the players (Dziedzic and Włodarczyk, 2018). While some methods also consider AI methods to evaluate the levels, this section focuses on literature that relies on analytical methods or player/level data.

A common use case for predicting difficulty is DDA, which aims to maximise player engagement by adjusting the difficulty during gameplay (Hunicke, 2005). DDA is used in numerous games, so it covers a wide variety of methods. Some methods are more reactive and adjust the content in set increments depending on the player's recent performance. An example of this is by Silva et al. (2015), where they change the strategy of the opponent AI in *DotA* if the relative performance (here, a linear combination of the level of the player, number of deaths and objectives describe) between the player and AI exceeds a given threshold. Pfau et al. (2020) use a λ parameter that changes depending on the success of previous playthroughs and is used to adjust the enemy health, spawn frequency and abilities. Nagle et al. (2016) consider four different difficulty adaptation methods where their DDA implementation change depending on the player's performance as measured by kills. Additionally, they examine how the Big Five personality traits (e.g. Roccas et al. (2002)) correlate with the duration and enjoyability of the different adjustment schemes.

Given that players are motivated differently, some approaches focus on modelling the skill of the players. For *Tetris*, Lora et al. (2016) use a clustering method based on the players' playstyle and decisions on where to place the first ten pieces to group people into three categories (newbie/average/expert). In games with a high degree of chance (e.g., casino games), Borm and van der Genugten (2001) note that the skill of the player should be measured over time and that the skill level of a player falls on a number between 0 (pure chance games) and 1 (pure skill games). Gonzalez-Duque et al. (2021) use a combination of prior knowledge of players and Bayesian optimisation methods that enable using priors to predict as quickly as possible the time a given player will use to solve Sudoku puzzles.

Other probabilistic methods have also been used to estimate the success of players. For example, Mourato et al. (2014) consider the player success rates depending on level parameters in platform games, such as gap sizes, timing windows and opponent speeds. This allows them to build a probabilistic model where the effect of each component on failing the level is used to estimate the final probability of completing the level. Bunian et al. (2017) consider a Hidden Markov Model (HMM) that captures a player's in-game behaviour and use this HMM to generate new behavioural features for predicting the expertise and the Big Five traits of the players.

Similarly, Xue et al. (2017) build a graph model that describes the probability of success and churn rate based on historical playthrough data. By knowing the historic churn and pass rate for a given random seed on a level and by treating the problem as a Markov Decision Process (see Section 2.2), their model serves the players with levels that maximise their engagement. In one of the included papers in this dissertation (Paper 1, Kristensen et al. (2021)), a statistical model is used to estimate the number of actions the general population requires to complete the level. From the parametric description of each level, the impact of changing the move limit can then be estimated.

A number of approaches break down the challenges into specific components that can be used for estimating the difficulty. Van Kreveld et al. (2015) consider a linear regression model that consists of puzzle level variables which can be used to predict a ground truth pass rate. The puzzle level variables are split into three categories: *initial* features that describe the initial setup (e.g., board size), *solution* features that describe the board state in its solved state, and *dynamic* features that capture aspects of the gameplay of the level (e.g., the minimum number of required moves). Wheat et al. (2016) also model the difficulty of platform games by different parameters such as platform size and speed. This is also the case in the work by Wehbe et al. (2017) where they use a Poisson regression to model the error rate of players depending on scroll speed, target size and jump task complexity, which can be used to estimate how much the error rate is expected to change as the level parameters change.

Aponte et al. (2011b) model the difficulty of the level and the skill of the player where the conditional probability of the player completing a level also depends on whether the degree they have learned the ability that certain challenges consist of. However, it is necessary to be able to measure these abilities. This shares some links with the Additive Factors Model (AFM) (Cen et al., 2006) which describes a learner's probability of success in a given task that requires a given ability, or *Knowledge Component* (KC). This has been applied in educational games by Harpstead and Aleven (2015) where they examine how children learn to balance in the game *Beanstalk*. Ultimately, such approaches rely on specifying specific skills of the players, which may not always be straightforward. In Paper 2 and 5 (Kristensen et al., 2022; Kristensen and Burelli, 2022) of this dissertation, we introduce a prediction method based on *Factorisation Machines* (FM) (Rendle, 2012). This method learns a latent representation of players and levels automatically and does not rely on specifying any abilities beforehand. It can be used to estimate the number of attempts a player is expected to spend to complete the level. Similar matrix factorisation methods have been suggested to be used for immediate content generation tailored to players (Zook et al., 2012).

Many of the methods above rely on leveraging existing player data for estimating the difficulty of the game content. While some of these methods can be generalised to new content (e.g., the linear regression model for puzzle games proposed by Van Kreveld et al. (2015)), not all games allow for an accurate estimate using these *initial* features. Another consideration is that in commercial games like *Lily's Garden*, new ideas and challenges are constantly introduced, which can quickly invalidate such approaches. The necessity to playtest novel content without observing players is a golden goose for game developers, and the methods that are used for developing such tools in this dissertation are the topic of the next section.

2.2 Reinforcement learning

Estimating the difficulty of a puzzle level by only observing the initial game board is hard and does not reveal "traps" or other dynamic aspects of the level that arise during gameplay. A large part of the research in this dissertation has, therefore, focused on creating a way to playtest levels to explore these dynamics to allow for a more comprehensive description of the level. As a first step, game designers often develop agents based on behaviour or heuristic behaviour trees, but as the game complexity grows over time due to new content and players, such tools can quickly become hard and cumbersome to update and maintain. Instead, a more feasible approach is one where the *playtesting agent* can adapt to content with minimal intervention. For this purpose, we look to the field of reinforcement learning (RL), which we will describe in this section.

2.2.1 Introduction

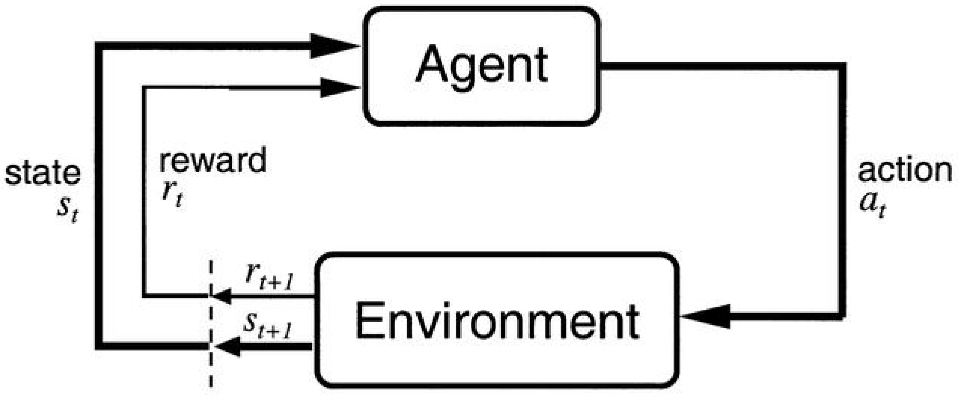


Figure 2.1: Basic reinforcement learning loop. Adapted from Sutton and Barto (2018).

The basic premise of reinforcement learning is that through interactions with the environment, an agent learns which actions maximise the long-term reward. The field has been thoroughly described in the textbook by Sutton and Barto (2018), but this section focuses on the background and concepts related to methods used in the included papers.

RL is commonly visualised as the loop shown in Fig. 2.1 where the agent with the *policy*, π_t, chooses an *action*, a_t, at time t given the *state* s_t. The agent then receives a reward, r_{t+1}, and observes the next state, s_{t+1}.

In this formulation, RL problems can be described as a *Markov decision process* (MDP) where the dynamics of the system can be expressed by the following parameters:

- Action a from the action space A, $a \in A$.
- State s from the state space S, $s \in S$.
- The initial state distribution, $p_0(s_0)$.
- A state transition density model, $p(s'|s,a)$, which describes the probability of transitioning to the new state, s', given state s and action a.

- A reward function, $r(s, a, s')$, that describes the acquired reward by going to state s' from state s after taking action a.

A common condition for describing the RL system as an MDP is that it should satisfy the *Markov property*, which describes a situation where the future state only depends on the current state and not the whole history. In a more formal description, the dynamics of the environment can be considered an MDP if the following is satisfied:

$$p(s_{t+1}|s_t, a_t) = p(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, \ldots, s_0, a_0)$$

In instances where the Markov property is not satisfied, such as when underlying or hidden information is not captured by the state, it is often referred to as a *partially observable Markov decision process* (POMPD). This is common in games where the player does not have access to the full state such as the random seed or other background information like stochastic processes. This can cause the RL algorithm to converge slowly or not at all, and in order to address that, the full trajectory is required to solve the problem.

2.2.2 Policy and value function

The goal of reinforcement learning is obtaining a *policy*, π, that maximises the reward. It is common to distinguish between a deterministic policy, $\pi(s) = a$, where the chosen action depends only on the state, or a stochastic policy where the action is sampled from a distribution, $a \sim \pi(a|s)$.

How well the learned policy is performing is evaluated by the expected future reward, or return, which is calculated by the sum of future discounted rewards:

$$R_t = \sum_{k=0}^{T} \gamma^k r_{t+k+1},$$

where T is the episode length and $\gamma \in [0, 1]$ is the discount factor. The discount factor is a mathematical addition that ensures that reward is finite in non-episodic problems. Smaller values of γ put a larger weight on immediate rewards, while a discount factor close to 1 also allows future rewards to have an impact on the return.

The *value function* is used to estimate the expected return, R_t, from following policy π given that system is in state s:

$$V_\pi(s) = \mathbb{E}_\pi\left[R_t|s_t = s\right]$$

Another relevant value is the Q-value which describes the expected future reward from taking action a in state s following policy π:

$$Q_\pi(s, a) = \mathbb{E}_\pi\left[R_t|s_t = s, a_t = a\right]$$

The relation between V_π and Q_π follows from the fact that $V_\pi(s)$ can be described as the sum of $Q_\pi(s, a)$ weighted by the probability of picking action a:

$$V_\pi(s) = \sum_{a \in A} Q_\pi(s,a)\pi(s,a) \tag{2.1}$$

A useful concept is the difference between the two value functions called the *advantage*:

$$A_\pi(s,a) = Q_\pi(s,a) - V(s)$$

The advantage describes how much better (or worse) action a is compared to the average performance.

2.2.3 Bellman equations

In order to tie the policy and value function descriptions into the MDP framework, we note that $Q_\pi(s,a)$ can be described as the sum of returns over all possible future states weighted by the transition probability:

$$Q_\pi(s,a) = \sum_{s' \in S} p(s'|s,a)[r(s,a,s') + \gamma V_\pi(s')], \tag{2.2}$$

where we have used the fact that the expected return depends on the immediate reward plus the discounted future rewards, $R_t = r_{t+1} + \gamma R_{t+1}$.

Combining Eq. 2.1 and Eq. 2.2 leads to the Bellman equations:

$$\begin{aligned} V_\pi(s) &= \sum_{a \in A} \pi(s,a) \sum_{s' \in S} p(s'|s,a)[r(s,a,s') + \gamma V_\pi(s')] \\ Q_\pi(s,a) &= \sum_{s' \in S} p(s'|s,a) \left[r(s,a,s') + \sum_{a \in A} \pi(s',a')\gamma Q_\pi(s',a') \right] \end{aligned}$$

These equations describe the value function in terms of the sum of all returns from future states weighted by the probability of visiting these states, which depend on both the transition probabilities and the policy.

2.2.4 Estimating the value function

At the initial step of a problem, there is no information regarding the expected return. For most RL algorithms, the goal is to learn an estimate for this through the value function, which is typically approached in three different ways: dynamic programming (DP), Monte-Carlo (MC) and temporal difference (TD). They can be visualised via *backup diagrams* (Fig. 2.2), which show the possible trajectories in the RL setting with each white-black node pair representing a state-action pair.

While we focus on TD methods in this dissertation, it is worthwhile to discuss DP and MC methods briefly. DP is a *model-based* approach where perfect knowledge of the system (i.e. transition probabilities of the MDP) allows the value function over all states to be computed. Through *policy iteration*, the estimated value function, V_{π_i} evaluates the policy, π_i, which is then in an iterative manner used to update the value function until they converge towards the

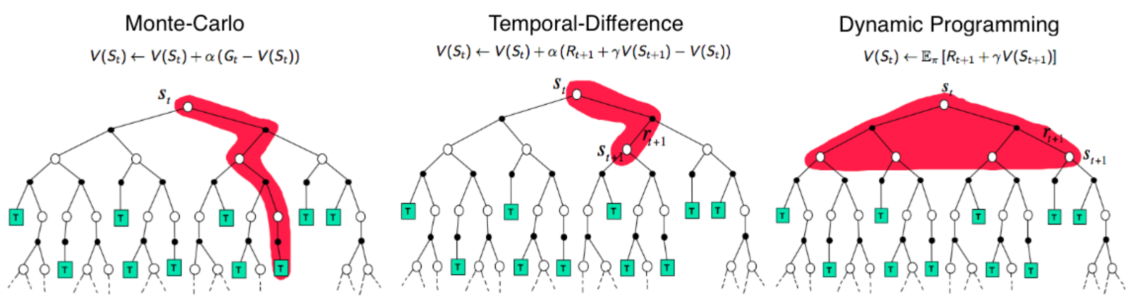

Monte-Carlo
$V(S_t) \leftarrow V(S_t) + \alpha(G_t - V(S_t))$
s_t
T
Temporal-Difference
$V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$
s_t
r_{t+1}
s_{t+1}
Dynamic Programming
$V(S_t) \leftarrow \mathbb{E}_\pi[R_{t+1} + \gamma V(S_{t+1})]$
s_t
r_{t+1}
s_{t+1}

Figure 2.2: Backup diagrams for the three main methods to learn the value function for RL problems. The white circles represent a state, s, and the black circles represent the possible actions, a, in that state. The red areas highlight which observations are used to update the value functions in each case. Adapted from David Silver's lecture 4 on RL; `https://www.davidsilver.uk/wp-content/uploads/2020/03/MC-TD.pdf`.

optimal policy, π^*, and the related value function, V^*. In practice, it is rare to have a full model of complex problems, and they can be computationally heavy, so MC and TD methods are generally used.

MC methods are typically *model-free* and rely on exploring the environment to learn the value function. An important aspect of these methods is that they only work for *episodic* problems – i.e. there is a terminal state. By exploring a given path in the backup diagrams until a terminal state is reached, the actual average reward can be used to update the value function. An advantage of MC approaches is that they utilise the empirical return which means that it is not necessary to model the environment.

TD methods are also *model-free*, but unlike MC methods, they do not rely on observing a full episode before updating the value function. Instead of using the actual return to update the value function, TD methods instead utilise an estimate of the discounted value of the next state. However, since this an approximation of the real return, there is an associated reward-prediction error, or *TD error* δ, which is given by

$$\delta = r(s, a, s') + \gamma V_\pi(s') - V_\pi(s) \tag{2.3}$$

This can be used to update the value function from the following equation:

$$V(s_t) \leftarrow V(s_t) + \alpha\delta,$$

where α is the step size.

Learning the policy in conjunction with the value function lead to an actor-critic structure, which is visualised in Fig. 2.3. The *critic* is parameterised by w and is used to estimate the value function, $V(s; w)$. The actor that is parameterised by θ with the policy π_θ can utilise the TD error from Eq. 2.3 to update the preference for a given action.

2.2.5 Exploration-exploitation dilemma

Common to model-free methods is the fact that there is a trade-off between *exploration* and *exploitation*: sufficient exploration is necessary to ensure possible better alternatives to the

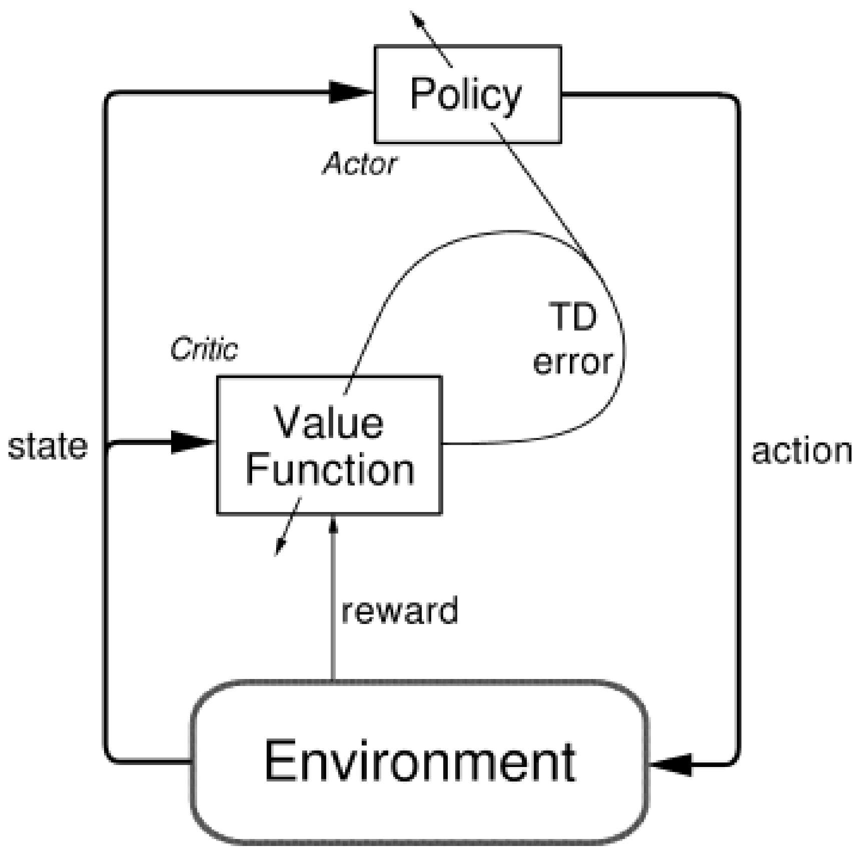


Figure 2.3: Actor-critic architecture. Adapted from Sutton and Barto (2018).

actions are found, but in order to maximise the returns and evaluate the current policy, it is necessary to follow and exploit the current policy.

Many algorithms differentiate between *on-policy* and *off-policy* methods. On-policy methods use the learned policy, π, to both collect observations and exploit the learnings. For this reason, π is often stochastic, which allows the chosen action to not always be the optimal one. Off-policy methods make use of a second policy called *behaviour policy*, which is different from the learned one and can be a heuristic-based one or even completely random.

2.2.6 Policy gradient methods

While the previous sections are primarily concerned with how the value function is updated, an essential aspect of RL is also how the policy can be updated. For this purpose, assume that the policy can be parameterised by θ, i.e. $\pi(a|s;\theta)$. We can define an objective function that focuses on achieving the highest return:

$$\mathcal{J}(\theta) = V_{\pi_\theta}(s_0),$$

where s_0 is the initial state. From this, it follows that the policy π_θ can be optimised by updating θ following a gradient ascent method. I.e.,

$$\theta \leftarrow \theta + \alpha \nabla_\theta \mathcal{J}(\theta),$$

where α is the learning rate and $\nabla\mathcal{J}(\theta)$ is the policy gradient. Given the fact that the collected rewards and states do not depend on θ, the policy gradient can also be written as

$$\nabla_\theta\mathcal{J}(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \ln \pi(a|s;\theta) Q_\pi(s,a)\right],$$

which is the *Policy Gradient Theorem* and is used in several policy gradient algorithms. An advantage of these methods is that they do not depend on the state space and thus the underlying MDP, which means they can be combined with model-free methods discussed previously. They are also often used in actor-critic architectures due to the ability to learn a value function simultaneously, which can assist in learning the policy.

2.2.7 Function approximation

Before introducing more specific RL methods used in this dissertation, it is helpful to introduce a common way to deal with complex RL problems. As discussed previously, the value functions and policies depend on the state, and we assumed each state was unique and tabular. However, in many RL problems, the state space may be very large. For example, the state space of a video game may be represented by the pixels forming a 2D image, but the state in two consecutive frames will be considered entirely unique, even if we know they should be similar. To deal with this, *function approximators* can be used to simplify the space, with artificial neural networks (NN) being a widespread approach and are referred to as *deep RL*. They are often also used as function approximators for the value functions and policy and may or may not share parameters in actor-critic methods.

Using NNs as function approximators has two significant difficulties. The first is that during *online* learning, where observations of the transitions are collected while learning, consecutive samples of the observations will be strongly correlated. Updating the weights of the NN with correlated data can quickly cause the method to become stuck in local minima. The other issue is that the problem is not stationary during learning, which is also true for other types of approximators that approximate the value function. Whenever the model parameters, θ, are updated, the target output of the value function also changes, which is unlike typical optimisation problems where the target stays the same. How they have been resolved will be discussed in Section 2.3.1.

2.2.8 Algorithms

In this dissertation, the main algorithm that is used for training a playtesting agent is Proximal Policy Optimization (PPO) (Schulman et al., 2017). However, there are many other policy gradient algorithms which sometimes employ similar algorithmic improvements (e.g. *Generalised Advantage Estimation* (GAE) (Schulman et al., 2015)). Weng (2018) provides a comprehensive overview, but in this section, the content is limited to natural policy gradient methods.

The basic idea behind natural policy gradients is that the updates to the model parameters, $\Delta\theta$, should be as large as possible with the minimal change in policy, π_θ, while still improving the policy. This is related to the fact that the problem is not stationary. If the new policy is very different from the previous one, the estimated value function is not representative of the new policy. By also searching for the largest update to the model parameters, natural policy gradients tend to converge much faster than normal policy gradients.

Trust Region Policy Optimization (TRPO)

In order to apply natural gradients to deep RL problems, Schulman et al. (2015) proposed *Trust Region Policy Optimization* (TRPO). The main idea is that the old policy function, $\pi_{\theta_{\text{old}}}$, should not diverge too much from the new policy, π_θ, which is enforced by ensuring the *Kullback-Leibler (KL) divergence*, D_{KL}, between the two policies stays small. By phrasing the problem as an *offline* learning problem where the behaviour policy is equal to $\pi_{\theta_{\text{old}}}$, the objective function follows a similar structure as an *importance sampling* problem (e.g., Levine et al. (2020)):

$$J(\theta) = \mathbb{E}_{s\sim\rho_{\pi_{\theta_{\text{old}}}}, a\sim\pi_{\theta_{\text{old}}}}\left[\frac{\pi_\theta(s,a)}{\pi_{\theta_{\text{old}}}(s,a)} Q_{\pi_{\theta_{\text{old}}}}(s,a)\right],$$

where ρ_π is the distribution of states visited following policy π. In practice, the objective function under the KL divergence constraint can therefore be written as

$$L(\theta) = J(\theta) - \lambda(D_{\text{KL}}(\pi_{\theta_{\text{old}}}||\pi_\theta) - \delta),$$

where δ is the permitted KL divergence and λ is the update step size. From this formulation, the updates to the model parameters then become

$$\Delta\theta = \frac{1}{\lambda} F(\theta_{\text{old}})^{-1} \nabla_\theta J(\theta),$$

where F is the *Fischer Information Matrix* which can locally approximate the KL divergence for similar distributions. This formulation follows the original definition of natural gradients (Amari, 1998), hence the link between TRPO and natural gradients.

Proximal Policy Optimization (PPO)

Two limitations to TRPO are its computational complexity and inefficiency when using complex network architectures. To overcome this, Schulman et al. (2017) introduced *Proximal Policy Optimization* (PPO). It uses a clipped surrogate function to prevent large updates and is given by

$$\begin{aligned} L^{\text{CLIP}}(\theta) = \mathbb{E}[\min(&r(\theta)\hat{A}_{\pi_{\theta_{\text{old}}}}(s,a), \\ &\text{clip}(r(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_{\pi_{\theta_{\text{old}}}}(s,a))], \end{aligned}$$

where $r(\theta) = \dfrac{\pi_\theta(s,a)}{\pi_{\theta_{\text{old}}}(s,a)}$, and the hat implies the empirical values.

In problems where the value function also needs to be evaluated, such as in actor-critic architectures, they propose additional terms in the loss function so it becomes

$$\begin{aligned} L^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}[&L^{CLIP}(\theta) \\ &- c_1 L^{\mathrm{VF}} \\ &+ c_2 S[\pi_\theta](s)], \end{aligned}$$

where $L^{\mathrm{VF}} = (V_\theta(s) - V_{\mathrm{target}})^2$ describes the value function loss, $S[\pi_\theta](s)]$ is an entropy bonus that encourage exploration by ensuring the policy does not only favour one action, and c_1 and c_2 are hyperparameters that affect the importance of each term.

2.3 Automated playtesting methods

As discussed previously, automated playtesting is becoming increasingly necessary for game developers, but these methods for playing games can take on many forms. In the *Artificial Intelligence and Games* book by Yannakakis and Togelius (2018), they consider three overall approaches for using game AI for playing games: planning-based, RL and supervised learning – or any combination of the three. While ad-hoc behaviour authoring methods such as Finite State Machines and Behaviour Trees are commonly employed in various games for AI, they can be very complex to create and hard to maintain and scale, especially for constantly changing games. In this section, we first give an overview of various methods developed for playing games before discussing how these methods have been used specifically for playtesting different aspects of games.

2.3.1 Learning to play games

Games have become a standard testbed for both RL and AI methods due to the ease of interfacing the algorithms with the game itself but also due to the diversity of games that allow researchers to tackle a wide range of challenges (Yannakakis and Togelius, 2018). We mainly discuss deep learning methods due to their ability to manage the complexity of many game environments and wide adoption in many game genres (Justesen et al., 2019), but other relevant methods will also be discussed.

A popular benchmarking framework for gameplaying algorithms is the Arcade Learning Environment (ALE) (Bellemare et al., 2013), which has over 50 Atari 2600 games available. This includes popular games like *Ms Pac-man* (a sequel to *Pac-man*) and *Space Invader* but also games such as *Montezuma's Revenge* that is recognised for being a hard exploration problem due to its sparse rewards. The framework allows to easily interface the games to a common RL setup and save the game state in the emulator, which is necessary for search-based methods.

An influential model-free approach is by Mnih et al. (2015) who introduced *deep Q-learning* (DQN) which is able to use the raw pixels to learn playing 49 Atari games in the ALE framework. DQN is based on standard Q-learning but uses artificial neural networks and convolutional neural networks (ConvNN) as function approximators. Due to the challenges of using NNs in RL, they introduced two critical components to the method: An *experience replay buffer* from which observations can be sampled during training of the neural network, which helps remove correlations between subsequent observations, and only *periodically updating the target* which helps with the non-stationarity that occurs when updating the model parameters. Further improvements, such as Prioritised Experience Replay (Schaul et al., 2015) and Duelling Q-networks (Wang et al., 2016), have been combined in the Rainbow DQN (Hessel et al., 2018), enabling super-human performance on most of the 57 Atari games. Similarly, *Agent57* (Badia et al., 2020) also achieves super-human performance on 57 Atari games by using a dual NN architecture for intrinsic and extrinsic rewards, a *meta-controller* for adaptive exploration and exploitation, and a larger window in the recurrent neural network (RNN) part of the architecture.

The other family of game-playing methods are planning or search-based methods, which have been very popular for board games such as Chess and Go (Lee et al., 2009). They differ from the above approach in that they are able to plan the next move by utilising a model-based approach or tree search to evaluate the value and impact of a given action. A popular search-based method is the Monte-Carlo Tree Search (MCTS) (Chaslot et al., 2021) which combines random sampling with tree search that allows for a good balance between accurately estimating the expected return and exploring the options. It has been used in a wide variety of contexts for games (Browne et al., 2012), but it relies on being able to simulate decisions, which can become prohibitively expensive in terms of computing or speed in complex games. *MuZero* (Schrittwieser et al., 2020) deals with this problem by instead *learning* a world model that can be queried rather than the environment itself, which the algorithm can then use for planning (e.g. MCTS). This enables the method to be used for both board games and Atari games and does not require any prior knowledge such as game rules. Other planning methods have also been used for playing Atari games. Lipovetzky et al. (2015) apply the Iterative Width planning method, a breadth-first planner, to the memory (RAM) state space of 54 ALE and find that it performs similar or better than the Monte Carlo based UCT (Kocsis and Szepesvári, 2006).

Evolutionary strategies (ES) have also been proposed for learning how to play Atari games (Hausknecht et al., 2014). In this work, they examine four different ways to evolve a neural network for learning a policy: a static network with only the weights evolved, CMA-ES (Hansen, 2006), NEAT (Stanley and Miikkulainen, 2002) and HyperNEAT (Gauci et al., 2008). Salimans et al. (2017) also compare CMA-ES to A3C (Mnih et al., 2016) for both Atari games and robotic tasks where they find that CMA-ES performs better in 28 out of 51 games. One advantage of ES is that they do not require calculating gradients like policy gradient methods, which makes them more tolerant to long time horizons where the discount factor of PG methods is necessary to reduce the variance of the gradient and ensure convergence. Additionally, since ES use the full, un-discounted reward, they are invariant to delayed rewards.

A concern with some of the methods mentioned above relates to their feasibility in practice. MCTS methods can be slow, and it is not always possible to simulate multiple decisions without breaking the game (i.e., inability to restore saved state). With ES, it may be hard to ensure similar performance across training sessions due to the stochastic process of evolution. In order to determine feasible approaches to use in the games made by the industrial partner of this dissertation, we look to tools that have been developed for the popular game development tool, Unity. An example is ML-Agents (Juliani et al., 2018) which utilises multiple algorithms including PPO, SAC (Christodoulou, 2019) and an imitation learning approach called GAIL

(Ho and Ermon, 2016). A notable addition to ML-agents is *curiosity* (Burda et al., 2018), which encourages the agent to explore by rewarding it for seeking out unexpected outcomes or emergent behaviour such as tool use (Baker et al., 2019). This can reduce the need for carefully designing a reward function, but it still has specific failure modes, such as the "noisy TV problem" where the agent is rewarded from stochasticity in the environment rather than from exploring.

Another issue is that many RL methods are prone to overfitting their environment (Kirk et al., 2021). In practice, this means that an agent may learn how to play on a specific level by simply recognising certain states and remembering the optimal actions. The agent will effectively have learned one specific strategy per instance, which means that it cannot play other levels where the states are entirely novel. To overcome this, several approaches have been suggested. Cobbe et al. (2019) propose using several standard methods known from artificial neural networks such as L2 weight regularisation and dropout. Laskin et al. (2020) utilise data augmentation techniques known from computer vision such as random translation, cropping and colour jitter can also improve generalisation, with similar ideas employed by Risi and Togelius (2020) who propose to use PCG methods to enhance the generality of ML methods.

In this dissertation, the developed playtesting agent is limited by multiple factors:

- There are no play traces.
- Tree-search is not possible due to technical challenges with saving and loading the game state.
- The implementation should be simple and easy to maintain.

Following these criteria, many of the methods mentioned above (imitation learning, MCTS or *MuZero*) are not feasible to use in the context of this dissertation. One possible choice of algorithm is PPO, which has proven to require little tuning regardless of the domain, and there is additional support through the ML-Agents framework. The challenge is that it can be brittle in terms of learning (Hämäläinen et al., 2018). Furthermore, many RL methods are sensitive to the exact details of the implementation (Ilyas et al., 2018; Huang et al., 2022). The work in Paper 3 and 4 explore how we implement PPO in practice.

2.3.2 Automated playtesting in games

Much of the previous discussion has concerned itself with learning how to beat the game. However, an equally important part for game designers who want to use simulation or AI-based methods for playtesting new content is to consider the gameplay from the players' point of view (Aponte et al., 2011b). As discussed by Roohi et al. (2018), player behaviour is not explained by purely rational and goal-driven decision-making, so a playtesting method that only focuses on winning will struggle to capture the full spectrum of player behaviour. A desirable aspect of any playtesting agent is, therefore, passing the Turing test of game playing bots (Hingston, 2009).

One immediate question is how agent behaviour can be likened with human behaviour. González-Duque et al. (2020) use one type of agent to explore the difficulty, as measured by the win rate, of platform levels that are generated from two descriptors. These results are used as priors for a Gaussian Process, which can allow quickly estimating the win rate of *different* agents (which serve as a proxy for human players). Gallego-Durán et al. (2018) use the NEAT algorithm for learning to play a 2D maze game and investigate if the learning curve of the students and playtesting agents were correlated. While they find a good correlation between the two as measured by the relative difference ($\sim 15\%$), they note that agents tend to be at a disadvantage initially since they learn from scratch, unlike the players who have experience from previous rounds. While not directly comparing the learning curves, Paper 4 in this dissertation (Kristensen et al., 2020) explores how different ways of training agents affect the agent's competence and, importantly, how the number of moves used by the agent correlates with the pass rate of the players. The strongest correlation is when only training the agent once by randomly sampling levels during training, while training the agent on a single level leads to a worse correlation because the agent learns different game mechanics at different speeds. Additionally, only considering a fraction of the best evaluation runs of the agent can lead to a higher correlation with human players, which has also been observed in other games and agents (Roohi et al., 2021).

Another way to test whether the agent is human-like is by analysing whether the selected actions of the agent resemble that of the players. Khalifa et al. (2016) consider the *action length* and *nil action length*, which describes how many repetitions of each action and the length of pauses between actions. Since humans have slower reaction times and require more time strategising than machines, players are expected to have more "sticky" actions and longer nil action lengths. A similar concern is considered for the *AlphaStar* bot (Vinyals et al., 2019) when playing Starcraft II online against other players, where the action rate of the agent has to be manually limited in order to ensure a fair match. Another way to compare behaviours is proposed by Holmgård et al. (2018), where they examine heatmaps of what parts of a map agents with different personas visit. Although they do not compare the behaviour with human players, it provides a way to explore correlations between behaviours which can be extended to include human players.

MCTS methods have been widely adopted for playtesting (Keehl and Smith, 2018; Stahlke et al., 2020; Baier et al., 2018; Poromaa, 2017), and ways to utilise them for playtesting multiple gameplay aspects and styles have been proposed. Variations of MCTS have been proposed by Guerrero-Romero et al. (2018) to create a team of agents that can explore different parts of the game, including a *winner* and *explorer* personality. In a similar vein, Holmgård et al. (2018) propose a variation of MCTS where the criteria for selecting which node to expand are determined through an evolutionary process, and the utility score is based on one of four hand-crafted player personas (such as the *treasure collector* whose score increases with the number of collect items). Mugrai et al. (2019) consider a similar approach specifically for puzzle games where four distinct goals (minimising/maximising score and available moves) lead to four distinct playing styles. Using pre-defined strategies for playing puzzle games has been explored by Shin et al. (2020) where an A2C agent (Mnih et al., 2016) learned to choose the optimal hand-crafted strategy, but the method is not easily adaptable to other games.

Rather than relying on an agent to learn from scratch, some methods utilise player data to assist in learning how to play. Zhao et al. (2020) consider imitation learning approaches along with other RL and planning based methods in four cases studies where they compare both the skill and style of the agents. Similarly, player data can also be used to treat the problem of picking human-like actions as a supervised prediction problem. Ortega et al. (2013) train a neural network to imitate players in Mario, but these methods are not easy to use with limited data available. Tastan and Sukthankar (2011) use *inverse reinforcement learning* (IRL) for teaching an agent to learn a reward function based on expert demonstration in the first-person shooter game *Unreal Tournament*. They find that the players estimate bots trained using IRL compared to the standard bots exhibit a more human-like behaviour. Ariyurek et al. (2019) also find that training playtesting agents using IRL lead to more human-like agents, even if goal of the agents and players is to find bugs rather than playing the game as intended. Dobre and Lascarides (2015) use human corpus data for biasing the node selection of the MCTS method in *Settlers of Catan* following the hypothesis that this can improve the performance in complex games. Gudmundsson et al. (2018) also use neural networks along with an MCTS approach in a commercial setting with an extensive data set where the agent learns to mimic the actions of the players. To account for how any trained agent learns differently than human players in terms of both goal and game mechanics, an additional modelling step is employed where the agent's pass rate is used along with level features as input to a logistic regression model that outputs the predicted pass rate on levels that are not available during training of the model.

This kind of two-step modelling can be beneficial, especially in an industrial setting, since it relies less on creating a playtesting agent that can play exactly like a human and enables an easier way to update predictions. As mentioned before, Gudmundsson et al. (2018) use the pass rate of a trained agent in combination with different puzzle level features to predict the average pass rate on levels without using any kind of player data for the tested levels. Similarly, Roohi et al. (2021) use a deep reinforcement learning agent to extract a baseline pass rate for each level, which is then combined into an "extended" model that simulates a cohort of players with varying skill, persistence and boredom attributes. The results show they can predict the churn and pass rate within 2%. Shaker et al. (2010) use two agents (the search-based A* (Togelius et al., 2010) and a heuristic agent) to play through procedurally generated Mario levels. By recording level descriptors, agent data, and the player experience through questionnaires, they use a multilayer perceptron to predict the reported levels of fun, frustration and challenge.

The challenge of creating a playtesting method that can be used in an industrial context requires it to be robust to change and accurate. In this dissertation, we consider a similar two-step modelling approach as described above in Paper 5 (Kristensen and Burelli, 2022) which is presented in Chapter 5. Crucially, though, we also investigate how this relates to the perceived difficulty of the player on a large range of players.

Part II

Main research

3 Modelling Difficulty Using Historic Observations

For some games, especially luck-based ones, deriving the difficulty from analysing the game is straightforward. For example, if we define difficulty as the probability of success, the chance of winning a fair coin toss is 50%. However, once the games become more complex, describing difficulty using a simple heuristic or formula becomes less straightforward. Instead, it relies on measuring how players perform on the tasks.

The fact that the difficulty of a task is often measured from the interactions of the players also means that modelling the difficulty when such data is available faces very different challenges compared to modelling the difficulty of entirely new content. In this chapter, we focus on how such historical playthrough data can be leveraged for modelling difficulty. This allows us to answer the first two research questions of how difficulty can be operationalised in puzzle games and how it can be predicted effectively for individual players.

The chapter contains two sections. Each section contains results from Paper 1 and Paper 2 of this dissertation. The full-length papers are attached at the end of this chapter. The first section introduces a way to model and operationalise difficulty on a per-level basis. The second section focuses on how it is possible to make personalised predictions, which can be used to estimate how many attempts any player is expected to use before completing a given level.

3.1 Per-level difficulty modelling

A common element in puzzle games is the existence of an action/move or time limit. In order to complete the level, the player must complete the task within this limit. This limit is one of the main parameters that designers can use to adjust the difficulty of a level, but since the game ends when this limit is reached, we cannot observe how many moves the player would have needed to complete the level if they fail. However, if we can model how this distribution of moves that players require to complete the level, it allows level designers to see beyond this limit and estimate how adjustments in the move limit ultimately affect difficulty.

3.1.1 Background

The level designers in Tactile Games usually use a simple rule-of-thumb heuristic that a change of the move limit by 1 changes the pass rate by 1.5 to 2.0%. However, there is data available about how many moves players spend to complete the level, which should allow a more data-driven approach.

The work by Kristensen et al. (2021) fills this gap by applying a more mathematical and theoretical approach to model players and difficulty. The study uses a statistical approach to answer the question of how many moves players require to complete a level, given that we can only observe up until the move limit. By using a statistical description of the player behaviour, we are able to estimate how the move distribution would look like if there were no move limit and subsequently also estimate the pass rate if there were a move limit, paving the way for possible ways to adjust the difficulty dynamically.

This way of modelling player behaviour on a play-by-play basis is one of the first steps for modelling difficulty. Subsequent works (Kristensen et al., 2022; Kristensen and Burelli, 2022) focus on predicting how many attempts a specific player requires to complete a given level. In order to understand how this attempt distribution looks for a given level and interpret the predictions, the results from this study provide a natural probabilistic explanation of the behaviour and facilitate understanding other observations regarding the nature of the data. The results, therefore, provide the basic building block of how our definition of difficulty can be observed and operationalised in practice.

3.1.2 Results and conclusion

A plausible distribution that could describe the move distribution is a negative binomial distribution. The distribution is described by two parameters and expresses the number of trials we observe before n failures occur, with the probability p of failure, although it should be noted that n can be any positive real number. By fitting a negative binomial distribution to the move distributions from 4000 available levels from the live commercial puzzle game, Lily's Garden, we find that in 85% of the levels, there is a good fit between the theoretical model and data. This strong result demonstrates how we can model players on this granular level.

An observation from the experiments also shows that by excluding attempts where booster items are used (e.g. additional moves or free actions that help the player finish the level goals), the move distribution is better described by the proposed negative binomial distribution. One reason is that when such attempts are included, there is a consistent spike in attempts that finished two moves before the move limit and a consistent dip in attempts on the last move. We interpret this as players trying to maximise their score in a sort of "photo finish" and/or following a safe strategy on the second-to-last move to ensure they complete the level. This observed behaviour has been important for understanding the shape of the attempt distribution in Paper 2 (Kristensen et al., 2022) and Paper 5 (Kristensen and Burelli, 2022). In those studies, we observe an increased number of players that complete levels with just one attempt compared to a more theoretical expectation.

A caveat to the method proposed in the paper is that it underestimates the difficulty of hard levels. This may be due to the move limit distribution not following a pure negative binomial distribution, which will require additional modelling and analysis of the players to account for. Additionally, the data is cleaned for attempts that used booster items, which account for around 15% of the playthroughs. This also changes the expected pass rate since only attempts that did not use any help – which are expected to fail more often – are included in the analysis.

The discrepancy between estimated and actual difficulty makes the method hard to use

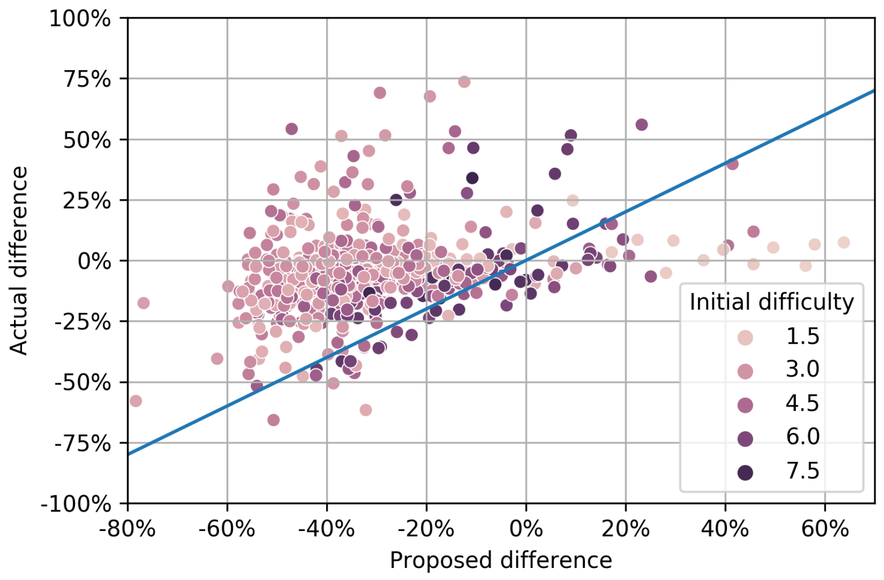


Figure 3.1: Scatterplot of how the difficulty, measured by the average attempts per complete, changed compared to the expected change. The blue line shows where the two metrics are equal.

in practice. Instead, a modified method has been used in practice where the percentage-wise expected change is used to suggest how many moves to add or remove to the move limit. The results of an A/B test that has the purpose of testing out the method for 300 levels are shown in Fig. 3.1, where it can be seen that the actual change in difficulty is not so large as expected. However, this is based on preliminary work and requires additional tuning to be used in practice.

Another limitation of this work is that it only considers players on an aggregate level and does not consider that players may learn new strategies during gameplay. From subsequent analyses, it was clear that the move distribution is not constant across the attempt number: when only looking at data from the first attempt of players, the mean and variance of the distribution were consistently lower when compared to the subsequent attempts. This suggests that the players who typically finish a level on their first attempt are more skilled than players who typically require more attempts.

Despite the limitations, an advantage of this approach is that rather than analytically breaking down the challenges into smaller components (e.g. Van Kreveld et al. (2015)) or employing specific domain knowledge (e.g. (Mourato et al., 2014)), this approach is game agnostic. The assumptions that lead to the proposed statistical distribution for capturing player behaviour are clearly defined, and they may apply to puzzle games and any other game with time- or action-limited challenges. The strength of this study comes from not just validating the method on a large sample of players from a live commercial puzzle game but also the generalisability to other games, which can help game designers better understand their player base.

3.1.3 Relevant paper(s)

The results and discussion are mainly based on Paper 1, *Statistical Modelling of Level Difficulty in Puzzle Games* (Kristensen et al., 2021). The paper was submitted to the 2021 IEEE Conference on Games as a full-length paper and has been accepted and presented. It is a part of a collaboration between the IT University of Copenhagen and Tactile Games and has been co-authored with the principal company supervisor, Arturo Valdivia, and the principal university supervisor, Paolo Burelli. I am responsible for conducting the experiments and authoring the introduction and related work sections in the paper. The methods section was mainly authored by Arturo Valdivia, and the results, discussion and conclusion sections were split between Arturo Valdivia and me. All co-authors have been involved in the discussion and editing of the final structure of the paper.

3.2 Personalised predictions

Due to the measured difficulty ultimately depending on the interaction between the players and the content, the difficulty can change over time depending on the player cohorts. This can lead to degraded performances over time for methods such as the one explored in Section 3.1. Building a more robust difficulty modelling method, therefore, involves taking individual player differences into account, which we will focus on in this section.

3.2.1 Background

The starting point of this research is determining which methods are viable for estimating a player's perceived difficulty of a level. A common approach for this purpose is to utilise DDA, which can significantly positively affect player enjoyment (Alexander et al., 2013). However, due to technical limitations and design choices, adjusting the game content on a game round-to-game round basis is not often feasible, as is commonly the goal for other DDA approaches (for a review, see Zohaib (2018)).

Kristensen et al. (2022) propose a method for personalised predictions based on Factorisation Machines (FM) (Rendle, 2012). This method is known from recommendation algorithms, where it is common to have sparse data in terms of user-item interactions. The model learns latent descriptions of players and levels, which can be used for predicting an individual player's perceived difficulty on a level but also provides interpretable model parameters that can be used for further modelling and other personalised and commercial applications by game designers.

The work presented in Section 3.1 considers how the probability of winning in a level depends on the move limit. This paper extends that line of thought of viewing the problem as a stochastic process. Given the assumption that the probability of winning is not the same for every player, another way to infer the individual pass rate is instead building a model that estimates the number of attempts a given player is expected to spend on a level, similar to the description of user skill and level difficulty by Aponte et al. (2011b). For this purpose, we rely on *Factorization Machines* (FM).

3.2.2 Results and conclusion

The paper seeks to answer three research questions related to the performance and interpretation of the FM method. The goals are predicting the number of attempts a player will spend on levels they have not yet encountered as well as how early on in a player's lifetime it is possible to differentiate the player from the average player. A baseline is provided for each level calculated as the average number of attempts of the players in the training data on the given level.

The results show that it generally requires between 10 and 30 observations of a player to predict better than the baseline on the levels they have not yet encountered. By using additional data about the player (e.g., average performance on previous levels and the use of booster items) and the level (e.g., specific mechanics), the predictive power of the models can be enhanced with fewer observations. This also enables using a random forest predictor, which works better than the FM approach with fewer observations. However, once more than 100 observations of a player are available, the FM method is more accurate.

A strength of using the FM for predicting the perceived difficulty is that it breaks down the difficulty into a player-level interaction. We find that the learned latent factors that describe this interaction between players and levels are closely related to player skill and level variance, which allows the learned model parameters to be used by the game designers for other types of personalised content, such as customised in-game offers or levels. Therefore, the suggested framework offers both the possibility to estimate and operationalise the perceived difficulty and also provide additional utility for the game designers, including the possibility of dynamic difficulty adjustments or timely intervention for players that are likely to get stuck.

One of the main limitations of the FM approach is that FMs suffer from a cold start problem. That is, if there are no observations of a player or level, it is impossible to learn the latent factors for that given entity. Without any other included information about the players or levels, the method is unable to predict the perceived difficulty for new players or on new levels. One way to overcome this is by including additional information about the players and levels, but further research is necessary to determine what type of information is helpful for these cold start predictions. This is explored in another paper included in this dissertation (Paper 5, Chapter 5).

3.2.3 Relevant paper(s)

The results and discussion are mainly based on Paper 2, *Personalized Game Difficulty Prediction Using Factorization Machines* (Kristensen et al., 2022). The paper has been submitted and accepted at UIST 2022. The presentation at the conference is expected to take place between 31st September 2022 and 2nd November 2022. It has been co-authored with three other people; Perttu Hämäläinen and Christian Guckelsberger from Aalto University and the principal university supervisor, Paolo Burelli, from the IT University of Copenhagen. I am responsible for conducting the experiments and contributing to most parts of the text except most of the introduction, which was mainly authored by Perttu Hämälåainen, and Section 2.1 on game difficulty, which was mainly authored by Christian Guckelsberger. All co-authors have contributed to different parts of the text and been involved in the discussion and editing of the final structure of the paper.

3.3 Paper 1: *Statistical Modelling of Level Difficulty in Puzzle Games*

Statistical Modelling of Level Difficulty in Puzzle Games

Jeppe Theiss Kristensen
IT University of Copenhagen/Tactile Games
Copenhagen, Denmark
jetk@itu.dk

Arturo Valdivia*
Tactile Games
Copenhagen, Denmark
arturo@valdivia.xyz

Paolo Burelli
IT University of Copenhagen/Tactile Games
Copenhagen, Denmark
pabu@itu.dk

***Abstract*—Successful and accurate modelling of level difficulty is a fundamental component of the operationalisation of player experience as difficulty is one of the most important and commonly used signals for content design and adaptation. In games that feature intermediate milestones, such as completable areas or levels, difficulty is often defined by the probability of completion or completion rate; however, this operationalisation is limited in that it does not describe the behaviour of the player within the area.**

In this research work, we formalise a model of level difficulty for puzzle games that goes beyond the classical probability of success. We accomplish this by describing the distribution of actions performed within a game level using a parametric statistical model thus creating a richer descriptor of difficulty. The model is fitted and evaluated on a dataset collected from the game Lily's Garden by Tactile Games, and the results of the evaluation show that the it is able to describe and explain difficulty in a vast majority of the levels.

***Index Terms*—player modelling, difficulty modelling, game design, dda, survival analysis**

I. Introduction

A central aspect of game design is difficulty and its effect on player experience – too easy and players are not sufficiently engaged; too hard and players become frustrated, causing them to quit the game. In games consisting of discrete tasks or levels, a common way to manage the difficulty is by controlling the resources available to the player to complete such task or level – *e.g.*, number of actions or time available to solve a puzzle. Balancing the correct number of resources available in the level to obtain a desired difficulty is a complex task that often relies on the ability of the designer to relate an abstract descriptor of difficulty to the behaviour of the players and the controllable components in the level.

For example, in the case of puzzle games that provide players with limited actions, or *moves*, to complete each level, such as match-3 or bubble shooter style games, a direct way to describe the difficulty is by measuring how many attempts it takes players on average to complete a level. This quantity is commonly referred to as *attempts-to-complete*, and its multiplicative inverse is what we call *completion rate*. This definition is useful for identifying levels in which players may feel stuck and thus stop playing, controlling the consumption rate of game content, or even enabling different monetisation

* Corresponding author.

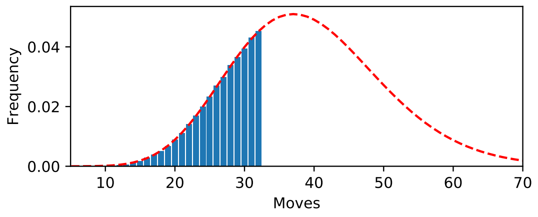


Fig. 1. Histogram over the number of actions spent by players to complete one of the levels in the data. The effect of the action limit near $M = 32$ can clearly be seen as a sharp cut-off in the distribution. If we are able to accurately estimate the full distribution (represented by the red curve), the completion rate using different action limits can be calculated.

strategies. However, such descriptor only considers the data in an aggregated way and thus lacks the granularity that may, for example, tell about the effect of changing the action limit or how close to finish a player was. This makes it relatively limited in it expressiveness, giving a designer little information on how to adjust the difficulty and thus turning the task of level adjustment into a trial-and-error procedure.

In the vast majority of currently published puzzle games, success or failure are not the only data available about the player behaviour in a game; often a summary of the actions performed and the resources used are tracked. If properly modelled, this information has the potential to be the basis of a much richer descriptor of level difficulty. In particular, the number of actions used by players in their attempts has both the benefit of describing their progress within a level and being directly related to an important level design aspect, move limit.

The number of actions used to complete a level depend on a number of factors, such as player skill, level setup and luck. This leads to a certain distribution of actions spent by players on each level (see Fig. 1). The central idea of this article is that, by modelling and understanding the nature of this action distribution, we may be able to not only evaluate the completion rate but also estimate the effect of design actions, such as changing the move limit, and gain a deeper understanding of the player challenges.

To achieve this, the model of the player behaviour needs to be both accurate and explainable. For this reason, in this research work, we have investigated the application of a parametric statistical model to represent the underlying action distribution. We discuss how this behaviour can be modelled using a negative binomial distribution and conduct an empirical study of the application of this modelling approach to a dataset from a popular mobile puzzle game – Lily's Garden by Tactile Games – and present and discuss the results of the study.

II. Related work

Flow [4] describes the psychological state where the difficulty of a task and user skill match which leads to an engaging gameplay experience. While difficulty can be broken down into multiple sub-components (e.g. cognitive, emotional, etc. [6]), in scenarios where it is necessary to operationalise difficulty, such as for dynamic difficulty adjustment or automated playtesting, it is common to use the probability of task success as an objective measure of difficulty [5], [7], [9], [13], [15], [17]. This interpretation is supported by Pedersen et al. [16] where the correlation between player emotions and level characteristics in a Super Mario Bros is investigated. Here, the biggest factor for feeling challenged was the completion rate of the levels or similar aspects of failure, such as number of deaths.

In this work we adopt a similar probabilistic definition: the difficulty of a level is given by the win probability, which empirically is the completion rate and can be computed as the number of times a given level has been completed over the total number of attempts on said level. However, while this aggregated description of difficulty as the completion rate is intuitive, it does not offer a deeper and actionable understanding of the problem, such as how imposing a time or action limit affects the completion rate or how close to finishing a player was. The nature of such data is censored since we do not have information about the complete playthrough, so to draw inspiration on how to deal with that, we can look to survival analysis [14].

Survival analysis is branch of statistics that focuses on estimating unseen, or censored, data and is commonly used to estimate a time until an event. There are multiple examples of using this approach to describe player behaviour using parametric distributions: Feng et al. [8] used a generalised Weibull distribution to model online session length, and Bauckhage et al. [2] tested various distributions, including a Weibull and Poisson-Gamma distribution, to estimate time until people lost interest in a game. A survival analysis approach has been used to describe gameplay related behaviour in [10], in which the authors investigated the operationalisation of perceived difficulty of levels in the game Flappy Bird. By using player and playtest AI data, they computed an empirical survival function, $S(x)$, which describes the distribution of attempts that reached a given length in a level. From this, the hazard function could then be used as an indicator of perceived difficulty.

The work presented in this article shares its nature with these last studies, in that we attempt to operationalise and abstract aspect of gameplay – i.e. level difficulty – using a parametric statistical distribution. The key points of departure are, that the model presented in this article is both built and evaluated on a large dataset of real player gameplay data; furthermore, we present a general framework to describe the operationalisation of difficulty, identify the appropriate distribution and evaluate its effectiveness.

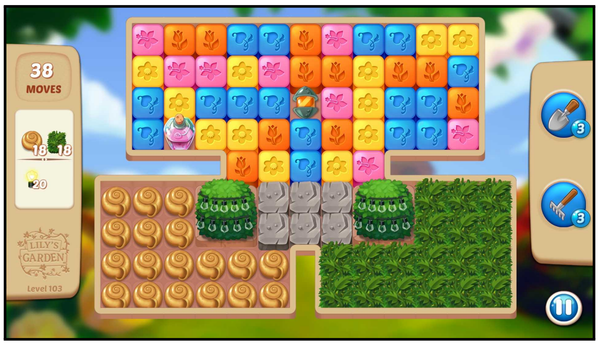


Fig. 2. An example of a level in Lily's Garden. The level goals are specified on the left side, and in-game boosters on the right side. These in-game boosters are very strong boosters that allow the player to complete the level more easily.

III. Methods

Let us start this section by briefly describing the puzzle game mechanics. Each level ℓ requires the player to collect a series of goals within a predetermined maximum number of actions, or moves, M_ℓ. Each move consists of collapsing groups of adjacent board pieces by tapping on one of them. Creating more powerful board pieces that clear a large area of the board is possible by matching groups of at least 5 board pieces at the same time. An example of a level is shown in Fig. 2.

If the player completes all of the level goals with no more than M moves, then we say that *the attempt was successful*, and the player passes to the next level. Consequently, each player can complete each level at most once. Now, if the player consumes all of the permitted number of moves M without completing the all of the level goals, then we say that *the attempt was a failure*. In this case the player can either spend a virtual currency to obtain some extra moves (*e.g.*, $+5$), or can decide to have one more attempt at the cost of a life. These lives regenerate automatically over time, and typically each player can get up to 5 of lives at any given time.

For this study we use data sample from $L = 4000$ levels which has been collected between 2020-06-01 and 2021-01-01. For each level, the available data for each attempt consist of the number of moves used and whether the attempt was successful or not. An initial data cleaning step is performed by excluding all incomplete attempts, *i.e.*, attempts which are terminated prematurely either due to a technical issue in the game, or simply because the player deliberately quits the game. We also exclude attempts using special in-game boosters which usually inflate the number of attempts finishing within $k = 0, 1, 2$ moves from the moves limit M_ℓ. The

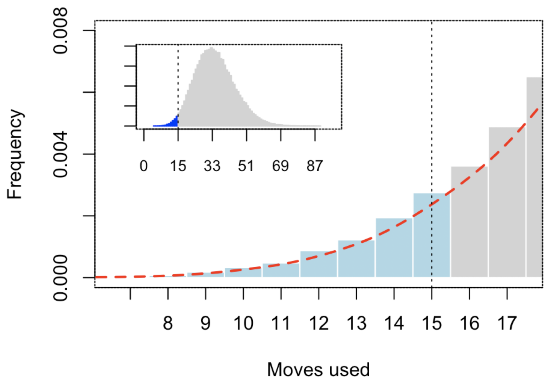


Fig. 3. Illustration of the observed frequencies of moves to complete a level. The vertical dotted line indicates that moves limit is set to $M_\ell = 15$. The fitted curve is marked with a dashed line. The subplot in the top-left suggests the almost linear growth in the observed frequencies leads to fitting left tail of the negative binomial distribution.

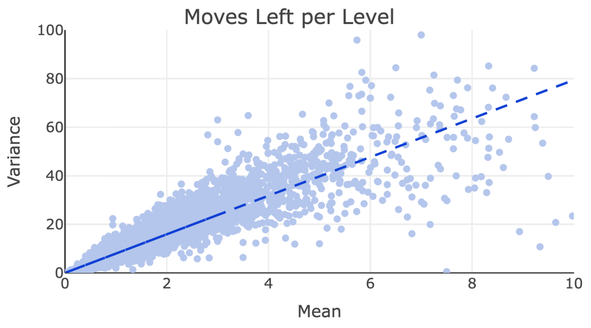


Fig. 4. Illustration of the linear relationship between the mean and the variance of number of moves left to complete level.

final input dataset consist of the frequency of moves used to complete the level (see Fig. 1) and the overall completion rate, which is defined as the percentage of successful attempts over the total number of attempts, with an average of 350,000 successful attempts per level.

The goal of the method is identifying a parametric distribution which can fit the number of moves used to complete a level to a *good degree*, *i.e.*, up the truncation point imposed by the moves limit M_ℓ^*. The fitted curve should match the observed frequencies, and the area under this curve should match the observed completion rate. As an illustration, Fig. 3 depicts the undesired situation where the fitted distribution is able to describe well the observed frequencies, but fails at matching the completion rate. We can expect this to occur for instance when the steady growth of the observed frequencies is almost linear and thus calibrated as the left tail of the distribution. These ideas are formalised below.

Remark. *Let us note here that for other types of games, the definition of the input dataset would be analogous, for instance, by interchanging the role of moves used to complete the level by the units of time taken to complete the task.*

A. Calibration of model parameters

Given a level ℓ with a move limit M_ℓ^*, let us denote by $\hat{F}_\ell$ the empirical distribution of moves used to complete the level. Let $\hat{c}_\ell$ be the observed level's completion rate, *i.e.*, the percentage of attempts that complete the level within a maximum of M_ℓ^* moves. As depicted in Fig. 1 the empirical move distribution $\hat{F}_\ell$ is truncated on the right by M_ℓ^*, but we assume that this data corresponds to a censored observation of an underlying non-truncated distribution F_ℓ. Let us assume that $\hat{F}_\ell$ and F_ℓ have probability density functions, and denote them by $\hat{f}_\ell$ and f_ℓ, respectively.

In these terms, our goal is to find a parametric model for the distribution F_ℓ, in such a way that following two conditions are met:

Condition 1. *The fitted distribution, F_ℓ, follows* closely *the empirical distribution, $\hat{F}_\ell$, all across the range $(0, M_\ell^*]$.*

Condition 2. *The quantity $F_\ell(M_\ell^*)$ approximates the observed completion rate $\hat{c}_\ell$.*

In this article we consider the Condition 2 as a validation step only; that is to say, we do not explicitly enforce this condition as part of the calibration algorithm. The rationale behind decision is that we aim at establishing here a baseline for how much can be explained by focusing only on fitting the truncated data. In other words, we are assessing the degree in which Condition 1 can ensure that Condition 2 is fulfilled as well.

Let us now describe calibration strategy for the model parameters. Given a parametric model for the distribution F_ℓ, we obtain the corresponding parameter set θ_ℓ by applying a Non-Linear Least Squares (NLLS) regression over the range $(0, M_\ell^*]$, which is were we can fully observe $\hat{F}_\ell$. Such a method requires an initial guess θ_0 of θ_ℓ as an input, which, if incorrectly chosen, may lead to a false negative due to a sub-optimal fit. In order to minimise this risk we choose the initial guess by solving the following optimisation problem:

$$\theta_0^*(\ell) := \underset{\theta_0 \in \Theta_\ell}{\arg\min}\ D(\hat{f}_\ell, f_\ell^{(\theta_0)}),$$

where Θ_ℓ denotes the search space for the initial guess θ_0; $f_\ell^{(\theta_0)}$ is the distribution we get from NLLS by using the initial guess θ_0; and D is a distance between the distributions $\hat{F}_\ell$ and $F_\ell^{(\theta_0)}$ over the range $(0, M_\ell]$. Here we shall use the Kolmogorov-Smirnov distance (see [1]) which in this case is simply given by

$$\begin{aligned} D(\hat{f}_\ell, f_\ell^{(\theta_0)}) &:= \max_{m \le M_\ell^*} \left| \hat{F}_\ell(m) - F_\ell^{(\theta_0)}(m) \right| \\ &= \max_{m \le M_\ell^*} \left| \sum_{m' \le m} \left(\hat{f}_\ell(m') - f_\ell^{(\theta_0)}(m') \right) \right|. \quad (1) \end{aligned}$$

Notice that in these terms Condition 1 can be rewritten as $D(\hat{f}_\ell, f_\ell^{(\theta_0^*(\ell))}) < \delta$, for a small enough δ, say 5%.

B. Requirements for the underlying parametric distribution

Our target distribution (*i.e.*, moves used to complete the level) takes only non-negative integer values. Consequently, in order to fit a parametric model we can use a non-negative integer-valued distribution (*e.g.*, negative binomial) or, alternatively, work with a discretization of a non-negative continuous distribution (*e.g.*, the gamma distribution).

In order to delimit the list of potential distributions we could use for our analysis, we start by looking at the pattern depicted by Fig. 4 which suggests that there is a strong linear relationship between the mean and the variance of number of moves left to complete level. More precisely: Let $M_\ell(n, i)$ be the number of moves left at the end of the n-th attempt of the i-th player to pass level ℓ. Each point of this graph corresponds to one of the levels $\ell = 1, ..., L$ in our sample ($L = 4000$), and the coordinates x and y axis equal the mean, μ_ℓ, and the variance, σ_ℓ, of $M_\ell(n, i)$, respectively, where n and i vary over all of the attempts that took place during the observation period. The dashed line shows the result of performing a linear regression of σ_ℓ^2 with respect to μ_ℓ with no intercept – *i.e.*, we consider a model of the form $\sigma_\ell^2 \approx \psi\mu_\ell$. The goodness of this fit (*i.e.*, $R^2 \approx 85\%$, p-value $< 10^{-16}$) suggests the aforementioned strong linear relationship between the mean μ_ℓ and the variance σ_ℓ^2 of M_ℓ. Further it implies a necessarily condition that our parametric model for M_ℓ should satisfy.

C. Negative binomial distribution as a baseline

Based on the above, is clear that the most natural non-trivial starting point is to consider a negative binomial distribution since it is a well-known non-negative integer-valued distribution exhibiting a linear relation between its mean and variance:

$$f_\ell(m) := \binom{m+n-1}{m}(1-p)^n p^m, \quad \text{for } m = 0, 1, 2, ...$$

As for the search space for the initial guess we shall use $\Theta_\ell := [1, 10M_\ell] \times [0.001, 0.999]$.

Two remarks are in order here: first of all, note that the negative binomial distribution is also referred to as the *Poisson-gamma distribution* since it is equivalent to a Poisson distribution with intensity parameter λ where the λ itself is allowed to be random by following a gamma distribution. Second, a more sophisticated approach would be to work with a discretization of a Tweedie distribution for which it is well-known that $\sigma_\ell^2 = \psi\mu_\ell^p$, or even a Poisson-Tweedie distribution for which $\sigma_\ell^2 = \mu_\ell + \psi\mu_\ell^p$ [3], [11]. However, we let this investigation for future work since our initial exploration (see Fig. 4) suggests that considering a *dispersion parameter* of $p = 1$ could provide already a very good starting point.

IV. Results

To estimate the validity of our approach, we tested it on 4000 levels from the puzzle game Lily's Garden: first, we analyse the overall results of fitted distribution parameters on all of the levels. In a second step, based the conditions described in the previous section regarding the fitted distributions, we discuss the goodness of the fit of the resulting model, thus evaluating the ability of the model to describe the player behaviour. Lastly, we validate whether the model is able to describe the levels' canonical definition of difficulty – *i.e.* completion probability – as well as the aforementioned behaviour.

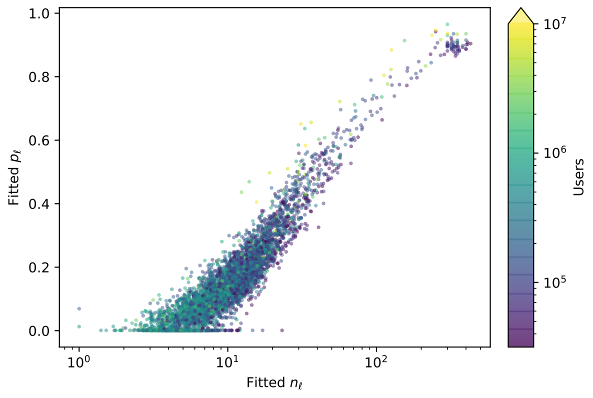


Fig. 5. Log-linear plot of the fitted parameters p and n for each level. The color indicates the number of users that have played the given level.

A. Distribution parameters

Figure 5 shows the fitted parameters obtained from the execution of the algorithm on $L = 4000$ levels from the puzzle game Lily's Garden. Each of these points represent the parameters (n_ℓ, p_ℓ) of a negative binomial model fitted to the distribution of moves used to complete each level $\ell = 1, 2, ..., L$. It can be seen that the majority (*i.e.*, 83%) of the levels fall within a central cluster defined by $0.001 < p_\ell \le 1$ and $1 \le n_\ell \le 200$. For this central cluster it is apparent that the parameters (n_ℓ, p_ℓ) follow a log-linear relationship $\log(n_\ell) = ap_\ell + b$ relationship ($R^2 = 87\%$), where a and b are global constants not depending on the level. This indicates that the level's move distribution can possibly be driven by a single parameter, which would enable level designers to easily compare levels to one another.

For this purpose, the so-called *scale parameter* (ϑ_ℓ) could be considered, which describes the spread of a distribution – *i.e.*, the larger the scale parameter, the more spread out the distribution. This numerical parameter is often considered in the context of a parametric family of probability distributions, and in the case of negative binomial distributions it is given by this simple expression

$$\vartheta_\ell := \frac{1 - p_\ell}{p_\ell}.$$

Notice that from this expression we can derive the (n_ℓ, p_ℓ) as

$$p_\ell = \frac{1}{1 + \vartheta_\ell}, \quad \text{and} \quad n_\ell = \exp\left(a\left(\frac{1}{1 + \vartheta_\ell}\right) + b\right).$$

There are also two other notable clusters. The first of these clusters is defined by $p_\ell = 0.001$ and consists of 15% of

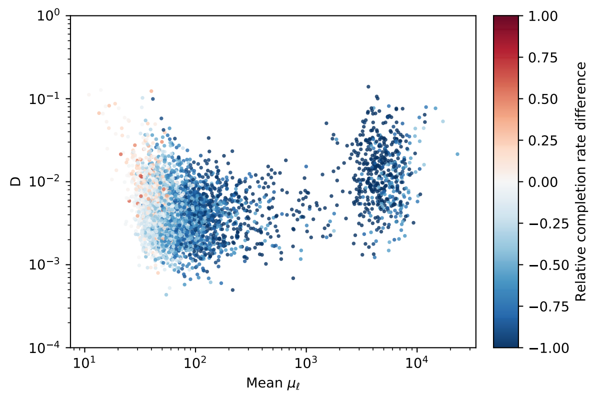


Fig. 6. Log-log plot of the Kolmogorov-Smirnov test statistic D and the mean of the fitted distributions. The colours show the relative difference between the expected and actual completion rate.

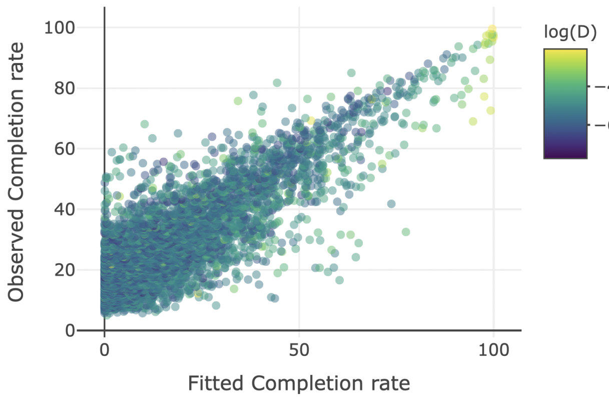


Fig. 7. Comparison between the observed completion rates and the fits obtained from the calibration algorithm.

sampled levels. The common feature for all instances in this cluster it that the parameter fitting threshold had been reached, which will be explored in more detail in Section IV-B. The second cluster is defined by $n_\ell > 200$ and consists of 2% of the levels in our sample. Inspecting the instances in this – high n_ℓ, high p_ℓ – cluster, we encountered either tutorial levels or levels with a specific type of game mechanic that channels the players to rather restrictive type of game play.

It is worth noting that, by design, the tutorial levels tend to exhibit a lower variance than the rest of the levels, either by fixing the random seed or overall layout and ideal strategy of the level. This reduced dependence on randomness may therefore also lead to a move distribution with smaller variance. In the same manner, we have observed that the levels containing the channelling mechanic that restricts gameplay lead to a less random play experience. Such information may be particularly useful to level designers since creating levels where the chance of winning is completely determined by chance removes any agency from players and are potentially not very fun to play. Being able to identify such levels can therefore provide a more quantitative measure of level randomness.

B. *Condition 1 and validity of fits*

The initial condition laid out in Section III-A states that the fitted distribution F_ℓ should closely follow the empirical distribution $\hat{F}_\ell$. To determine whether this is true, we use the Kolmogorov-Smirnov distance D as defined by Eq. (1). To give an overview of the link between the distribution parameters and D, Fig. 6 plots D against the mean ($\mu_\ell = n_\ell \frac{1-p_\ell}{p_\ell}$) of the fitted distribution, and coloured by the relative difference $(c_\ell - \hat{c}_\ell)/\hat{c}_\ell$. What we find is that 99% of the levels satisfy $D < 5\%$, meaning that the fitted distributions describe the empirical data very well in many cases and thus fulfil **Condition 1**.

One thing to note is that in some cases, the parameter boundaries were reached during the fitting process. This was observed to happen in around 15% of the levels and typically lead to $p_\ell = 0.001$. These levels appear in the right-most cluster in Fig. 6 and are defined by $\mu_\ell > 10^3$. This was typically observed to happen when the empirical move distribution only exhibited a steadily increasing trend, leading to instances where only using the tail of the distribution would best describe this simple behaviour. We consider those examples bad fits due to the method not converging and exclude them for the rest of the analysis in the next section.

Before moving on to the next part of the analysis, we first attempt to isolate what differentiates the levels that show a good fit from the other ones. Specifically, we first investigate whether different game mechanics influence the move distribution. For this purpose, we use a logistic regression to model whether the level fit converged or not in order to estimate the impact of specific board pieces. The results indicate that timing mechanics generally lead to a better fit while one specific spawning mechanic (*i.e.*, the collect goals first appear after interacting with the spawner) lead to a worse fit.

One thing that is worth noting is that the data used for this analysis disregarded attempts that used various in-game help items (*i.e.*, extra moves, boosters, etc). If a player finds a level to be difficult or frustrating, subsequent attempts by the player may be disregarded because they use helping items, distorting the move distribution. A number of observations support this hypothesis: When only considering the move distribution of the second attempt of players, the fraction of levels that successfully converged increased by about +5%. Additionally, the fraction of attempts in which players used in-game boosters and help items were up to +18% more frequent in non-converging examples than convergent ones; thus, more attempts are ignored on average for non-converging examples. In our data processing step, these attempts were filtered out because they exhibited a clear artificial alteration of the curve, especially in the last two moves of the levels. Part of the explanation for the divergent fits can therefore also be related to the data.

C. *Condition 2: completion rate comparison*

The second condition states that the expected completion rate, $\hat{F}_\ell(M_\ell^*)$, should approximate the observed completion

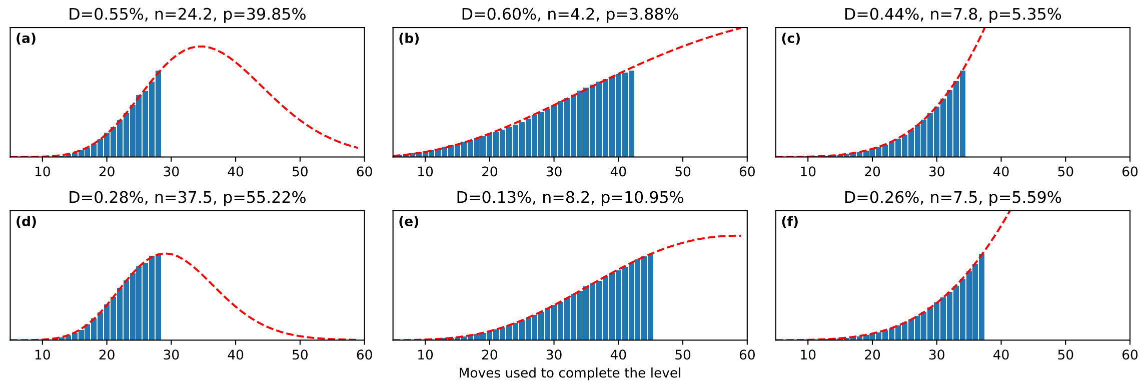


Fig. 8. Subplots in the top (**a-c**) and bottom rows (**d-f**) correspond to instances when the observed completion rate $\hat{c}_\ell \approx 20\%$ and $\hat{c}_\ell \approx 40\%$, respectively. The first, second and third columns exemplify cases we found a good (**a** and **d**), medium (**b** and **e**) and low (**c** and **f**) agreement between the observed and fitted completion rates, respectively.

rate, $\hat{c}_\ell$. In order to assess this condition, we first notice that the two values are strongly correlated as exhibited by their Pearson's correlation coefficient of $\rho = 83\%$. Further, Fig. 7 suggests that the observed and fitted completion rates are related to each other by means of the linear relationship

$$c_\ell \approx 1.035\hat{c}_\ell - 0.104 \tag{2}$$

with an adjusted coefficient of determination of $R^2 = 75\%$. Equation (2) suggests that the completion rates tend to be underestimated, especially at low completion rates (*i.e.*, for very hard levels where the average player will need the equivalent of 8 or more attempts are needed to complete the level). Based on these arguments we can consider the **Condition 2** has been met as well.

In practice level designers typically work with ranges of the completion rate rather than point estimates, so that they can classify the levels in classes (*e.g.*, "easy", "very hard"). Consequently, the current results are positive and very promising. One could however also look at point estimates of the completion rates, for instance under the light of the *absolute percentage error* given by $\varepsilon_\ell := |c_\ell/\hat{c}_\ell - 1|$. By doing this, we have observed that the median value of ε_ℓ revolves around the 49%, and it goes down to 23% when adjusting according to Equation (2).

In order to get a better understanding on what leads to the aforementioned underestimation (*i.e.*, cases where $c_\ell < \hat{c}_\ell$), we exemplify in Fig. 8 what happens in a series of scenarios: Scenario 1, as illustrated by subplots **a** and **d**, corresponds to cases where the relative error between $\hat{c}_\ell$ and c_ℓ is small. Scenario 2 in subplots **b** and **e** show cases where error is medium. And finally Scenario 3 in subplots **c** and **f** corresponds cases with a major underestimation. In Scenario 3, it can be seen that it is only the tail of the distribution that is used to describe the data. A similar phenomenon was also observed in the cases where the fitting method did not converge: Due to the available player data and steady increase in completions, only the tail is required to describe this relatively simple behaviour. However, contrary to those cases, these levels are in more of a continuum: It is more likely to underestimate at low completion rates where more data is censored, while for higher values of $\hat{c}_\ell$ (like in Scenario 1 and Scenario 2) we have more information about the distribution is available which further constrains f_ℓ.

In order to see if there are any specific game mechanics that may cause a difference between the completion rates, a similar method as section IV-B is used. Instead of using a logistic regression for predicting whether it was a good fit or not, a linear regression is used to predict the difference between expected and actual completion rate. The results are similar to the findings in the previous section regarding successful fitting: Levels with timing or other gameplay restrictive mechanics lead to a higher expected completion rate. Interestingly, board pieces with colour-matching mechanics tend to lead to too low expected completion rates. A way to possibly interpret this is that goals which can be completed at a steady pace (such as colour mechanics) lead to a more steadily increasing ramp-like distribution, leading to completely underestimating the completion rate due to more degrees of freedom in the fitting. Timing mechanics, on the other hand, may require more planning that appear as a more constrained minimum number of moves spent which leads more defined distribution around a given move and less variance that may be detrimental to the modelling method. That said, there are additional factors not considered (such as level topology), so more work is required to establish any link between the completion rate difference and game mechanic.

V. Discussion and Future Work

In 85% of the levels we are able to find a negative binomial distribution that describes the player data well. Additionally, we are able to derive estimations of different game play features, such as level randomness and board piece descriptors, that can give additional insights to the game designers. That said, there are still some open questions to address about the current approach related to the modelling and possible use-cases, which will be discussed in this section.

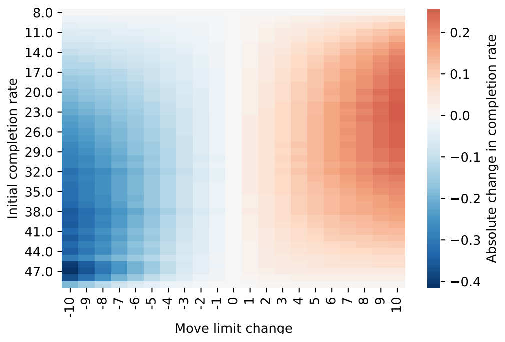


Fig. 9. 2D plot of how the completion rate is expected to change depending on the initial completion rate and the change in moves limit. The initial completion rates are binned in bins spanning 2%, and the new expected completion rate is adjusted by the trend from Fig. 7.

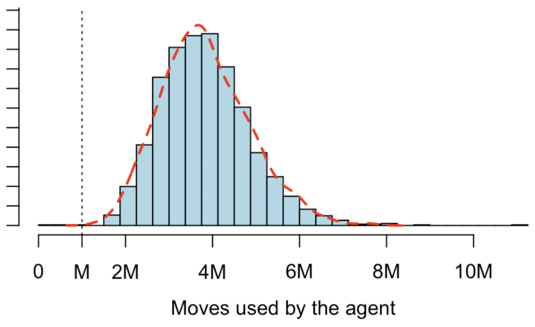


Fig. 10. Histogram of the moves used by a certain AI agent to complete a specific Lily's Garden. The dashed line represents the negative binomial fit.

A. *Changing the move limit*

One of the discussed use-cases of modelling these distributions is that level designers can estimate what changing the move limit would mean for the completion rate. To examine how the completion rate is affected by changes in the move limit, Fig. 9 shows how the predicted absolute change in expected completion rate depends on the initial completion rate and change in move limit. As a rule of thumb, the completion rate seems to change on average by 2% with slightly lower sensitivity at high or low completion rates. From discussions with level designers, this is consistent with their commonly used heuristic.

Another insight is that adding or removing moves is an asymmetric operation, where the rate of change is bigger when removing moves. While this is also expected since the negative binomial distribution itself can be asymmetric and can potentially have a long right tail (and thus less sensitive to adding moves), it suggests that game designers need to be more careful when removing time or actions to increase difficulty because of this asymmetric change.

One possible limitation of this argument is that it assumes that the distribution parameters will stay the same if the move limit changes. However, this is not necessarily true since players may change their behaviour when closer to the move limit. For instance, a common strategy is to set up powerful board piece combinations and fire them off in the end to maximise the score (regardless of whether there is an explicit score or not).

B. *Player skill*

The perceived level difficulty depends not just on level randomness but also the player skill. So far the level randomness was linked to the variance of the fitted distributions, but logically the move distributions should also be affected by the skills that have played the level. Indeed, this phenomenon is something that level designers experience in their day to day work: As more players reach older levels, the completion rate slowly changes, which makes it necessary to have a constant maintenance of all levels.

As a next step, investigating how the level difficulty changes over time in a longitudinal study using different player cohorts may provide meaningful insights on player skill and also model how this affects the distribution parameters. This can then be used for a more proactive and automatic approach to difficulty adjustment that ensures a coherent play experience for both old and new users.

C. *Playtesting*

Playtesting is crucial for game developers since this process provides a reliable way to identify bugs and potential design flaws in a safe environment before going to market. This process, however, tends to be so expensive and slow that game developers are increasingly starting to automate this by means of AI agents using, for instance, reinforcement learning techniques (*e.g.* [12] and references therein). This context also provides an interesting set-up to gain deeper insight into the techniques derived in the present article and further potential applications.

Indeed, given that playtest agents are trained in the same environment as human players, we can for instance let the agent play using all of the normal rules and game mechanics but without the constraint on the move limit, M_ℓ. Figure 10 shows the distribution of moves generated by one of the playtest agents considered in [13] when testing a given level. This particular agent performed sub-par with respect to the average human player, but what is relevant is that we are able to visualise its whole move distribution even beyond the limit M. We can thus fit our proposed negative binomial distribution across the whole $(0, 10M]$ range, *i.e.*, without truncation.

In line with our expectations, we get a really good fit as described by a Kolmogorov-Smirnov distance of $D = 1.8\%$. This sparks the following question regarding playtest agents: *Can exhibiting a negative binomial distribution be regarded as a necessary condition to declare that the AI agent is playing in a human-like manner?*

Finally we highlight another connection with the results reported in [13]. In that work the authors report that the 5% best runs of the agent on a given level were the strongest predictor of the actual completion rate. Clearly this 5^{th} percentile is a quantity that can be derived explicitly as formula of the fitted negative binomial parameters. In this sense we can also study whether the fitting procedure proposed here can be further used as post-processing strategy to estimate completion rates from data generated by playtest agents with sub- or even super-human performance.

D. Other games

In this work we have investigated the application of the proposed method to a mobile puzzle game; however, there is nothing in our assumptions that rules out that the same distribution can be used not just for similar puzzle games with discrete moves and action limit but also other genres such as platform or even competitive games.

Generally, puzzle games tend to be very focused on solving the level goal as fast as possible. Although some games also provide a score, it is a limited number of factors (randomness and skill) that affect the distribution of moves. However, in other game genres, there may be other factors and incentives for playing: in platform games, players are encouraged to explore and test out different strategies, and in competitive games, players may want to beat their opponent as fast as possible, with randomness playing less of a role than the relative skill of players. A promising venue for future research is therefore using this modelling approach across genres to test and validate its generalisability to different player behaviours.

VI. Conclusion

In this research work we set out to determine a richer way of describing the level difficulty in puzzle games. Specifically, we propose that the move frequency distribution of the players for completing a level follows a negative binomial. Using data from 4000 levels from the game Lily's Garden as a case study, the results showed that:

- The negative binomial is able to describe the move distribution of around 85% of the levels, and the method can easily be extended to other types of games.
- Describing the levels is possible using a single parameter – that is the scale parameter ϑ – that describes the spread of the distribution.
- This more detailed description of the difficulty enables: (*i*) estimating the effect of changing the move limit; (*ii*) estimating the level randomness; and (*iii*) identifying deviations in player behaviour on a level.

In the remaining $\sim 15\%$ of the cases where the method does not converge; the main issue is due to the data only exhibiting an increasing trend which leads to the method only using a very small part of the distribution to match it. Similarly, the method also tends to underestimate the observed completion rate, $\hat{c_\ell}$, especially towards low completion rates. A possible avenue for future research is therefore to extend on this model and include $\hat{c_\ell}$ as a parameter in the modelling rather than a constraint. This has the promise of not only improving the predictions of the method but also ultimately enable estimating player skill and dynamically adjust difficulty to ensure an optimal player experience.

VII. Acknowledgements

This work has been supported by the Innovation Fund Denmark and Tactile Games. We also thank Arnau Escapa and Rasmus Berg Palm for fruitful discussions.

References

[1] Kolmogorov-Smirnov test - Encyclopedia of Mathematics.

[2] Christian Bauckhage, Kristian Kersting, Rafet Sifa, Christian Thurau, Anders Drachen, and Alessandro Canossa. How players lose interest in playing a game: An empirical study based on distributions of total playing times. In *2012 IEEE Conference on Computational Intelligence and Games, CIG 2012*, pages 139–146, 2012.

[3] Wagner H. Bonat, Bent Jørgensen, Célestin C. Kokonendji, John Hinde, and Clarice G. B. Demétrio. Extended poisson–tweedie: Properties and regression models for count data. *Statistical Modelling*, 18(1):24–49, 2018.

[4] Mihaly Csikszentmihalyi and Mihaly Csikzentmihaly. *Flow: The psychology of optimal experience*, volume 1990. Harper & Row New York, 1990.

[5] Simon Demediuk, Marco Tamassia, William L. Raffe, Fabio Zambetta, Xiaodong Li, and Florian Mueller. Monte Carlo tree search based algorithms for dynamic difficulty adjustment. In *2017 IEEE Conference on Computational Intelligence and Games, CIG 2017*, pages 53–59. Institute of Electrical and Electronics Engineers Inc., 10 2017.

[6] Alena Denisova, Paul Cairns, Christian Guckelsberger, and David Zendle. Measuring perceived challenge in digital games: Development & validation of the challenge originating from recent gameplay interaction scale (corgis). *International Journal of Human-Computer Studies*, 137:102383, 2020.

[7] Miguel González Duque, Rasmus Berg Palm, David Ha, and Sebastian Risi. Finding Game Levels with the Right Difficulty in a Few Trials through Intelligent Trial-and-Error, 1 2020.

[8] Wu-chang Feng, Francis Chang, Wu-chi Feng, and Jonathan Walpole. A traffic characterization of popular on-line games. *IEEE/ACM Transactions On Networking*, 13(3):488–500, 2005.

[9] Stefan Freyr Gudmundsson, Philipp Eisen, Erik Poromaa, Alex Nodet, Sami Purmonen, Bartlomiej Kozakowski, Richard Meurling, and Lele Cao. Human-like playtesting with deep learning. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–8. IEEE, 2018.

[10] Aaron Isaksen, Dan Gopstein, Julian Togelius, and Andy Nealen. Exploring Game Space of Minimal Action Games via Parameter Tuning and Survival Analysis. *IEEE TRANSACTIONS ON GAMES*, 10(2), 2018.

[11] B. Jørgensen. *The Theory of Dispersion Models*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis, 1997.

[12] Jeppe Theiss Kristensen and Paolo Burelli. Strategies for using proximal policy optimization in mobile puzzle games. In *International Conference on the Foundations of Digital Games*, pages 1–10, 2020.

[13] Jeppe Theiss Kristensen, Arturo Valdivia, and Paolo Burelli. Estimating player completion rate in mobile puzzle games using reinforcement learning. In *2020 IEEE Conference on Games (CoG)*, pages 636–639. IEEE, 2020.

[14] Elisa T Lee and John Wang. *Statistical methods for survival data analysis*, volume 476. John Wiley & Sons, 2003.

[15] J. Derek Lomas, Kenneth Koedinger, Nirmal Patel, Sharan Shodhan, Nikhil Poonwala, and Jodi L. Forlizzi. Is difficulty overrated? the effects of choice, novelty and suspense on intrinsic motivation in educational games. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, page 1028–1039, New York, NY, USA, 2017. Association for Computing Machinery.

[16] Christopher Pedersen, Julian Togelius, and Georgios N Yannakakis. Modeling player experience for content creation. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(1):54–67, 2010.

[17] Su Xue, Meng Wu, John Kolen, Navid Aghdaie, and Kazi A. Zaman. Dynamic difficulty adjustment for maximized engagement in digital games. In *Proceedings of the 26th International Conference on World Wide Web Companion*, WWW '17 Companion, page 465–471, Republic and Canton of Geneva, CHE, 2017. International World Wide Web Conferences Steering Committee.

3.4 Paper 2: *Personalized Game Difficulty Prediction Using Factorization Machines*

Personalized Game Difficulty Prediction Using Factorization Machines

Jeppe Theiss Kristensen
jetk@itu.dk
IT University of Copenhagen
Digital Design
Copenhagen, Denmark

Christian Guckelsberger
christian.guckelsberger@aalto.fi
Aalto University
Department of Computer Science
Espoo, Finland

Paolo Burelli
pabu@itu.dk
IT University of Copenhagen
Digital Design
Copenhagen, Denmark

Perttu Hämäläinen
perttu.hamalainen@aalto.fi
Aalto University
Espoo, Finland

ABSTRACT

The accurate and personalized estimation of task difficulty provides many opportunities for optimizing user experience. However, user diversity makes such difficulty estimation hard, in that empirical measurements from some user sample do not necessarily generalize to others.

In this paper, we contribute a new approach for personalized difficulty estimation of game levels, borrowing methods from content recommendation. Using factorization machines (FM) on a large dataset from a commercial puzzle game, we are able to predict difficulty as the number of attempts a player requires to pass future game levels, based on observed attempt counts from earlier levels and levels played by others. In addition to performance and scalability, FMs offer the benefit that the learned latent variable model can be used to study the characteristics of both players and game levels that contribute to difficulty. We compare the approach to a simple non-personalized baseline and a personalized prediction using Random Forests. Our results suggest that FMs are a promising tool enabling game designers to both optimize player experience and learn more about their players and the game.

CCS CONCEPTS

• **Human-centered computing → User models**; *Human computer interaction (HCI).*

KEYWORDS

Factorization Machines, games, player modelling

1 INTRODUCTION

Understanding and estimating task difficulty is a fundamental problem in Human-Computer Interaction (HCI). While good user interface design aims to minimize the difficulty of completing tasks and achieving goals, we might also be interested in introducing challenges of just the right difficulty – neither too easy nor too hard – to e.g. support learning [54] or create enjoyable video games [8, 49]. Estimating how difficult a challenge is for the target user is hard due to the diversity of factors affecting difficulty. On a macro level, the difficulty of making decisions can be boiled down to skill, the available time, and the inherent difficulty of the decision [2]. However, only the available time is straightforward to measure in general, and in HCI, executing decisions can pose additional perceptual-motor difficulties.

This paper investigates difficulty in the particular setting of casual mobile puzzle games where the player progresses in the game by completing discrete challenges, or levels. In this context, a common operationalization of difficulty is the chance of the player completing the level, or pass rate. This is typically calculated as the count of successful attempts divided by the total attempts on the level. The inverse represents the average number of attempts per complete. Averaged over all players, this is a useful difficulty metric for game designers since it gives a tangible measure of how much time a player spends on a level before proceeding, which can help identify where the player might feel stuck [14] as a potential cause for churn. However, a significant drawback of this measure is that it does not account for individual differences in skill. Simply looking at the aggregate population, therefore, runs the risk of alienating anybody but the average player who, in many cases, is neither representative nor the most important prediction target. A more useful approach for game designers should thus take both individual player and level information into account.

This challenge of accounting for individual difficulties has been approached in multiple ways. A common practice is to utilize *dynamic difficulty adjustment* (DDA) in which the predictions are used to automatically adjust game parameters (e.g. number of enemies) and tailor the game difficulty to maximize aspects such as retention or monetization of individual players. However, this kind of fine-grained control over the levels is not always possible due to technical aspects (difficult to implement, uncertainty about parameters' effect on difficulty, etc.) or level and game design choices (levels requiring a specific strategy or visuals, difficulty curve must follow a certain pattern, etc.). Moreover, fully automated difficulty adjustment might not be attractive for game designers if they want to retain some control over the player experience. An alternative approach that can be of practical use should therefore also capture and explain player-level interactions and generate knowledge for the game designers that allows them to be more proactive.

The main objective of this paper is to showcase such a framework that game designers can use to both understand the interaction between players and levels and to estimate level difficulty for individual players. Rather than updating the estimates between game rounds, as is common in many DDA approaches, the goal of this framework is to leverage daily play session data to inform the offline work of level designers and help them understand the player base. To achieve this, we use *factorization machines* (FMs) which are especially known from recommendation systems [20, 39, 40, 42]. FMs allow predicting user-content interactions by estimating latent variables that describe each user and each piece of content.

To better understand how FMs can be applied for difficulty estimation, we investigate the following research questions:

RQ1: How do FMs compare to other difficulty prediction methods?
RQ2: How many observations of a player are necessary before it is possible to discern them from the average player?
RQ3: What do the FM model latent variables mean or represent?

Contribution. In summary, we examine FMs as a novel approach for personalized game level difficulty prediction. Using a large dataset of 700k players from the commercial puzzle game Lily's Garden, we compare FMs against both a naive non-personalized baseline and personalized predictions using Random Forests (RF). Our results support the use of FMs as a promising tool that clearly outperforms the other methods, especially if augmented with similar additional features as in the RF regression.

2 RELATED WORK

To contextualize our contribution, we first resolve ambiguity around the concept of difficulty, and survey related work on operationalizing and quantifying difficulty in videogames. We then survey existing player difficulty prediction models.

2.1 Game Difficulty

Game difficulty is a highly ambiguous concept [12], with at least two meanings [11]. Firstly, it can denote an *intrinsic attribute* of a game, characterizing a game-internal task based on its objective and the barriers that prevent potential players from achieving it. Secondly, it can describe a *relational attribute* between the game and the player, characterizing the player's experience of the task based on their individual skill and history. Often, researchers use the notion of *perceived difficulty* to convey the second, experiential meaning. To complicate things further, difficulty is often used synonymously with *challenge*. However, we can draw a subtle distinction based on the concepts' *valence* [11]: players typically consider a game task difficult, if it causes them frustration and discomfort; challenging tasks in contrast are stimulating and convey a feeling of being in control over the outcome [26]. Here, we primarily use the notion of difficulty, but without appealing to its negative valence.

Difficulty can be actively sought by players as a goal experience [6], or it can form the foundation [36] of other experiences. Famously, perceived difficulty contributes to player *enjoyment*, as investigated by Alexander et al. [1]. Another such goal experience is *flow* – a state in which a player feels engulfed in the task and loses track of time and worries [8, 10]. It results from exposing a player to difficulties that are optimal with respect to their individual skills. Self-determination theory [45, 50] posits that optimal difficulty satisfies players' intrinsic need for feelings of competence and in effect yields motivating gameplay. Amongst others, flow and intrinsic motivation contribute to player engagement [4], which, similar to enjoyment, constitutes a core game design objective.

Given the many ways through which difficulty impacts player experience, it is unsurprising that game designers and researchers have sought ways to operationalize difficulty for use in design-time quality control or runtime optimization. In the particular case of puzzle games, Pusey et al. [37] proposed several objective measures to quantify difficulty, including the number of incorrect attempts, the number of actions used, and the time taken to solve a puzzle. The average number of attempts that players spend to complete a level, or inversely the pass rate, has been used to assess difficulty in puzzle games such as Angry Birds [43] and Lily's Garden [24]. A player's experienced number of successes and failures also constitutes a major component of perceived difficulty, as empirically identified by Denisova et al. [11] in the development of a questionnaire to assess perceived difficulty in games. Similar to previous work, we operationalize an individual player's perceived difficulty as the average attempts per complete. Given that players cannot repeat levels in this study, this is equivalent to the number of attempts they require to complete a specific level.

2.2 Predicting Difficulty

Previous research has approached player difficulty prediction in numerous ways depending on the application and research purpose. Common application areas of difficulty prediction include DDA and automated playtesting for quality assurance. We next identify several shortcomings of existing work based on selected examples and highlight how our contribution overcomes them.

Existing approaches typically predict difficulty aggregated over a player population rather than individual experiences (e.g. [18, 22, 24, 32, 43, 51]). An example of this is the work by Gudmundsson et al. [18] where an AI game-playing agent was used to extract a preliminary estimate of the level pass rate. This estimate was then combined with level features to create a binomial regression model to predict the overall pass rate on the level. We consider the focus on aggregated predictions a shortcoming, as the perceived difficulty is affected by individual skill and experience, and an aggregate prediction is thus likely to predict the various goal experiences that perceived difficulty contributes to less accurately. In contrast to existing approaches, this paper focuses on predicting individual player difficulty, thus moving one step closer to predicting perceived difficulty as a relational attribute between one player and the game.

Existing studies that focus more on individual player difficulty prediction often suffer from practical and technical barriers: they are either operating on a very small scale (e.g. [29, 46, 47, 53]), only consider toy problems (e.g. [17]), or both (e.g. [16, 21]). Gonzalez-Duque et al., [16] for instance, use a Bayesian optimization approach to reliably estimate the time it would take a player to complete a Sudoku level given the number of pre-filled digits or a simple puzzle level given two level descriptors after having observed the player for 5 or 15 game rounds, respectively. However, the limited number of play traces (<300) in each game lead to a large variance in the results. It is uncertain how the approach would scale in a live game with several player cohorts and continuous updates to the game. In our study, we use data from more than 700,000 players over a 6 month period to demonstrate the applicability of our approach to a large-scale, complex commercial game system.

A common approach in many DDA methods is to consider the recent history of players in order to adapt the game content. However, not all methods are able to take advantage of the wealth of information that is available about the players and levels (e.g. [5, 16, 27, 55, 56]). For instance, Xue et al. [55] use a probabilistic graph that estimates the probability of winning, failing and churning based only on the player's progress and the current number of attempts; however, they do not leverage more descriptive features, such as average playtime, that have been found to be correlated with player engagement [13, 23]. In our approach, we aim to leverage a maximum range of player and level data, combined with high-cardinality data such as individual player-level interactions.

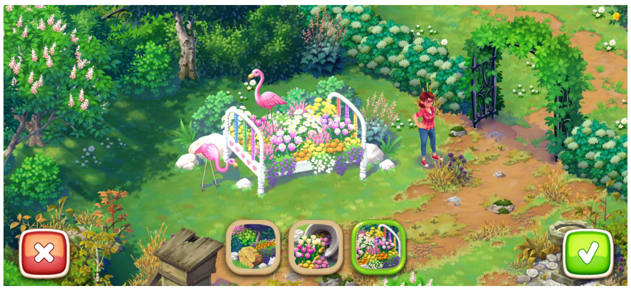
Especially in the domain of educational games, it is common to explicitly model a learning curve and introduce new elements through tutorials to build up player competencies for solving more complex tasks later [28]. The Additive Factors Model [7] is a popular method in this context for determining a player/student's chance of success on a task that requires certain skills [19]. However, many related approaches require labelling the required skills beforehand, and more specialized domain models can be hard to validate and generalize [15]. In this work, we do not assume any specific structure of player skill and knowledge and instead learn latent representations which we can interpret afterward.

As last two shortcomings, we note that, especially in live game operations, it is not desirable to have a black-box prediction method that requires expert knowledge to operate [30], as in related work that utilizes deep learning methods (e.g. [31, 33, 35]). In addition, methods that require complex implementations (e.g. [52]) are also undesirable, as any unexpected behavior may be hard to troubleshoot and make game designers lose trust in the system. Moreover, little to no knowledge is generated that enables game designers to make informed decisions about their games. In this work, we demonstrate that factorization machines (FMs) afford ease of use, do not require any special input other than data about the player and the number of attempts they spent on a given level, and afford straightforward interpretation that game designers may be able to use for other tasks such as personalized offers or churn prediction.

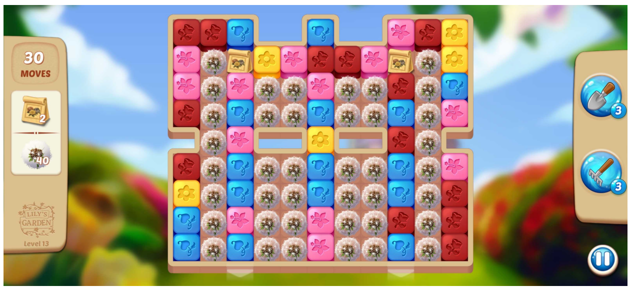
To determine feasible approaches for difficulty prediction in games, it is worth looking beyond games. A similar problem to difficulty prediction is student grade prediction [38, 48]. Sweeney et al. [48] use a number of methods to predict the grades for new students in future courses, ranging from average course grades, over Random Forest (RF) regressors, to FMs. They found that FMs performed the best when not including any other information than the student-course interaction, while with additional information, FMs and Random Forest regressors performed similarly well. We are inspired by this work and the ability of FMs to capture high-cardinality data like user-item-context interactions. Consequently, we adopt a similar strategy and compare three different approaches to modeling player difficulty, including RF and FM.

3 CASE STUDY: LILY'S GARDEN

We study the use of FMs for predicting individual player difficulty in the commercial game Lily's Garden by Tactile Games. Released in early 2019, Lily's Garden is a casual mobile puzzle game with more than 6,000 levels and one million daily active users worldwide.



(a) Metagame garden scene, with Lily being offered three flowerbed decoration options. Realizing an option and progressing in the storyline requires points, which the player collects by solving puzzles.



(b) Example of a puzzle level. The goal is to collect certain board pieces, shown on the left side, within a limited number of moves.

Figure 1: Lily's Garden: interplay of meta and puzzle game.

The gameplay has two main components (Fig. 1). In a narrative-driven meta game, the player is confronted with an abandoned garden in which they can unlock new areas, make decorative choices and progress in the story by spending points. These points can be acquired by solving successive puzzle game levels unlocked at specific times in the storyline. This study focuses on predicting individual player difficulty for these puzzle levels.

To complete a puzzle, the player must collect specific goal pieces on the board within a given move limit. The core gameplay consists of tapping on board piece clusters to clear them, destroy adjacent pieces, and hereby collect the goal pieces. By tapping on clusters with more than five, eight or ten pieces, the player can create power pieces capable of clearing large parts of the board. In some levels, forging such power pieces is strictly necessary to succeed, and more advanced strategies involve their combination for enhanced effects.

The game implements a free-to-play model. The player can use in-game and real currency to buy help in the form of power pieces and other boosters. Additionally, there are certain game events that provide such boosters for free. Moreover, players can purchase an additional five moves for the current attempt if they fail to complete all the goals within the initial move limit. While purchases are the main monetization avenues of the game, the designers ensure that every level can be completed without bought assistance.

For our model, we adopt the existing operationalization of level difficulty as the average number of attempts that players require to complete a level. An analysis of player data from Lily's Garden

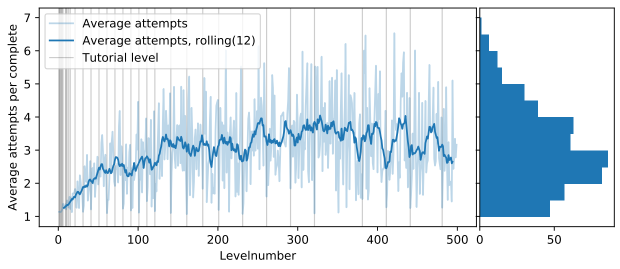


Figure 2: The average number of attempts per level completion for the first 500 levels. The difficulty trend is illustrated as a moving mean with a centered window of size 12 – twice the length of typical designed level sequences.

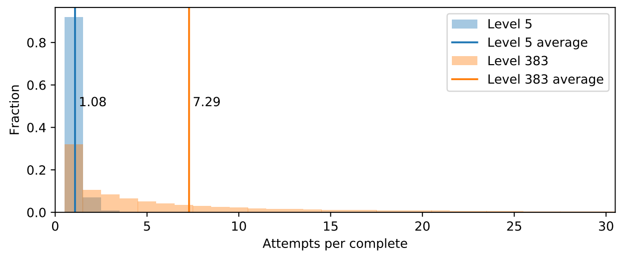


Figure 3: Comparison of player attempt distributions for a tutorial (level 5) and a hard level (level 383).

(Sec. 5 provides an overview of this data) shows the average number of attempts over the whole level range (Figure 2). The first few levels (< 10) contain multiple tutorial levels where the gameplay is streamlined and players are restricted to certain moves. After these levels with almost guaranteed wins, the difficulty slowly ramps up – a common design pattern to engage players early on [28].

To understand what level design aspects can affect (intrinsic) difficulty, we interviewed the team of level designers of Lily's Garden and identified a number of candidate level features that could be relevant for predicting difficulty. This includes quantifiable features such as the move limit (higher limits lead to easier levels [25]), the number of goals, and the entropy of the pieces' color distribution (the closer to uniform, the harder the level due to power pieces being harder to create). Other features, such as the level layout, board piece complexity, or reliance on power pieces, are harder to quantify but will still affect the difficulty in non-trivial ways. Defining descriptors that can fully encapsulate these intricacies is therefore rather challenging and can lead to less expressive and accurate difficulty prediction models.

In addition to the strong fluctuations in the average number of attempts over the level range, a more detailed analysis also highlights large differences between individual completion rates both with respect to the players and the levels (Fig. 3). In early, less (intrinsically) difficult levels with fewer than 1.5 attempts per complete on average, most people (>90%) require 1 attempt, with the remaining players requiring a little more. For later levels with greater (intrinsic) difficulty, we find a large variance in attempts and a long tail distribution extending to more than 30 attempts per complete. These individual differences, paired with strong fluctuations in average difficulty, make Lily's Garden an ideal, challenging candidate for studying personalized player difficulty prediction.

4 METHODS

In accordance with existing work on puzzle games (Sec. 2.1), we operationalize difficulty by the number of attempts a player will spend on the level. We consequently frame our individual player difficulty prediction task as a regression problem, where the target is to estimate the number of attempts a specific player will spend on a specific level. For this purpose, we compare four different methods:

- Naive baseline (**NB**): Average attempts by other players.
- Random Forest regression (**RF**): Ensemble prediction from multiple decision trees that utilize aggregated player behavior data over the observed levels.
- Factorization Machines (**FM**): A general regression model that uses a feature embedding to describe interactions between variables (e.g. user-item interactions).
- Factorization Machines with Relational Data (**FM+feat**): As **FM**, but also includes the descriptive variables used in the RF method (e.g. user-item-feature interactions).

To answer our first research question (Sec. 1), we compare these models based on their prediction error. We moreover analyze how this error changes based on the number of levels that we observed the players for. In other words, we identify the number of required observations to push the error below a certain threshold and thus answer our second research question.

4.1 Naive Baseline

Given that much related work focuses on predicting difficulty for a player population rather than individuals, we chose the player-average number of attempts per level complete as our naive prediction baseline. We calculate this non-personalized prediction on the players' data from our training set as illustrated in Fig. 2 using a linear regression model,

$$\hat{y} = w_0 + \sum_{\ell=1}^{L} w_\ell x_\ell, \tag{1}$$

where $w_0 = 0$, w_ℓ is the attempts on level ℓ averaged over all other players, L is the total number of levels, and $x_\ell \in \{0, 1\}$.

This non-personalized baseline is what game designers currently use in practice for estimating level difficulty. Hence, any improvements over this baseline can directly inform game designers of the compared methods' benefits.

4.2 Random Forest Regression

As mentioned in Sec. 2.2, a Random Forest (RF) regression model has previously been shown to deliver comparable performance to FMs [48] in a related task. Consequently, we train an RF regressor on player and level features in our comparison.

RF is an ensemble method based on multiple random trees, i.e., decision trees $h(\mathbf{x}; \theta_t), t = 1, ..., T; \theta_t$ with *i.d.d.* random vectors, where the nodes of each tree are split using a random set of features and subsets of the data. For regression problems, the split is decided based on which feature leads to the largest decrease in the absolute or squared error. The final prediction then combines the predictions from the trees into an average prediction, $y = \hat{h}(\mathbf{x})$. This ensemble approach enables modeling more complex non-linear behavior and is less likely to overfit compared to a single random tree.

We use the Random Forest regressor implementation from the *scikit-learn* library (version 1.0.2) [34] with the following hyperparameters and settings: `n_estimators=150`, `max_depth=None`, `min_samples_split=2`, `min_weight_fraction=0.0`, `max_features="auto" [=n_features]`, `max_leaf_nodes=None`, `min_impurity_decrease=0.0`. Due to the size of our data, we employ an incremental training method[1].

4.3 Factorization Machine

Factorization machines (FMs) are a class of factorization models that can be used as a general predictor for classification, regression, and ranking tasks [39]. They are similar to linear regression models but instead model second-order terms as an interaction between variables using a feature embedding:

$$\hat{y} = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v_i}, \mathbf{v_j} \rangle x_i x_j, \tag{2}$$

where w_0 is the 0th order term or global bias, w_i is the first-order term and describes the bias of the i'th variable, and $\mathbf{v_i}$ is a second-order feature embedding vector of the i'th variable. $\langle \mathbf{v_i}, \mathbf{v_j} \rangle$ describes the interaction between two variables as the dot product:

$$\langle \mathbf{v_i}, \mathbf{v_j} \rangle = \sum_{f=1}^{k} v_{i,f} \cdot v_{j,f},$$

where k is the number of latent factors and a hyperparameter that must be chosen.

This learned embedding is what enables modeling unseen interactions in the data, which also makes FMs widely used for recommendation systems, including games recommendations [3, 9], where user-item interactions are typically very sparse.

The input data is not restricted to the rating-user-item format. It is possible to use other contextual information [41], including

- One-hot encoding of previous user interactions on other items.
- User and item descriptors (both numeric and categorical, e.g. age or labels).
- Implicit feedback data.

Adding additional features typically increases the data set complexity by $n \times f$, where n is the dataset length and f is the number of additional features. However, using FMs with a *relational data* block structure to make use of repeated patterns, as suggested by Rendle et al. [41], can greatly reduce the computational complexity and make the method scale to very large datasets. Additionally, Rendle et al. found that FMs with such relational data showed a consistent performance increase over FMs without relational data.

We use the original implementation of LibFM by Steffen Rendle [40]. We train each model for 1000 iterations using Markov Chain Monte-Carlo (MCMC) with an initial standard deviation of 1 to sample v_i for FM models and 0.1 for FM+feat models. Based on brief experiments and the nature of the attempt distribution data, we do not include a global bias term w_0.

5 DATA

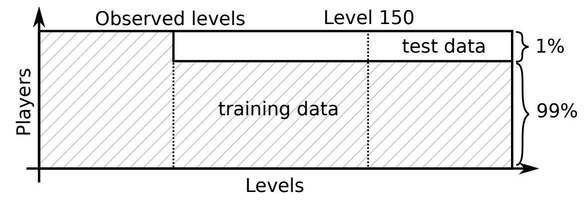


Figure 4: Train/test data split illustration. All data of 99% of players is used for training. Additionally, the training utilizes initial observations ("Observed levels") from the 1% of players who constitute the test set.

The data used in this study was collected from 2021-06-01 to 2021-11-30 from the game Lily's Garden and consists of 759,382 players who have all, in this period, played the game beginning with the first level and at least up to level 200. In free-to-play mobile games, it is common to have a large churn rate at the beginning of the game from players that do not interact meaningfully with the game, so this condition ensures both, that the whole history of each player is complete, and that the included players share the same minimum engagement level. Due to the long tail of the attempts distribution (Fig. 3), for numerical stability, we truncate attempts with more than 30 attempts to 30, which affects 0.34% of the data.

We split the data to match a realistic use case: We select 1% of the players to represent "new" players to test the methods on, and the remaining 99% of the players are considered "old" players who have started playing earlier and have already progressed far in the game. We use this old player data for training all models.

Additionally, as illustrated in Fig. 4, FM training also utilizes some initial observations of the new players. This corresponds to the model being periodically updated to improve its predictions as new players progress through the game and more observations become available. It is also necessary for FM: without observation of the given player during training, the model does not learn the bias and embedding of this player (cold start problem).

In our performance evaluations, we report the results with different numbers of initial observations. The results are always computed from levels after 150 to maintain a consistent test set even when the number of initial observations changes.

The RF and FM+feat methods require feature vectors that describe the players and levels. These features were selected based on domain knowledge from level designers and are shown in Table 1. RF methods do not deal well with high-cardinality data, so it is not possible to one-hot encode players and levels in this case. Instead, the player features for all players are aggregated and averaged across the first n observed levels. This means that any prediction of a player depends on their early performance in the game and does not take recent observations into account. Otherwise, players are not comparable through their features due to each player being at

[1]https://github.com/garethjns/IncrementalTrees

Type of feature	Name	Description
Player features	Attempts	Number of attempts on levels
	Moves used	Number of moves used relative to the move limit when completing the level
	Pre-game boosters	Boosters that can be used before starting the level
	In-game boosters	Boosters that can be used while playing the level
	Powerpieces, total	Number of power pieces created while playing the level
	Powerpieces, combos	Number of power piece combinations created while playing the level
	Rockets, solo	Number of rockets created and used on their own
	Rocket-bomb combo	Number of rocket-bomb combos created and used
	Rocket-magic combo	Number of rocket-magic combos created and used
	Bomb-magic combo	Number of bomb-magic combos created and used
Level attributes	Attempts	Average number of attempts on level by players in training set
	Color entropy	Entropy of color spawning weights; $S = -\sum_i p_i \log p_i$
	Colors	Number of unique colors in the level
	SpreadingBlocker, cg	Levels with a spreading blocker as collect goal (cg)
	LayerCake, cg	Levels with a specific blocker with 3 hitpoints as a collect goal
	ConsecutiveBlocker, cg	Levels with a blocker that requires two attacks in a row as a collect goal
	MegaMultiColorBlocker	Levels with a large blocker that requires matching multiple colors to remove
	Teleport	Levels with a teleport mechanic that transports pieces around the board

Table 1: Features investigated for RF and FM+feat. The player features are aggregated means on the first n observed levels.

different stages of the game with different types of levels, difficulties, required strategies, etc. The level attributes are static and do not change depending on the number of observations.

6 RESULTS

In the following, we first describe the prediction task to put the baseline prediction and error metric in context. We then identify how many observations are necessary to beat the baseline to answer our first and second research questions. Lastly, in order to answer our third research question, we provide an interpretation of the model parameters and identify key game levels for understanding the model and thus providing valuable insights for the level designers.

6.1 Baseline Prediction and Error

Both the RF and FM models have been optimized using root mean square error (RMSE). However, the underlying distribution of attempts, as shown in Fig. 3, follows a geometric-like distribution, where the most common value is 1, and especially hard levels exhibit a long tail that drives the average attempts up. This long tail on hard levels can yield a large RMSE on said levels, leading to the hard levels having a large effect on the model optimization. To provide a more complete picture of the models' performance and support model comparisons on easier levels, we also report the mean absolute error (MAE) next to RMSE.

Crucially, we cannot expect this baseline nor any of our methods to yield a close-to-zero prediction error. This is because each outcome of a level playing attempt is also governed by aleatoric uncertainty, which the model cannot account for. Each prediction captures the expected value for a given player-level combination.

Calculating the baseline error by aggregating all data points, we find the prediction errors across the whole level range to be $\text{RMSE}_{\text{all}} = 3.86$ and $\text{MAE}_{\text{all}} = 2.33$. The errors after level 150

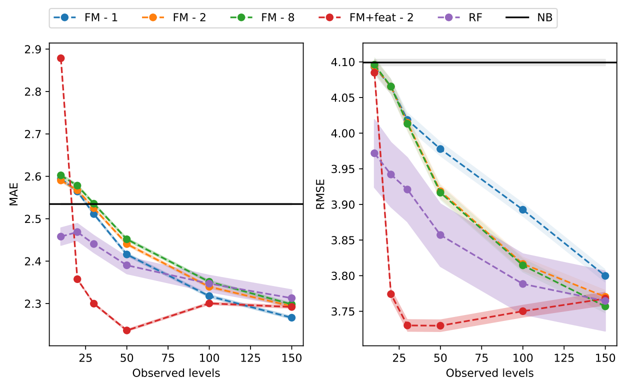


Figure 5: MAE and RMSE on test levels $\ell > 150$ for different observed level counts. The shaded area shows the 95% confidence interval around the means.

are $\text{RMSE}_{\ell>150} = 4.10$ and $\text{MAE}_{\ell>150} = 2.53$. The larger error on $\ell > 150$ is due to these levels being generally (intrinsically) more difficult (Fig. 2) and thus having a larger attempt variance.

6.2 Effect of Observed Level Count

Before being able to differentiate between players meaningfully, we require sufficiently many discernible observations of their gameplay. The first 10 levels introduce the core gameplay, and new mechanics are then introduced in every tenth tutorial level (21, 31, 41, ... see Fig. 2). We, therefore, compare the methods when trained at 6 points of a player's progress: at 10, 20, 30, 50, 100 and 150 levels. To avoid information leaking between training and test sets, we evaluate the predictions on the test users after level 150.

We tested our FMs with 1, 2, 4, 8, 16, and 32 factors, but we leave out the 4, 16, and 32-factor models in the presented results for clarity of visualization since they do not alter or further inform our conclusions. The MAE and RMSE of all tested models are shown in Fig. 5, along with the 95% confidence intervals. The FM models without additional features (i.e., excluding FM+feat-2) all have similar performance, with the 1-factor model performing better in terms of MAE and the other FM models performing better in terms of RMSE. This suggests that using a single latent factor is not enough to capture the large variance in high-difficulty levels (see Fig. 3), but it makes the model less likely to overfit and perform worse on easier levels. The RMSE plot shows that these FM models are on par with the baseline prediction for as little as 10 observed levels, and, as more levels are observed, the prediction further improves over the baseline.

The predictions can be further improved while requiring fewer observations by including additional features as described in Table 1. This holds for both the RF and FM+feat models. The RM+feat-2 model shows superior performance, but its error increases after 50 observed levels. We consider this an effect of overfitting to the additional data since the training error for all the FM+feat models appears to be around $\text{RMSE}_{\text{train}} = 3.2$.

This highlights the importance of feature engineering, and while more sophisticated features could be utilized, the basic FMs presented here are game-agnostic and scale easily, providing a feasible method for game designers to employ. Our results also show that, even when additional information is available and included as features in the RF model, FMs are still able to extract more relevant information from the player-level-feature interactions.

To explore why early predictions can be improved by including additional data comprising more fine-grained descriptions of players, we analyze the feature importances from the RF model. Fig. 6 shows how the four most important features change depending on the level observation count. We find that the model utilizes additional information other than the number of attempts: early on, more fine-grained behavior data such as the average number of moves is more important compared to later on, where the average number of player attempts becomes increasingly more important. However, with more observed levels, all models reach a similar degree of performance. While the FM+feat-2 model performance appears to stagnate, the FM approaches reach similar, if not better, performance. This stagnation may be caused by the feature aggregation, where the possible level of detail and stand-out behaviors are more washed out with an increasing number of observations. Other features, such as power piece and booster usage, all have a relative importance of around 0.05, i.e., they are not uninformative. However, they are strongly correlated, which may reduce their joint importance, as mentioned in Sec. 4.2.

We also analyzed how the accuracy of predictions varies depending on the specific level they are computed for. To this end, we plotted the MAE of the FM-2 model relative to the naive baseline, as shown in Fig. 7. We omit a similar plot for the other models as they exhibited similar behavior. We find that the predictions immediately after the last observed level are more accurate, with performance deteriorating compared to the baseline on later levels. We conclude that, firstly, the FM prediction accuracy drops at later levels, and, secondly, more observed levels lead to a slower decline in predictive performance, meaning a higher observed level horizon allows a model to be used farther into the future.

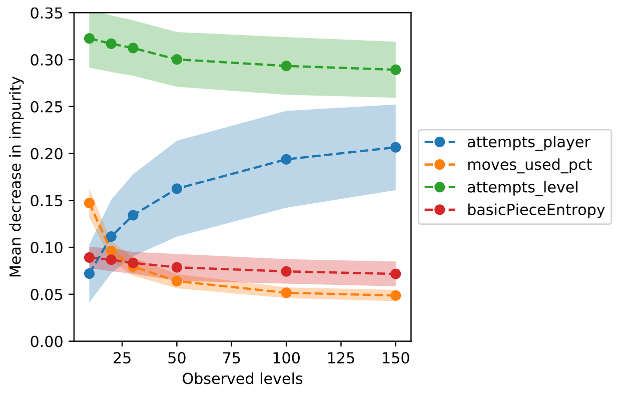


Figure 6: Mean feature importances and one standard deviation for the Random Forest (RF) regressor. Only features with a relative significance above 0.05 are shown.

The results presented in this section allow us to answer the first two research questions: Firstly, the basic FM model without additional information fares worse than the RF model with additional data until after 100 observed levels, where they converge to similar performance. However, the FM with the same additional data as the RF outperforms all other models after 20 observed levels. Secondly, the number of observations that are necessary to discern a player from the average players depends on how detailed the available information is: with only the aggregate number of attempts, the FM method requires between 10 and 30, while more fine-grained data can enable predictions after 10 observations for all approaches.

6.3 Interpreting FM Model Parameters

To answer the third research question, we investigate the found FM model parameters in detail. As already mentioned, the FM models with two or more factors seem to capture other aspects than the 1-factor model, and the FM-2 method yields comparable performance as the RF method after 100 observed levels. We thus limit the analysis to the 2-factor model with 100 observed levels.

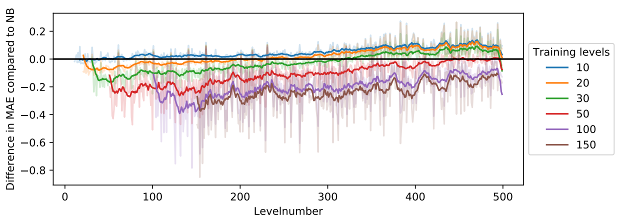


Figure 7: MAE difference between the prediction and baseline for the FM-2 model on the test user predictions for different observed level counts. The lines show the rolling average over 12 levels. Similar behavior was observed for the other models but is not shown for clarity.

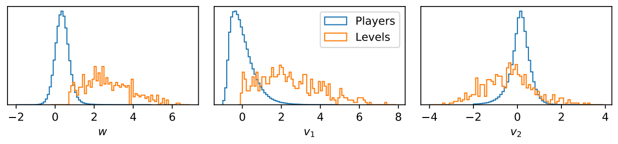


Figure 8: Histogram of the FM model parameters (100 observed levels, 2 latent factors).

FMs use two main parameters (Sec. 4): the variable biases, w, and the latent factors that describe second-order interactions, $\mathbf{v}$ (ignoring subscripts for specific users/items/attributes). As a first step, we consider histograms of the model parameters in Fig. 8 and separate the distributions by player and level parameters. We find that the distributions are distinctly different between players and levels for all three parameters. We consequently differentiate between levels and players in the following examination of each parameter before discussing their relationship to the player-level interactions.

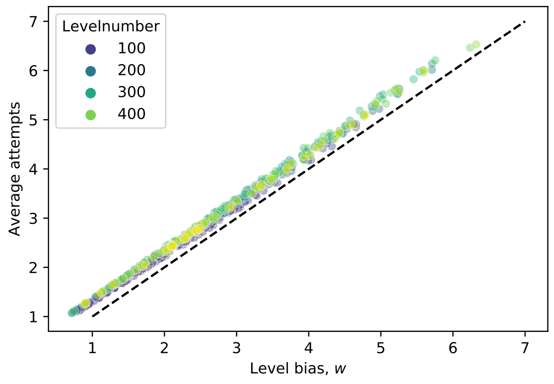


Figure 9: The average attempts on the level over the level bias, w. The black dashed line is the diagonal w = attempts.

Variable bias, w: For the levels, this variable is strongly correlated ($\rho_{\text{Spearman}} = 0.99$) with the average number of attempts on that level (Fig. 9). However, the w parameter for the levels is not sufficient to match the underlying distribution of attempts, which is not surprising since that would correspond to the baseline model. As for the players, w does not correlate strongly with any player-related metrics. This suggests that the main first-order term comes from the levels, supporting an interpretation of w as a baseline level difficulty. The remaining variance must therefore be explained by the second-order interaction terms between player and level.

First latent factor, $\mathbf{v}_1$: For both the levels and players, this variable is also strongly correlated with the average number of attempts ($\rho_{\text{Spearman}} = 0.98$ and 0.70, respectively), as well as the variance of the attempts ($\rho_{\text{Spearman}} = 0.99$ and 0.73, respectively). We note that for the levels, v_1 is strictly positive, with the exception of 5 tutorial levels which are only slightly negative. This suggests that v_1 captures a variance effect for each level, where the sign of the interaction depends solely on the player. The amplitude then reflects the consistency of the level/player: a value close to 0 for the level signals little variation between the players (it may only require one simple strategy to win, e.g. in tutorials), while a large v_1 suggests a large variation (e.g. it may require multiple strategies, or come with high aleatoric uncertainty). For the player, v_1 is more reflective of their skill level: for $v_1 < 0$, we expect a player to use consistently fewer attempts than their peers, especially on harder levels where skill and strategy matter more. Conversely, $v_1 > 0$ indicates that the player struggles to employ winning strategies.

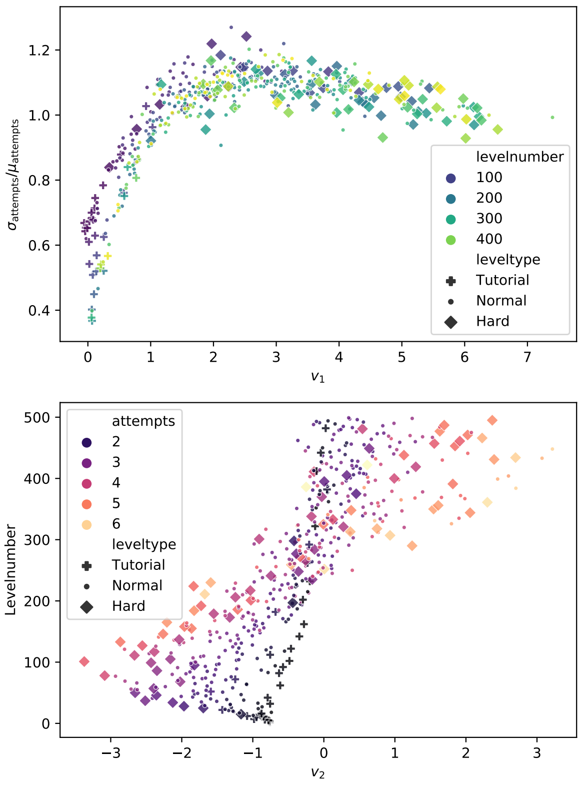
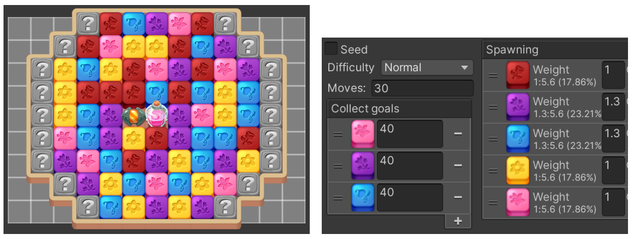


Figure 10: Top: Normalized variance versus the latent factors, v_1. Bottom: The level number plotted against the second latent factor, v_2. Note that level designers assign the *hard* label based on their own initial estimates, so it does not always reflect whether a level requires many attempts on average.

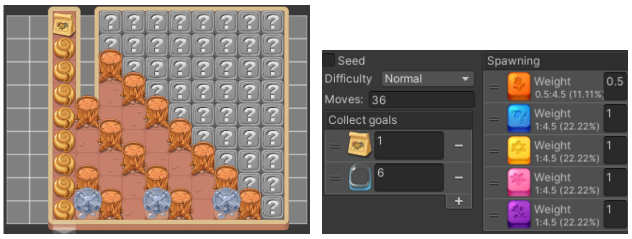
To support this interpretation, we investigated levels at the extreme values of v_1. Since v_1 is strongly correlated with the average number of attempts to complete a level, which in turn is strongly correlated with the attempt variation on said level, we show the attempt variation normalized by the average attempts as a function of v_1 in Fig. 10. At small values of v_1, we find the tutorial levels which generally have a high move limit and do not require any special strategies. Conversely, at large v_1 are the levels with the greatest difficulty and variance compared to the mean. As an illustrative example, we show level 383 in Fig. 11, which has the highest v_1 among all levels. There are multiple gameplay reasons why this level might have high variance and difficulty:



Seed
Difficulty Normal
Moves: 30
Collect goals
40
40
40

Spawning
Weight 1:5.6 (17.86%) 1
Weight 1.3:5.6 (23.21% 1.3
Weight 1.3:5.6 (23.21% 1.3
Weight 1:5.6 (17.86%) 1
Weight 1:5.6 (17.86%) 1

(a) Level 5. The level is a tutorial level and has the 7th lowest v_1. It does not require any special strategies to complete.



Seed
Difficulty Normal
Moves: 36
Collect goals
1
6

Spawning
Weight 0.5:4.5 (11.11% 0.5
Weight 1:4.5 (22.22%) 1
Weight 1:4.5 (22.22%) 1
Weight 1:4.5 (22.22%) 1
Weight 1:4.5 (22.22%) 1

(b) Level 383. The level is hard and has the highest v_1. In order to complete the level, the first level goal requires removing all the shells to the left, so the seed bag drops down to the bottom, while the second goal requires attacking either of the three bubble spawners at the bottom and then destroying six bubbles. The shells and tree stumps are both hard blockers, requiring power pieces for their removal.

Figure 11: Examples of a tutorial (*a*) and hard level (*b*). The left-hand side shows the level layout, and the right-hand side shows the move limit, level objectives, and color distributions. The pieces marked with '?' are assigned a random color at the start given by the weight distribution shown to the right.

(1) The level contains blockers, which can only be destroyed with power pieces. Less skilled players will have a harder time creating the necessary pieces.
(2) The spawning weights on piece colors are almost uniform, making it harder to create power pieces.
(3) There is one specific power piece combination (2 magics, requiring 2 combinations of 10 pieces) that can easily win a level. More skilled players may have learned this strategy.
(4) The board itself is larger than average, allowing more possible strategies (e.g., focus on the top/bottom/left/right side).

All these aspects lead to the perceived level difficulty being highly variable and dependable on the player's skill and game knowledge, supporting the interpretation that v_1 expresses how large an impact the player's skill can have on a given level. In contrast, level 5 in Fig. 11 has the 7th lowest v_1 and requires a much more simple strategy to win, affording little variation in individual player difficulty.

Second latent factor, $\mathbf{v_2}$: This factor is strongly negatively correlated with the level number, as shown in Fig. 10. For levels below 250, v_2 tends to be negative and becomes positive beyond this threshold. There is nothing in the gameplay that changes drastically in the first 500 levels, but since the level number is linked to how many players have played the level, we interpret v_2 to capture aspects that are more temporal and related to the underlying data distribution: there are more observations on early levels, and different players have played later levels. It is thus likely that FM models with more factors find such spurious connections in the data.

For a deeper interpretation of v_2, we note that that v_2 is inversely correlated with player features as compared to v_1. Since v_1 can be interpreted as player consistency and skill, v_2 appears to capture similar aspects but with the opposite sign, although it should be noted that the sign of the factors is arbitrary, as the dot product between user and content factors stays unchanged if one changes the sign of both factors. Since v_2 for the levels is centered around 0, the effect on the prediction depends on the player and their progress: a negative v_2 for a player means they spend more attempts on early levels but fewer attempts later on, while a positive v_2 signals a reduction in predicted attempts early on and more attempts later on. The v_2 factor can therefore also be related to a temporal shift in a player's playstyle compared to their peers. Crucially, this interpretation is less clear than for the first latent factor as v_2 appears to capture aspects more connected to the data collection rather than the player-level relation.

7 DISCUSSION

Overall, our results from this case study suggest that FM outperforms both the naive baseline and RF, especially if augmented with similar features as RF. Although RF performs better with fewer observations than non-augmented FM, the latter is game-agnostic, can be informative about the inter-dependency between players and levels, and does not require expert knowledge on relevant player and level descriptors. If such information is available, it can be utilized by FMs in addition to the player-level data. Crucially though, there remain some caveats and limitations regarding the practical use of FMs and their generalizability, which we further elaborate on below.

7.1 New Players and Content

One issue with FMs is the cold start problem where the model does not learn model parameters (w, $\mathbf{v}$) for a specific player if they were not included in the training data (e.g. if a model is trained during the night and the player starts playing the next day). This can be mitigated by including additional player information, as we tested in this study. While this increases computational complexity, the capacity to describe the player with a set of features that can be calculated immediately enables predictions without retraining the whole model.

While we did not extend our research to include unseen levels, the same cold start problem also exists in that case. We expect this to be more difficult to mitigate since information about the historical average number of attempts on a level – a very strong predictor for the individualized predictions – is unavailable, and the entropy of the color distribution (which *is* available) only explains a small percentage of the variation. More work is therefore necessary to identify relevant level features, but some immediate options could be to include AI playtest agent data since this data can be strongly correlated with the player pass rate [24, 43, 44]. We advocate FMs as a straightforward extension to the current AI playtesting approaches that will be able to capture the temporal and cohort differences, unlike previous prediction models.

7.2 Utility to Game Designers

A lot of work on DDA focuses on immediate adjustments to the game during play. This may be possible using an FM model and relational data; however, the strength of the FM approach is not just accurate predictions but also the modeling of further player characteristics. The latent factors were identified to be related to the players' skill and consistency over time, and this information can be relevant to many other features of the game. For instance, it could be used to cluster people together for in-game tournaments or groups so that they match in skill level and the competition is fair. It can also help level designers in proactively maintaining the level database by identifying problematic levels and bottlenecks before a new player cohort reaches this point. Ideally, in-game purchase data can also be incorporated into the process to provide estimates for both expected difficulty and monetization effectiveness and let level designers make informed decisions on necessary changes.

Before a tool like this can be implemented in a live game, though, there are a number of practical challenges that need to be considered first. Level design is very often an iterative process where minor adjustments are repeatedly performed on the level. Any change to the level will lead to a different difficulty, and the challenge is to convey this to the model. While some changes are easy to quantify, such as an alteration of the move limit or goals, others can be more tricky: even minor changes in the layout (e.g., assigning specific colors to start pieces or creating a hole in the level) can have a large impact on the difficulty, creating almost an entirely new level. We argue that FMs should be able to deal with both cases: if the change is easily quantifiable, it can be added as relational data, and if not, the level can be counted as a completely new level. While this would discard the information that two levels are very similar and increase data sparsity, FMs generally perform very well on sparse data compared to other algorithms, making them a very good candidate to use in the game industry.

7.3 Sensitivity to Randomness

As noted earlier, due to the stochastic nature of Lily's Garden, it is not possible to predict the exact number of attempts of a player on a level. An alternative could be to predict the probability of success per attempt, similar to Xue et al. [55]. Approaching the optimization problem from this perspective would require replacing the RMSE and MAE for evaluating the models with a measure such as the Poisson deviance, which is more appropriate for strictly positive counting data. However, for this study, we focus on RMSE and MAE, as these measures are more canonical and easier to interpret by game designers.

7.4 Generalizability to Other Games and Domains

Our study is based on one specific puzzle game, but we expect our approach to work for other games as well, including other types of puzzle games, different genres (e.g. platformers or first-person shooters), or even different types of games (e.g. educational games). FMs are task-agnostic and can be applied to any game where the following requirements are met:

(1) The variable of interest, e.g., pass rate or time taken, can be predicted as a multiplicative (or divisive) combination of user and content latent variables such as skill and difficulty.
(2) Multiple users are exposed to the same units of content such as puzzles or levels, allowing for inference of the latent variables.

Many games meet these requirements, but there are notable exceptions, such as highly/completely random games, where the player has little effect on the outcome, and procedurally generated games, where the generated content can be unique for each player. Some instances of procedural content generation might still allow modeling latent player-content relations, e.g. different players struggling with different enemies. However, this needs to be evaluated on a per-game basis.

While we expect the model parameters to capture similar aspects across games, we do not consider it feasible to directly transfer a trained model to another game unless the games are very similar in terms of both gameplay and the distribution of predictor variables. However, if the same players play multiple games, all the multi-game user-content interactions could, in principle, be combined into one big FM dataset. This is a potential direction for future research. It should be noted, though, that games can have different types of challenges, such as emotional, cognitive, and physical challenges [12]. To model all the challenge and skill facets of multiple games, one will likely need more FM factors than our default 2.

7.5 Generalizability to Difficulty and/or Skill Varying Over Time

Our approach does not explicitly take into account players' learning and skill improvement during play. However, Fig. 7 demonstrates that the model performs well more than 100 levels into the future. Predicting beyond such a safe horizon is typically not necessary, as the model can be retrained periodically to include new observations of each player progressing through the game and gaining in skill. For instance, in our case, it would be feasible to retrain daily, as most players consume far less than 100 levels per day (median = 7, 90th quantile = 31), and the training takes only 14 hours on an Intel Cascade Lake *c2-standard-16* Google Cloud machine.

Generalizability to dynamically varying skill and difficulty could also be improved with additional FM input features. An obvious choice is to utilize difficulty features such as a level's move limit or the amounts and types of enemies. It should also be possible to utilize player statistics such as total playtime, to allow FM to explicitly model skill acquisition over time. Future work is needed to investigate whether and how much such extra features improve prediction accuracy.

8 CONCLUSION

We have considered the problem of predicting personalized perceived difficulty and specifically how many attempts a player will spend on completing a level in a commercial mobile puzzle game. To this end, we compared four approaches: a simple non-personalized baseline, a Random Forest regressor, and Factorization Machines (FMs), the latter with and without game-specific features.

In terms of prediction performance, both the RF and FM methods achieved a lower MAE than the baseline after 10 and 30 observations of the players, respectively, answering our second research question about how many observations are necessary for being able to discern between players. The FM method that utilizes the same features as the RF model achieves better performance than all the other models already after 20 observations. All models had a lower RMSE than the baseline after 10 observations, indicating that any kind of personalized prediction can improve difficulty estimates.

A deeper analysis allowed us to answer our last research question about the FM parameters and their correlation with player and level characteristics. The first-order term, w, captures level difficulty, while the first latent factor, v_1, captures a player-level interaction that quantifies the player skill and level randomness: for levels, v_1 was strictly positive (except for 5 tutorial levels) and can be thought of as a level variance, whereas for players, both the magnitude and sign of v_1 indicate how sensitive and consistent a player is to the stochastic elements of a level and can thus be thought of as an expression of skill. Lastly, v_2 captures signals related to the underlying data distribution and temporal changes in player behavior. For levels, v_2 is strongly correlated with the level number. The latent factor of the player, therefore, describes whether they perform better or worse over the progress of levels compared to their peers and, in a sense, how well they learn how to play. Including more latent factors led to overfitting and was thus not analyzed in more detail.

Overall, we find that FMs have multiple advantages over other prediction approaches: they outperform other typical approaches even when only relying on game-agnostic player-level-attempt data but have the option to utilize more fine-grained data to further improve performance; the FM model parameters provide interpretable results; FMs are scalable to large amounts of data. FMs therefore show a large potential for difficulty modeling not just for research but also in an industrial context where such advantages are requirements.

9 FUTURE WORK

We have focused on estimating how many attempts a player will spend on completing a level, but there are a number of extensions to the model that may make it even more useful for game designers.

The model performance clearly improves with additional information, especially when only a few player observations are available. A relevant line of future investigation is therefore to identify the features that have a large impact on the model performance and optimize the feature selection to prevent overfitting. Additionally, including AI playtest agent data may also improve the predictions on levels with very few or no player-level observations.

The same player-level data could also be used in future work to predict variables other than the number of attempts. Other useful prediction targets could include churn, the probability of purchasing a continue, or the number of actions required to complete a level. Furthermore, the learned model parameters might be useful for other modeling approaches such as personalized offers, churn prediction, or sampling criteria for A/B testing new content to ensure enough player variation of the content.

ACKNOWLEDGMENTS

Many thanks to our anonymous reviewers for their extremely valuable feedback. CG has been partly funded by the Academy of Finland Flagship program *Finnish Center for Artificial Intelligence* (FCAI). We thank Tactile Games for supporting this work, and especially the level design team for sharing their expertise and insights.

REFERENCES

[1] Justin T. Alexander, John Sear, and Andreas Oikonomou. 2013. An investigation of the effects of game difficulty on player enjoyment. *Entertainment Computing* 4, 1 (Feb. 2013), 53–62. https://doi.org/10.1016/j.entcom.2012.09.001

[2] Ashton Anderson, Jon Kleinberg, and Sendhil Mullainathan. 2017. Assessing human error against a benchmark of perfection. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 11, 4 (2017), 1–25.

[3] Syed Muhammad Anwar, Talha Shahzad, Zunaira Sattar, Rahma Khan, and Muhammad Majid. 2017. A game recommender system using collaborative filtering (GAMBIT). In *2017 14th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*. 328–332. https://doi.org/10.1109/IBCAST.2017.7868073

[4] Jeanne H Brockmyer, Christine M Fox, Kathleen A Curtiss, Evan McBroom, Kimberly M Burkhart, and Jacquelyn N Pidruzny. 2009. The Development of the Game Engagement Questionnaire: A Measure of Engagement in Video Game-Playing. *Journal of Experimental Social Psychology* 45, 4 (2009), 624–634.

[5] Sara Bunian, Alessandro Canossa, Randy Colvin, and Magy Seif El-Nasr. 2017. Modeling individual differences in game behavior using HMM. (2017).

[6] Paul Cairns. 2016. Engagement in Digital Games. In *Why Engagement Matters: Cross-Disciplinary Perspectives of User Engagement in Digital Media*, Heather O'Brien and Paul Cairns (Eds.). Springer International Publishing, Cham, 81–104. https://doi.org/10.1007/978-3-319-27446-1_4

[7] Hao Cen, Kenneth Koedinger, and Brian Junker. 2006. Learning factors analysis–a general method for cognitive model evaluation and improvement. In *International conference on intelligent tutoring systems*. Springer, 164–175.

[8] Jenova Chen. 2007. Flow in Games (and Everything Else). *Commun. ACM* 50, 4 (apr 2007), 31–34. https://doi.org/10.1145/1232743.1232769

[9] Germán Cheuque, José Guzmán, and Denis Parra. 2019. Recommender Systems for Online Video Game Platforms: the Case of STEAM. In *Companion Proceedings of The 2019 World Wide Web Conference*. ACM, San Francisco USA, 763–771. https://doi.org/10.1145/3308560.3316457

[10] Mihaly Csikszentmihalyi and Mihaly Csikzentmihaly. 1990. *Flow: The psychology of optimal experience*. Vol. 1990. Harper & Row New York.

[11] Alena Denisova, Paul Cairns, Christian Guckelsberger, and David Zendle. 2020. Measuring perceived challenge in digital games: Development & validation of the challenge originating from recent gameplay interaction scale (CORGIS). *International Journal of Human-Computer Studies* 137 (May 2020), 102383. https://doi.org/10.1016/j.ijhcs.2019.102383

[12] Alena Denisova, Christian Guckelsberger, and David Zendle. 2017. Challenge in Digital Games: Towards Developing a Measurement Tool. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. ACM, Denver Colorado USA, 2511–2519. https://doi.org/10.1145/3027063.3053209

[13] Anders Drachen, Eric Thurston Lundquist, Yungjen Kung, Pranav Rao, Rafet Sifa, Julian Runge, and Diego Klabjan. 2016. Rapid prediction of player retention in free-to-play mobile games. (2016).

[14] Tobias Drey, Fabian Fischbach, Pascal Jansen, Julian Frommel, Michael Rietzler, and Enrico Rukzio. 2021. To Be or Not to Be Stuck, or Is It a Continuum?: A Systematic Literature Review on the Concept of Being Stuck in Games. *Proceedings of the ACM on Human-Computer Interaction* 5, CHI PLAY (Oct. 2021), 1–35. https://doi.org/10.1145/3474656

[15] Ilya Goldin, April Galyardt, et al. 2018. Most of the Time, It Works Every Time: Limitations in Refining Domain Models with Learning Curves. *Journal of Educational Data Mining* 10, 2 (2018), 55–92.

[16] Miguel Gonzalez-Duque, Rasmus Berg Palm, and Sebastian Risi. 2021. Fast Game Content Adaptation Through Bayesian-based Player Modelling. In *2021 IEEE Conference on Games (CoG)*. 01–08. https://doi.org/10.1109/CoG52621.2021.9619018

[17] Miguel González-Duque, Rasmus Berg Palm, David Ha, and Sebastian Risi. 2020. Finding Game Levels with the Right Difficulty in a Few Trials through Intelligent Trial-and-Error. In *2020 IEEE Conference on Games (CoG)*. 503–510. https://doi.org/10.1109/CoG47356.2020.9231548

[18] Stefan Freyr Gudmundsson, Philipp Eisen, Erik Poromaa, Alex Nodet, Sami Purmonen, Bartlomiej Kozakowski, Richard Meurling, and Lele Cao. 2018. Human-like playtesting with deep learning. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 1–8.

[19] Erik Harpstead and Vincent Aleven. 2015. Using Empirical Learning Curve Analysis to Inform Design in an Educational Game. In *Proceedings of the 2015*

Annual Symposium on Computer-Human Interaction in Play (London, United Kingdom) *(CHI PLAY '15)*. Association for Computing Machinery, New York, NY, USA, 197–207. https://doi.org/10.1145/2793107.2793128

[20] Fuxing Hong, Dongbo Huang, and Ge Chen. 2019. Interaction-aware factorization machines for recommender systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 3804–3811.

[21] Martin Jennings-Teats, Gillian Smith, and Noah Wardrip-Fruin. 2010. Polymorph: dynamic difficulty adjustment through level generation. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*. 1–4.

[22] Ildar Kamaldinov and Ilya Makarov. 2019. Deep reinforcement learning in match-3 game. In *2019 IEEE conference on games (CoG)*. IEEE, 1–4.

[23] Jeppe Theiss Kristensen and Paolo Burelli. 2019. Combining Sequential and Aggregated Data for Churn Prediction in Casual Freemium Games. In *2019 IEEE Conference on Games (CoG)*. 1–8. https://doi.org/10.1109/CIG.2019.8848106 ISSN: 2325-4289.

[24] Jeppe Theiss Kristensen, Arturo Valdivia, and Paolo Burelli. 2020. Estimating Player Completion Rate in Mobile Puzzle Games Using Reinforcement Learning. In *2020 IEEE Conference on Games (CoG)*. 636–639. https://doi.org/10.1109/CoG47356.2020.9231581

[25] Jeppe Theiss Kristensen, Arturo Valdivia, and Paolo Burelli. 2021. Statistical Modelling of Level Difficulty in Puzzle Games. In *2021 IEEE Conference on Games (CoG)*. IEEE, 1–8.

[26] R Lazzaro. 2004. Why we play games: 4 keys to more emotion. *Proc. Game Developers Conference 2004*. https://cir.nii.ac.jp/crid/1572543025651858816

[27] Jiayu Li, Hongyu Lu, Chenyang Wang, Weizhi Ma, Min Zhang, Xiangyu Zhao, Wei Qi, Yiqun Liu, and Shaoping Ma. 2021. A Difficulty-Aware Framework for Churn Prediction and Intervention in Games. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. ACM, Virtual Event Singapore, 943–952. https://doi.org/10.1145/3447548.3467277

[28] Conor Linehan, George Bellord, Ben Kirman, Zachary H. Morford, and Bryan Roche. 2014. Learning curves: analysing pace and challenge in four successful puzzle games. In *Proceedings of the first ACM SIGCHI annual symposium on Computer-human interaction in play*. ACM, Toronto Ontario Canada, 181–190. https://doi.org/10.1145/2658537.2658695

[29] Diana Lora, Antonio A Sánchez-Ruiz, Pedro A González-Calero, and Marco A Gómez-Martín. 2016. Dynamic difficulty adjustment in tetris. In *The Twenty-Ninth International Flairs Conference*.

[30] Tobias Mahlmann, Anders Drachen, Julian Togelius, Alessandro Canossa, and Georgios N. Yannakakis. 2010. Predicting player behavior in Tomb Raider: Underworld. In *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games*. 178–185. https://doi.org/10.1109/ITW.2010.5593355 ISSN: 2325-4289.

[31] Hee-Seung Moon and Jiwon Seo. 2020. Dynamic difficulty adjustment via fast user adaptation. In *Adjunct Publication of the 33rd Annual ACM Symposium on User Interface Software and Technology*. 13–15.

[32] Fausto Mourato, Fernando Birra, and Manuel Próspero dos Santos. 2014. Difficulty in action based challenges: success prediction, players' strategies and profiling. In *Proceedings of the 11th Conference on Advances in Computer Entertainment Technology*. 1–10.

[33] Dvir Ben Or, Michael Kolomenkin, and Gil Shabat. 2021. DL-DDA-Deep Learning based Dynamic Difficulty Adjustment with UX and Gameplay constraints. In *2021 IEEE Conference on Games (CoG)*. IEEE, 1–7.

[34] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[35] Johannes Pfau, Jan David Smeddinck, and Rainer Malaka. 2020. Enemy within: Long-term motivation effects of deep player behavior models for dynamic difficulty adjustment. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–10.

[36] Christopher Power, Paul Cairns, Alena Denisova, Themis Papaioannou, and Ruth Gultom. 2019. Lost at the edge of uncertainty: Measuring player uncertainty in digital games. *International Journal of Human–Computer Interaction* 35, 12 (2019), 1033–1045.

[37] Megan Pusey, Kok Wai Wong, and Natasha Anne Rappa. 2021. The Puzzle Challenge Analysis Tool. A Tool for Analysing the Cognitive Challenge Level of Puzzles in Video Games. *Proceedings of the ACM on Human-Computer Interaction* 5, CHI PLAY (Oct. 2021), 1–27. https://doi.org/10.1145/3474703

[38] Zhiyun Ren, Xia Ning, Andrew S. Lan, and Huzefa Rangwala. 2019. Grade Prediction with Neural Collaborative Filtering. In *2019 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. 1–10. https://doi.org/10.1109/DSAA.2019.00014

[39] Steffen Rendle. 2010. Factorization machines. In *2010 IEEE International conference on data mining*. IEEE, 995–1000.

[40] Steffen Rendle. 2012. Factorization Machines with libFM. *ACM Transactions on Intelligent Systems and Technology* 3, 3 (May 2012), 1–22. https://doi.org/10.1145/2168752.2168771

[41] Steffen Rendle. 2013. Scaling factorization machines to relational data. *Proceedings of the VLDB Endowment* 6, 5 (2013), 337–348.

[42] Steffen Rendle, Zeno Gantner, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2011. Fast context-aware recommendations with factorization machines. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. 635–644.

[43] Shaghayegh Roohi, Christian Guckelsberger, Asko Relas, Henri Heiskanen, Jari Takatalo, and Perttu Hämäläinen. 2021. Predicting Game Engagement and Difficulty Using AI Players. *Proceedings of the ACM on Human-Computer Interaction* 5, CHI PLAY (Oct. 2021), 1–17. https://doi.org/10.1145/3474658 arXiv: 2107.12061.

[44] Shaghayegh Roohi, Asko Relas, Jari Takatalo, Henri Heiskanen, and Perttu Hämäläinen. 2020. Predicting game difficulty and churn without players. In *Proceedings of the Annual Symposium on Computer-Human Interaction in Play*. 585–593.

[45] Richard M. Ryan, C. Scott Rigby, and Andrew Przybylski. 2006. The Motivational Pull of Video Games: A Self-Determination Theory Approach. *Motivation and Emotion* 30, 4 (2006), 347–363.

[46] Anurag Sarkar and Seth Cooper. 2017. Level difficulty and player skill prediction in human computation games. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, Vol. 13. 228–233.

[47] Mirna Paula Silva, Victor do Nascimento Silva, and Luiz Chaimowicz. 2015. Dynamic Difficulty Adjustment through an Adaptive AI. In *2015 14th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*. 173–182. https://doi.org/10.1109/SBGames.2015.16 ISSN: 2159-6662.

[48] Mack Sweeney, Huzefa Rangwala, Jaime Lester, and Aditya Johri. 2016. Next-Term Student Performance Prediction: A Recommender Systems Approach. *arXiv:1604.01840 [cs]* (Sept. 2016). https://doi.org/10.5281/zenodo.3554603 arXiv: 1604.01840.

[49] Penelope Sweetser and Peta Wyeth. 2005. GameFlow: a model for evaluating player enjoyment in games. *Computers in Entertainment (CIE)* 3, 3 (2005), 3–3.

[50] April Tyack and Elisa D Mekler. 2020. Self-Determination Theory in HCI Games Research: Current Uses and Open Questions. In *Proc. Conference on Human Factors in Computing Systems (CHI)*. ACM, 1–22.

[51] Marc Van Kreveld, Maarten Löffler, and Paul Mutser. 2015. Automated puzzle difficulty estimation. In *2015 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 415–422.

[52] Matheus Weber and Pollyana Notargiacomo. 2020. Dynamic Difficulty Adjustment in Digital Games Using Genetic Algorithms. In *2020 19th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*. IEEE, 62–70.

[53] Daniel Wheat, Martin Masek, Chiou Peng Lam, and Philip Hingston. 2016. Modeling perceived difficulty in game levels. In *Proceedings of the Australasian Computer Science Week Multiconference*. 1–8.

[54] Robert C Wilson, Amitai Shenhav, Mark Straccia, and Jonathan D Cohen. 2019. The eighty five percent rule for optimal learning. *Nature communications* 10, 1 (2019), 1–9.

[55] Su Xue, Meng Wu, John Kolen, Navid Aghdaie, and Kazi A. Zaman. 2017. Dynamic Difficulty Adjustment for Maximized Engagement in Digital Games. In *Proceedings of the 26th International Conference on World Wide Web Companion - WWW '17 Companion*. ACM Press, Perth, Australia, 465–471. https://doi.org/10.1145/3041021.3054170

[56] Alexander Zook, Stephen Lee-Urban, Michael R. Drinkwater, and Mark O. Riedl. 2012. Skill-based Mission Generation: A Data-driven Temporal Player Modeling Approach. In *Proceedings of the The third workshop on Procedural Content Generation in Games - PCG'12*. ACM Press, Raleigh, NC, USA, 1–8. https://doi.org/10.1145/2538528.2538534

4 Developing Playtesting Agents for New Content

One obstacle with estimating difficulty on entirely new levels is that there is little to no information about how players interact with the content. Game designers typically solve this *cold start* scenario by playtesting the content themselves or releasing it to a smaller player population before the full release. However, both playtesting methods can be relatively time-consuming and expensive, so the question is how this can be made fast and efficient.

This chapter addresses how such a method can be developed for puzzle games. In doing so, we contribute to answering the second research question of how we can predict the difficulty of levels that have little or no player data. In the following section, we introduce the playtesting method developed during this PhD project to explore the dynamics of the game levels. This will form the basis for predicting the difficulty of new levels in Chapter 5, which will combine methods from the previous chapter and this chapter.

4.1 Automated playtesting

Automated playtesting methods are becoming increasingly common and necessary to explore the complex systems present in today's games. In the case of puzzle games, it provides a way to test out multiple aspects of the gameplay, such as finding solutions that level designers did not think of or estimating the minimum of moves required to complete a level. Therefore, these methods play a crucial part in evaluating and predicting player behaviour on new content. The results in this section detail how such an agent has been developed for playtesting puzzle levels and the challenges of extracting relevant gameplay information.

4.1.1 Background

Creating a playtesting method for the games developed by Tactile Games requires considering the design choices of the games, which puts certain restrictions on viable methods.

One consideration is that the games were never designed with automated testing and tree search in mind, which in effect means that it is impossible to save and load the full state of the gameboard. In addition, as with any piece of software, there are bugs and unexpected behaviours that can further complicate employing playtesting methods, such as crashes or playthroughs not terminating when there are no available moves. MCTS and other similar methods that rely on simulating a whole episode and being able to reset to the current state easily are not viable in this setting.

Another significant limitation is the fact that there are no play traces of players. Recording the whole game state at each move for 1 million daily players requires efficient logging, and it has only recently been made possible in Tactile Games' games to retrace previous playthroughs accurately. This means that the playtesting agent can not be trained using imitation learning (e.g. Juliani et al. (2018); Tastan and Sukthankar (2011)) or supervised learning approaches (e.g. Ortega et al. (2013)).

Another consideration is the time it takes to evaluate the levels. If the level designers want immediate feedback on a new level or other changes, it should not take longer than 5 minutes to provide the relevant feedback. Alternatively, given that around 50 new levels are released bi-weekly, the evaluation could consider the levels in bulk just before the release time, with final, minor adjustments being suggested by a difficulty modelling method (e.g. Kristensen et al. (2021)). The results by Gudmundsson et al. (2018) showed that using 1,000 attempts compared to 100 attempts could reduce the variance. As mentioned by Poromaa (2017), this approach requires around 1M simulations of the environment in order to capture 100 attempts, and in the current implementation of the simulator, this would take around 2.5 hours.

The playtest agent introduced in Paper 3 (Kristensen and Burelli, 2020) assumes that similar limitations can generally be present in other types of games. It is based on PPO, and given that the performance can depend a lot on the implementation (Ilyas et al., 2018; Huang et al., 2022), the paper's focus is to demonstrate several strategies that allow the algorithm to work robustly in a typical industrial setting. Additionally, having full access to the game itself allows us to implement a number of improvements, such as no move limit on the levels and a more informative state space (i.e. outputting game mechanics rather than pixels, similar to the approach by Volz et al. (2018) for Mario levels). Paper 4 (Kristensen et al., 2020) further explores how the agent can be trained in a realistic scenario where time and accuracy are essential factors and evaluates the agent's performance by considering the correlation with the players' pass rate in new levels.

4.1.2 Results and conclusion

Paper 3 (Kristensen and Burelli, 2020) provides three suggestions for implementing an RL agent based on PPO in Lily's Garden:

1. Shuffling the colour layers in the state space.
2. Early termination of episodes.
3. Use of a soft action mask.

Since the mechanics pertaining to the colours are the same, the algorithm should not treat the colours differently. One reason this can improve learning is that it is a kind of data augmentation that can help the model learn to generalise and not get stuck in local minima (Risi and Togelius, 2020).

Early termination is found to be a crucial addition to ensure stability in learning during training. The idea is similar to what Liu et al. (2019) found for Angry Birds, where they prevent collecting harmful trajectories in the replay buffer by restarting the level early if certain game states are reached. In the study by Kristensen and Burelli (2020), it is also found that without this early stopping, the agent tends to learn policies that are overfitted to poor observation and only repeat one action, and typically an invalid one. Early stopping allows the agent to fill the replay buffer with meaningful actions early on, enabling more gradual learning towards the goal.

The problem of selecting the same action can be helped by employing an action mask. Since the simulator does not provide an action mask (which has since been implemented for Chapter 5), a hand-crafted action mask valid for the first 11 levels is used instead. However, while this leads to learning a good policy faster, the algorithm fails catastrophically within a few episodes. As an alternative, the action mask was included as a “soft” action mask where it was included in the state space instead.

The study also provides suggestions for the choice of hyperparameters, including an entropy coefficient of 0.01 to encourage exploration and an experience buffer size between 128 and 256 to allow for whole episodes to be included.

While these strategies help the agent learn how to play the first 11 levels, there are some limitations to this work. Firstly, an issue with shuffling the colours is that it slows down learning. A more efficient way could include invariance to colour in the neural network itself, such as including the permutation-invariant attention neuron (Tang and Ha, 2021). Another issue that is not considered is that imposing time limits when resetting early without including this information in the game state can lead to suboptimal policies (Pardo et al., 2018). Lastly, using action masks in reinforcement learning can be tricky, especially in policy gradient methods, since the action mask needs to be considered in the gradient used to update the policy model parameters (Huang and Ontañón, 2020). The last two issues have been considered in Paper 5 (Kristensen and Burelli, 2022) where complete information about the time and move limit has been included in the state space, and the action mask is both fully valid and implemented correctly in PPO.

Kristensen and Burelli (2020) only considers the first 11 levels, which are very introductory in nature. This limited scope means that it is hard to conclude whether the method is viable for modelling player behaviour on the more complex levels. Kristensen et al. (2020) therefore extend the analysis to include the first 120 levels where the first 100 are used for training. Furthermore, three ways of training the agent are considered:

1. **One-step curriculum**: The agent is trained on randomly sampled levels during training.
2. **One-step level**: The agent is only trained on the specific level to evaluate.
3. **Two-step**: The agent is first trained on randomly sampled levels, then trained again on the specific level to evaluate.

The results highlight the difficulty of using AI for playtesting: while the best performing approach in terms of moves taken to complete the levels is the two-step approach, the correlation with player data is lower than with the other approaches. Conversely, the one-step curriculum approach performs the worst out all RL agents in terms of moves but the best in terms of player correlation. Furthermore, the strongest correlation with player data is when only the 5% best evaluation runs are considered, with similar results being found subsequently by Roohi et al. (2021).

The one-step curriculum model is the strongest contender for being used in an industrial setting since it can be used immediately without additional training and shows the strongest correlation with player behaviour. However, there is still room for improvement since a more competent agent could speed up playtesting further by completing levels faster and thus allowing for more playthroughs, provided its performance is consistent and still correlated with player data. A key is ensuring the agent learns to generalise (Cobbe et al., 2019) and possibly create a curriculum that does not rely on random sampling to allow it to learn to play more challenging levels (Justesen et al., 2018).

4.1.3 Relevant paper(s)

The results and discussion are based on Paper 3, *Strategies for Using Proximal Policy Optimization in Mobile Puzzle Games* (Kristensen and Burelli, 2020), and Paper 4, *Estimating Player Completion Rate in Mobile Puzzle Games Using Reinforcement Learning* (Kristensen et al., 2020).

Paper 3 was submitted to the 2020 Foundations of Digital Games conference as a full-length paper and has been accepted and presented. It has been co-authored with the principal university supervisor, Paolo Burelli. I am responsible for conducting the experiments and contributing to all of the text. All co-authors have been involved in the discussion and editing of the final structure of the paper.

Paper 4 was submitted to the 2020 IEEE Conference on Games as a short paper and has been accepted and presented. It has been co-authored with the principal university supervisor, Paolo Burelli, and the principal company supervisor, Arturo Valdivia. I am responsible for conducting the experiments and contributing to all of the text. All co-authors have been involved in the discussion and editing of the final structure of the paper.

4.2 Paper 3: *Strategies for Using Proximal Policy Optimization in Mobile Puzzle Games*

Strategies for Using Proximal Policy Optimization in Mobile Puzzle Games

Jeppe Theiss Kristensen
IT University of Copenhagen/Tactile Games
Copenhagen, Denmark
jetk@itu.dk

Paolo Burelli
IT University of Copenhagen/Tactile Games
Copenhagen, Denmark
pabu@itu.dk

***Abstract*—While traditionally a labour intensive task, the testing of game content is progressively becoming more automated. Among the many directions in which this automation is taking shape, automatic play-testing is one of the most promising thanks also to advancements of many supervised and reinforcement learning (RL) algorithms. However these type of algorithms, while extremely powerful, often suffer in production environments due to issues with reliability and transparency in their training and usage.**

In this research work we are investigating and evaluating strategies to apply the popular RL method Proximal Policy Optimization (PPO) in a casual mobile puzzle game with a specific focus on improving its reliability in training and generalization during game playing.

We have implemented and tested a number of different strategies against a real-world mobile puzzle game (Lily's Garden from Tactile Games). We isolated the conditions that lead to a failure in either training or generalization during testing and we identified a few strategies to ensure a more stable behaviour of the algorithm in this game genre.

***Index Terms*—reinforcement learning, ppo, player agent, player modelling, playtesting, autonomous agent**

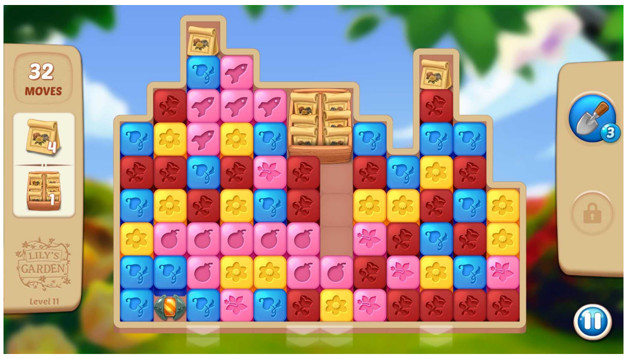


Fig. 1. Level 11 from Lily's Garden. The left hand side shows number of moves left to finish the level, and the board pieces below indicate which and how many pieces to collect before completing the level. Collecting the objectives is done by clearing them off the board, which can be done by clicking on two or more basic pieces of the same color, or using power pieces that clear an entire row/area. Power pieces can be created by matching 5 or more basic pieces.

I. Introduction

Human game testing is an expensive and slow process. It usually requires the full attention of the testers, and there are limitations on how fast humans can operate. Game developers are therefore increasingly starting to use automated play testing. However, developing and implementing such methods in practice has its problems – the methods tend to require a very specific setup for one game, and trying to adapt it to other environments may sometimes break the algorithm and render it useless. In this paper we therefore set out to explore both novel and common strategies for ensuring a stable implementation of a reinforcement learning (RL) play-testing agent in a mobile puzzle game in a production setting.

A popular choice for creating play testing tools is reinforcement learning, and research in this field is moving fast. Novel algorithms and updates to current state-of-the-art methods are constantly being introduced in the latest publications, showing better performance on typical frameworks such as the Arcade Learning Environment [3].

However, contrary to these kind of one-shot evaluations, adapting these methods in a production environment in a company requires additional considerations – such as ease-of-use and long-term reliability. Unlike these benchmark games, production games are updated frequently, and it can not be expected to be possible to draw on expert knowledge at any time in case something goes wrong. Until more focus has been put on strategies on *how* to use these methods, adoption of these methods in the industry will be slow at best.

In this research work we focus on the challenges of implementing the popular RL method Proximal Policy Optimization (PPO) [30], a widely used algorithm available in various RL libraries (OpenAI Baselines/stable-baselines [6], [15], TF-Agents [9], Unity ML-Agents [18]), in a mobile puzzle game called Lily's Garden by Tactile Games (Fig. 1). While other RL methods may also work in this environment, we choose to only focus on PPO since it is one of the two main algorithms implemented in Unity ML-Agents and thus widely accessible to game developers that use Unity.

Our contribution is two-fold:

- We explore different setups for training an agent in a mobile puzzle game and determine a set of hyperparameters and setups that enable the agent to some extend play both seen and unseen levels competently.
- We highlight that the impact of some PPO variations are not fully understood and can easily lead to unexpected learning behaviours. We then suggest strategies for avoid-

ing such behaviours and ensure a more stable training.

This paper is structured as follows: First we introduce the game environment that we will use for testing. Next we present the basics of the PPO algorithm and discuss the specific implementation we use. This is followed by the experiments section where we present the various setups we tested and highlight the main difficulties and problems encountered during training of the agent. Lastly we discuss which methods and strategies that are feasible to employ in a production setting and identify areas that need improvement.

II. Related work

When it comes to creating agents for playing games, reinforcement learning (RL) and deep learning methods have started to become a staple and have been used to play a large variety of games, ranging from arcade games to first-person shooter games [19].

Each genre has its own challenges, and some approaches work better than others in different settings. It is therefore relevant to consider which approaches that have been used for play-testing in similar game genres.

In Atari games, some of the state-of-the-art approaches using pixel data or memory-features as input are deep Q-learning (DQN) [25], [26] and variations thereof (such as Rainbow [14]), and actor-critic approaches like PPO [30] and soft actor-critic [11] in [4].

The *MuZero* algorithm introduced by Schrittwieser et al. [29] uses a combination of tree-search planning and a learned model of the environment and is capable of playing Go, Chess, Shogi and the Atari games. However, how to deal with stochastic transitions was not examined.

As for approaches used specifically on puzzle games, other approaches have also been directly applied. Gudmundsson et al. [10] treat the task as a classification problem and train a convolutional neural network on player data. Their method beats state-of-the-art Monte-Carlo Tree Search algorithms in terms of difficulty prediction and training time and has been used actively for a year by the time of publication. However, this method requires play-through data which may not always be available. Mugrai et al. [27] use a MCTS method with an evolutionary strategy where the fitness function is used to mimic specialised player personas/strategies with different goals, such as maximising score or minimising moves used. This aspect of creating human-like agents is indeed important if they are to be used as a play-testing tool, which is also highlighted by Zhao et al. [36] where the agents are evaluated by considering both skill and style. A comparison of three popular methods (DQN, PPO and A3C) in a custom match-3 game is done by Kamaldinov et al. [21] which shows that the A3C method achieves the highest accumulative reward while the PPO and DQN methods perform worse than random. They use a custom match-3 environment, though, so it is not clear if these results reflect real-world results in puzzle games. An example of training an agent using actual games levels can be seen in the Unity blogpost [33], where an agent for playing Snoopy Pop using ML-Agents in [33] is attempted using a actor-critic method (SAC, [11]) and imitation learning (GAIL, [16]). Although a slightly different genre, it shows that the training can be sped up using sample efficient methods and a player to guide the agent initially. However, a similar approach is not efficient in games like Lily's Garden since in those cases it is not necessary to simulate physics. Furthermore, there are more than 1500 levels available so deciding which levels to train on or alternatively have a player play through all of them is not scalable.

When it comes to using such automated systems in a production setting, reliability and accessibility of the algorithm are critical components. The less interference required, the better, and when something goes wrong, identifying the points of failure easily is important so it can be fixed quickly and not waste resources. RL systems, especially PPO approaches, tend to be the antithesis of these requirements: they tend to be brittle [12], and the stability tends to be implementation-dependant [17]. It is therefore important to consider not just the algorithms but also the strategies of how a play-testing tool should be developed.

Such a tool also needs to be able to generalise to new levels, and one problem that appears in many RL papers is overfitting to an environment [35]. Ways to diagnose and improve the generalisation in deep RL systems have been examined is various works [28], [35]. Farebrother et al. [7] find that dropout and $\ell 2$ regularisation with a DQN method improve generalisation. This is also supported by the findings by Cobbe et al. [5] where data augmentation, batch normalisation and stochasticity were also found to improve generalisation in an implementation of PPO. Adding entropy regularisation also helps find smoother solutions but is very environment dependent [1]. Variations in the levels by using procedural content generation methods can also improve generalisation and help learn more difficult levels [20]. Avoiding undesirable and dangerous actions may also help the agent learn more efficiently because of better and safer exploration strategies [34]. Having the system learn which actions to eliminate has been the focus in some recent works [2], [31], [34]. In addition to learning action blocking, Kenton et al. [22] also use an ensemble model in both a DQN and PPO setup. While the DQN method showed improvements, the PPO experiments showed little improvement compared to the baseline.

III. Environment

In this paper we focus on one game, Lily's Garden[1]. It is a free to play casual puzzle mobile game where you progress through the main story by completing levels. The main gameplay is matching similar colored pieces and thereby collecting objectives (collectgoals), which must be done before running out of moves. The game board has a maximum size of 13 by 9, and in each position, board pieces with various attributes may be placed. The basic pieces can be destroyed/collected if two or more of the same color are next to each other and will create power pieces if 5 or more are

[1]Android, Apple

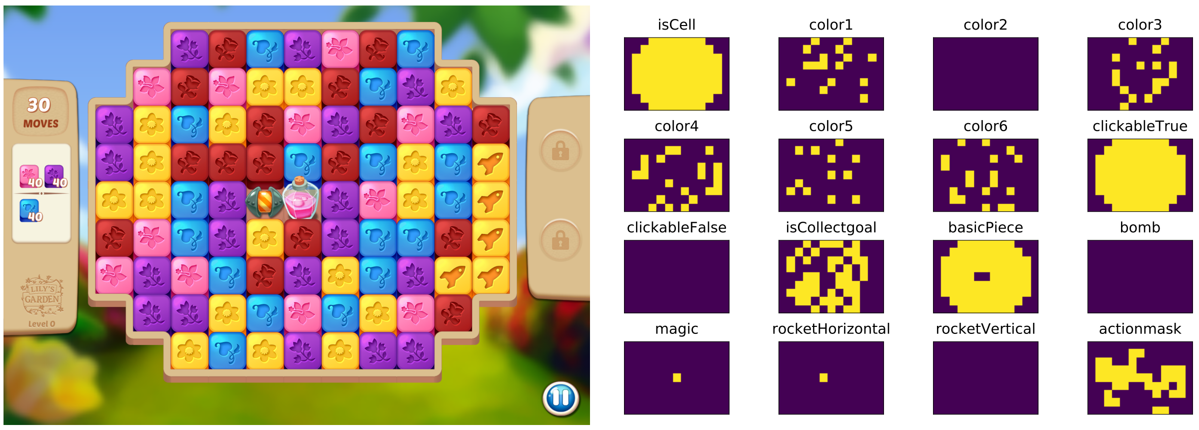


Fig. 2. Example of how an in-game level looks like and how the game board is represented using different channels corresponding to certain board piece attributes. Note that the last channel, the action mask, is only included in certain experiments.

next to each other. The power pieces can be clicked at any time and destroy everything in for example a line or circle around the position. Lastly there are unclickable board pieces, or blockers, that can be removed by matching basic pieces next to it or sometimes only by using a power piece. An example of a level is shown in Fig. 1.

We set up an OpenAI gym environment that connects to headless version of the game (no graphical interface) which, for speed purposes, allows us to play through levels without rendering any graphics. We define a rich reward function where the reward is calculated at each step as: $r = c_{\text{collection}} n + c_{\text{completion}} - 0.1$, where $c_{\text{collection}} = 0.05$, n is the number of collected collectgoals and $c_{\text{completion}} = 1$ if all collectgoals have been collected. The negative term, -0.1, is added to encourage the agent finishing the level faster as to not get a large negative accumulated reward. Given that a typical level has around 50 collectables and requires up to 25 moves to complete, the expected final reward is $R \approx 50 \cdot 0.05 - 25 \cdot 0.1 + 1 = 1$ (not considering discount).

Since each board piece may be of the same type (e.g. basic or blocker) but different attributes (e.g. color or gravity), one-hot encoding each board piece by the unique combination of attributes may lead to a very large and sparse representation, as seen in [10]. Instead we choose to represent the observation space by using layers that correspond to the attributes of all the board pieces in a given position (see Fig. 2). Specifically, we represent the following attributes with a layer giving a total of 15 channels:

- ISCELL: used to define shape of game board
- COLOR: one-hot encoding of 6 unique colors
- ISCOLLECTGOAL: if board piece is a collectgoal
- ISCLICKABLE[TRUE/FALSE]: one layer for clickable, another for non-clickable since a non-clickable piece may be on top of another
- ID: one-hot encoding of BASICPIECE, ROCKETHORIZONTAL, ROCKETVERTICAL, BOMB and MAGIC

This approach also has an advantage when it comes to generalizability for future versions because the observation space will not depend on graphics updates and new types of board pieces are typically made up of a combination of existing attributes. The action space consists of $9 \times 13 = 117$ discrete actions, corresponding to each square of the game board.

IV. METHODS

The typical reinforcement learning problem consists of an agent that interacts with an environment and receives a reward depending on the action. This loop may then continue indefinitely or until the episode ends. The main purpose of the algorithm is then to learn a behaviour that maximises the accumulated reward [32].

In the original form, PPO refers to a family of policy gradient methods that optimize a (clipped) surrogate objective function using multiple minibatch updates per data sample. However, the exact implementations in various libraries may be slightly different because of other additions such as value scaling or batch normalisation [17]. Common for them all is the suggested function to optimise, which is the sum of several loss functions and is given by

$$L^{CLIP+VF+S}(\theta)_t = \hat{\mathbb{E}}_t \left[L^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](\theta) \right], \tag{1}$$

where $L^{CLIP}(\theta)$ is the clipped surrogate objective function, $L_t^{VF}(\theta)$ is the value function squared-error loss, S is an entropy bonus and c_1 and c_2 are coefficients. The $L^{CLIP}(\theta)$ term ensures that the policy updates will not be too large, and the $L_t^{VF}(\theta)$ term is to ensure that the loss from both policy and value functions of the neural networks are accounted for. The S entropy term encourages a more random policy (i.e. more exploration) so a larger entropy coefficient c_2 will encourage more exploration.

A. *Implementation*

Since we want to investigate strategies for implementing PPO in a production environment, we choose to go with a widely used code library. Some of the notable RL libraries are

OpenAI Baselines[2] and Unity ML-Agents[3]. For the following experiments we choose to use stable-baselines, which is a fork from OpenAI Baselines but follows the same algorithmic implementation of PPO.

We test out three different strategies which will be described in this section. These strategies are:

- Color shuffling (CS)
- Resetting
- Action mask

Color shuffling refers to swapping the color channels in the observations randomly. While color shuffling is done in the post-training evaluation for all models to simulate how levels are designed, we want to test how effective it is to also include this strategy during training. It should also help prevent overfitting – even though it is random which board pieces that drop down and replace cleared pieces, the initial setup are usually predetermined (see Fig. 4) which may lead to strong overfitting.

Resetting the environment commonly happens at the end of episodic environments, which in this case could be when the level is completed or failed. However, the level move limit is subject to change because of design considerations, and we already add a penalty at each step to encourage it finishing faster. Imposing a move limit does therefore not make much sense. What we do try with the reset strategy, though, is imposing a total step limit, which includes both valid and invalid moves. The reasoning behind this strategy, similar what is given in [24] using restarts in Angry Birds, is that the agent is prevented from exploring useless states that it will not learn anything from.

Before deciding what the maximum episode length could be, two things should be considered. One is how the typical PPO implementation samples observations. In our case, we sample 256 observations before training on these minibatches. This means that if we reset after 256 total steps, we may end up with a full minibatch of bad training samples, which is undesirable. Secondly the typical steps required to pass a level is generally around 50. Levels that require more steps are rare since it would be very frustrating for players to almost finish a level but ultimately fail after, say, 100 steps rather than 50. We therefore choose to reset after 100 steps which should ensure at least some good observations and still allow the agent to complete a level.

Using an action mask during training is the last strategy we explore. While we give a penalty for selecting an invalid action, preventing the agents from selecting certain catastrophic or invalid actions may lead to more efficient learning. The question is how this limitation should be implemented. We use two different approaches for creating action masks in the following experiments – a hard and a soft action mask.

With the hard action mask, the invalid actions are completely masked when sampling from the policy distribution. In practice, this is done by adding the mask to the logits of policy distribution, where valid actions have a value of 0 and invalid action a value of $-\infty$. This is slightly different than in the ML-Agents library where a small probability ϵ is added to the action probabilities which prevents ∞ values but also allow invalid actions to be taken, albeit with a very low probability. The way sampling is done in the stable-baselines library is by using a Gumbel-max trick.[4] Specifically, noise following a Gumbel distribution (computed by taking the negative logarithm twice of uniformly distributed noise) is added to the logits which ensures the sampling will follow the underlying probabilities of the actions.

The soft action mask is a kind of forward model of the environment. Specifically we add the action mask to the observation space as an additional channel, as illustrated in the last panel in Fig. 2. The reason for calling this a soft action mask is because it does not directly prevent invalid actions from being taken although it might significantly reduce the probability. The soft action mask model is denoted with V2.

Since the game simulator does not provide a method for getting the action mask, we define it ourselves. It follows the basic rules that an action is valid if at least two BASICPIECES are adjacent and of the same color, or if there is a power piece in the cell. While this is not true for later levels, it is sufficient for the first 11 levels that we test on.

Lastly, we also want to evaluate if it makes a difference to continue training after the learning curves have plateaued since shorter training times allow for quicker iterations and thus easier testing in a production environment. These long-trained models are denoted in the post-training evaluation figures with (late).

V. Experiments

We carried out a number of experiments to test the performance of the PPO algorithm in our environment. We used the PPO2 implementation from the Python RL library stable-baselines [15] and a custom CNN policy (Fig. 3).

Each of the experiments are evaluated similarly to [5] where the trained agent is tested on unseen levels in order to evaluate

TABLE I
OVERVIEW OF MODELS AND USED ENTROPY COEFFICIENT (EC) AS WELL AS WHICH TRAINING STEP CHECKPOINT USED FOR POST-TRAINING EVALUATION. THE SECTION IN WHICH THE RESULTS OF SAID MODELS ARE ALSO SHOWN. CS: COLOR SHUFFLE.

Model	**EC**	**Step** ($\times 10^6$)	**Section**
Baseline	0	0.35	VI-A
Baseline	0.001	0.20	VI
Baseline	0.01	11	VI-A
CS	0.001	0.20	VI-A
CS	0.01	14	VI-A
CS+reset	0.01	0.35	VI-B
CS+reset+mask	0.01	6.5	VI-C
CS+reset+maskV2	0.01	4.0	VI-C
CS+reset (late)	0.01	14	VI-B
CS+reset+maskV2 (late)	0.01	14	VI-C

[2]https://github.com/openai/baselines

[3]https://github.com/Unity-Technologies/ml-agents

[4]https://github.com/hill-a/stable-baselines/blob/a57c80e0636582995d602309d2ea5547c0d58e61/stable_baselines/common/distributions.py#L323

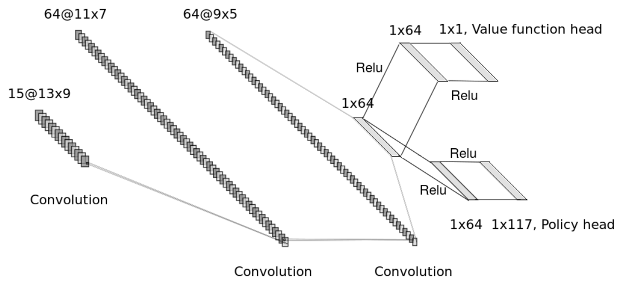


Fig. 3. The network architecture of the agent.

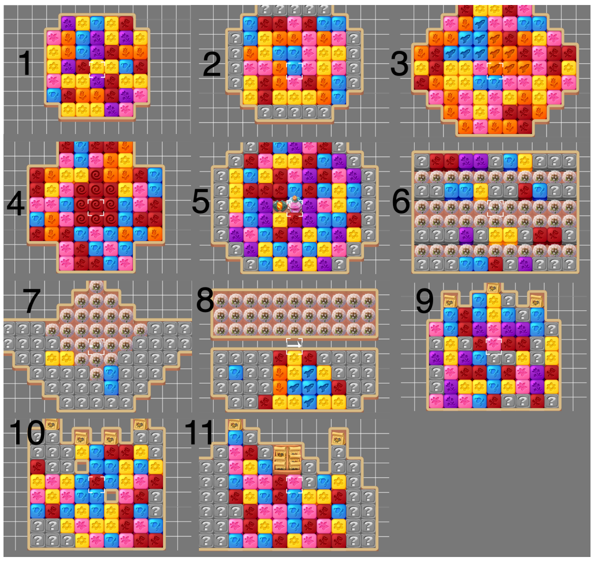


Fig. 4. Levels used for the experiment. Board pieces with question marks are assigned a random color on level start, while every other board piece is hardcoded.

its ability to generalize. This is done by training on 5 chosen levels (1, 3, 5, 7 and 9) selected randomly and uniformly during training and validated using an additional 6 levels (2, 4, 6, 8, 10 and 11). With the exception of level 11, these levels include two unique blockers, and splitting the levels accordingly ensures both that the tutorial levels and trained on and both the training and test sets will include at least one level containing any of the blockers. Level 11 has a third unique blocker so we include that level in the evaluation to see how the agent performs with completely unseen mechanics. An overview of the levels is shown in Fig. 4.

A. Evaluation Metrics

During training we consider the accumulated reward/learning curve as the evaluation statistic. For the post-training evaluation we do not want to only estimate if the agent can finish the level within the actual in-game max moves but also how competent it is compared to a random agent. We therefore allow up to 2000 total steps and do not use an action mask. We also shuffle the colors during evaluation for all models in order to simulate actual in-use performance, since the different colors of the board pieces only affect the aesthetics of the game and are used interchangeably.

We will consider two post-training evaluation metrics:

Competence is the reciprocal average index (starting with 1) of the first valid action after sampling actions using the action probabilities without replacement. Taking the reciprocal value corresponds to estimating the average valid step percentage and can be thought of as a proxy for how well the agent understand the basic match-2 mechanic of the game.

Level completion percentage is calculated by imposing the level move limit on the agent. We also include actual player data. It should be noted that the player completion percentage is estimated by taking the number of level completions over total number of level attempts. However, the level attempts include successes, failures *and* abandoning the game, where the latter may happen if the game for example crashes, other technical failures or simply just giving up on a level. Abandoning the game typically happens less than 5% of the time, though, so this effect should be minor.

B. Model Setup

We did a preliminary analysis training various models with different hyperparameters to find a stable configuration. While we also experimented with reward shaping and state representations, the key changes required to get the PPO algorithm to work with Lily's Garden was changing the minibatch size, number of steps per update and number of actors. We found that setting nminibatches to 64 (default: 4), n_steps to 256 (default: 128) and the number of parallel actors is set to 8 gave a good balance between speed and stability of the algorithm. This is not unexpected as these changes from default ensure a smoother gradient and faster and more stable training [13] and thus more stable training.

We use a custom neural network setup as shown in Fig. 3. It uses three 2x2 convolutions with filter size 64 and leaky relu activations, which are fed into two fully connected 64 layer for the actor and critic heads respectively.

The above hyperparameters are kept the same throughout the experiments except for the entropy coefficient, which will be discussed further in Section VI and VI-A. That setup will serve as a baseline model where no special strategies for training are used. For the other experiments, we use the three aforementioned strategies in Section IV-A.

VI. Results

The learning curves for every model can be seen in Fig. 5, the valid move percentages in Fig. 6 and the completion rate in Fig. 7. Table I shows at which step each model was evaluated as well as in which of the following sections they are discussed further.

In this section, we only consider the baseline model with an entropy coefficient (EC) of 0.001. The other two baseline models are discussed in the next section.

Looking at the learning curve of the Baseline EC: 0.001 model in Fig. 5, it can be seen that the agent quickly learns as reflected in the increase in episode rewards. However,

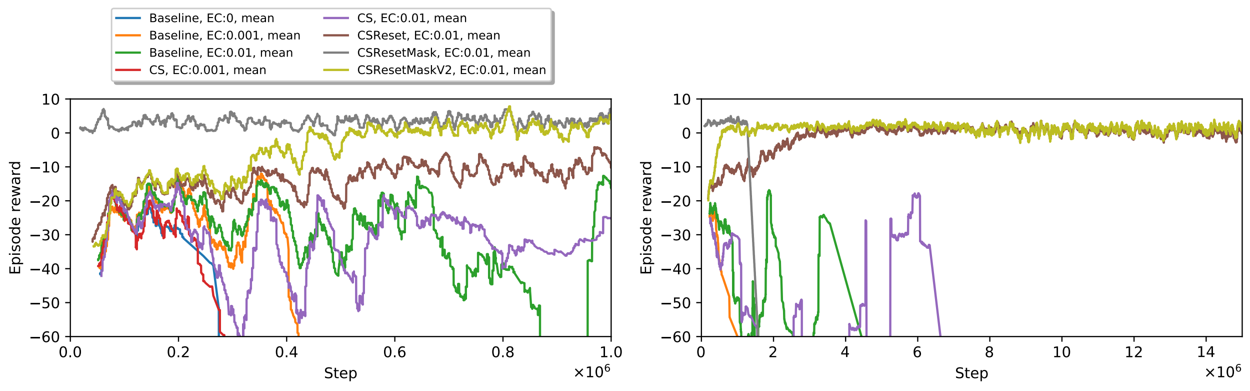


Fig. 5. Learning curves for the tested model. The left figure shows a zoomed in version on the first 1 million steps. CS refers to models trained with color shuffling.

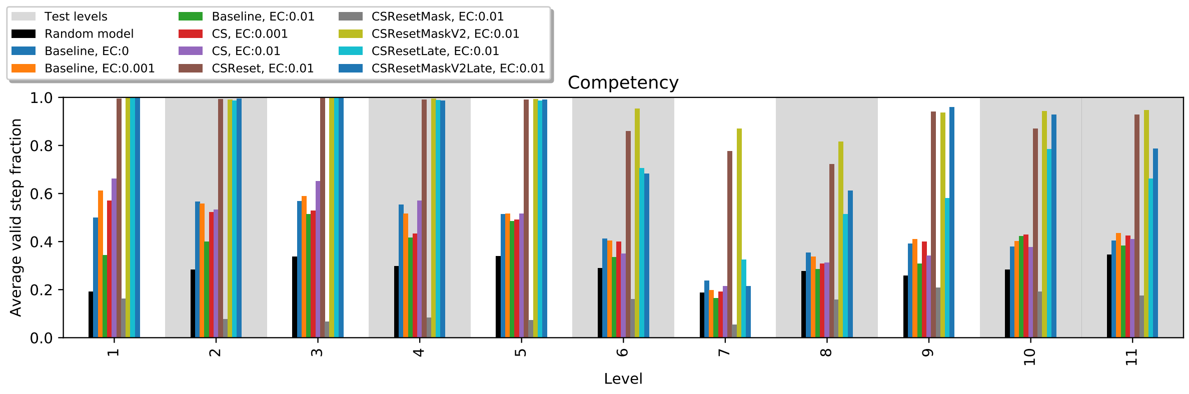


Fig. 6. Number of valid moves per level for each model. CS refers to models trained with color shuffling. The grey shaded levels are the unseen test levels

after 400.000 steps, the episode reward sharply decreases and completely breaks the training. The same behaviour was also observed in other experiments during the initial analysis. This happens when the action entropy becomes sufficiently low which indicates is that the agent ends up picking the same bad action and fills up the training samples with bad observations. The problem is further compounded by the fact that invalid actions do not change the state of the game and we do not do anything to prevent the algorithm from selecting invalid actions, leading to identical training data samples and thus broken learning.

On Fig. 6 and 7 it can be seen that while the agent generally picks valid actions and completes the levels more often than the random agent, it does not reach human-like performance on both seen and unseen levels after level 2.

A. *Generalisation*

The observation that the agent does not reach human level performance and sometimes also get stuck on an invalid move may indicate that the agent does not explore sufficiently. One way to increase exploration with a PPO algorithm is to increase the entropy coefficient which adds an entropy bonus to the loss function (c_2 in Eq. (1)). Three different configurations were tested: 0.0, 0.001 and 0.01, where 0.001 is the default value.

Generally the learning curves are very similar but the higher the entropy coefficient is, the longer the agent can be trained for and the less likely it is to encounter catastrophic learning behaviours.

Adding color shuffling should also help the agents generalise because it adds randomness. Indeed, the completion percentage for the CS, EC:0.01 model on level 3 and 4 is better than any of the baseline models and comparable on the other levels. While it should be noted that the model had been training for longer, this was made possible because of the higher entropy coeifficient and more environment stochasticity. Color shuffling therefore seems to be a viable strategy in addition to a high entropy coefficient.

B. *Max Episode Length*

Using strategies that add randomness and increase exploration are not enough to prevent the agent from sampling the same move over and over again as evidenced by the previous experiments. We therefore try the strategy of resetting the

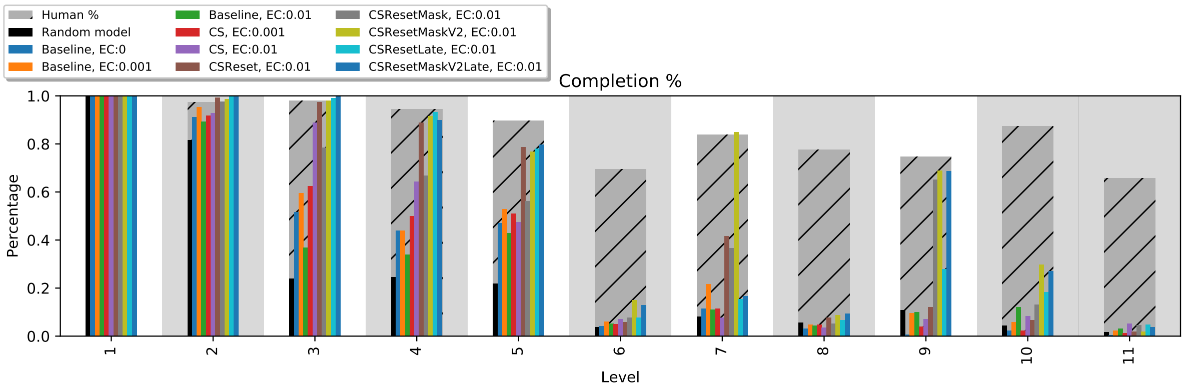


Fig. 7. Completion rate for each model given the level limit. CS refers to models trained with color shuffling. The unseen test levels are shown with a grey shaded area as in 6, while the human completion rate is shown as the large grey hatched bar.

environment to break the loop if a bad learning behaviour happens.

It should be noted that it is difficult to compare the learning curves of agents trained with reset and those without, since resetting ensures that it is not possible to accumulate large negative rewards from choosing the same invalid action over and over again. However, the learning curves are still useful for verifying that the agent is improving and not encountering catastrophic learning behaviour.

Using the reset strategy has a large positive impact on the learning. None of the agents that employ this strategy run into the same loop of selecting the same action all the time which enables the agent to train longer and learn more, with the exception of the CSResetMask model which will be described in the next section. Resetting the environment after a number of steps is therefore a good strategy that leads to more stable learning.

C. Action Masks

Since none of the other experiments directly prevent invalid actions to be taken, the agent has to first learn to infer which moves that are valid. We therefore test two different ways of adding this information – a hard mask and a soft mask, dubbed V2, as described in Section IV-A.

Using a hard action mask very quickly leads to high rewards which makes sense since invalid actions lead to a -0.5 penalty but are never taken now. However, as far as stability goes, the training completely fails after around 1.5 million steps, as seen in the sharp drop in the learning curve on Fig. 5.

Unlike what was seen in many of the previous experiments when the entropy becomes very low/zero, it now receives undefined rewards, indicating something with the algorithm itself is failing. What is happening is that the action probability distribution from the policy is 100% of an invalid action, and 0 on the rest, but because of the hard action mask, the final logits distribution is filled with $-\infty$. Taking the maximum of this vector then leads to unexpected behaviour. This is supported by the fact that the trained agent is actually not very competent (even worse than random, Fig. 6) and thus tend to select invalid actions first.

The picture is completely different when using it as a soft action mask. Looking at Fig. 5, the CSResetMaskV2 agent is both stable during training and learns faster compared to the CSReset agent (i.e. they reach the same learning plateau after 0.5M and 4M steps, respectively). It also has better completion rate and competency on both test and training levels than any of the other approaches.

VII. Discussion

The most effective strategy for training seems to be resetting the environment after a number of total steps. Color shuffling together with an increased entropy coefficient are also strategies that help the agent learn despite slowing down the training. Shifting towards more exploration and less exploitation in games like Lily's Garden therefore seems to be beneficial.

Some of the strategies did not work very well, though, like using a hard action mask or training for too long. This gives rise to some concerns if used in a production environment and will be discussed below.

A. Dealing With Invalid Actions

The main issue encountered throughout the experiments was invalid actions, which may be very specific to our environment and implementation of PPO. For example, in ML-Agents a small probability ϵ is added to the raw probabilities ensuring that there will be no $-\infty$ when converting to logits. This avoids the hard action mask problem, but it can be argued that it is not a hard action mask anymore. Other ways to deal with sampling the same action over and over could be to use an epsilon-greedy approach or by sampling the way we did it in post-training evaluation but this significantly slows down the training.

While this problem with invalid actions may be a very specific problem to our environment, it still highlights a possible issue that may arise in other similar games where some actions do not progress the game. Additionally, if a

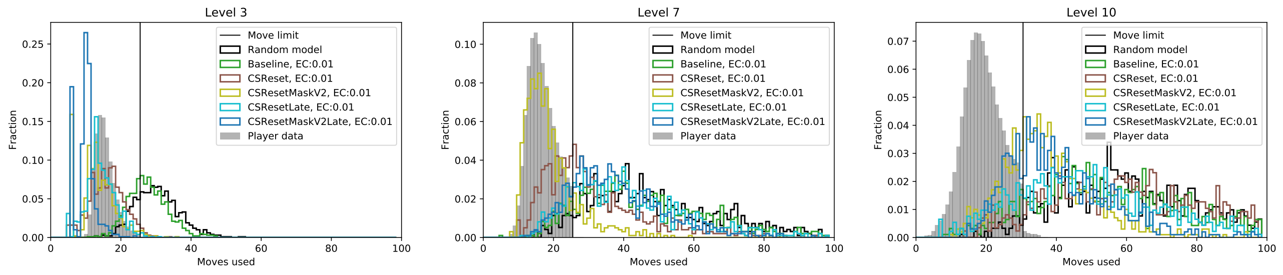


Fig. 8. Histograms of how many moves required to finish selected levels for selected agents, including the random agent in black, that show super-, normal and sub-human performance. The grey shaded area is actual player data and is the distribution that we want to mimic. A sharp cut-off can be seen in the player data distributions which is aligned with the move limit. The reason a small tail can be seen in level 10 is because players are able to purchase an additional 5 moves if they fail, but only a fraction of players choose to do so. The normalisation is therefore also not completely comparable since the agents are allowed to play past the move limit.

hard action mask is being used, the algorithm runs the risk of masking out every action, leading to unexpected behaviour. This is an issue since in a number of research papers on play-testing agents it is not clear how the action masking is actually being done even though it has a huge impact on the training of the agent. This adds further complexity to understanding the algorithms and reflects the thoughts in [17] that the implementation matters.

One question is whether using the action mask is practical in the long run since it does require some kind of modelling of the environment. Additionally, while the levels considered in this paper do not have very complex game mechanics, later levels include mechanics that prevent certain actions. While the environment could be configured to return a proper action, this may prove computationally and developer-time intensive and therefore not viable in the long run. However, interestingly enough it was found out during the evaluation process that the action mask in some very specific cases allowed invalid actions. Whether that is because a bug in the game or action mask modelling is not clear, but it was interesting that this was not a problem when using the soft action mask. This suggest that using even a imperfect forward model of the environment still improves learning.

B. *Usefulness in Production*

There are two things to consider before judging if the agent is actually useful to level designers.

The first question is whether the level designers would be able to rely on the agent or not. For that to be the case, the more consistent and performant it is, especially on unseen levels, the better. However, what is observed is that the completion rate is worse on unseen levels. This limits the usefulness to level designers since the new levels will obviously not have been encountered before. One solution for this could be to allow the agent to train on the unseen levels. To see whether this is feasible in a production setting requires further testing.

One other thing to take note of is the fact that the completion rate is low despite picking valid action most of the time. This suggests that the agents learn how to play the game but not how to play it optimally. This may be a consequence of the reward function, though – a relatively big penalty is given for selecting invalid moves compared to collecting objectives. The first thing the agents learn is therefore how to not take an invalid action. Learning new things, such as going after objectives, is secondary and would require more training without overfitting. The best way to achieve this would be to introduce more levels, which should help with generalising and making the agent more consistent.

The second thing to consider is that it must play like a human and not superhuman, so the estimated difficulty matches with how players perceive it. While the completion percentage used in the post-training evaluation already reflects this aspect, it does not tell the whole story. Another way to judge how human-like the agent behaves is by looking at the distribution of moves required to finish the level (Fig. 8) and comparing with human data. This kind of visualisation is also more useful to level designers since it can be used to determine the move limit. However, none of the models are consistent in being super/sub-human which must be addressed first.

C. *Future Work*

When an agent first tries to learn how to play these puzzle games, it first needs to figure out how to do a valid move. As revealed by using an action mask, it learns much faster if something guides it initially. One way could therefore be to use imitation learning to first teach it how to do the basics. This also has the added benefit that it may be easier to guide the agent to play more like a human which would make the tool more useful to level designers. It would require some time and effort to set this up in practice, though, both in regards to implementing it in production code but also the time level designers would have to spend training the agent. Evaluating which approach is more time-effective should therefore not only include computation time but also the human resources required. However, an imitation learning module has been added to ML-Agents and may provide a good starting point.

The post-training evaluations show that the agents play some levels well but struggle with others. It therefore seems like a better strategy to spend more time training on the difficult levels rather rather than continuing selecting the levels randomly. An idea could be an automated approach like in [8] where the training examples that yield the most learning are chosen. This would also open up for training on more levels which should help generalisation of the agent on unseen levels. One thing to keep in mind before training on many new levels and mechanics, though, is that the agent may be prone to catastrophic forgetting [23] where previously learned behaviours are completely forgotten.

VIII. Conclusion

In this research paper we have successfully adapted the popular RL method PPO to a production grade puzzle game for training play-testing agents. Crucial to this success, not considering hyper-parameter tuning, was introducing a reset strategy where the environment is reset after a fixed number of steps. This ensured a more stable training, enabling the models to learn more. Other strategies also improved other aspects of the training – color shuffling improved generalisability, and introducing an action mask as a partial forward model of the environment in the observation greatly improved training speed, though the latter may not always be feasible in other types of games.

When we experimented with a hard action mask that was added to the logits of the action probabilities, the algorithm completely broke down. This happened because all the valid actions from the model were practically 0 while the invalid actions were all 0 because of the action mask, effectively masking out every action and leading to unexpected behaviour. Various RL libraries use a similar method but it should be used with great caution. A better approach would be to include the action mask in the observations and thus serving as a partial forward model.

IX. Acknowledgements

This work has been supported by the Innovation Fund Denmark and Tactile Games.

We thank Hunter Park (https://github.com/H-Park) and Kenneth Tang (https://github.com/ChengYen-Tang) for discussions on the various implementations.

We also thank Rasmus Berg Palm (ITU) for helpful comments for the manuscript.

References

[1] Zafarali Ahmed, Nicolas Le Roux, Mohammad Norouzi, and Dale Schuurmans. Understanding the impact of entropy on policy optimization. 2018.

[2] Mohammed Alshiekh, Roderick Bloem, Ruediger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe Reinforcement Learning via Shielding. 2017.

[3] Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *IJCAI International Joint Conference on Artificial Intelligence*, 2015-Janua:4148–4152, 2015.

[4] Petros Christodoulou. Soft Actor-Critic for Discrete Action Settings. 2019.

[5] Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying Generalization in Reinforcement Learning, 2018.

[6] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. Openai baselines. https://github.com/openai/baselines, 2017.

[7] Jesse Farebrother, Marlos C. Machado, and Michael Bowling. Generalization and Regularization in DQN. 2018.

[8] Alex Graves, Marc G. Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. Automated Curriculum Learning for Neural Networks. *34th International Conference on Machine Learning, ICML 2017*, 3:2120–2129, 2017.

[9] Sergio Guadarrama, Anoop Korattikara, Oscar Ramirez, Pablo Castro, Ethan Holly, Sam Fishman, Ke Wang, Ekaterina Gonina, Neal Wu, Efi Kokiopoulou, Luciano Sbaiz, Jamie Smith, Gábor Bartók, Jesse Berent, Chris Harris, Vincent Vanhoucke, and Eugene Brevdo. TF-Agents: A library for reinforcement learning in tensorflow. https://github.com/tensorflow/agents, 2018. [Online; accessed 25-June-2019].

[10] Stefan Freyr Gudmundsson, Philipp Eisen, Erik Poromaa, Alex Nodet, Sami Purmonen, Bartlomiej Kozakowski, Richard Meurling, and Lele Cao. Human-Like Playtesting with Deep Learning. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–8, 2018.

[11] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. *35th International Conference on Machine Learning, ICML 2018*, 5:2976–2989, 1 2018.

[12] Perttu Hämäläinen, Amin Babadi, Xiaoxiao Ma, and Jaakko Lehtinen. PPO-CMA: Proximal Policy Optimization with Covariance Matrix Adaptation. 2018.

[13] Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, S. M. Ali Eslami, Martin Riedmiller, and David Silver. Emergence of Locomotion Behaviours in Rich Environments. 2017.

[14] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, pages 3215–3222, 2018.

[15] Ashley Hill, Antonin Raffin, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, Rene Traore, Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. Stable baselines. https://github.com/hill-a/stable-baselines, 2018.

[16] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in Neural Information Processing Systems*, pages 4572–4580, 2016.

[17] Andrew Ilyas, Logan Engstrom, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Are Deep Policy Gradient Algorithms Truly Policy Gradient Algorithms? pages 1–40, 11 2018.

[18] Arthur Juliani, Vincent-Pierre Berges, Esh Vckay, Yuan Gao, Hunter Henry, Marwan Mattar, and Danny Lange. Unity: A General Platform for Intelligent Agents. pages 1–18, 2018.

[19] Niels Justesen, Philip Bontrager, Julian Togelius, and Sebastian Risi. Deep Learning for Video Game Playing. *IEEE Transactions on Games*, PP(c):1–1, 2019.

[20] Niels Justesen, Ruben Rodriguez Torrado, Philip Bontrager, Ahmed Khalifa, Julian Togelius, and Sebastian Risi. Illuminating Generalization in Deep Reinforcement Learning through Procedural Level Generation. 6 2018.

[21] Ildar Kamaldinov and Ilya Makarov. Deep Reinforcement Learning in Match-3 Game. In *2019 IEEE Conference on Games (CoG)*, number 4, pages 1–4. IEEE, 8 2019.

[22] Zachary Kenton, Angelos Filos, Owain Evans, and Yarin Gal. Generalizing from a few environments in safety-critical reinforcement learning. pages 1–16, 2019.

[23] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences of the United States of America*, 114(13):3521–3526, 2017.

[24] Tommy Liu, Jochen Renz, Peng Zhang, and Matthew Stephenson. Using Restart Heuristics to Improve Agent Performance in Angry Birds. 5 2019.

[25] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning. pages 1–9, 12 2013.

[26] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2 2015.

[27] Luvneesh Mugrai, Fernando Silva, Christoffer Holmgard, and Julian Togelius. Automated playtesting of matching tile games. *IEEE Conference on Computatonal Intelligence and Games, CIG*, 2019-Augus, 2019.

[28] Charles Packer, Katelyn Gao, Jernej Kos, Philipp Krähenbühl, Vladlen Koltun, and Dawn Song. Assessing Generalization in Deep Reinforcement Learning. 2018.

[29] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model. 2019.

[30] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. 2017.

[31] Mathieu Seurin, Philippe Preux, and Olivier Pietquin. I'm sorry Dave, I'm afraid I can't do that" Deep Q-learning from forbidden action. 2019.

[32] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning series. MIT Press, 2018.

[33] Erving Teng. Training your agents 7 times faster with ML-Agents - Unity Technologies Blog.

[34] Tom Zahavy, Matan Haroush, Nadav Merlis, Daniel J. Mankowitz, and Shie Mannor. Learn What Not to Learn: Action Elimination with Deep Reinforcement Learning. *Advances in Neural Information Processing Systems*, 2018-Decem(NeurIPS):3562–3573, 9 2018.

[35] Amy Zhang, Harsh Satija, and Joelle Pineau. Decoupling Dynamics and Reward for Transfer Learning. 2018.

[36] Yunqi Zhao, Igor Borovikov, Ahmad Beirami, Jason Rupert, Caedmon Somers, Jesse Harder, Fernando de Mesentier Silva, John Kolen, Jervis Pinto, Reza Pourabolghasem, Harold Chaput, James Pestrak, Mohsen Sardari, Long Lin, Navid Aghdaie, and Kazi Zaman. Winning Isn't Everything: Training Human-Like Agents for Playtesting and Game AI. *arXiv: e-prints*, pages 1–16, 2019.

4.3 Paper 4: *Estimating Player Completion Rate in Mobile Puzzle Games Using Reinforcement Learning*

Estimating Player Completion Rate in Mobile Puzzle Games Using Reinforcement Learning

Jeppe Theiss Kristensen
IT University of Copenhagen/Tactile Games
Copenhagen, Denmark
jetk@itu.dk

Arturo Valdivia
Tactile Games
Copenhagen, Denmark
arturo@tactile.dk

Paolo Burelli
IT University of Copenhagen/Tactile Games
Copenhagen, Denmark
pabu@itu.dk

***Abstract*—In this work we investigate whether it is plausible to use the performance of a reinforcement learning (RL) agent to estimate the difficulty measured as the player completion rate of different levels in the mobile puzzle game Lily's Garden.**

For this purpose we train an RL agent and measure the number of moves required to complete a level. This is then compared to the level completion rate of a large sample of real players.

We find that the strongest predictor of player completion rate for a level is the number of moves taken to complete a level of the ∼5% best runs of the agent on a given level. A very interesting observation is that, while in absolute terms, the agent is unable to reach human-level performance across all levels, the differences in terms of behaviour between levels are highly correlated to the differences in human behaviour. Thus, despite performing sub-par, it is still possible to use the performance of the agent to estimate, and perhaps further model, player metrics.

***Index Terms*—reinforcement learning, ppo, player agent, player modelling, playtesting, autonomous agent**

I. Introduction

Automatic testing of games has long been one of the objectives of research in game artificial intelligence. The ability to use an agent to test new content and mechanics has the potential to dramatically reduce the cost of production of games and improve the game designers' workflow.

Over the years, autonomous game-playing agents have been developed using different techniques ranging from rule based systems to machine learning methods such as supervised learning or reinforcement learning. While the objective in large parts of these agents is to play the game optimally – *i.e.* to solve the game – many efforts have been put also into creating agents that behave in a way that resembles as closely as possible the way a human player would play [4].

The purpose of human-like agents is to play a game competently and behave in a way that is indistinguishable from a human player to an outside observer; essentially, being able pass a game version of the Turing test [10]. Agents of this kind are ideal candidates to evaluate game content as the observation of their behaviour in the game can give a more realistic feedback to a game designer.

Creating human-like agents for game-play testing has been explored in a number of other works. Holmgård et al. [5] explore creating such agents using MCTS and evolving node selection criteria in order to generate procedural player personas. Mugrai et al. [7] developed this method further and tested it on a puzzle game comparing its scores to the ones of a small number of human-players that participated in their experiment.

Shin et al. [9], in order to try to mirror human players' behaviour, train an RL agent to learn which of 5 predefined human-like strategies to pick before picking a valid action matching the preferred strategy.

Lastly, actual player play-traces can also be used to learn how actual players play, which was demonstrated by Gudmundsson et al. [2], where a convolutional neural network action selection policy is learned from the play-traces. However, one issue with using playtraces is that these data are not always available, for example for a newly released game with little or no player data, or technical limitations such as cost of storage or tracking issues. Thus, an agent trained using reinforcement learning may be a more viable solution.

The initial results on using RL agents for playing Lily's Garden in Kristensen et al. [6] show that it is indeed possible to use an RL agent in this setting. In this research work we expand upon this line of investigation by including more levels and consider the next step for estimating player level completion rate using agent performance. We investigate how to train and use autonomous agents for estimating the player completion rate of a number of levels in the game Lily's Garden by Tactile Games[1]. For this purpose, we developed a set of Proximal Policy Optimisation (PPO) based reinforcement learning agents [8] and evaluate how the number of steps taken by the agent for completing the levels relate to the behaviour of a sample of ∼900,000 players.

II. Method

Before outlining the experimental approach, in this section we first present the RL setup that we use for the experiments followed by a description of the evaluation method.

A. Reinforcement Learning Setup

To serve as a test bed for our agent, we use a custom environment of Lily's Garden, detailed in previous work [6]. The game board representation is a $(13\times9\times m)$ array, where the

[1]https://tactilegames.com/lilys-garden/

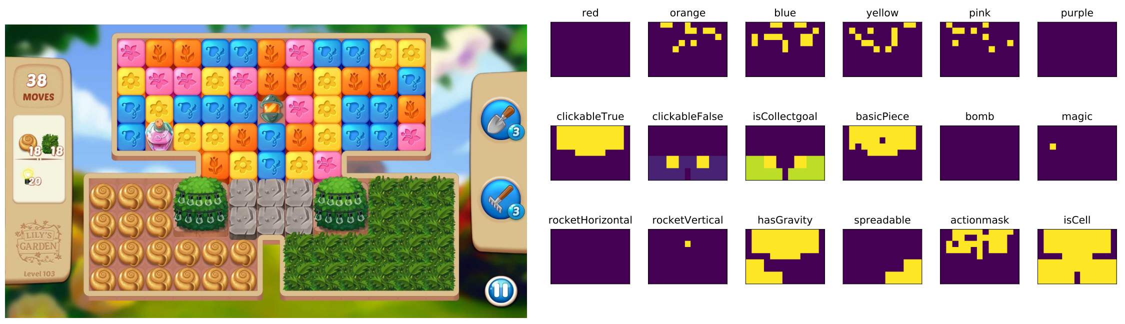


Fig. 1. Example of how the game-board of level 103 looks like *left)* in-game and *right)* how it is represented in our custom environment. The channels are not one-hot encoded but use the hit points of the board piece, hence the different colours in the CLICKABLEFALSE and ISCOLLECTGOAL channels.

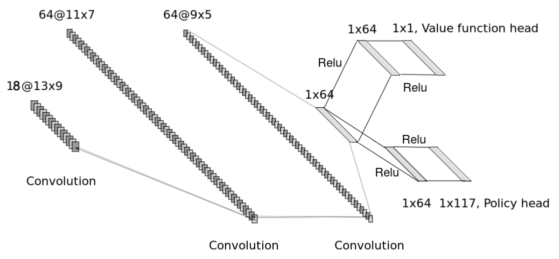


Fig. 2. Agent CNN policy setup. Each convolution uses a 2x2 kernel.

different board piece attributes are encoded in the *m* channels (see Fig. 1).

Our RL agent implementation is also based on our previous work in [6] and follows the on-policy implementation of PPO available in OpenAI Baselines [1] and stable-baselines [3] where multiple agents can collect state-action-reward observations simultaneously. The architecture of the policy and value networks is shown in Fig. 2: three convolutional layers for feature learning are in common between the two networks, with separate value and policy heads.

In all the experiments, we use an entropy coefficient of 0.01, a learning rate of $1e-4$ and run 8 agents simultaneously. The other hyper-parameters are set to $n_{\text{minibatches}} = 64$ (default 4), $n_{\text{steps}} = 256$ (default 128) and otherwise default values.

Additionally, based on our previous research [6], we use three strategies to improve the stability of the training:

- Colour shuffling, where the colour channels are randomly permuted during the training. This is done to help the agent to generalise patterns regardless of the colours.
- Resetting after 100 total steps, to prevent the agent from getting stuck on an invalid move and filling the training data buffer with useless observations.
- Adding an action mask to the observations, which serves as a partial forward model leading to faster initial training.

B. Estimating player success rate

When level designers wish to evaluate a level, the player completion rate is often then used as a proxy for difficulty. Since players can only complete the level once before progressing, the completion rate can be directly calculated as the number of level completions over total attempts across all users. It is important to note, though, that this is not necessarily the inherent difficulty of a level and may change in time depending on which cohorts of players that have reached a given point.

In our game environment of Lily's Garden, the agent is able to take an unlimited number of moves per level. The move distribution is therefore different compared to player data, where there is a sharp cut-off after the move limit, see Fig. 3 *left*. In order compare these two distributions and estimate the player completion rate, we record the max number of moves number of the best $x\%$ agent runs and then calculate the Spearman correlation of this number with the player completion rate. Because the number of moves spent by the agent to complete a level is invariant level move limit is, while the player completion rate is very much determined by this limit, we normalise the agent moves with the level move limit.

III. EXPERIMENTS

The objective of this research project is to help determine how an agent can be used in a production setting in a mobile gaming company where not only accuracy but also speed is important. For that purpose, we explore here three scenarios:

- One-step training on curriculum
- One-step training on the target level
- Two-step training, first on a curriculum and then on the target level

In each scenario the aforementioned training and in-game behaviour performance are collected and compared to the human behaviour. In addition to that, we benchmark these agents against two baselines:

- *Random agent* which picks a random valid action every time it has to make a move.
- *Greedy agent* which mimics a play style that prioritises clicking on valid pieces belonging to the biggest clusters.

Out the three aforementioned training scenarios, we start with the *One-step training on curriculum* because it offers one desirable feature for production usage: a previously trained

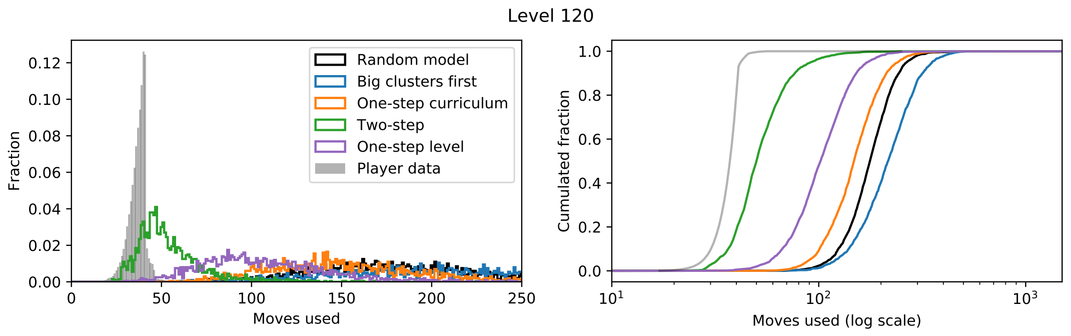


Fig. 3. *Left)* Distribution of number of moves required for each model to complete level 120 compared to actual player data. The sharp drop-off in the player distribution of because of the level move limit. Additional steps after the move limit can be purchased using in-game currency. *Right)* Cumulative move distribution, or level completion curves, plotted on a log scale.

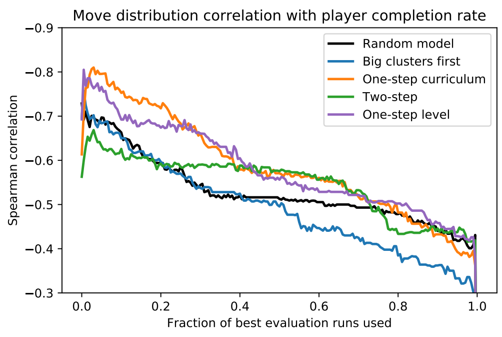


Fig. 4. Spearman correlation between player completion rate and normalised moves required to finish each level as a function of fraction of best evaluation runs used.

agent is used in order to estimate the difficulty of new unseen levels, without requiring any further training.

The training curriculum here should be chosen in such a way that the agent is subsequently able to generalise across new levels and mechanics. Thus, in particular, the curriculum has to include a variety of different types of levels and be representative of different game mechanics (*e.g.*, *rock blockers*, *grass spreading blocks*). Furthermore, for evaluation the agent has to be exposed to a set of equally different unseen levels.

To choose the levels for the curriculum, we note that at the beginning of Lily's Garden, the new mechanics are typically introduced once every ten levels (*i.e.*, on levels 21, 31, 41, ...). The following nine levels at each interval generally mix the new mechanic with previously introduced mechanics. And so for the training, we include the first hundred levels which introduces eight new board pieces in addition to the base mechanics of the game. These levels are uniformly randomly sampled and trained on for at least one epoch over 35M *steps* – *i.e.*, the equivalent of a human player tapping on the screen. We remark here that from our empirical observations, this number of steps ensures that the learning has plateaued and each level will have been trained on for multiple epochs.

To evaluate the agent performance, we test on the 20 levels following the ones used in training (*i.e.*, levels 101 to 120), which include the previous mechanics plus two new mechanics: *teleporters*, which move board pieces to other parts of the game board, and *containers*, which are 2x2 blockers with 10 hit points. For the results we also include the levels 13, 23, ..., 93 to test the performance on previous levels which are not tutorial levels.

In the second scenario considered, *One-step training on evaluation level*, the agent is trained every time from scratch directly on the new target level. This is done through 1M steps. Two potential drawbacks of this approach are apparent: one is that solution may require longer training time before reaching a level of competency when compared to the other scenarios based on curricula. Secondly, it may also lead to poor generalisation. However, if good accuracy is obtained and the training can still performed in reasonable time, then this approach may still offer a viable solution to used on a production environment.

The final scenario we consider, *Two-step training*, is inspired by how players typically learn: At first the player has some previous general knowledge of how to play the game but no specific knowledge on how to beat certain level or game mechanic. Then, after having played through the level a number of times, the player may finally learn a winning strategy and complete the level. In this setting, the training for the first step is analogous to what was is done in the setting of the *One-step training on curriculum* scenario, while for the second step we proceed as analogously to the *One-step training on evaluation level* scenario.

IV. Results and Discussion

Before answering the question of whether we can correlate the agent behaviour to the players', we first examine which agents that acquire the highest proficiency, as measured by the least amount of average moves spent to complete a level. A representative example of the level of proficiency the different agents acquire can be seen by considering the move distributions in Fig. 3. The most proficient agent comes from the two-step training approach, followed by the one-step training on target level and then finally one-step training on a curriculum.

All the training scenarios lead to agents performing better than the random and greedy agents. Despite the fact that some agents only perform slightly better than random move-wise, time-wise the trained agents are much faster during evaluation because the random agent attempts to take many invalid actions before finally choosing a valid one, which increases the runtime of the evaluation.

One thing that is worth noting is that one-step curriculum leads to the least proficient agent. The agents trained on a single level demonstrate that it is possible for the agent to almost play optimally, so this suggests that something in our way of training on a curriculum – randomly sampling levels after an epoch – may prevent the agent from becoming more proficient. Improving this could be done by developing a more intelligent curriculum well as adding changes to the RL algorithm to ensure that no catastrophic forgetting will occur, where the agent forgets how to play previously learned levels.

That being said, a high proficiency does not necessarily mean that the performance of the agent is correlated with the player completion rate. Indeed, looking at Fig. 4 it can be seen that the one-step curriculum approach shows the highest correlation, despite being the least proficient agent. It can also be seen that the highest correlation occurs when only considering $\sim$5% of the best runs. With the one-step curriculum scenario both being the most practical approach, due to not spending any time on additional training, and also showing the highest correlation, this is a very promising result towards using this approach in a production setting.

The least correlated approach is the two-step training, which shows an even worse correlation than random. This might be due to the fact that it is able to completely memorise some levels, while on other levels the agent is still learning. This mix of memorisation and proficiency may then lead to very uncorrelated behaviours. This could also explain why in the one-step training on target levels still shows a correlation; the agent in this scenario has simply not trained long enough on a single level to memorise it, so only the agent proficiency matters.

Why the correlation with player completion rate is highest when only considering the 5% best runs is not immediately clear but may be linked with the long tail of the move distribution: the longer the game goes on for, the more spread out the point of completion is due to an inherent randomness in the levels, leading to a lower correlation. Conversely, there is a certain minimum number of moves required to finish many levels, so having a good run and finishing early leads to a much tighter distribution.

The results so far suggest that there is a correlation between the agent behaviour with player completion rate. Unlike other works that try to model and predict the precise player metric (*c.f.*, [2]) using the rank correlation can instead be used to give an estimate on how a level is compared to other levels. For example, it may show that a certain level is one of the top 10% most difficult levels.

These initial results are promising but also have some limitations. Only 120 levels were included in this analysis, which contain around 60% of the game mechanics. Whether these correlations extend to the remaining mechanics and whether the agent is able to deal with them need to be further investigated before using it in a production setting. Additionally, we only consider the move distribution for the correlation. However, it may be possible to utilise additional agent or level data for our estimates. Not only could this possibly lead to a more robust estimate, but it could also help the level designer understand the effects of changing various aspects of a level.

V. Conclusion

We have examined a number of scenarios in which an RL agent can be trained and used to predict the level difficulty in a mobile puzzle game. The results – based on $\sim$60% of the game mechanics – demonstrate that the two-step training scenario leads to the most proficient agent, while with the one-step curriculum the agent attains the largest correlation to real players' completion rates. The latter scenario is also arguably the most practical one in a production scenario.

By considering the best $\sim$5% of the runs of the agent and record the max number of moves required to finish the level, the difficulty of the level, as measured by the player completion rate, can be estimated in terms of how it ranks compared to other levels.

Because the results shown in this research work are only for a limited subset of levels, future work should look into whether this correlation holds for the remaining levels and possibly attempt a more modelling-based approach.

References

[1] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[2] Stefan Freyr Gudmundsson, Philipp Eisen, Erik Poromaa, Alex Nodet, Sami Purmonen, Bartlomiej Kozakowski, Richard Meurling, and Lele Cao. Human-Like Playtesting with Deep Learning. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–8. IEEE, 8 2018.

[3] Ashley Hill, Antonin Raffin, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, Rene Traore, Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. Stable baselines. https://github.com/hill-a/stable-baselines, 2018.

[4] Philip Hingston. A Turing Test for Computer Game Bots. *IEEE Transactions on Computational Intelligence and AI in Games*, 1(3):169–186, 9 2009.

[5] Christoffer Holmgard, Michael Cerny Green, Antonios Liapis, and Julian Togelius. Automated Playtesting with Procedural Personas with Evolved Heuristics. *IEEE Transactions on Games*, 1502(c):1–1, 2018.

[6] Jeppe Kristensen and Paolo Burelli. Strategies for using proximal policy optimization in mobile puzzle games. 2020.

[7] Luvneesh Mugrai, Fernando Silva, Christoffer Holmgard, and Julian Togelius. Automated playtesting of matching tile games. *IEEE Conference on Computatonal Intelligence and Games, CIG*, 2019-Augus, 2019.

[8] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv e-prints*, pages 1–12, 7 2017.

[9] Yuchu Shin, Jaewon Kim, Kyohoon Jin, and Youngbin Kim. Playtesting in Match 3 Game Using Strategic Plays via Reinforcement Learning. *IEEE Access*, 8:1–1, 2020.

[10] Julian Togelius, Georgios N. Yannakakis, Sergey Karakovskiy, and Noor Shaker. *Assessing Believability*, pages 215–230. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

5 Agent-Assisted Game Difficulty Prediction

In Chapter 3, we discussed ways to model and operationalise difficulty using historical player data, but we did not consider how to apply it to content without any player data. In Chapter 4, we discuss ways to create an automated playtesting agent that can play through puzzle levels, but we did not explore how it can be used to model difficulty and predict the number of attempts players will spend on new content. In this chapter, we investigate how we can combine the two lines of research by using historical data and playtesting data for operationalising difficulty on *any* content. The results from this chapter help answer both of the main research questions of this dissertation.

5.1 Difficulty prediction using AI agents

Creating a playtesting method that can play the level is one thing. Another question is whether the behaviour of this agent correlates with the player. The work in this dissertation that focuses on creating a playtesting agent did find a correlation between agent performance and human players, but a direct translation of the agent performance to player performance is not sufficiently accurate for the level designers. A more robust approach in an industrial setting is to use a two-step approach where data is first collected by *some* playtesting method and then subsequently used to model and operationalise the estimated difficulty. The results in this section focus on how this is done in practice.

5.1.1 Background

The previous work presented in Chapter 3 requires observing players on a given level before being able to model the difficulty of said level. Therefore, the question of how these methods can be applied in situations where there is little or no player data is not examined. Similarly, much of the existing literature regarding predicting the difficulty of new levels does not consider the vital aspect investigated in Section 3.2, which is how the individual player perceives the difficulty. Instead, these methods commonly compare the agent's performance to the human averages (e.g. Gudmundsson et al. (2018); Mugrai et al. (2019); Horn et al. (2018); Roohi et al. (2021); Kristensen et al. (2020)).

This section focuses on investigating how these two issues of having no player data and not considering personalisation can be combined. We hypothesise that combining personalised predictions with data from an agent that learns how to play the levels makes a more accurate estimate of perceived difficulty possible. For this, we employ an improved version of the playtesting agent described in Section 4.1 where we limit the number of possible random seeds of the levels and remove colour shuffling. This allows the agent to learn the optimal strategy on each level fully, and while this is a different way to apply playtesting in practice compared to the recommendations in Section 4.1.2, the study provides insight into how the optimal strategy is connected to player behaviour. We then carry out a comparative study in Paper 5 (Kristensen and Burelli, 2022) between the methods in Section 3.2 using data from the playtesting agent in order to answer the following questions:

- What model and combination of data are the most accurate for personalised difficulty predictions in the scenario where the prediction is made on existing content and cold start scenarios?
- Does personalised difficulty predictions lead to better accuracy compared to cohort predictions in the two scenarios?

5.1.2 Results and conclusion

In the first experiment, we test how well we can predict the number of attempts a given player will spend on both existing and new puzzle levels. In addition to the FM method, we also consider neural networks and random forest predictors. We find that the FM method outperforms the other methods for predictions on existing levels, and the accuracy is increased further by including additional data about the level mechanics and player behaviours. For personalised predictions on new content, all methods experience a decrease in performance, with the FM method that includes agent data experiencing the smallest decrease. These results highlight the FM method's effectiveness in capturing individual differences and the benefit of using playtesting data when no other data source is available.

Since the level designers typically take decisions based on the average difficulty of the level (i.e. average number of attempts), we run a second experiment to simulate such a scenario. We split the players into ten cohorts and predict the average difficulty on a level for each cohort. This is done two ways: by calculating the aggregated average based on the personalised predictions and by aggregating the data on a cohort level first and then building a new prediction model for predicting the average.

The results in this second experiment show that for predicting the average difficulty of existing content with historical data, cohort predictions generally perform better than personalised predictions. This is not surprising for two reasons: First, since the cohorts all have very similar averages on each level and the models are able to learn to recognise which specific set of features belongs to each level, the models essentially overfit to the data and learn the average from the training data. Second, the models for personalised predictions are optimised for individual predictions and not the cohort averages, so the predictions that lead to the smallest RMSE in these two cases are not necessarily the same.

For predictions in the cold start scenario with no historical data, the errors are larger but still better than the baseline prediction. The best performing model for both the cohort-based and personalised predictions is the NN with all categories of data included (player features, level descriptions and agent features). The errors for personalised predictions are similar to the cohort predictions, but personalised predictions generally exhibit a larger variance while cohort predictions tend to predict more towards the global average.

These results reveal that level designers gain little from using personalised predictions for cohort predictions on new levels and, in this case, can rely on a NN method for estimating the difficulty of new content. The FM approach is useful for personalised predictions on existing content, and the latent features of the players can still be interpreted in terms of player skill and consistency, as discussed in Section 3.2.

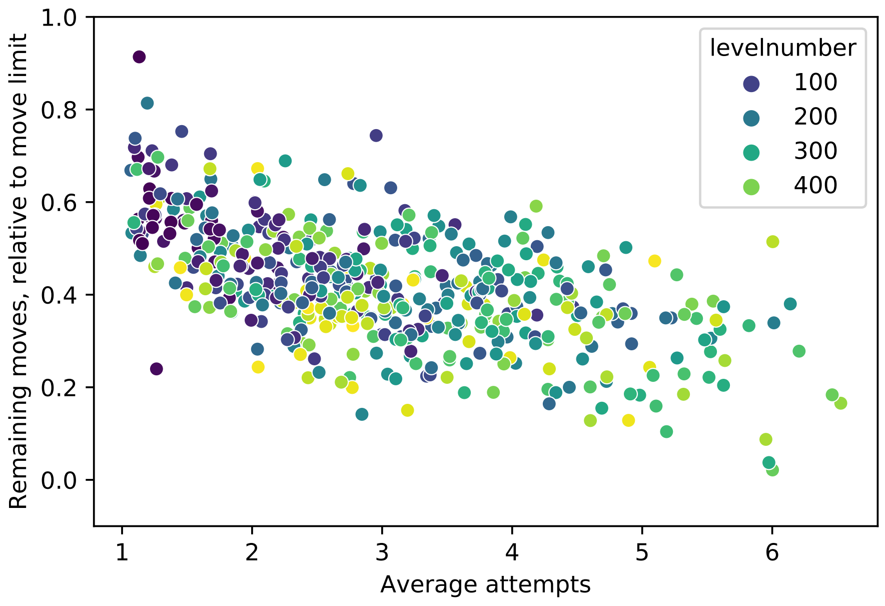


Figure 5.1: Performance of the improved agent on the first 499 levels.

The main suggested path for improving predictions is changing how the playtesting agent is used. As shown by the agent performance in Fig. 5.1, the agent's competence leads to the agent completing the levels with several moves left, which is much better than many human players and thus also not reflective of the full range of player experiences. Possible ways to improve this are by including more random seeds rather than only the eight used in this study and possibly training other agents with different reward functions to explore other types of playstyles.

5.1.3 Relevant paper(s)

The results and discussion are based on Paper 5, *Difficulty Modelling in Puzzle Games* (Kristensen and Burelli, 2022).

Paper 5 has been submitted to Transactions on Games in July 2022. As of the time of writing, it is currently under review. It has been co-authored with the principal university supervisor, Paolo Burelli. I am responsible for conducting the experiments and contributing to most of the text. All co-authors have been involved in the discussion and editing for the final structure of the paper.

5.2 Paper 5: *Difficulty Modelling in Puzzle Games*

Difficulty Modelling in Puzzle Games

Jeppe Theiss Kristensen and Paolo Burelli

***Abstract*—Difficulty is one of the key drivers of player engagement and it is often one of the aspects that designers tweak most to optimise the player experience; operationalising it is, therefore, a crucial task for game development studios. A common practice consists in creating metrics out of data collected by player interactions with the content; however, this allows estimation only after the content release and does not consider the characteristics of potential future players.**

In this article, we present a number of potential solutions for the estimation of difficulty under such conditions, and we showcase the results of a comparative study intended to understand which method and which types of data that perform better in different scenarios.

The results reveal that models that are trained on a combination of cohort statistics and simulated data produce the most accurate estimations of difficulty across all scenarios. Furthermore, among these models, artificial neural networks show the most consistent results.

***Index Terms*—Difficulty, reinforcement learning, factorization machines.**

I. Introduction

Among the many characteristics of a game that affect the player experience, difficulty is generally considered to be one of the most impactful. As theorised by Csikszentmihalyi regarding the concept of *flow* [1], if the game is too easy, the player risks not feeling challenged, becoming bored and potentially quitting the game. On the other hand, in case it is too hard, the player might feel stuck, become frustrated and, again, potentially quit the game. It is therefore not surprising that an important focus for many game studios is being able to estimate difficulty.

For the mobile game industry, which makes up more than 50% of the $175.8B annual revenue in the games industry [2]–[4], this is an active area of research, and in particular for puzzle games. These types of games are the most popular subgenre of mobile games making up around 15% of all installs, and in such a competitive market, it is not enough for game studios to simply acquire new users by having a polished game with low to no cost to enter. Ensuring players *stay* engaged in the game is equally crucial for the game to be a commercial success, which puts pressure on game studios to both have content that is adequately challenging for new players to pique their interest but also release new content for more experienced users that can keep the game experience fresh and exciting. The challenge for any difficulty prediction framework is therefore two-fold: not only should it work on previously released content, where rich information about other players is available but also on novel content where player data is more sparse or non-existent. Furthermore, as difficulty is a concept that emerges from the interaction between players and content, the estimation is going to be affected not only by the content type but also by the target player or cohort of players.

Jeppe Theiss Kristensen and Paolo Burelli are with the IT University of Copenhagen, Denmark. Emails: {jetk, pabu}@itu.dk

Manuscript received XXXX.XX.XX; revised XXXX.XX.XX

As a first step in the process of building a difficulty modelling framework, it is necessary to operationalise difficulty so it can be quantified and measured. A common gameplay element in many puzzle games is the existence of discrete tasks, or levels. A direct way to quantify the difficulty of a level is by using the average number of attempts the players require to complete the level. This metric has been linked to player churn, and it gives the level designers a clear measure of how much time players are expected to spend on the challenges. However, the measurement very much depends on the players that have played through the level at the time of measuring. This *perceived* difficulty can vary depending on player cohorts and their skill, so using this average metric for all players can be misleading and does not fully inform the level designers about the *intrinsic* difficulty of the level [5].

Another challenge for difficulty estimation is the complexity of the gameplay. Some aspects of a puzzle, such as winning strategies or "traps", may not be immediately apparent through a static analysis of the level. While it may be possible to infer the level's complexity by using historic data of previous players' performances, this is not possible with new content. Instead, it typically requires multiple level designers to manually playtest the content to overcome this *cold start* problem, which can be an expensive and time-consuming process. This not only limits the volume of new content but also the quality due to the level designers' own skills and biases. A framework for difficulty prediction can therefore benefit from a component that can explore the dynamics of the game to determine the intrinsic difficulty but also possibly model how individual players respond to the intrinsic difficulty.

One or more models able to estimate perceived difficulty in these different scenarios have multiple potential applications: they could be employed to give immediate feedback to designers when working on new puzzles, they could be used to guide a procedural content generation algorithm intended to create content at a specific level of difficulty, or they could be used to adapt content so that it would provide a specific level of challenge to a specific player or cohort.

With these challenges and applications in mind, in this research work, we investigate various difficulty modelling frameworks that can address some of the issues and shortcomings that previous approaches have revealed. The two main research questions we aim to answer are:

RQ1: Is it possible to accurately estimate the perceived difficulty of a puzzle for a specific player? What combination of model and data is most effective at this task?

RQ2: Is it possible to accurately estimate the perceived difficulty of a puzzle for a specific cohort of players? Can we use the same models and methods used of personalised predictions? What combination of model and data is most effective at this task?

On one hand, with the first research question (RQ1), we aim at investigating personalised models of difficulty that can be used for adaptive gameplay. On the other hand, with the second research question (RQ2), we aim at investigating models that are suitable for either cohort-level adaptation or to support procedural content generation and manual content design.

In order to answer these two questions, we have designed two experiments in which we break the questions down and explore how different difficulty prediction methods work in various scenarios. Using data from a commercial puzzle game, we investigate the impact of using different types of data, methods and granularity for difficulty prediction, which contributes to the body of knowledge for developing a complete and feasible framework for both personalised and cohort difficulty predictions. While the research builds on results from a puzzle game, the methods and results have the potential to be extended to other types of games with similar characteristics – i.e. linear progression and discrete repeatable challenges – that can help game studios and researchers alike for building a more comprehensive model of difficulty in games.

II. Related work

Difficulty modelling and prediction can be approached in many different ways depending on the use case. In this paper, we root our analysis in the practical application of modelling difficulty in a commercial context in which both the player population and game experience change constantly. In order to properly analyse a framework that can work in such a context, we first discuss the definition of difficulty and give an overview of how it has previously been modelled in various contexts. In addition to that, we also examine how various playtesting methods have been used for evaluating new content and relate that to the approach presented in this paper.

A. Predicting difficulty

In order for any measurement of difficulty to be meaningful, it should be reflective of the player experience. However, there can be multiple aspects that can change this perception, which include uncertainty and player skill [6]. It is therefore highly individual how challenging a player finds a puzzle, so a useful notion is *perceived* difficulty, where difficulty is described as a *relational attribute* between the game and the player [5], [7]. The importance of the individual perception of difficulty can be understood through the lens of *flow*; a state where the player loses track of time and worries [1], [8]. To reach this state, the presented difficulty of a task should be optimal in terms of player skill, which means it should satisfy the players' intrinsic needs for feeling competent and lead to an engaging game experience [9]–[11]. With flow and intrinsic motivation being major contributors to player engagement [12], a core design object of the difficulty prediction framework is, therefore, to account for individual or cohort differences.

Pusey et al. [13] describe a number of measurable metrics for puzzle games that quantify difficulty, which include the number of actions and time taken to complete a puzzle as well as the number of incorrect attempts. Indeed, using the average number of attempts players spend to complete the level, or inversely the pass rate, is a common way to operationalise difficulty in puzzle games (e.g. in Angry Birds [14], Lily's Garden [15], Candy Crush [16]). As noted by Denisova et al. [6], the player's perceived difficulty is strongly linked to the number of successes and failures, so in this paper, we adopt a similar definition of difficulty as the number of attempts spent to complete a given level.

The objective of predicting the number of attempts has been approached in a number of different ways and with different applications. One use-case is dynamic difficulty adjustment (DDA) [17], [18] where the game content is adjusted so the perceived difficulty for a player follows an optimal or predefined goal. Gonzalez-Duque et al. [19] consider recent play history and use Bayesian methods for estimating how much time a player requires for completing a sudoku puzzle or platform level. A more direct approach is presented by Xue et al. [20], where they use the pass rate associated with different random seeds for each level to serve personalised content. Deep learning methods have also been used for modelling players and optimising difficulty and engagement [21], [22].

While DDA has the promise of increasing engagement [23], [24], not all games are designed with this kind of dynamic adjustment in mind and designers instead find it sufficient with more ad-hoc or daily predictions. For this purpose, parametric-based approaches have been applied for estimating the probability of success [25]–[27]. An example is the work by Wheat et al. [28] where four categories of data (e.g. level features and player behaviour) are considered. In this work, a random forest classifier had the best accuracy when predicting how difficult each player perceived levels in a custom platform game. However, in more complex games with large, diverse player bases, these parametric approaches may not be able to fully capture the complex inter-relationship between players and levels, leading to limited applicability. Matrix factorization methods, known from recommender systems and their ability to deal with sparse data, have been used specifically for predicting the perceived difficulty of each player in games [5], [29].

In this work, we build on the prediction methods from Kristensen et al. [5] by extending the analysis in two ways. The main problem is that matrix factorization methods suffer from a cold start problem where new players or levels are not possible to include during training. This leads to the model not learning latent representation of the new users and levels, which in turn makes predictions on new content impossible. To address this, we experiment with including different categories of data, similar to Wheat et al. [28] in addition to testing out both a neural network and a random forest model for difficulty prediction.

B. Playtesting agents

While player data and level data have been used for estimating the difficulty of levels, these categories of data do not

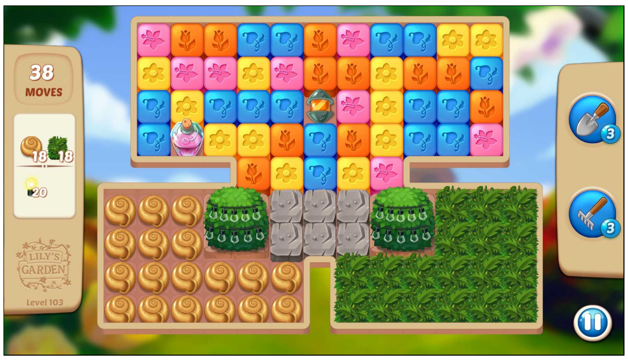


Fig. 1. Example puzzle level in Lily's Garden

include any kind of dynamic information about the levels and are bound to be only an approximation of the gameplay. It is, therefore, necessary to extract data from playing through the level, and with the complexity of games today, more intelligent solutions than simple heuristics-based methods are increasingly necessary. The methods have also become mature enough to assist the whole game design process, ranging from finding bugs, edge cases and general playability [30]–[33], to modelling players and adjusting difficulty [14], [15], [34]–[36].

In the specific case of puzzle games, (deep) reinforcement learning (RL) and Monte-Carlo Tree Search (MCTS) based approaches are among the most common ones employed to estimate the difficulty of levels [16], [37]–[41]. In one of the most recent examples, Kristensen et al. [37] use an implementation of the RL method Proximal Policy Optimisation (PPO) [42] to develop an agent capable of playing through puzzle levels in the commercial puzzle game Lily's Garden. Further work showed that despite the agent performing subpar compared to human players, the agent performance is strongly correlated to actual player performance [15]. As noted by Zhao et al. [43], the goal of creating playtesting agents is not necessarily to outperform humans but rather to capture facets of the gameplay related to skill and style that can be related to player behaviour. With a similar perspective, Gudmundsson et al. [16] demonstrated that by post-processing the statistics produced by an agent playing a commercial puzzle game, it is possible to better capture these facets and produce a more accurate estimation of difficulty.

Inspired by these approaches to agent-based testing and parametric difficulty modelling, in this study we propose a combined method built on the RL approach by Kristensen et al. [15], [37]. The method and its components are chosen and evaluated through a comparative study intended to showcase how different combinations of methods and data perform at the different difficulty estimation tasks described by the two research questions defined in the previous section. The aim of the study is to conduct a systematic analysis of the difficulty modelling problem in the context of puzzle games and establish a benchmark for future works in the field.

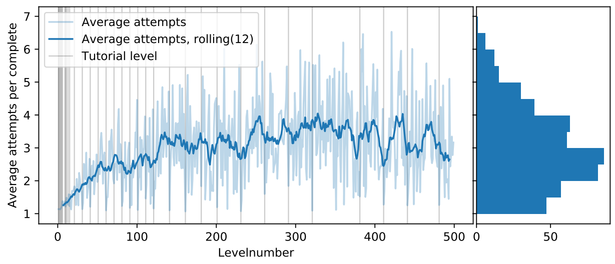


Fig. 2. The average attempts per complete over the first 500 levels investigated. A rolling mean with window size 12 is also shown to visualise the trend. The vertical grey bars indicate tutorial levels. Adapted from Kristensen et al. (2022) [5].

III. Case study: Lily's Garden

In this paper, we employ data from the free-to-play mobile puzzle game Lily's Garden by Tactile Games as a case study to model difficulty in a realistic commercial scenario. The game was released in early 2019 and has close to a million daily active users worldwide as of today. It is among the top 10 grossing puzzle games in the US [44] and serves as a representative sample of games in this genre both in terms of gameplay and general characteristics.

Lily's Garden is a puzzle game with an overarching narrative in which the player can unlock decorative pieces and story plots by completing puzzle levels. These puzzle levels are blast-type puzzles and contain a gameboard that can be up to 13 by 9 in size. An example of a level is shown in Fig. 1. The core gameplay consists of tapping on basic board piece clusters of the same colour to remove the pieces themselves as well as adjacent non-basic pieces. By tapping on larger clusters, more powerful pieces can be created which can clear larger areas on the board. In order to complete the level, the player must clear a number of objectives within a given move limit. As a part of the free-to-play model the game employs, it is also possible for the player to use booster items that can affect the gameboard directly or add additional moves, which can be acquired from in-game events and purchases. However, all levels are designed to be possible to complete without using any boosters.

There are currently more than 6000 available levels, but in this study, we limit the modelling and predicting difficulty to the first 500 levels. Using the definition of difficulty as the average number of attempts per complete, the difficulty over the level range can be seen in Fig. 2. The first 100 levels or so contain multiple tutorials and easy levels to properly onboard new players and engage them in the background story. Subsequently, the difficulty stabilises at around 3.2 attempts per complete, with easier levels typically being completed in just one attempt and more difficult levels requiring upwards of an average of 7 attempts.

While the average varies around 3 attempts, there are large individual differences. As an example of this, Fig. 3 shows the distribution of attempts on an easy and hard level and their averages. Players most commonly only spend one attempt on a level but due to both the random elements of the game and player skill, hard levels sometimes require upwards of 30

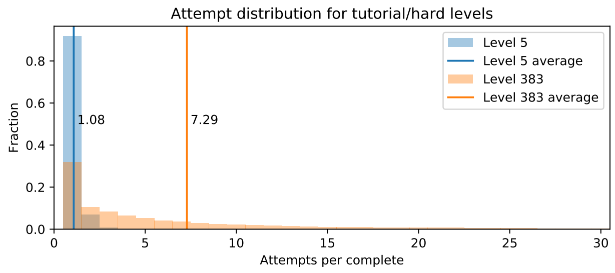


Fig. 3. Two examples of attempt distribution on an easy and a hard level (level 5 and 383, respectively). Adapted from Kristensen et al. (2022) [5].

attempts or more. This long-tailed distribution appears similar (but not equal) to a geometric distribution where the mean (here the average number of attempts) is proportional to the inverse pass rate and the variance is proportional to the mean and pass rate.

IV. Methods

In a similar approach as Kristensen et al. [5], we treat the task of predicting the number of attempts a player will spend on a level as a supervised regression task. We employ a two-step approach similar to what is used by Gudmundsson et al. [16] where we first extract relevant data from a playtest agent and secondly train a prediction model. In the following section, we describe the playtest agent as well as the different types of regression methods that will be used in the experiments. The specifics and hyperparameters of each setup are described in Appendix A.

A. Playtest agent

To explore the dynamics of each puzzle in the game, we employ a reinforcement learning agent where we follow the learnings from previous work on creating a playtesting agent for Lily's Garden [15], [37]. In order to capture the game numerically, each level is described by a 3D array of size $13 \times 9 \times n$, where $n = 27$ and represents various board piece mechanics. These mechanics include possible colours, whether the piece is clickable or not and if it is a goal piece. If a board piece is present in a given position, its hit-points are added to the position in each of the channels that match the mechanics of the piece. In addition to the game board, we also provide information about the number of moves taken, the moves left as well as how many total goal pieces remain after the match. The possible actions correspond to a position on the game board, meaning there are $13 \times 9 = 117$ possible actions, with invalid moves being masked out. A more comprehensive description is given at the website https://aicompetition.tactilegames.com/environment.

Differently from our previous works aiming to create a single playtesting agent that learns to generalise its strategies to any level, we train one agent for each level. We, therefore, do not employ colour shuffling, and we use 8 fixed random seeds for each level which allows us to test 8 different level configurations and learn the optimal strategy for each.

We allow the agent to take up to 100 moves per episode, which is around double the average normal move limit of 42. In order to encourage the agent to learn to finish the level as quickly as possible, we reward the agent for completing the level and give an additional bonus for completing it before the normal move limit, or a penalty if it spends more than the move limit. Additionally, we also introduce a small reward for creating power piece combinations, which can be key to winning some levels. The final reward function is as follows:

- $+0.1$ for each goal collected.
- $+0.8$ if completing the level.
- $\max(-0.8; +0.05 \cdot [M - n])$ on level completion, where n is the number of steps taken and M is the move limit.
- $+0.1$ for each power piece combination used.

B. Prediction methods

We test out three different methods for this problem: a neural network (**NN**) and a random forest (**RF**) approach which generally work well with dense data, and a factorization machines (**FM**) method which extracts a latent representation of interactions that enables modelling sparse, high-dimensionality data such as user-item interactions.

1) Random forest: Random forest is an ensemble prediction method that utilises multiple decision trees that are created using a random subset of the features and data. This enables non-linear modelling and predictions, and it has been used for the same task in previous work [5].

RFs do generally not work very well with high-cardinality data such as unique user ids and levels since splits on individuals is not generalisable and can easily lead to overfitting. Ways to ensure the model can generalise to new data is by limiting the maximum depth and number of trees and using a smaller subset of features for building the trees.

2) Artificial neural networks: Artificial neural networks are another way to model non-linear processes. Although they are more of a black box approach than RF, their performance is usually comparable to or better than RF methods, and they can be trained incrementally while keeping the model size constant. This is unlike RF methods which either need to be fully retrained or incrementally retrained by adding new individual trees to the ensemble.

NNs can be prone to overfitting due to their large number of parameters of the models. In order to improve their ability to generalise, using regularisation and/or adding dropout layers to the model are commonly employed strategies [45].

3) Factorization machines: Factorization machines are a general purpose method that can be used for both classification and regression tasks and belong to a family of matrix factorization methods where entities are described by an embedded latent vector that enables modelling sparse interaction between the entities [46]. Specific to FMs, the features can consist of both high-cardinality data such as user and level ids, as well as dense data such as content tags or level descriptors. The formula is given by

$$\hat{y} = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j, \qquad (1)$$

Type of feature	Name	Description
Historic data	Average attempts (level)	Average attempts on the level based on playthroughs by old players
	Average attempts (player)	Average attempts by the player on the first 100 levels
Player features	Moves used	Number of moves used relative to the move limit when completing the level
	In-game boosters	Boosters that can be used during the level
	Pre-game boosters	Boosters that can be used before starting the level
Level attributes	Board size	Number of available board slots
	colour entropy	Entropy of colour spawning weights; $S = -\sum_i p_i \log p_i$
	colours	Number of unique colours in the level
	Board pieces	Multiple features that each describe the number of board pieces on the initial board
	Collect goals	Multiple features that each describe the number of collectgoals
Agent features	Training steps	Number of training steps until minimum average length was achieved
	Move limit	Move limit of the specific level
	Game length *(min + std)*	Average number of moves used to complete level across 8 seeds
	Completion rate *(min + std)*	Fraction of times the agent finished the level within the movelimit/100 moves across all 8 seeds
	Action entropy *(min + std)*	RL agent action entropy
	Training losses *(min + std)*	RL policy and value losses

TABLE I
FEATURES INVESTIGATED FOR THE DIFFERENT MODELS. THE PLAYER FEATURES ARE AGGREGATED MEANS ON THE FIRST 100 LEVELS. THE VALUES OF THE AGENT FEATURES ARE AVERAGED FROM THE LATEST 100 EPISODES/PLAYTHROUGHS. FOR SOME OF THE AGENT FEATURES, WE INCLUDE BOTH THE VALUE AT THE TIME STEP WHERE THE SMALLEST AVERAGE NUMBER OF STEPS WAS ACHIEVED (DENOTED *min*), AND THE STANDARD DEVIATION OF THE FEATURE BETWEEN TIME STEPS 1M TO 9M (DENOTED *std*).

.

where w_0 is a global bias, w_i is the bias of the i'th variable, $\mathbf{v}_i = [v_1, \ldots, v_k]$ is the latent representation of the i'th variable using k latent factors, and $\langle \mathbf{v}_i, \mathbf{v}_j \rangle$ is the dot product between the i'th and j'th which captures the second order interactions of the variable.

Matrix factorization methods, such as FMs, are particularly prone to *cold start* problems. In order to learn a latent representation of a given item, it needs to be present in the training data, but that is not possible with unreleased content (e.g. new levels in puzzle games). One possible way to deal with this in FMs is by including additional data, such as tags or other item features, which is possible to learn latent representations from.

V. DATA

The exact type of data that is available will depend on the game. However, we can define four categories that the data can belong to, following the approach by Wheat et al. [28]:

- *Historic data* that considers the average number of attempts by previous players, either on a per-level basis or per-player basis.
- *Player data* that capture aspects related to the player's skill and purchase tendency.
- *Level data* that capture specific types of game mechanics and descriptors of the level.
- *Agent data* that captures more dynamic information about the level and is extracted using a playtesting method.

Each of these categories will have an impact on the performance of the prediction model, but they may not be fully informative on their own. For example, player data is necessary to include for personalised predictions, but on its own, it does not say anything about the level itself or how easily the player can handle specific gameplay elements. Similarly, agent data may reveal some intrinsic difficulty/ground truth pass rate, but without level data, it is hard to account for biases in the agent algorithm (i.e. certain mechanics may be easier to learn for an agent compared to a human [16]), and personalised predictions are impossible without player data. Lastly, historic data may be informative, but it is not available on new content or for new players. We, therefore, experiment with combining the data in a number of ways to test the importance of each data category. We consider the following combinations:

- **Historic:** Only player-level-attempt information. For FMs, this is implicitly learned in the variable biases, while for the RF and NN approaches, it needs to be calculated explicitly. Only available for predictions on existing content.
- **Player+level:** Both player and level data, possible to use for personalised predictions.
- **Agent:** Only agent data, possible to use for non-personalised predictions.
- **All:** Player, level and agent data combined, possible to use for non-personalised or personalised predictions. It does not include features that are explicitly derived from historic observations.

A. Lily's Garden case study data

The data used in this study consists of the number of attempts each player has spent on a given level and is recorded between 2021-06-01 and 2021-12-01. Only players that have started playing the game in this interval and played at least 200 levels are included in the data set. We use data that come from the four different categories of data mentioned previously. An overview of these features is shown in Table I.

The player data consist of aggregate player data over the first 100 levels. We consider the average number of moves they have used to complete the levels as well as booster usage on the first 100 levels. These features capture the general

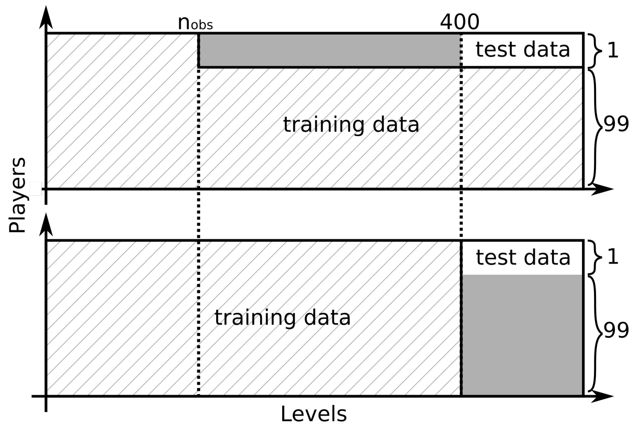


Fig. 4. Train/test split of the data for personalised predictions on levels with historic data and on levels in the cold start scenario. In this study we use $n_{\text{obs}} = 100$. The greyed out area is ignored observations for ensuring the experiments are evaluated on the same content. Adapted from [5].

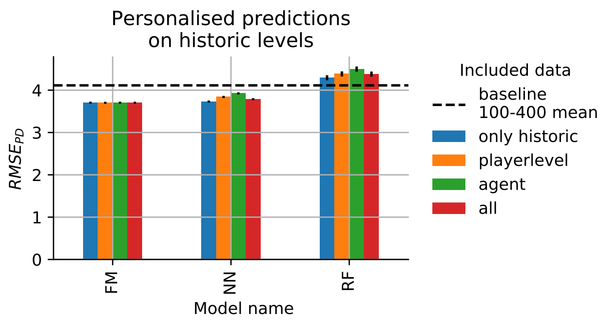


Fig. 5. RMSE of personalised predictions on content that has been played by other players.

competence of the players and their willingness to pay to progress, which are both strong predictors for how many attempts they will use on future levels [5].

The level attributes describe the size of the board and the goals and board pieces present in the level. Additionally, the number of colours and the entropy of the colour distribution are also included, which can affect how easy it is to make stronger combinations and thus the difficulty of the level. Unlike previous work where some of the level features consisted of historic data such as the average number of attempts on the level, for this study we only include features that are possible to determine before any players played the given level. This is necessary since in the experiments we consider the scenario with new levels where no historic data is available.

Lastly, for the agent data, we include information about how well the agent learned to play the level and how easily it learned to do so. We do this by recording some features at the point during training at which the average number of moves taken by the agent is the lowest as well as the standard deviation of the features between the time steps 1M to 9M.

VI. Experiment A: personalised predictions

The first research question (RQ1) is aimed at investigating the modelling of the perceived difficulty of a puzzle level for a specific player. We follow a similar approach as Kristensen et al. [5] where we use data from the commercial puzzle game, Lily's Garden, that is described in Section III. However, we extend the study by also considering a cold start scenario where there are no observations available for a given level. We denote the two cases as *playthrough/historic data* (PD) and *cold start* (CS).

For creating the data sets for training and testing, we split the data in two different ways, which are shown in Fig. 4. In the first case, we split the players into train/test sets with a 99-1 split but include observations of the players from the test set in the training set up until the first n_{obs} levels. In the second case, we split by level number, where the first 400 levels are used for training and the last 99 levels are used for testing.

The nature of the prediction task, where very difficult levels can have a large variance in attempts as shown previously in Fig. 3, means the best performing models in terms of root mean squared error (RMSE) and mean absolute error (MAE) may be different. Since the models are optimised using RMSE and we want the method to be more robust towards more difficult levels, we choose to use RMSE as the reported metric. We calculate it from the same group of players above level 400 in all cases.

We test out the three methods described in Section IV-B, and we train them using the four different combinations of data outlined in Section V. In the first case, we include a naive baseline prediction for each level which is calculated as the average number of attempts on the given level by the players in the training set. In the second case, we use a constant prediction of $y_{\text{baseline}} = 3.23$ that is calculated as the average number of attempts on levels 100-400.

A. Personalised predictions with historic data available

The performance of the methods in terms of RMSE in the first case is shown in Fig 5. The best-performing model is the FM model which only includes the player and level data. FMs, therefore, appear more robust in identifying possible bottlenecks for an individual player, where they possibly require a large number of attempts and may feel stuck. This can be used to the game designers' advantage by preemptively providing assistance to the players if, for example, the estimated perceived difficulty exceeds a certain threshold.

Another observation to note is that, while the NN and FM approaches perform better than the baseline, the RF model appears to perform worse than all other methods. A crucial difference between this study and the study by Kristensen et al. (2022) [5] is that the level features do not include the historic average attempts due to the fact that this information would not be available when estimating the difficulty of unreleased levels. Without such a strong predictor, the RF model does not appear to be able to infer a meaningful average prediction. Furthermore, both the FM and NN approaches do not employ average attempts as input, but they may be able to infer the average attempts on each level through either the complexity of the network or the variable biases, w_i, of the FM model.

The impact of using different types of data as inputs to the FM method is shown in Fig. 6. The results are in line

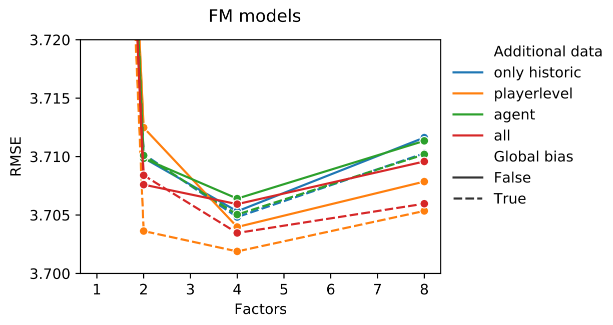


Fig. 6. RMSE of the tested FM models for personalised predictions on current levels as a function of number of factors. The colours indicates what additional datba that are included in the data set. The dashed line indicates whether w_0 is included in the model or not (see Eq. 1).

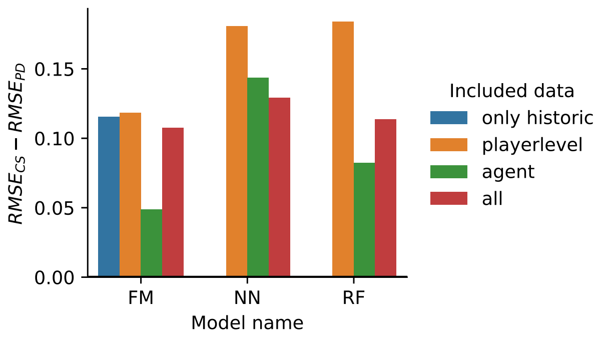


Fig. 7. Difference in RMSE for personalised predictions between the scenario where data about player playthroughs (**PD**) is available and the cold start (**CS**) scenario. A positive difference indicates that the error in the cold start scenario is higher.

with previous studies [5] in which the RMSE and MAE are around 3.8 and 2.3, respectively, and augmenting the data with player and level data leads to a minor improvement. An extension to the previous study also includes agent data. While agent data does not improve the predictions significantly for the FM method, it leads to a worse performance when combining it with the player and level data categories. A closer inspection of the training process shows that the performance on the test set worsens after just 50 iterations when including all data categories, while in the other cases, a performance drop happens after around 950 iterations. This suggests that a likely cause for the worse performance is the tendency of the model to overfit. Additionally, in a high-dimensional space with strongly correlated parameters, the Gibbs sampler used in the MCMC optimiser for the FM method may not converge [47], which is the case with the colour entropy and colours of the level attributes, and the minimum and standard deviation of the game length.

In terms of other model parameters, four factors seem to work the best for optimising the RMSE but at the cost of increasing the MAE. However, the difference in performance when using between two and four factors is not significant and does not contradict the results from previous work where two factors were concluded to be sufficient. Using additional data does also not warrant the use of more than four factors and can in fact worsen the performance. There is not a clear pattern of whether using a global bias improves the predictions, although the best-performing model does include a global bias.

B. Predictions in cold start scenarios

In the second case, the performance of the models is very similar as seen in Fig. 5. Excluding the models with only historic data since this is not available in the cold start scenario, all three FM models perform better than the comparable NN and RF models. Similarly, only the RF models perform worse than the constant baseline. The same parameters for the FM methods (4 factors, both global and variable bias included) lead to the best performance of the tested FM configurations.

Since the personalised predictions in both cases have very similar RMSE, we look at the difference between the errors in the two cases in Fig. 7 to better compare the performances. The results show that for all the methods, the error in the cold start scenario is higher regardless of data type. This is not surprising since the information from other players' performance on a level should be a strong predictor of the perceived difficulty of other players. In the FM approach, including agent data does lead to a smaller gap in error in the two scenarios, which suggests that agent data does contain important predictors for predicting difficulty that the FM model learns to utilise. This is also true for the other methods, but the difference between the two scenarios for the NN and RF model is still larger than in the FM case.

It is notable that the performance of the models is similar and that the FM method with no included data performs better than a constant baseline. Since the FMs reduce to a simple linear function that consists of a global bias and bias term for the players according to Eq. 1 in cold start scenarios, it suggests the learned biases for individual players are sufficient for capturing the players' general performance on both existing and new content. This is valuable information for game designers since it may be possible for them to use these learned biases for more individualised content.

VII. Experiment B: Cohort Predictions

The second research question is regarding the possibility of doing cohort predictions. Level designers are typically interested in knowing the average number of attempts on the levels, and since different cohorts reach the levels at different times, the measured average difficulty may change over time depending on the cohort. This experiment simulates this by comparing the aggregated average predictions per level with the average attempts on each level for the cohort in the test set. Similar to the previous experiment, we consider both the scenario where there is historic data from previous players' playthroughs on a given level and a cold start scenario. We use the same baselines as in the previous experiment.

We consider two ways of predicting the average number of attempts on a cohort level. The first approach is using the personalised predictions from the first experiment and computing the aggregated average on each level. For the second approach, since individual estimates may be noisy or

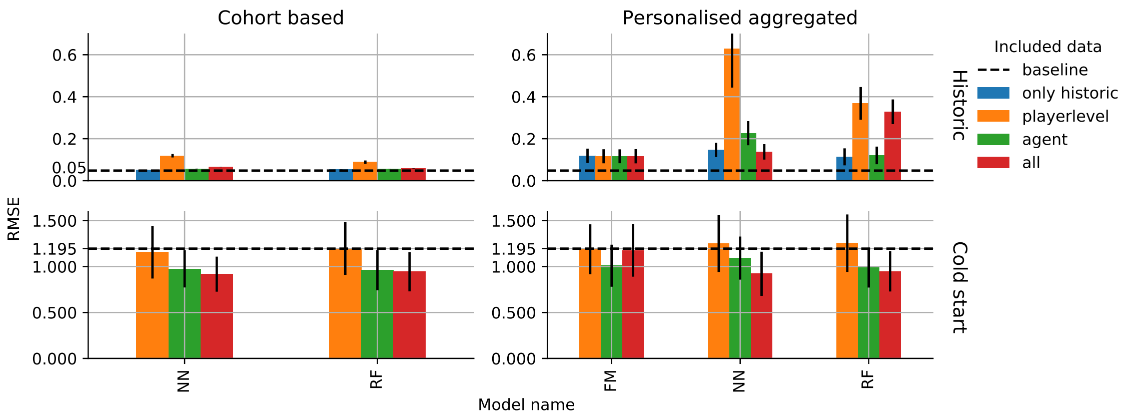


Fig. 8. RMSE of the different approaches for predicting the average difficulty of a level for a cohort. The left plots show the cohort-based per-level predictions, and the right plots show the aggregated average of the personalised predictions for the full data set. The top row is in the case where observations on the examined levels exist, while the bottom row is the cold start scenario. Note that historic data is not available in the cold start scenario.

inaccurate, we group the players into 10 random cohorts and use the aggregated average values of the player features to describe these cohorts. The regression target for each cohort is then their average number of attempts on the given level. This allows us to directly use the tested methods to predict the average number of attempts on a per-level basis of the cohort of players in the test set. These two approaches will be denoted by how the initial predictions are grouped (i.e. **personalised** or **cohort**) but we note that in both cases, the final estimates are on a cohort level.

The results of using the different investigated methods are shown in Fig. 8. The errors for all models when predicting the difficulty of existing content is very low, which is not surprising since the cohorts are generally not very different as demonstrated by the very small RMSE of the baseline. The personalised errors are generally larger than the cohort-based predictions, which is most likely due to the fact that these models are optimised to minimise the RMSE for each player. The optimal model, in this case, is not necessarily one that captures the average number of attempts by the whole cohort but rather one that can account for the high within-level variance. This is clear for the FM models, in which using only 1 factor leads to an error similar to the baseline but more factors lead to higher RMSE in this experiment where we show the results from the 4-factor models.

In the cold start scenario, both the NN and RF models are the best performing models and appear to have similar performance for both personalised and cohort predictions. Additionally, including all possible categories works the best for the NN and RF methods, similar to what was found in the first experiment. This is not the case for FM with all data included, though, which seems to suffer from the same type of overfitting seen previously. The model parameters for the best performing FM models were similar as well, where including both a global and variable bias as well as using 2-4 latent factors lead to the best performance.

Generally, including only player and level data is not sufficient by themselves to perform better than the baseline. Only when including agent data it is possible to have more accurate predictions than a simple constant average. By looking at the feature importances for the RF models, we also find that the agent features are on average 5-6 times more impactful than the level features, with the minimum game length and completion rate being among the most important. These results highlight the importance of having a playtest agent that can capture dynamic aspects of a level for enabling the prediction models in cold start scenarios. It also supports the conclusion by Gudmundsson et al. [16] that combining agent performance with a level features is useful in order to be able to account for differences in how players and AI agents learn and behave.

Another notable result is that cohort predictions are on-par with personalised predictions for estimating the average level's difficulty in the cold start scenario. To visualise the differences between the two approaches, Fig. 9 shows the predictions from each method compared to the actual average number of attempts. Both methods tend to overestimate the difficulty of easy levels and underestimate the difficulty of the hard levels and tend to predict the mean number of attempts. This suggests that they are both underfitted or that the features are not informative enough to generalise to new levels.

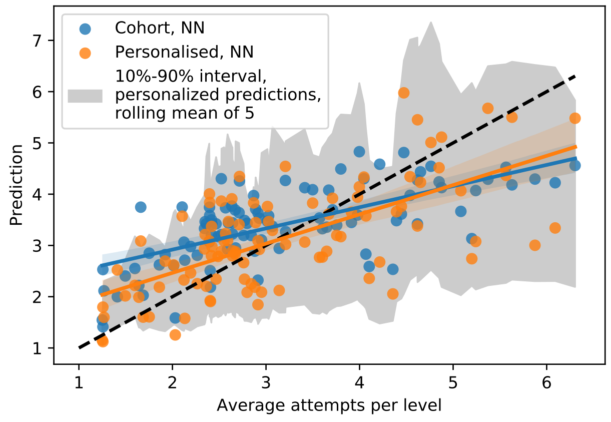


Fig. 9. Example of predictions of the NN method with cohort and personalised predictions using all categories of data in the cold start scenario.

The personalised predictions tend to have a larger variance, especially at higher difficulties, which can be seen from the 80% prediction interval. This is not completely unexpected due to the high variance in attempts between players on harder levels, as shown in Fig. 3.

VIII. Discussion

From the results of the experiments, it is clear that the tasks of creating personalised difficulty predictions for players on content with available player playthrough data and without require different approaches. While a part of this is related to generalisability and overfitting, a deeper discussion about the data and models can help understand in which circumstances one approach works better than another. In this section we therefore focus on three topics: the difference between models for old and new content, the feasibility of using any of these models in practice and playtest agent data.

A. *Choice of model*

One complication with a combined difficulty framework that can work on both old and new content is that the available data for the two scenarios are inherently different. For new content, there are no direct observations of how many attempts actual players spend on a level. This *cold start problem* can especially be an issue for factorization-based approaches, such as FMs, since the bias and latent factors associated with new content can not be properly inferred with no observations. While the inclusion of additional data in FM methods, such as tags [48] or player/level/agent data, can alleviate this problem, we still observe that the predictions tend to stay close to the global average and not capture the full range of behaviours of players on new levels. This suggests that the contribution of the additional data to the model prediction is limited, and further work on ensuring the included data is impactful is necessary.

For the RF and NN approaches, the cold start problem does not exist to the same extent. FMs learn latent descriptions for each item, but the RF and NN methods instead only utilise the included data features. This makes these latter models not rely on modelling specific items, at the cost of not having as accurate personalised predictions, as seen in the first experiment It is therefore also not surprising that the difference between personalised and cohort predictions on new content is insignificant with those two methods: the features with high predictive power (mainly agent data) are the same, so the information that is possible to extract is the same. The data for the personalised model training is therefore essentially an oversampled dataset of the full dataset.

It is worthwhile to note that, for the RF method, while the results from the second experiment for both personalised and cohort predictions were promising, the results from the first experiment showed that the RF method performs worse than the baseline for predicting the individual perceived difficulty. The inability of the RF method to capture individual differences may be due to the greedy nature of the algorithm and the fact that the chosen player features were too uninformative compared to both agent and level data. More complex player data, such as recent history or more detailed behavioural data, may improve this approach, but this is equally likely to benefit the NN and FM approaches. Ultimately, both regarding training and the ability to capture complex behaviours, a neural network based approach is more feasible for personalised predictions on both old and new content, though at the cost of interpretability. For accurate, interpretable methods on existing content, an FM approach is better.

B. *Difficulty modelling in practice*

An important question about this approach is whether these results are reflective of how these methods would work in practice. To answer that, we consider some aspects regarding how we collect and split the data.

The data that is used in this study was collected over a 6 month period, but there is no information about when the players started or stopped playing or which cohort they might belong to. In order to enable the level designers of a live game to be proactive, a more realistic approach would be to consider whether the player has been active recently or not. While the approach used in this study made it possible to compare it with previous work (Kristensen et al. (2022) [5]), an idea for a future study is using a rolling window where the training data only includes data before a given date, and the test data only includes after this date up until the prediction date where the level designers would use the results.

Another consideration regarding the way the data is split is also whether the scenario of having historic data or the cold start scenario is necessary. While the cold start scenario is representative of completely new content, there are often other concerns when creating new levels, such as whether the puzzle *feels* fun or if the solution(s) are clear. This is a much harder problem to solve and requires the level designers to play through the levels regardless. This also means that, in practice, there is a smooth transition from a complete cold start scenario to a scenario with *some* historic data available – and not just by the level designers but also by other players due to pre-releases to certain regions or A/B tests. Since FMs should work well with such sparse data, an interesting line of research would therefore be to investigate how many observations of attempts on a level are necessary to beat the baseline. This question of how many observations are necessary has been explored by Kristensen et al. [5] for players, but it would be valuable to learn whether the results are true for levels too.

Lastly, we note that in order to address this cold start scenario, we experimented with treating the agent as a player. It is a way that could help alleviate the cold start problem in FMs where enriching item data by adding new ratings (e.g. [49]) can augment very sparse observations. However, initial experiments did not show promising results, and an even stronger tendency to predict the global average was observed. Using agent behaviour as a direct stand-in for a player, therefore, requires more finesse than simply oversampling observations of the agent, and possible ways to improve the agent will be discussed next

C. *Playtesting agent role in difficulty estimation*

While the agent data had a large positive impact on the performance of the models, there are two major aspects that can be considered for further work: extracting relevant level and agent behavioural data, and diversifying the agent.

In this study, the agent was trained on a single level at a time. Only 8 different random seeds were used, which limits the number of configurations of a given level that could be explored. Given that the random seed can greatly affect the chance of success [20], this not only means that the chosen seeds may have been more favourable to some levels but also the agent data may not be fully representative of the complete player experience where thousands of seeds are used. While an option could be to include more seeds during training, the RL algorithm can be very sensitive to hyperparameter tuning and such curricula [37]. An interesting avenue for future research is therefore finding an optimal balance in this trade-off between finding a representative sample while still enabling learning optimal strategies on the level, which leans into the *curriculum learning* domain [50].

A more diverse level sample may be necessary to ensure the data is distinct and informative. For example, the pass rate of the agent was 93% (or approximately 1.1 attempt per complete) across all levels, which also means in a majority of the cases, the agent only spent 1 attempt on all 8 seeds. This leads to degeneracy in the levels where they are indistinguishable from one another in the agent data, despite having different player pass rates. This can also explain the issue of many of the predictions trending towards the global average since an otherwise informative feature does not allow the algorithms to easily differentiate between levels especially the easier and medium difficulty ends of the scale. A possible way to overcome this is by including multiple agents that are trained with different reward functions (similar to [38]) which could allow for more comprehensive data regarding the gameplay.

With the ultimate goal of being able to employ this difficulty framework in a live game, the main concern is whether the accuracy on especially new content is accurate enough for the level designers to rely on. Gudmundsson et al. [16] consider the same business problem and present a method based on a similar two-step approach that employs a playtest agent to extract an agent pass rate and subsequently use in a prediction model. They mentioned that King has used it for two of their games and reported the best model has a 4.0% to 6.6% MAE on the pass rate. Translating our results from attempts to pass rate, our method has a 8.1% MAE. However, it is not possible to translate the result directly since they do not report the absolute magnitude of the win rates, which means if they have more hard levels with low, but similar, pass rates, the reported MAE is also expected to be lower. Additionally, they considered two different games which share many similar mechanics but still observed a difference in the predicted performance. While it is therefore hard to make a direct comparison, the results in this study can still inform the level designers about the scale of difficulty, and further ideas could involve classifying the levels into easy, medium and hard difficulties rather than directly predicting the number of attempts. Furthermore, the variation of results within the experiment by Gudmundsson et al. [16] and between that and this experiment, albeit minor, can be studied further to investigate whether there are specific differences in the player base or the characteristics of the games or the methods employed that lead to the observed results.

IX. Conclusion

In this study we considered two questions: is it possible to estimate the perceived difficulty of a puzzle level for a given player on both existing newly generated (human or otherwise) levels, and is it possible to do this on a cohort level? To answer these questions, we tested out three different prediction methods for estimating the number of attempts each player is expected to spend in order to complete a given level using data from a live commercial puzzle game. Additionally, we experimented with including and combining data belonging to four categories: historic data, player and level data, as well as agent data collected by a reinforcement learning playtesting agent

For personalised predictions on existing content, with previous players' data, the factorization machines approach performed significantly better than the random forest and neural network methods. However, including all three types of data for the factorization machine led to worse performance most likely due to overfitting and bad optimisation conditions. The neural network approach worked better than the baseline, while the random forest performed worse. This is explained by the fact that the NN method is able to capture the historic average number of attempts through the complexity of its network and sequential training that enables better handling of large data sets.

To estimate the average difficulty of new content, in a cold start scenario, the neural network approach worked the best for both personalised and cohort-based predictions, while the factorization machines method did not work very well due to the cold start problem common in factorization methods.

It was also found that including the data collected by a playtest agent is necessary for the methods to perform better than the global average. However, since the agent consistently performed better than most players, it did not allow to differentiate easily between the easy to medium difficulty levels.

References

[1] M. Csikszentmihalyi and M. Csikzentmihaly, *Flow: The psychology of optimal experience.* Harper & Row New York, 1990, vol. 1990.

[2] "Gaming market - growth, trends, covid-19 impact, and forecasts (2022-2027)," https://www.mordorintelligence.com/industry-reports/global-gaming-market, accessed: 2022-07-08.

[3] "Global games market to generate $175.8 billion in 2021; despite a slight decline, the market is on track to surpass $200 billion in 2023)," https://newzoo.com/insights/articles/global-games-market-to-generate-175-8-billion-in-2021-despite-a-slight-decline-the-market-is-on-track-to-surpass-200-billion-in-2023/, accessed: 2022-07-08.

[4] "How much is the gaming industry worth in 2022? (revenue & stats)," https://earthweb.com/how-much-is-the-gaming-industry-worth/, accessed: 2022-07-08.

[5] J. Kristensen, C. Guckelsberger, P. Burelli, and P. Hämäläinen, "Personalized game difficulty prediction using factorization machines," *User Interface Software and Technology*, vol. 2022, 2022.

[6] A. Denisova, P. Cairns, C. Guckelsberger, and D. Zendle, "Measuring perceived challenge in digital games: Development & validation of the challenge originating from recent gameplay interaction scale (CORGIS)," *International Journal of Human-Computer Studies*, vol. 137, p. 102383, May 2020. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1071581919301491

[7] A. Denisova, C. Guckelsberger, and D. Zendle, "Challenge in Digital Games: Towards Developing a Measurement Tool," in *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. Denver Colorado USA: ACM, May 2017, pp. 2511–2519. [Online]. Available: https://dl.acm.org/doi/10.1145/3027063.3053209

[8] J. Chen, "Flow in games (and everything else)," *Commun. ACM*, vol. 50, no. 4, p. 31–34, apr 2007. [Online]. Available: https://doi.org/10.1145/1232743.1232769

[9] R. M. Ryan, C. S. Rigby, and A. Przybylski, "The Motivational Pull of Video Games: A Self-Determination Theory Approach," *Motivation and Emotion*, vol. 30, no. 4, pp. 347–363, 2006.

[10] A. Tyack and E. D. Mekler, "Self-Determination Theory in HCI Games Research: Current Uses and Open Questions," in *Proc. Conference on Human Factors in Computing Systems (CHI)*. ACM, 2020, pp. 1–22.

[11] J. T. Alexander, J. Sear, and A. Oikonomou, "An investigation of the effects of game difficulty on player enjoyment," *Entertainment Computing*, vol. 4, no. 1, pp. 53–62, Feb. 2013. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1875952112000134

[12] J. H. Brockmyer, C. M. Fox, K. A. Curtiss, E. McBroom, K. M. Burkhart, and J. N. Pidruzny, "The Development of the Game Engagement Questionnaire: A Measure of Engagement in Video Game-Playing," *Journal of Experimental Social Psychology*, vol. 45, no. 4, pp. 624–634, 2009.

[13] M. Pusey, K. W. Wong, and N. A. Rappa, "The Puzzle Challenge Analysis Tool. A Tool for Analysing the Cognitive Challenge Level of Puzzles in Video Games," *Proceedings of the ACM on Human-Computer Interaction*, vol. 5, no. CHI PLAY, pp. 1–27, Oct. 2021. [Online]. Available: https://dl.acm.org/doi/10.1145/3474703

[14] S. Roohi, C. Guckelsberger, A. Relas, H. Heiskanen, J. Takatalo, and P. Hämäläinen, "Predicting game difficulty and engagement using ai players," *Proceedings of the ACM on Human-Computer Interaction*, vol. 5, no. CHI PLAY, pp. 1–17, 2021.

[15] J. T. Kristensen, A. Valdivia, and P. Burelli, "Estimating player completion rate in mobile puzzle games using reinforcement learning," in *2020 IEEE Conference on Games (CoG)*, 2020, pp. 636–639.

[16] S. F. Gudmundsson, P. Eisen, E. Poromaa, A. Nodet, S. Purmonen, B. Kozakowski, R. Meurling, and L. Cao, "Human-like playtesting with deep learning," in *2018 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 2018, pp. 1–8.

[17] R. Hunicke, "The case for dynamic difficulty adjustment in games," in *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*, 2005, pp. 429–433.

[18] M. Zohaib, "Dynamic difficulty adjustment (dda) in computer games: A review," *Advances in Human-Computer Interaction*, vol. 2018, 2018.

[19] M. Gonzalez-Duque, R. B. Palm, and S. Risi, "Fast game content adaptation through bayesian-based player modelling," in *2021 IEEE Conference on Games (CoG)*, 2021, pp. 01–08.

[20] S. Xue, M. Wu, J. Kolen, N. Aghdaie, and K. A. Zaman, "Dynamic Difficulty Adjustment for Maximized Engagement in Digital Games," in *Proceedings of the 26th International Conference on World Wide Web Companion - WWW '17 Companion*. Perth, Australia: ACM Press, 2017, pp. 465–471. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3041021.3054170

[21] D. B. Or, M. Kolomenkin, and G. Shabat, "Dl-dda-deep learning based dynamic difficulty adjustment with ux and gameplay constraints," in *2021 IEEE Conference on Games (CoG)*. IEEE, 2021, pp. 1–7.

[22] J. Pfau, J. D. Smeddinck, and R. Malaka, "Enemy within: Long-term motivation effects of deep player behavior models for dynamic difficulty adjustment," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020, pp. 1–10.

[23] T. Constant and G. Levieux, "Dynamic Difficulty Adjustment Impact on Players' Confidence," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. Glasgow Scotland Uk: ACM, May 2019, pp. 1–12. [Online]. Available: https://dl.acm.org/doi/10.1145/3290605.3300693

[24] J. Li, H. Lu, C. Wang, W. Ma, M. Zhang, X. Zhao, W. Qi, Y. Liu, and S. Ma, "A Difficulty-Aware Framework for Churn Prediction and Intervention in Games," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. Virtual Event Singapore: ACM, Aug. 2021, pp. 943–952. [Online]. Available: https://dl.acm.org/doi/10.1145/3447548.3467277

[25] F. Mourato, F. Birra, and M. P. dos Santos, "Difficulty in action based challenges: success prediction, players' strategies and profiling," in *Proceedings of the 11th Conference on Advances in Computer Entertainment Technology*, 2014, pp. 1–10.

[26] M. Van Kreveld, M. Löffler, and P. Mutser, "Automated puzzle difficulty estimation," in *2015 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 2015, pp. 415–422.

[27] J. T. Kristensen, A. Valdivia, and P. Burelli, "Statistical modelling of level difficulty in puzzle games," in *2021 IEEE Conference on Games (CoG)*. IEEE, 2021, pp. 1–8.

[28] D. Wheat, M. Masek, C. P. Lam, and P. Hingston, "Modeling perceived difficulty in game levels," in *Proceedings of the Australasian Computer Science Week Multiconference*, 2016, pp. 1–8.

[29] A. E. Zook and M. O. Riedl, "A temporal data-driven player model for dynamic difficulty adjustment," in *Eighth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2012.

[30] S. Ariyurek, A. Betin-Can, and E. Surer, "Automated video game testing using synthetic and humanlike agents," *IEEE Transactions on Games*, vol. 13, no. 1, pp. 50–67, 2019.

[31] A. Latos, "Automated playtesting on 2d video games. an agent-based approach on nethackclone via iv4xr framework," Master's thesis, Utrecht University, 2022.

[32] S. A. Dukkancı, "Level generation using genetic algorithms and difficulty testing using reinforcement learning in match-3 game," Master's thesis, Middle East Technical University, 2021.

[33] J. Bergdahl, C. Gordillo, K. Tollmar, and L. Gisslén, "Augmenting automated game testing with deep reinforcement learning," in *2020 IEEE Conference on Games (CoG)*. IEEE, 2020, pp. 600–603.

[34] S. Stahlke, A. Nova, and P. Mirza-Babaei, "Artificial players in the design process: Developing an automated testing tool for game level and world design," in *Proceedings of the Annual Symposium on Computer-Human Interaction in Play*, 2020, pp. 267–280.

[35] B. Horn, J. A. Miller, G. Smith, and S. Cooper, "A monte carlo approach to skill-based automated playtesting," in *Fourteenth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2018.

[36] J. Pfau, A. Liapis, G. Volkmar, G. N. Yannakakis, and R. Malaka, "Dungeons & replicants: automated game balancing via deep player behavior modeling," in *2020 IEEE Conference on Games (CoG)*. IEEE, 2020, pp. 431–438.

[37] J. T. Kristensen and P. Burelli, "Strategies for using proximal policy optimization in mobile puzzle games," in *International conference on the foundations of digital games*, 2020, pp. 1–10.

[38] L. Mugrai, F. Silva, C. Holmgård, and J. Togelius, "Automated playtesting of matching tile games," in *2019 IEEE Conference on Games (CoG)*. IEEE, 2019, pp. 1–7.

[39] Y. Shin, J. Kim, K. Jin, and Y. B. Kim, "Playtesting in match 3 game using strategic plays via reinforcement learning," *IEEE Access*, vol. 8, pp. 51 593–51 600, 2020.

[40] I. Kamaldinov and I. Makarov, "Deep reinforcement learning in match-3 game," in *2019 IEEE conference on games (CoG)*. IEEE, 2019, pp. 1–4.

[41] E. R. Poromaa, "Crushing candy crush: predicting human success rate in a mobile game using monte-carlo tree search," Master's thesis, KTH, School of Computer Science and Communication (CSC), 2017.

[42] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv e-prints*, pp. 1–12, 7 2017. [Online]. Available: http://arxiv.org/abs/1707.06347

[43] Y. Zhao, I. Borovikov, F. de Mesentier Silva, A. Beirami, J. Rupert, C. Somers, J. Harder, J. Kolen, J. Pinto, R. Pourabolghasem *et al.*, "Winning is not everything: Enhancing game development with intelligent agents," *IEEE Transactions on Games*, vol. 12, no. 2, pp. 199–212, 2020.

[44] "Appbrain: Google play ranking: The top grossing puzzle games in the united states," https://www.appbrain.com/stats/google-play-rankings/top_grossing/puzzle/us#, accessed: 2022-07-08.

[45] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[46] S. Rendle, "Factorization Machines with libFM," *ACM Transactions on Intelligent Systems and Technology*, vol. 3, no. 3, pp. 1–22, May 2012. [Online]. Available: https://dl.acm.org/doi/10.1145/2168752.2168771

[47] A. Justel and D. Peña, "Gibbs sampling will fail in outlier problems with strong masking," *Journal of Computational and Graphical Statistics*, vol. 5, no. 2, pp. 176–189, 1996. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/10618600.1996.10474703

[48] Y. Juan, Y. Zhuang, W.-S. Chin, and C.-J. Lin, "Field-aware factorization machines for ctr prediction," in *Proceedings of the 10th ACM conference on recommender systems*, 2016, pp. 43–50.

[49] F. Tahmasebi, M. Meghdadi, S. Ahmadian, and K. Valiallahi, "A hybrid recommendation system based on profile expansion technique to alleviate cold start problem," *Multimedia Tools and Applications*, vol. 80, no. 2, pp. 2339–2354, 2021.

[50] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone, "Curriculum learning for reinforcement learning domains: A framework and survey," *arXiv preprint arXiv:2003.04960*, 2020.

[51] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html

[52] F. Pardo, A. Tavakoli, V. Levdik, and P. Kormushev, "Time limits in reinforcement learning," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 4045–4054. [Online]. Available: https://proceedings.mlr.press/v80/pardo18a.html

[53] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

Appendix A
Method Parameters

A. Playtest agent setup

We use an implementation of PPO from Stable-Baselines3 [51]. We use a three-layer convolutional neural network (ConvNN) with 32 filters each, stride of 1, kernel sizes of [3, 2, 2] and paddings of sizes [1, 0, 0]. Additionally, to make the method time- and progress-aware, we concatenate the flattened output from the ConvNN of size 64 with a vector with information about the progress and limit for both episode length and goals, as suggested in [52]. The critic network head has two fully connected layers of size 128, and the actor network head has two fully connected layers of size 32. All activation functions are exponential linear units (ELUs).

We use the following hyperparameters:

- `ent_coef`: 0.01
- `learning_rate`: 3e-4
- `batch_size`: 512
- `n_steps`: 256
- `n_epochs`: 4
- `clip_range_vf`: 1.0
- `gamma`: 0.99
- `gae_lambda`: 0.95

B. Random forest setup

We use the random forest regressor implementation from the scikit-learn library version 1.0.2 [53]. Due to the size of the data we also incrementally train the model using IncrementalTrees[1]. We use the following hyperparameters for the RF (*non-personalized/personalized*) setup (default parameters unless otherwise noted):

- `n_estimators`: 150/260
- `max_depth`: 8/8
- `max_features`: 3/3

[1]https://github.com/garethjns/IncrementalTrees

C. Neural network setup

We use a simple fully connected feed-forward neural network with three layers and 256 and 64 neurons for personalised and non-personalised predictions each, respectively. To improve generalisation, we also employ 0.1 dropout layers and an L2 regularisation of strength 1×10^{-5} during training.

D. Factorization machines setup

We use the implementation of `LibFM` [46] version 1.4.2 with a MCMC inference method for model parameters. We use an initial standard deviation for initialising $\mathbf{v}$ of 0.1. We also use the `meta` option to assign players and levels into separate groups, which allows for a more complex regularisation of parameters by assigning different regularisation parameters for each group.

Biography Section

Jeppe Theiss Kristensen received his MSc degree in astronomy in 2016 from Aarhus University.

He is currently an industrial Ph.D. student at the IT University of Copenhagen and collaborates with an industrial partner, Tactile Games.

His research focus involves player modelling and reinforcement learning for playtesting methods in puzzle games.

Paolo Burelli received his MSc degree in information technology in 2007 from the University of Udine and his PhD degree in 2012 from the IT University of Copenhagen.

Dr. Burelli is currently an Associate Professor at the IT University of Copenhagen and a member of the Games Technical Committee of the IEEE Computational Intelligence Society.

He researches computational models of user experience and behaviour in digital environments and, throughout his experiences in the academic world and in the game industry, he has published more than 30 articles on this and related topics.

Part III

Epilogue

6 Discussion and Conclusions

This dissertation's central research theme concerns how difficulty in puzzle games can be modelled and operationalised. A vital component of that work has been considering its applicability in an industrial context where technical and design restrictions, together with a focus on enabling informed decisions, have guided this work. In this chapter, the contributions to the research fields of player modelling and AI in games will be presented as well as the possibilities for improving and extending the work. A final summary will be given at the end.

6.1 Contributions

This dissertation is concerned with two research questions: how player data can be employed to model and operationalise difficulty, and how this is possible with little to no player data. Through the five included papers, the research questions have been illuminated and have led to the following contributions.

6.1.1 Modelling player behaviour in puzzle games

As an initial step for modelling and operationalising difficulty, we propose to model the number of moves players spend to complete different levels as a negative binomial distribution (see Section 3.1). This statistical interpretation provides a way to directly estimate the difficulty measured by the pass rate by asking what the probability of spending fewer moves than the move limit is, given the fitted negative binomial distribution. Furthermore, it allows level designers to operationalise difficulty as a function of the move limit, supporting more data-driven decisions on their behalf.

In order to validate the hypothesis that a negative binomial distribution can describe the move distribution, we consider data from 4,000 levels from the game Lily's Garden by Tactile Games. From this, we consider two criteria for determining the goodness of fit.

First, we use a Kolmogorov-Smirnoff distance D to check how different the actual and modelled distribution are and find that 99% of the levels are well described ($D < 5\%$). However, we also find that the fitting method reaches the parameter threshold in around 15% of the cases. This typically happens when the distribution exhibits a steady increase which leads to fitting to the tail end of the distribution.

The second criterion is comparing the estimated pass rate with the actual pass rate, where we find that the estimated pass rate generally underestimates the actual pass rate. However, these two metrics are strongly correlated, and the difference can be corrected when using this approach in practice.

Two other valuable insights for level designers have also come from this research. The first insight is related to the data itself, where excluding players that use booster items is crucial for good fits. For a majority of the levels, when players who use booster items are included in the data, the move distributions show a distinct "photo finish" behaviour on the last two moves where players save up big combos they can fire off just before the move limit – but still with one move to spare.

The second insight is related to the distribution that highlights certain aspects of difficulty. Where previous work sometimes models human behaviour with a symmetric normal distribution (e.g. Mourato et al. (2014)), this work proposes an asymmetric distribution. This means that when adjusting the move limit, the pass rate is not equally affected by either adding or removing moves. Furthermore, we find evidence that the recorded moves used by the playtesting agent follow a similar distribution. Given the long tail of the distribution, it follows that the agent will sometimes struggle to complete a level, which can lead to slower learning since it rarely reaches the goal.

While these results focus on puzzle games, we expect they can be extended to similar games (such as time taken to complete other challenges), and we provide the mathematical foundation to test if that is the case.

6.1.2 Personalised difficulty prediction

An essential part of the player experience is the perceived difficulty of the player. To account for that when modelling and operationalising difficulty, we propose using Factorisation Machines, known from recommender system literature, to estimate the number of attempts the individual player will use to complete a level. The strength of this method is that it describes the interaction between players and levels by learning latent descriptions of both, which allows the method to estimate the perceived difficulty of players on levels they have not yet encountered. While a similar approach has been considered by Zook and Riedl (2012), we can utilise additional data and provide an interpretation of the learned latent factors in this novel approach.

We consider the first 499 levels of Lily's Garden to validate our approach and use 700,000 players in our study (see Section 3.2). We also collect data regarding the level mechanics and the player performance and purchase history, which allows a broader range of methods to be tested and measure the impact of including additional data.

We treat the problem of personalised predictions as a regression problem. We test out an FM and RF method for this purpose and compare them with a baseline computed by the current level average, representing the current way level designers use difficulty. Additionally, we investigate how the number of levels a player has played affects the accuracy of the personalised predictions. We calculate the mean absolute error (MAE) and root mean squared error (RMSE) for evaluation.

Both methods perform better than the baseline after 20 to 30 observations. Including additional data can improve the predictions even further. The RF method performs better with fewer observations, but once more than 100 observations of the player are included, the FM method has comparable or better performance. An additional benefit of the FM method is that the latent factors can be interpreted as player skill and level variance, which the level designers can use for other customised content.

The investigation is extended in Chapter 5 where we consider the possibility of using personalised predictions in cold start scenarios where we have no historical player data on the levels, which has, to our knowledge, not been considered in previous works in similar contexts (e.g. Gudmundsson et al. (2018); Roohi et al. (2021); Mugrai et al. (2019)). In this extended study, we also include a NN method and data from a playtesting agent. The results show that the NN method is a better approach for both cohort and personalised predictions, especially when the agent data is combined with player and level data.

6.1.3 Robust playtesting agent

The dissertation has introduced an RL-based playtesting agent that can be used to play through puzzle levels (see Chapter 4). It uses an implementation of PPO, (Schulman et al., 2017) which helps ensure the learned policy does not fail catastrophically. We have also identified other additions from this research that ensure more stable learning, including shuffling the colours, early resetting and correctly applying action masks.

In order to use the agent for playtesting purposes, we have explored plausible approaches for training the agent and employing it in practice. Training the agent once on a random curriculum of levels leads to the highest correlation with player pass rates, which is also the most feasible approach to use in an industrial context since it is faster to evaluate new levels. Any additional training leads to more variance in the agent's skill and thus a lower correlation, which highlights that consistency is key when using playtesting agents. Furthermore, the agent does not need to emulate human behaviour exactly since the difference needs to be adjusted for in a subsequent step regardless, which we do by testing personalised predictions using the three approaches described in Chapter 5.

Another observation is that in some instances, when the agent struggles to complete a level, an alternative is to only consider the top 5-10% best evaluations of the level. Similar results have been observed in subsequent work by Roohi et al. (2021), which corroborates the hypothesis that the performance of the agent may exhibit a large variance (see Section 3.1) that can cause outliers in the data and thus worse correlation with player behaviour.

Lastly, reducing the stochasticity of the environment, such as limiting the number of random seeds, can significantly help the agent learn the optimal strategies for even the hard levels. This can provide the basis for further development of the agent where a curriculum can be built for the agent or train multiple agents with different playstyles.

6.2 Limitations and future work

The research of this dissertation has led to actionable contributions that have already been used in an industrial context. In order to further improve the impact of these methods, there are a number of questions and limitations that could be addressed in future work. Each paper has put forth suggestions for future work, and in this section, we group them by topic and more critically discuss them.

6.2.1 Pass rate as difficulty metric

The pass rate, or inversely the average attempts per complete, is widely adopted in both literature and game studios. It is often this type of data that game companies share with researchers (e.g. Roohi et al. (2021); Poromaa (2017)), and it has a straightforward interpretation: the average attempts metric gives a tangible measure that level designers can use to compare their own performance with and is strongly correlated with the amount of time players will spend on the levels. User studies also typically find that the perception of difficulty is correlated with the number of successes compared to failures (e.g. Alexander et al. (2013); Denisova et al. (2020)). The question is whether this metric is helpful on its own. In the following discussion, three additional metrics will be discussed which could further improve the applicability of any difficulty modelling framework.

Perceived challenge and uncertainty

While we have focused on operationalising difficulty as the average number of attempts or pass rate, as discussed by Denisova et al. (2020), the perceived *challenge* can depend on multiple things, where the difficulty is just one of the aspects. Another aspect that can affect the perceived challenge is uncertainty, and in puzzle games, many elements can contribute to uncertainty about the outcome.

One example is the chance of spawning different coloured basic pieces. The more uniform this colour distribution is, the more difficult it is to create large combinations for creating power pieces, and the more difficult the level is, which has also been experimentally confirmed through A/B tests at Tactile Games. However, this information is not available to the player, leading to uncertainty about the optimal strategy.

Similarly, other game mechanics can contribute to uncertainty, such as a board piece that can spread and cover basic pieces. This mechanic can force the player to change strategy or ruin otherwise carefully laid plans, increasing the perceived difficulty and leading to high frustration, as evidenced by user reports in Tactile Games gathered by the player care employees.

The number of attempts a player spends to complete a given level is therefore only one aspect of the perceived challenge. An enhanced framework for modelling the player experience could include ways to capture this uncertainty. A possible future research direction is how this can be estimated from player statistics. One idea is extending the research of modelling the move distributions (Kristensen et al., 2021) and using the move distribution variance as a proxy for uncertainty. This could also be extended with the results from the FM model (Kristensen et al., 2022) to further model individual players and individual differences. Lastly, it is also an option to utilise a playtesting agent to test and evaluate the variance of stochastic elements in the game design.

Churn rate

Another important metric for game designers to consider is the churn rate since it is a direct sign of low engagement of the players, and high retention of players is crucial for monetisation. Roohi et al. (2021) adopt a more complete view of difficulty prediction where churn rate is also considered in the modelling process. By simulating the progression of a group of players with varying degrees of skill, persistence and boredom – the latter two affecting churn rate – on the first 168 levels of Angry Birds Dream Blast, their extended model allows them to explain the observed player churn and pass rates accurately. Similarly, Xue et al. (2017) use a probabilistic progression graph to model the players' trajectories in various games, where the player at a given level is assigned separate transition probabilities for churning, failing the level or progressing to the next level. Since this satisfies the Markov property (Section 2.2), they can use dynamic programming methods to optimise the player progression for maximal player retention.

In order to better capture how the difficulty changes depending on cohort level, an extension to the work in this dissertation could be to include churn. This could also enable optimising the level funnel, which is particularly relevant in special events (such as Christmas releases) where there is limited time to complete the content. Results from A/B tests conducted by Tactile Games where the ordering of the levels was randomised have not revealed any significant differences. However, a more complete framework like the ones discussed above could enable a more analytical approach to optimise the ordering. Churn prediction has also been the focus in previous work (Kristensen and Burelli, 2019), but if these predictions can not be used to recommend actionable decisions, such as adjusting the difficulty, they are hard to use in practice. Therefore, an interesting line of research is how we can combine personalised difficulty predictions from this dissertation with personalised churn probabilities. Rather than relying on simulations of player groups (e.g. Roohi et al. (2021)) or relying on historical data (e.g. Xue et al. (2017)), such an approach has the promise to enhance the player experience even further.

Near-win attempts

One last metric to consider is being able to measure how close the player is to winning. A strong psychological hook to get players to continue playing or purchasing boosters is if they feel they were close to winning and can win next time/using a booster now. This is in the company's interest for optimising revenue, but for players, it is also a matter of having the agency for deciding whether they complete the level or not to ensure a positive experience (Juul, 2009), regardless of whether it is from skill or other kinds of resources. Earlier work hinted that such player behaviour is measurable, such as the "photo finish" observed in the move distribution (Section 3.1), but future work could look more closely at which types of levels and mechanics that lead to the player feeling close to winning.

6.2.2 Impact of booster items on predictions

Whenever data needs to be extracted for analysis, it is necessary to make certain choices as to how to filter it. These choices can significantly affect the results that follow from analyses and modelling based on this data, making it harder for researchers to compare results.

One example is the work in Section 3.1 where we exclude players that used boosters that added moves to the level and other in-game items. This is around 15% of the player base per level, which is not an insignificant amount, and more importantly, the filtered data is different from what Tactile Games typically use for their day-to-day operations where they do not apply the same filtering. Using boosters makes it easier for the player to pass the level, and indeed,

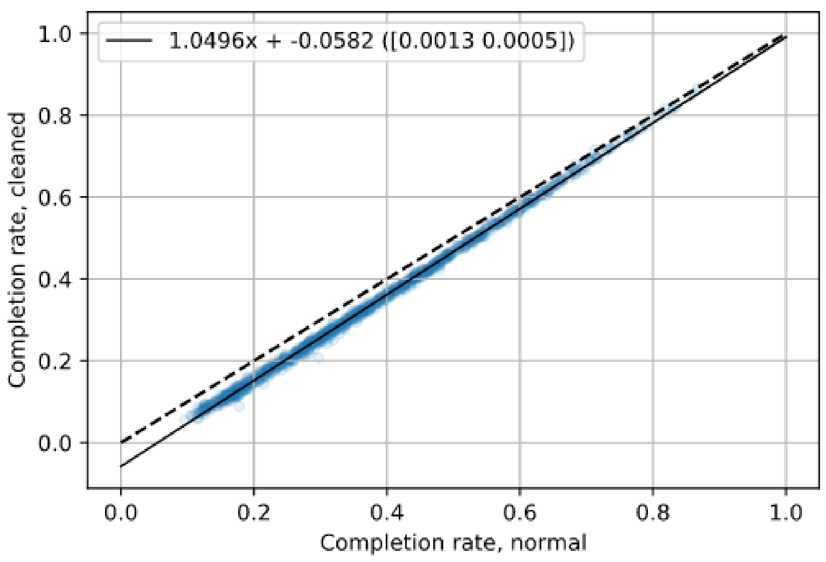


Figure 6.1: The pass rate (=completion rate) of levels when excluding players that used booster items (cleaned) compared to the pass rate of the whole player base (normal). The cleaned pass rate is consistently lower than the normal pass rate, but there is a strong correlation between the two.

the computed pass rate, or completion rate, of the filtered data compared to the unfiltered, normal data consistently exhibits a lower pass rate (see Fig. 6.1).

Some players hold the belief that using boosters is cheating, while others consider it a normal part of the game. Neither viewpoint is invalid, but excluding one or the other group leads to a difference in the data, which means that different results and conclusions can be drawn from each instance. Any sort of data filtering will, therefore, bias the results. In the case with pass rates shown in Fig 6.1, this bias can be adjusted by a simple linear correlation, but this may not be the case in other instances or use cases.

For future research, a recommendation is to provide a clear description of the filtering of the data and a discussion of its effect on the results. This will help researchers and industry transfer the learnings from one study to another and increase the impact of the work.

6.2.3 Improving the playtesting agent

The results in Chapter 4 show that training one playtesting agent to play all the levels competently is a challenging problem. The agent used in Chapter 5 is able to learn to play most levels competently, but it was necessary to limit the number of random seeds and only train it on a single level at a time. This section discusses possible ways to enable the training of a single agent that can play any new level competently.

Planning methods

In many similar works (e.g. Gudmundsson et al. (2018); Roohi et al. (2021); Mugrai et al. (2019); Holmgård et al. (2018)), a commonly employed method is MCTS. The complexity of puzzle games is comparable to board games like Chess and Go, in which planning and search-based methods work well and tend to outperform deep RL methods like PPO (e.g. Roohi et al. (2021); Mugrai et al. (2019)). One of the reasons that the dissertation focused on developing a playtesting agent based on deep RL, and more specifically PPO (see Chapter 4), is that search-based methods are not technically viable in the games by Tactile Games mainly due to not being able to save and load game states.

In order to overcome this limitation but still leverage the strength of planning approaches is to utilise *world models*. For example, Ha and Schmidhuber (2018) use a recurrent neural network (RNN) in combination with a variational autoencoder to predict the future state and allow taking the recent history into account, which can be helpful when working with partially observable systems. *MuZero* also learns a kind of world model that allows the method to perform tree-search in this learned world model. Both methods (using an RNN and a world model to perform tree-search) are possible ways to improve the current approach. Including an RNN in the network architecture may also allow the agent to be more consistent in its strategies (such as focusing on creating power pieces), and the world model can overcome the technical limitations that exclude any search-based methods.

Learning to play hard levels

In Chapter 5, it is noted that in order to ensure the agent can learn to finish hard levels, it is necessary to reduce the stochasticity of the environment by limiting the number of random seeds and using the same colours throughout the training. However, as discussed in Chapter 4, (re)training the agent on the level that is to be evaluated is undesirable since it reduces the correlation with player metrics and takes more time to provide a difficulty estimate of the level.

This is related to the field of *generalisability in reinforcement learning* and *curriculum learning* (Narvekar et al., 2020), and there are multiple possible options that can be explored, such as reward shaping or data augmentation. For the immediate next steps, we propose to develop a better curriculum. We already used a curriculum for training the agent described in Chapter 4, albeit an inefficient one – namely, a random sampling curriculum. Furthermore, since it is not technically possible to employ PCG methods (Justesen et al., 2018; Hald et al., 2020) or adjust the content efficiently in the games by Tactile Games, we have to rely on the available content.

One reason why a random curriculum is inefficient is that harder levels tend to require more moves or more advanced strategies to complete. By introducing too hard levels early on, the probability that the agent reaches the goal within the reset limit is low, which slows down learning by not letting the agent receive the reward signal in the end. A possible line of research could therefore be how to optimally select a curriculum that ensures the agent learns how to play hard levels and which method leads to the largest correlation with human data. An option is to use a curriculum that is guided by the measured human difficulty, where levels are sampled from a given range of difficulty depending on the agent's current performance. A more intelligent sampling could also be tested, such as a multi-armed bandit approach similar to Graves et al. (2017) or Mysore et al. (2019). The learned level sampling strategy could then be one that improves and maximises the correlation between the agent and human behaviour.

One aspect to keep in mind is that this kind of sequential learning in artificial neural networks is prone to catastrophic forgetting (Kirkpatrick et al., 2017), where the performance on previously learned tasks deteriorates as new tasks are learned. As a first step, implementing a number of common regularisation techniques in the network architecture (e.g. $L2$ weight decay and dropout Cobbe et al. (2019)) can help with this, in addition to simplifying the environment by implementing a permutation invariant layer to account for colour interchangeability (e.g. Tang and Ha (2021)) could be employed to address these issues.

Diversity in play styles and correlation with players

The reward function of an agent significantly affects what the agent learns, and while multiple agents with different reward functions are a possibility to increase playtesting coverage, the question still remains *what* reward function(s) is useful for a playtesting scenario.

One family of methods that has not been explored in this dissertation is *curiosity*-driven learning (Pathak et al., 2017). The core idea is that the agent receives an intrinsic reward (contrasted to an extrinsic reward from the environment) for taking actions that lead to unexpected game states, which in turn promotes exploration and emergent strategies. In our case, it may bias the agent towards learning strategies that clear the entire board since the following board state will be completely random. Crucially, though, we will not need to shape the reward function to create power pieces or utilise other heuristics, which we were generally unsuccessful with in previous work (Kristensen et al., 2020).

Another possible approach is to condition the applied policy on not just the state but also other factors. For example, *DIAYN* (Eysenbach et al., 2018) learns a policy that is conditioned on a latent variable, z, which represents distinct skills. A more direct approach for conditioning the policy is by adjusting the reward function, which could consist of multiple terms with different coefficients (similar to de Woillemont et al. (2021)). By providing these coefficients as a part of the state space and changing the coefficients during training, it is possible to condition the policy on these coefficients. Future work could look into which terms are efficient for learning (e.g. pieces cleared as a proxy for curiosity) and which configurations lead to the most player-like behaviour.

6.2.4 Using a playtesting agent in practice

Using a playtesting agent in practice is not just a question of whether it is possible to use it for estimating the difficulty of new levels. It also requires maintenance when, for example, the game gets updated, or the inevitable bugs appear. Given the complexity of the problem, this section will discuss some of the challenges and questions that need to be answered in future research to move the method closer to a "production ready" state.

Maintainability

A reasonable question is whether the playtesting agent that makes up half of this PhD is accessible for game developers to utilise and transfer to new games. To answer this question, we can consider aspects regarding *explainability* and *ease of use.*

A common thing that happened when discussing how the agent works with different stakeholders at Tactile Games was that they understood it as a tree-search-based method. While it is an innocent misunderstanding, it can materialise itself as wrong or inefficient code and time-consuming back-and-forth communications until the misunderstanding has been cleared up. Furthermore, a common question when checking how the agent played through a level was why it took a particular move and what its strategy was. Especially in business contexts, understanding *why* an algorithm does something is just as important as the performance itself. An interesting line of research, for this reason, is *explainable reinforcement learning*, which seeks to provide explanations for the agent's choices (for a survey, see Puiutta and Veith (2020)). As a first step, an idea is a post-hoc analysis of the agent's behaviour. Specifically, we can supplement the difficulty predictions with a heatmap of the actions over the course of playtesting a given level, which can provide the level designers with information about what the agent focuses on and, thus, its strategy.

The other aspect of maintainability is how easy it is to update the agent. This has been the topic for Kristensen and Burelli (2020) where we consider different strategies to make the learning more robust, but with more than 6,000 levels in some games and regular new mechanics and updates, a key focus is being able to add new information to the agent while still keeping its performance. There are two ways we imagine this can work in practice. One is by simply retraining the agent from scratch with the new information, and the key to this approach is to ensure that the learning utilises a good curriculum as described in Section 6.2.3. The other idea is to employ methods from the *transfer learning* domain (for a survey in RL, see Zhu et al. (2020)).

There are multiple ways we can approach transfer learning in practice. One way is by adding new channels and layers to the neural network while keeping the initial weights the same. The newly added weights can then be updated from subsequent training on the new levels or other curricula. Another way is to use the existing agent and train the new agent since any game updates are not expected to invalidate the old agent completely. This can be done through imitation learning approaches (e.g. *GAIL*) or by treating it as a supervised learning task where the goal is to have the new network output the same as the old one. The most efficient way can be explored in future research.

Open questions

During the development of the playtesting agent, there have been several decisions that may have had an impact on the performance of the agent but did not warrant deeper investigations. In the following part of the text, we ask whether these decisions indeed impact the agent and propose to investigate these aspects before using the playtesting agent in practice.

As described by Pardo et al. (2018), time limits in RL matter. It can affect what the agent learns and how it plays the level. In this work, we have proposed using an early resetting limit of 100 actions, but this is more than the typical level move limit. While the human move limit has been included as a part of the reward function and generally encourages the agent to finish as fast as possible, the question is whether this very high reset limit leads the agent to learn very different strategies than human players. Testing out different reset limits or even adjusting it during training can therefore be a way to answer this question.

During the training of an agent, especially with on-policy ones, it is common to use a stochastic policy to allow it to explore properly. On the other hand, during evaluation, it is common to switch to a greedy deterministic policy where the agent fully exploits its learning. However, this may not be ideal in our case. Since two actions may lead to the exact same outcome – e.g. if two basic pieces are stacked on top of each other, tapping either of them leads to the same outcome even though it is two distinct actions – the policy may assign equal importance to both actions. However, this also means that for larger clusters, the relative importance of each action corresponding to tapping on one of the pieces in the cluster is lower than the actions associated with smaller clusters. A greedy, deterministic policy may therefore become biased towards tapping on smaller clusters, which may, in turn, also make it rarer to create powerful combos. The question is, therefore, whether using a stochastic or a deterministic policy is better for evaluating the levels. A direct way to test this is to compare the performance between the two policies and measure the typical size of the tapped clusters and moves taken to complete the level.

One limitation of the agent is that it cannot use booster items. This partly simplifies the RL problem, and the level designers playtest and design all levels to be possible to complete without any such tools. However, boosters are an integral part of the game that some players are happy to use, so to increase the coverage of the agent, the question is how this can be incorporated into the playtesting method. Technically, this could be achieved by modifying the reward function where a penalty is given if the agent chooses to use boosters. The magnitude of the coefficient of this penalty term could also help mimic players with different "budgets" and could be an extension to the proposed experiment by using varying coefficients for each reward function term discussed in Section 6.2.3. The action space would also need to be appropriately modified by, for example, making it a multi-discrete action space that allows the agent to use the in-game boosters on any location of the game board. Some boosters also require to be activated before the level starts, but there are frequent events that add such boosters for free (e.g. "hot streaks"). A solution to that is to add these booster power pieces to the initial board setup and consider it as a part of the initial board state distribution. More complex boosters, such as unlimited lives, may lead the players to take more risks and sub-optimal play styles, but they are more difficult to include, and we do not expect it affects the coverage of the agent by not considering this aspect.

6.3 Summary

This dissertation has approached the topic of operationalising difficulty in puzzle games by considering challenges related to utilising historical player data and employing automated playtesting methods for modelling and operationalising difficulty. The work includes a statistical description of the number of moves spent on each level, introducing a framework for modelling the interaction between players and levels, a novel reinforcement learning implementation for automated playtesting and a novel approach for combining personalised predictions with playtesting agent data to predict the number of attempts players will spend on new content. Each of the contributions is based on data from the commercial puzzle game, *Lily's Garden*, and is evaluated on large samples of players, which solidifies the significance of the results. In addition to the proposed methods, multiple findings have come from this work. The negative binomial used to describe the move distribution provides a probabilistic explanation of the performance of both players and agents and improves the operationalisation of using the average number of attempts as a difficulty metric. Additionally, from the game-agnostic Factorisation Machine algorithm for personalised difficulty prediction, the results show that the learned latent descriptions of players and levels capture aspects of player skill and level variance. Lastly, the process of developing a reinforcement learning agent highlights the various challenges with both using reinforcement learning for puzzle game environments and correlating the agent performance with player behaviour. The methods in this dissertation overcome multiple of these challenges and provide clear recommendations on how difficulty can be operationalised for both existing and new content and possible directions for future work.

Bibliography

Ernest Adams. *Fundamentals of game design.* Pearson Education, 2014.

Justin T. Alexander, John Sear, and Andreas Oikonomou. An investigation of the effects of game difficulty on player enjoyment. *Entertainment Computing*, 4(1):53–62, February 2013. ISSN 18759521. doi: 10.1016/j.entcom.2012.09.001. URL `https://linkinghub.elsevier.com/retrieve/pii/S1875952112000134`.

Brendan Z. Allison and John Polich. Workload assessment of computer gaming using a single-stimulus event-related potential paradigm. *Biological Psychology*, 77(3):277–283, 2008. ISSN 0301-0511. doi: https://doi.org/10.1016/j.biopsycho.2007.10.014. URL `https://www.sciencedirect.com/science/article/pii/S030105110700186X`.

Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2): 251–276, 1998.

Maria-Virginia Aponte, Guillaume Levieux, and Stéphane Natkin. Difficulty in videogames: an experimental validation of a formal definition. In *Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology*, pages 1–8, 2011a.

Maria-Virginia Aponte, Guillaume Levieux, and Stephane Natkin. Measuring the level of difficulty in single player video games. *Entertainment Computing*, 2(4):205–213, 2011b. ISSN 1875-9521. doi: https://doi.org/10.1016/j.entcom.2011.04.001. URL `https://www.sciencedirect.com/science/article/pii/S1875952111000231`. Special Section: International Conference on Entertainment Computing and Special Section: Entertainment Interfaces.

Sinan Ariyurek, Aysu Betin-Can, and Elif Surer. Automated video game testing using synthetic and humanlike agents. *IEEE Transactions on Games*, 13(1):50–67, 2019.

Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Zhaohan Daniel Guo, and Charles Blundell. Agent57: Outperforming the atari human benchmark. In *International Conference on Machine Learning*, pages 507–517. PMLR, 2020.

Hendrik Baier, Adam Sattaur, Edward J Powley, Sam Devlin, Jeff Rollason, and Peter I Cowling. Emulating human play in a leading mobile card game. *IEEE Transactions on Games*, 11(4): 386–395, 2018.

Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autocurricula. *arXiv preprint arXiv:1909.07528*, 2019.

M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47: 253–279, jun 2013.

Joakim Bergdahl, Camilo Gordillo, Konrad Tollmar, and Linus Gisslén. Augmenting automated game testing with deep reinforcement learning. In *2020 IEEE Conference on Games (CoG)*, pages 600–603. IEEE, 2020.

Peter Borm and Ben van der Genugten. On a relative measure of skill for games with chance elements. *Top*, 9(1):91–114, 2001.

Barbaros Bostan and Sertaç Öğüt. Game challenges and difficulty levels: lessons learned from rpgs. In *International simulation and gaming association conference*, pages 1–11, 2009.

Jeanne H Brockmyer, Christine M Fox, Kathleen A Curtiss, Evan McBroom, Kimberly M Burkhart, and Jacquelyn N Pidruzny. The Development of the Game Engagement Questionnaire: A Measure of Engagement in Video Game-Playing. *Journal of Experimental Social Psychology*, 45(4):624–634, 2009.

Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A Survey of Monte Carlo Tree Search Methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, 3 2012. ISSN 1943-068X. doi: 10.1109/TCIAIG.2012.2186810. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6145622%0Apapers3://publication/doi/10.1109/TCIAIG.2012.2186810http://ieeexplore.ieee.org/document/6145622/`.

Sara Bunian, Alessandro Canossa, Randy Colvin, and Magy Seif El-Nasr. Modeling individual differences in game behavior using hmm. In *Thirteenth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2017.

Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A Efros. Large-scale study of curiosity-driven learning. *arXiv preprint arXiv:1808.04355*, 2018.

Hao Cen, Kenneth Koedinger, and Brian Junker. Learning factors analysis–a general method for cognitive model evaluation and improvement. In *International conference on intelligent tutoring systems*, pages 164–175. Springer, 2006.

Guillaume Chaslot, Sander Bakkes, Istvan Szita, and Pieter Spronck. Monte-carlo tree search: A new framework for game ai. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 4(1):216–217, Sep. 2021. URL `https://ojs.aaai.org/index.php/AIIDE/article/view/18700`.

Petros Christodoulou. Soft Actor-Critic for Discrete Action Settings. *arXiv e-prints*, pages 1–7, 10 2019. URL `http://arxiv.org/abs/1910.07207`.

Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *International Conference on Machine Learning*, pages 1282–1289. PMLR, 2019.

Tom Cole, Paul Cairns, and Marco Gillies. Emotional and functional challenge in core and avant-garde games. In *Proceedings of the 2015 annual symposium on computer-human interaction in play*, pages 121–126, 2015.

Anna Cox, Paul Cairns, Pari Shah, and Michael Carroll. Not doing but thinking: The role of challenge in the gaming experience. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, page 79–88, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450310154. doi: 10.1145/2207676.2207689. URL `https://doi.org/10.1145/2207676.2207689`.

Mihaly Csikszentmihalyi. *Flow: The psychology of optimal experience*, volume 1990. Harper & Row New York, 1990.

Pierre Le Pelletier de Woillemont, Rémi Labory, and Vincent Corruble. Configurable agent with reward as input: A play-style continuum generation. In *2021 IEEE Conference on Games (CoG)*, pages 1–8. IEEE, 2021.

Alena Denisova, Christian Guckelsberger, and David Zendle. Challenge in Digital Games: Towards Developing a Measurement Tool. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, pages 2511–2519, Denver Colorado USA, May 2017. ACM. ISBN 978-1-4503-4656-6. doi: 10.1145/3027063.3053209. URL `https://dl.acm.org/doi/10.1145/3027063.3053209`.

Alena Denisova, Paul Cairns, Christian Guckelsberger, and David Zendle. Measuring perceived challenge in digital games: Development & validation of the challenge originating from recent gameplay interaction scale (CORGIS). *International Journal of Human-Computer Studies*, 137:102383, May 2020. ISSN 10715819. doi: 10.1016/j.ijhcs.2019.102383. URL `https://linkinghub.elsevier.com/retrieve/pii/S1071581919301491`.

Mihai Sorin Dobre and Alex Lascarides. Online learning and mining human play in complex games. In *2015 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 60–67. IEEE, 2015.

Tobias Drey, Fabian Fischbach, Pascal Jansen, Julian Frommel, Michael Rietzler, and Enrico Rukzio. To Be or Not to Be Stuck, or Is It a Continuum?: A Systematic Literature Review on the Concept of Being Stuck in Games. *Proceedings of the ACM on Human-Computer Interaction*, 5(CHI PLAY):1–35, October 2021. ISSN 2573-0142. doi: 10.1145/3474656. URL `https://dl.acm.org/doi/10.1145/3474656`.

Dagmara Dziedzic and Wojciech Włodarczyk. Approaches to measuring the difficulty of games in dynamic difficulty adjustment systems. *International Journal of Human–Computer Interaction*, 34(8):707–715, 2018.

Tomáš Effenberger, Jaroslav Čechák, and Radek Pelánek. Measuring difficulty of introductory programming tasks. In *Proceedings of the Sixth (2019) ACM Conference on Learning@ Scale*, pages 1–4, 2019.

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation matters in deep rl: A case study on ppo and trpo. In *International conference on learning representations*, 2019.

Laura Ermi and Frans Mäyrä. Analyzing immersion. *Worlds in play: International perspectives on digital games research*, 21:37, 2007.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

Francisco Gallego-Durán, Rafael Molina-Carmona, and Faraón Llorens-Largo. Estimating the difficulty of a learning activity from the training cost for a machine learning algorithm. In *Proceedings of the Sixth International Conference on Technological Ecosystems for Enhancing Multiculturality*, pages 654–659, 2018.

Pablo García-Sánchez, Alberto Tonda, Antonio M. Mora, Giovanni Squillero, and Juan Julián Merelo. Automated playtesting in collectible card games using evolutionary algorithms: A case study in hearthstone. *Knowledge-Based Systems*, 153:133–146, 2018. ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2018.04.030. URL `https://www.sciencedirect.com/science/article/pii/S0950705118301953`.

Jason Gauci, Kenneth O Stanley, et al. A case study on the critical role of geometric regularity in machine learning. In *AAAI*, pages 628–633, 2008.

Miguel Gonzalez-Duque, Rasmus Berg Palm, and Sebastian Risi. Fast game content adaptation through bayesian-based player modelling. In *2021 IEEE Conference on Games (CoG)*, pages 01–08, 2021. doi: 10.1109/CoG52621.2021.9619018.

Miguel González-Duque, Rasmus Berg Palm, David Ha, and Sebastian Risi. Finding game levels with the right difficulty in a few trials through intelligent trial-and-error. In *2020 IEEE Conference on Games (CoG)*, pages 503–510, 2020. doi: 10.1109/CoG47356.2020.9231548.

Camilo Gordillo, Joakim Bergdahl, Konrad Tollmar, and Linus Gisslén. Improving playtesting coverage via curiosity driven reinforcement learning agents. *arXiv preprint arXiv:2103.13798*, 2021.

Alex Graves, Marc G. Bellemare, Jacob Menick, Rémi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. In *34th International Conference on Machine Learning, ICML 2017*, volume 3, pages 2120–2129. International Machine Learning Society (IMLS), 2017. ISBN 9781510855144.

Stefan Freyr Gudmundsson, Philipp Eisen, Erik Poromaa, Alex Nodet, Sami Purmonen, Bartlomiej Kozakowski, Richard Meurling, and Lele Cao. Human-like playtesting with deep learning. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–8. IEEE, 2018.

Cristina Guerrero-Romero, Simon M Lucas, and Diego Perez-Liebana. Using a team of general ai algorithms to assist game design and testing. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–8. IEEE, 2018.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

A. Hald, J.S. Hansen, J. Kristensen, and P. Burelli. Procedural Content Generation of Puzzle Games using Conditional Generative Adversarial Networks. In *ACM International Conference Proceeding Series*, 2020. ISBN 9781450388078. doi: 10.1145/3402942.3409601.

Perttu Hämäläinen, Amin Babadi, Xiaoxiao Ma, and Jaakko Lehtinen. PPO-CMA: Proximal Policy Optimization with Covariance Matrix Adaptation. *arXiv e-prints*, 10 2018. URL `http://arxiv.org/abs/1810.02541`.

Nikolaus Hansen. The cma evolution strategy: a comparing review. *Towards a new evolutionary computation*, pages 75–102, 2006.

Erik Harpstead and Vincent Aleven. Using empirical learning curve analysis to inform design in an educational game. In *Proceedings of the 2015 Annual Symposium on Computer-Human Interaction in Play*, CHI PLAY '15, page 197–207, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450334662. doi: 10.1145/2793107.2793128. URL `https://doi.org/10.1145/2793107.2793128`.

Matthew Hausknecht, Joel Lehman, Risto Miikkulainen, and Peter Stone. A neuroevolution approach to general atari game playing. *IEEE Transactions on Computational Intelligence and AI in Games*, 6(4):355–366, 2014.

Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, pages 3215–3222, 2018.

Philip Hingston. A turing test for computer game bots. *IEEE Transactions on Computational Intelligence and AI in Games*, 1(3):169–186, 2009.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in Neural Information Processing Systems*, pages 4572–4580, 2016. ISSN 10495258.

Christoffer Holmgård, Michael Cerny Green, Antonios Liapis, and Julian Togelius. Automated playtesting with procedural personas through mcts with evolved heuristics. *IEEE Transactions on Games*, 11(4):352–362, 2018.

Britton Horn, Josh Aaron Miller, Gillian Smith, and Seth Cooper. A monte carlo approach to skill-based automated playtesting. In *Fourteenth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2018.

Dayana Hristova. Dynamic difficulty adjustment (dda) in first person shooter (fps) games, 2017.

Shengyi Huang and Santiago Ontañón. A closer look at invalid action masking in policy gradient algorithms. *arXiv preprint arXiv:2006.14171*, 2020.

Shengyi Huang, Rousslan Fernand Julien Dossa, Antonin Raffin, Anssi Kanervisto, and Weixun Wang. The 37 implementation details of proximal policy optimization. In *ICLR Blog Track*, 2022. URL `https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/`. https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/.

Robin Hunicke. The case for dynamic difficulty adjustment in games. In *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*, pages 429–433, 2005.

Andrew Ilyas, Logan Engstrom, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Are deep policy gradient algorithms truly policy gradient algorithms? *CoRR*, abs/1811.02553, 2018. URL `http://arxiv.org/abs/1811.02553`.

Arthur Juliani, Vincent-Pierre Berges, Esh Vckay, Yuan Gao, Hunter Henry, Marwan Mattar, and Danny Lange. Unity: A general platform for intelligent agents. *CoRR*, abs/1809.02627, 2018. URL `http://arxiv.org/abs/1809.02627`.

Niels Justesen, Ruben Rodriguez Torrado, Philip Bontrager, Ahmed Khalifa, Julian Togelius, and Sebastian Risi. Illuminating generalization in deep reinforcement learning through procedural level generation. *arXiv preprint arXiv:1806.10729*, 2018.

Niels Justesen, Philip Bontrager, Julian Togelius, and Sebastian Risi. Deep learning for video game playing. *IEEE Transactions on Games*, 12(1):1–20, 2019.

Jesper Juul. Fear of failing? the many meanings of difficulty in video games. *The video game theory reader*, 2(237-252), 2009.

Jesper Juul. *Half-real: Video games between real rules and fictional worlds*. MIT press, 2011.

Misaki Kaidan, Tomohiro Harada, Chun Yin Chu, and Ruck Thawonmas. Procedural generation of angry birds levels with adjustable difficulty. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 1311–1316. IEEE, 2016.

Ildar Kamaldinov and Ilya Makarov. Deep reinforcement learning in match-3 game. In *2019 IEEE conference on games (CoG)*, pages 1–4. IEEE, 2019.

Oleksandra Keehl and Adam M Smith. Monster carlo: an mcts-based framework for machine playtesting unity games. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–8. IEEE, 2018.

Ahmed Khalifa, Aaron Isaksen, Julian Togelius, and Andy Nealen. Modifying mcts for human-like general video game playing. *IJCAI International Joint Conference on Artificial Intelligence*, 2016-January:2514–2520, 2016. ISSN 1045-0823. 25th International Joint Conference on Artificial Intelligence, IJCAI 2016 ; Conference date: 09-07-2016 Through 15-07-2016.

Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. A survey of generalisation in deep reinforcement learning. *arXiv preprint arXiv:2111.09794*, 2021.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences of the United States of America*, 114(13):3521–3526, 2017. ISSN 10916490. doi: 10.1073/pnas.1611835114.

Julia Kneer, Malte Elson, and Florian Knapp. Fight fire with rainbows: The effects of displayed violence, difficulty, and performance in digital games on affect, aggression, and physiological arousal. *Computers in Human Behavior*, 54:142–148, 2016. ISSN 0747-5632. doi: https://doi.org/10.1016/j.chb.2015.07.034. URL `https://www.sciencedirect.com/science/article/pii/S0747563215300455`.

Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.

Jeppe Kristensen and Paolo Burelli. Personalized game difficulty prediction using factorization machines. *Transaction on Games*, 2022, 2022.

Jeppe Kristensen, Christian Guckelsberger, Paolo Burelli, and Perttu Hämäläinen. Personalized game difficulty prediction using factorization machines. *User Interface Software and Technology*, 2022, 2022.

Jeppe Theiss Kristensen and Paolo Burelli. Combining Sequential and Aggregated Data for Churn Prediction in Casual Freemium Games. In *2019 IEEE Conference on Games (CoG)*, pages 1–8, August 2019. doi: 10.1109/CIG.2019.8848106. ISSN: 2325-4289.

Jeppe Theiss Kristensen and Paolo Burelli. Strategies for using proximal policy optimization in mobile puzzle games. In *International conference on the foundations of digital games*, pages 1–10, 2020.

Jeppe Theiss Kristensen, Arturo Valdivia, and Paolo Burelli. Estimating player completion rate in mobile puzzle games using reinforcement learning. In *2020 IEEE Conference on Games (CoG)*, pages 636–639, 2020. doi: 10.1109/CoG47356.2020.9231581.

Jeppe Theiss Kristensen, Arturo Valdivia, and Paolo Burelli. Statistical modelling of level difficulty in puzzle games. In *2021 IEEE Conference on Games (CoG)*, pages 1–8. IEEE, 2021.

Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *Advances in neural information processing systems*, 33:19884–19895, 2020.

Chang-Shing Lee, Mei-Hui Wang, Guillaume Chaslot, Jean-Baptiste Hoock, Arpad Rimmel, Olivier Teytaud, Shang-Rong Tsai, Shun-Chin Hsu, and Tzung-Pei Hong. The computational intelligence of mogo revealed in taiwan's computer go tournaments. *IEEE Transactions on Computational Intelligence and AI in games*, 1(1):73–89, 2009.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Conor Linehan, George Bellord, Ben Kirman, Zachary H. Morford, and Bryan Roche. Learning curves: analysing pace and challenge in four successful puzzle games. In *Proceedings of the first ACM SIGCHI annual symposium on Computer-human interaction in play*, pages 181–190, Toronto Ontario Canada, October 2014. ACM. ISBN 978-1-4503-3014-5. doi: 10.1145/2658537.2658695. URL `https://dl.acm.org/doi/10.1145/2658537.2658695`.

Nir Lipovetzky, Miquel Ramirez, and Hector Geffner. Classical planning with simulators: Results on the atari video games. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

Tommy Liu, Jochen Renz, Peng Zhang, and Matthew Stephenson. Using restart heuristics to improve agent performance in angry birds. In *2019 IEEE Conference on Games (CoG)*, pages 1–8, 2019. doi: 10.1109/CIG.2019.8848039.

J Derek Lomas, Kenneth Koedinger, Nirmal Patel, Sharan Shodhan, Nikhil Poonwala, and Jodi L Forlizzi. Is difficulty overrated? the effects of choice, novelty and suspense on intrinsic motivation in educational games. In *Proceedings of the 2017 CHI conference on human factors in computing systems*, pages 1028–1039, 2017.

Diana Lora, Antonio A Sánchez-Ruiz, Pedro A González-Calero, and Marco A Gómez-Martín. Dynamic difficulty adjustment in tetris. In *The Twenty-Ninth International Flairs Conference*, 2016.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2 2015. ISSN 0028-0836. doi: 10.1038/nature14236. URL `http://dx.doi.org/10.1038/nature14236http://www.nature.com/articles/nature14236`.

Volodymyr Mnih, Adria Puigdomenech Badia, Lehdi Mirza, Alex Graves, Tim Harley, Timothy P. Lillicrap, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. *33rd International Conference on Machine Learning, ICML 2016*, 4:2850–2869, 2016.

Antonio Miguel Mora, Francisco Aisa, Pablo García-Sánchez, Pedro Ángel Castillo, and Juan Julián Merelo. Modelling a human-like bot in a first person shooter game. *International Journal of Creative Interfaces and Computer Graphics (IJCICG)*, 6(1):21–37, 2015.

Fausto Mourato, Fernando Birra, and Manuel Próspero dos Santos. Difficulty in action based challenges: success prediction, players' strategies and profiling. In *Proceedings of the 11th Conference on Advances in Computer Entertainment Technology*, pages 1–10, 2014.

Luvneesh Mugrai, Fernando Silva, Christoffer Holmgard, and Julian Togelius. Automated playtesting of matching tile games. *IEEE Conference on Computatonal Intelligence and Games, CIG*, 2019-Augus, 2019. ISSN 23254289. doi: 10.1109/CIG.2019.8848057.

Siddharth Mysore, Robert Platt, and Kate Saenko. Reward-guided curriculum for robust reinforcement learning. *preprint*, 2019.

Aniket Nagle, Peter Wolf, and Robert Riener. Towards a system of customized video game mechanics based on player personality: Relating the Big Five personality traits with difficulty adaptation in a first-person shooter game. *Entertainment Computing*, 13:10–24, March 2016. ISSN 18759521. doi: 10.1016/j.entcom.2016.01.002. URL `https://linkinghub.elsevier.com/retrieve/pii/S1875952116000045`.

Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey. *arXiv preprint arXiv:2003.04960*, 2020.

Juan Ortega, Noor Shaker, Julian Togelius, and Georgios N. Yannakakis. Imitating human playing styles in Super Mario Bros. *Entertainment Computing*, 4(2):93–104, 4 2013. ISSN 18759521. doi: 10.1016/j.entcom.2012.10.001.

Fabio Pardo, Arash Tavakoli, Vitaly Levdik, and Petar Kormushev. Time limits in reinforcement learning. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4045–4054. PMLR, 10–15 Jul 2018. URL `https://proceedings.mlr.press/v80/pardo18a.html`.

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. *34th International Conference on Machine Learning, ICML 2017*, 6:4261–4270, 2017.

Radek Pelánek. Difficulty rating of sudoku puzzles by a computational model. In *Twenty-Fourth International FLAIRS Conference*, 2011.

Johannes Pfau, Jan David Smeddinck, and Rainer Malaka. Enemy within: Long-term motivation effects of deep player behavior models for dynamic difficulty adjustment. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–10, 2020.

Erik Ragnar Poromaa. Crushing candy crush: predicting human success rate in a mobile game using monte-carlo tree search. Master's thesis, KTH, School of Computer Science and Communication (CSC), 2017.

Erika Puiutta and Eric Veith. Explainable reinforcement learning: A survey. In *International cross-domain conference for machine learning and knowledge extraction*, pages 77–95. Springer, 2020.

Megan Pusey, Kok Wai Wong, and Natasha Anne Rappa. The Puzzle Challenge Analysis Tool. A Tool for Analysing the Cognitive Challenge Level of Puzzles in Video Games. *Proceedings of the ACM on Human-Computer Interaction*, 5(CHI PLAY):1–27, October 2021. ISSN 2573-0142. doi: 10.1145/3474703. URL `https://dl.acm.org/doi/10.1145/3474703`.

Steffen Rendle. Factorization Machines with libFM. *ACM Transactions on Intelligent Systems and Technology*, 3(3):1–22, May 2012. ISSN 2157-6904, 2157-6912. doi: 10.1145/2168752.2168771. URL `https://dl.acm.org/doi/10.1145/2168752.2168771`.

Sebastian Risi and Julian Togelius. Increasing generality in machine learning through procedural content generation. *Nature Machine Intelligence*, 2(8):428–436, 8 2020. ISSN 2522-5839. doi: 10.1038/s42256-020-0208-z. URL `http://www.nature.com/articles/s42256-020-0208-zhttp://arxiv.org/abs/1911.13071`.

Sonia Roccas, Lilach Sagiv, Shalom H. Schwartz, and Ariel Knafo. The big five personality factors and personal values. *Personality and Social Psychology Bulletin*, 28(6):789–801, 2002. doi: 10.1177/0146167202289008. URL `https://doi.org/10.1177/0146167202289008`.

Shaghayegh Roohi, Jari Takatalo, Christian Guckelsberger, and Perttu Hämäläinen. Review of intrinsic motivation in simulation-based game testing. In *Proceedings of the 2018 chi conference on human factors in computing systems*, pages 1–13, 2018.

Shaghayegh Roohi, Christian Guckelsberger, Asko Relas, Henri Heiskanen, Jari Takatalo, and Perttu Hämäläinen. Predicting game difficulty and engagement using ai players. *Proceedings of the ACM on Human-Computer Interaction*, 5(CHI PLAY):1–17, 2021.

Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

Jesse Schell. *The Art of Game Design: A Book of Lenses.* A K Peters/CRC Press, London, 2019. ISBN 978-1138632059.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-Dimensional Continuous Control Using Generalized Advantage Estimation. *arXiv e-prints*, art. arXiv:1506.02438, June 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv e-prints*, pages 1–12, 7 2017. URL `http://arxiv.org/abs/1707.06347`.

Noor Shaker, Georgios Yannakakis, and Julian Togelius. Towards automatic personalized content generation for platform games. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 6(1):63–68, Oct. 2010. URL `https://ojs.aaai.org/index.php/AIIDE/article/view/12399`.

Yuchul Shin, Jaewon Kim, Kyohoon Jin, and Young Bin Kim. Playtesting in match 3 game using strategic plays via reinforcement learning. *IEEE Access*, 8:51593–51600, 2020.

Mirna Paula Silva, Victor do Nascimento Silva, and Luiz Chaimowicz. Dynamic Difficulty Adjustment through an Adaptive AI. In *2015 14th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, pages 173–182, November 2015. doi: 10.1109/SBGames.2015.16. ISSN: 2159-6662.

Samantha Stahlke, Atiya Nova, and Pejman Mirza-Babaei. Artificial players in the design process: Developing an automated testing tool for game level and world design. In *Proceedings of the Annual Symposium on Computer-Human Interaction in Play*, pages 267–280, 2020.

Kenneth O Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.

Matthew Stephenson and Jochen Renz. Agent-based adaptive level generation for dynamic difficulty adjustment in angry birds. *arXiv preprint arXiv:1902.02518*, 2019.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, Cambridge, MA, 2018.

Yujin Tang and David Ha. The sensory neuron as a transformer: Permutation-invariant neural networks for reinforcement learning. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021. URL `https://openreview.net/forum?id=wtLW-Amuds`. `https://attentionneuron.github.io`.

Bulent Tastan and Gita Sukthankar. Learning policies for first person shooter games using inverse reinforcement learning. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 7(1):85–90, Oct. 2011. URL `https://ojs.aaai.org/index.php/AIIDE/article/view/12430`.

Julian Togelius, Sergey Karakovskiy, and Robin Baumgarten. The 2009 mario ai competition. In *IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE, 2010.

Marc Van Kreveld, Maarten Löffler, and Paul Mutser. Automated puzzle difficulty estimation. In *2015 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 415–422. IEEE, 2015.

Rodrigo Vicencio-Moreira, Regan L Mandryk, and Carl Gutwin. Now you can compete with anyone: Balancing players of different skill levels in a first-person shooter game. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 2255–2264, 2015.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

Vanessa Volz, Jacob Schrum, Jialin Liu, Simon M Lucas, Adam Smith, and Sebastian Risi. Evolving mario levels in the latent space of a deep convolutional generative adversarial network. In *Proceedings of the genetic and evolutionary computation conference*, pages 221–228, 2018.

Peter Vorderer, Tilo Hartmann, and Christoph Klimmt. Explaining the enjoyment of playing video games: The role of competition. In *Proceedings of the Second International Conference on Entertainment Computing*, ICEC '03, page 1–9, USA, 2003. Carnegie Mellon University.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003. PMLR, 2016.

Rina R Wehbe, Elisa D Mekler, Mike Schaekermann, Edward Lank, and Lennart E Nacke. Testing incremental difficulty design in platformer games. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 5109–5113, 2017.

Lilian Weng. Policy gradient algorithms, Apr 2018. URL `https://lilianweng.github.io/posts/2018-04-08-policy-gradient/`.

Daniel Wheat, Martin Masek, Chiou Peng Lam, and Philip Hingston. Modeling perceived difficulty in game levels. In *Proceedings of the Australasian Computer Science Week Multiconference*, pages 1–8, 2016.

Su Xue, Meng Wu, John Kolen, Navid Aghdaie, and Kazi A. Zaman. Dynamic Difficulty Adjustment for Maximized Engagement in Digital Games. In *Proceedings of the 26th International Conference on World Wide Web Companion - WWW '17 Companion*, pages 465–471, Perth, Australia, 2017. ACM Press. ISBN 978-1-4503-4914-7. doi: 10.1145/3041021.3054170. URL `http://dl.acm.org/citation.cfm?doid=3041021.3054170`.

Georgios N. Yannakakis and Julian Togelius. Experience-driven procedural content generation. *IEEE Transactions on Affective Computing*, 2(3):147–161, 2011. doi: 10.1109/T-AFFC.2011.6.

Georgios N. Yannakakis and Julian Togelius. *Artificial Intelligence and Games*. Springer International Publishing, Cham, 2018. ISBN 978-3-319-63518-7. doi: 10.1007/978-3-319-63519-4. URL `http://link.springer.com/10.1007/978-3-319-63519-4`.

Yunqi Zhao, Igor Borovikov, Fernando de Mesentier Silva, Ahmad Beirami, Jason Rupert, Caedmon Somers, Jesse Harder, John Kolen, Jervis Pinto, Reza Pourabolghasem, et al. Winning is not everything: Enhancing game development with intelligent agents. *IEEE Transactions on Games*, 12(2):199–212, 2020.

Zhuangdi Zhu, Kaixiang Lin, and Jiayu Zhou. Transfer learning in deep reinforcement learning: A survey. *arXiv preprint arXiv:2009.07888*, 2020.

Mohammad Zohaib. Dynamic difficulty adjustment (dda) in computer games: A review. *Advances in Human-Computer Interaction*, 2018, 2018.

Alexander Zook, Stephen Lee-Urban, Michael R. Drinkwater, and Mark O. Riedl. Skill-based Mission Generation: A Data-driven Temporal Player Modeling Approach. In *Proceedings of the The third workshop on Procedural Content Generation in Games - PCG'12*, pages 1–8, Raleigh, NC, USA, 2012. ACM Press. ISBN 978-1-4503-1447-3. doi: 10.1145/2538528.2538534. URL `http://dl.acm.org/citation.cfm?doid=2538528.2538534`.

Alexander E Zook and Mark O Riedl. A temporal data-driven player model for dynamic difficulty adjustment. In *Eighth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2012.