Dungeons & Replicants: Automated Game Balancing via Deep Player Behavior Modeling

Johannes Pfau
Digital Media Lab
University of Bremen
Bremen, Germany
jpfau@tzi.de

Antonios Liapis
Institute of Digital Games
University of Malta
Msida, Malta
antonios.liapis@um.edu.mt

Georg Volkmar
Digital Media Lab
University of Bremen
Bremen, Germany
gvolkmar@tzi.de

Georgios N. Yannakakis
Institute of Digital Games
University of Malta
Msida, Malta
georgios.yannakakis@um.edu.mt

Rainer Malaka
Digital Media Lab
University of Bremen
Bremen, Germany
malaka@tzi.de

***Abstract*—Balancing the options available to players in a way that ensures rich variety and viability is a vital factor for the success of any video game, and particularly competitive multiplayer games. Traditionally, this balancing act requires extensive periods of expert analysis, play testing and debates. While automated gameplay is able to predict outcomes of parameter changes, current approaches mainly rely on heuristic or optimal strategies to generate agent behavior. In this paper, we demonstrate the use of *deep player behavior models* to represent a player population ($n = 213$) of the massively multiplayer online role-playing game *Aion*, which are used, in turn, to generate individual agent behaviors. Results demonstrate significant balance differences in opposing enemy encounters and show how these can be regulated. Moreover, the analytic methods proposed are applied to identify the balance relationships between classes when fighting against each other, reflecting the original developers' design.**

***Index Terms*—Automated game testing, balancing, deep learning, generative player modeling, imitation learning, video games**

I. Introduction

Due to its steady growth in popularity and accessibility, the video game industry has evolved to a multi-billion dollar branch that surpassed all other entertainment industry sectors including TV, cinema and music[1]. Along with this development, player demands for content and mechanics are ramping up to extents that even large companies struggle to manage [1]. Next to core content production, the majority of computational and labour effort is put on the detection of gameplay and experience bugs (e.g., 80% of the 50 most popular games on the major distribution platform *Steam*[2] require critical updates after launch [2]). While automated routines for the detection and reporting of critical errors and solvability become more popular in the industry [3], [4], balancing remains one of the most difficult and time-consuming phases of the game design process. The availability of versatile in-game units, character classes, factions or roles between which players are able to choose from has become indispensable for many successful titles, yet the balancing of these appears to open up an incessant effort. Even prominent titles of competitive online games that launched years ago still undergo persistent balance patches (e.g. StarCraft II [5], Overwatch [6] or Guild Wars 2 [7]). Following the definition of Sirlin [8], a game is *"balanced if a reasonably large number of options available to the player are viable"* (where viability sets the requirement of having many meaningful choices throughout a game), while *"players of equal skill should have an equal chance at winning"*. Together with frequently desired asymmetrical configuration possibilities of these options, this inherently leads to combinatorial explosions, which can become hazardous for the enjoyability of the game and the satisfaction of its players. Even worse, Hullett et al. highlight that balancing issues most of the time *"only become apparent after many months of play"* [9]. Compared to straightforward fixable bugs, glitches and solvability aspects, the trouble with balancing issues is that they do not only appear during the launch of a newly published game. Instead, balancing is an ongoing, repetitive task that is heavily influenced by the perceptions of the player community: *"after each patch, often the discussion begins again, factoring in new balancing or abilities for each class"* [10]. In the game industry, balancing is most often approached through long-term expert analysis, excessive human play-testing, and persistent debates with the community. Meanwhile, recent applied machine learning techniques have become very successful in outperforming human capabilities of playing, e.g. in Atari games [11], classical board games such as chess, shōgi and Go [12], [13] or real-time strategy games (RTS) such as StarCraft II [14]. While computer-controlled agents employing these approaches might also be suitable for automated game testing, their utility for automated balancing is arguably limited, given that optimal or super-human proficiency is not representative for the population of human players the game should be tailored for [15].

In this paper, we apply *Deep Player Behavior Modeling* (DPBM) [16] to automated game balancing. Within DPBM, individual decision making from game states is mapped to a preference distribution of actions via machine learning, approximating the replication of individual players. In contrast

This work was funded by the German Research Foundation (DFG) as part of Collaborative Research Center (SFB) 1320 EASE - Everyday Activity Science and Engineering, University of Bremen (http://www.easecrc.org/), subproject H2.

[1]https://newzoo.com/insights/trend-reports/newzoo-global-games-market-report-2019-light-version/

[2]https://store.steampowered.com/

978-1-7281-4533-4/20/$31.00 ©2020 IEEE

to optimal or generalized models, the DPBM approach allows for the consideration of many (potentially viable) playing styles that players can employ instead of reducing it to a global decision making module. In previous work, DPBM showed to be successful in generating agents capable of offering challenges on the same proficiency level [17] and convinced other players that they replicated individual behavior believably [18].

In this work, we used a dataset of the popular massively multiplayer online role-playing game (MMORPG) *Aion* [19] consisting of atomic decision making that was recorded throughout 6 months and 213 players in one-versus-one situations [17]. From this, we generated DPBM-driven agents for all players and benchmarked their proficiency against heuristic NPCs in a two-dimensional study setup that manipulated the offensive and defensive capabilities of the latter. While that study gave initial insights about the basic versatility of classes in player versus environment (PvE) settings, a subsequent investigation examined how all player replicas playing against each other in a player versus player (PvP) situation. For the empirical assessment of the resulting proficiencies, we utilized a metric that approximates the quality of single benchmark performances in terms of effectiveness and efficiency. Evaluating the capabilities for automated game balancing and individual proficiency estimation, we aim to answer the following research questions:

- *Can imbalances between in-game classes be detected through generative player modeling with respect to PvE and PvP?*
- *Can generative player modeling elevate automated game testing to turn design specifications into optimized parameter constellations?*

We hypothesize that agents that are representative of individual players' decision making are able to detect differences in performance between classes and resemble the population closer compared to generalized or random agents. Under these conditions, DPBM should provide a viable technique to map behavioral patterns to proficiency scores and to inform automated game balancing empirically. This work contributes to games user research and game development in academia and industry by introducing a novel technique capable of enhancing game testing processes with the potential of reducing the associated effort. In addition, a proficiency metric is constructed and presented that allows for the comparison of benchmark results. Effectively, this indicates the added value of assessing the replicated player population against generalized or random models.

II. Related Work

The implementation of automatic simulations of video game play has become a viable and efficient alternative or improvement to tedious and non-exhaustive human testing for the purpose of finding critical errors, solvability investigations or parameter tuning. The majority of scientific approaches focuses on detecting logical bugs or game crashes, such as Radomski et al. [20] or Varvaressos et al. [21] who identified violations of manually defined constraints via simulated play. Buhl et al. [3] highlight the utility of autonomous testing routines in everyday continuous integration and continuous delivery pipelines by contrasting the amount of encountered bugs against previous developments without them. Zheng et al. [22] designed a game playing agent utilizing deep reinforcement learning, while Chan et al. [23] made use of a neuroevolution approach that on top of playing was able to report on the constellation and sequence of actions that lead to game malfunctions. Furthermore, Bécares et al. [24] mapped human tester playthrough records to semantic replay models using Petri nets and Iftikhar et al. [25] and Schaefer et al. [26] introduced frameworks for autonomously testing generic games of the platformer or puzzle genre, respectively.

A number of studies tackle solvability, such as those of Powley et al. [27], Shaker et al. [28] or Volkmar et al. [29] that aided the level design of (procedurally generated) games by assuring potential solutions are feasible. Schatten et al. [30] simulated large-scale dynamic agent systems to test quest solvability in MMORPGs. Within the scope of point-and-click adventure games, Pfau et al. [4] established a generic adventure solver traversing these via reinforcement learning and reporting crashes, dead-ends and performance issues. Van Kreveld et al. [31] and Southey et al. [32] assessed difficulty or interestingness approximations of levels or mechanics by machine learning of descriptive in-game metrics.

Regarding balancing, scientific approaches often build on simulations that iteratively assess balance criteria and dynamically tune in-game parameters based on the former. Jaffe et al. [33], García-Sanchez et al. [34] and De Mesentier Silva et al. [35] applied this paradigm to board or card games, which was amplified by Mahlmann et al. [36] by introducing procedurally generated cards on top of these simulations. In other genres, Beau and Bakkes [37] utilized Monte-Carlo Tree Search for balancing units of Tower Defense games, Morosan and Poli [38] tweaked difficulty specifications in RTS and Arcade games after neuroevolution agents assessed these and Leigh et al. [39] dynamically balanced strategies though the coevolution of two competing agents playing a Capture The Flag game.

Closely related to the approach outlined in this paper, Holmgård et al. [40] conflated atomic player behavior into procedural personas to simulate and test different play styles in a Dungeon Crawler game and Gudmundsson et al. [41] utilized atomic choices in order to predict the difficulty of various levels of a Match-3-Puzzle game. Nonetheless, even if some approaches process some kind of human player input, incorporating actual information about *individual* and *atomic* player behavior has not been tackled yet. Generative player modeling has the potential to fuse automatic simulation methods with behavioral information, giving the developers the opportunity to receive practically immediate insights on which player strategies are popular, dominant and/or may require rework. Further generative player modeling is able to inform developers on how parameter tuning will likely alter the outcome of strategies before presenting it to the community, how implemented dynamic difficulty approaches

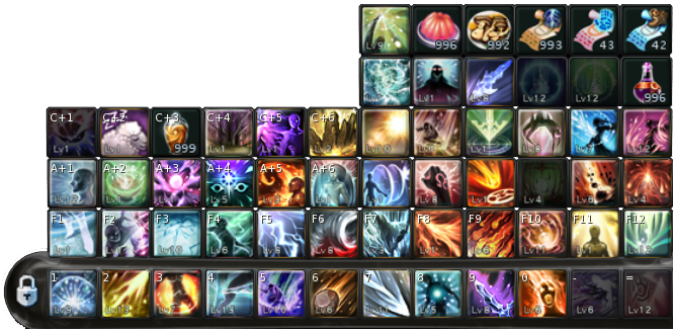

Fig. 1. Exemplary arrangement of a subset of skills available to the *Sorcerer* class in *Aion*. Additionally, context-dependent skills (when the player or a target opponent is in a particular condition) can be activated.

can be informed about parameter thresholds, and how to automatically balance game mechanics after large-scale permutations of classes, setups, parameters and behavior in all stages of development.

III. Approach

This section details our decisions for the selected game environment, the recorded data structure and the modeling approach.

A. Game Environment

To select a representative game within a genre that considerably suffers from the aforementioned balancing issues, we chose the MMORPG *Aion* in which a typical set of in-game classes is available. *Melee* classes (Gladiator, Templar, Assassin) mainly deal close-combat damage, in contrast to *Magic* classes (Sorcerer, Spiritmaster, Gunner) or Rangers. *Heal* classes (Cleric, Bard) deal less damage but offer additional support, while Chanters excel at the latter. Even if many in-game situations involve multi-player constellations, all classes are able to perform on their own in principle. Combat is mainly fought out by activating skill actions that harm the opponent(s) and/or benefit the player character (cf. Fig. 1). Depending on the sequencing of these skills and their contextual usage, individual players execute diverse strategies. Even if these strategies rarely maximize efficiency, they resemble situational preferences that emerge in personal play styles, such as improving own offensive or defensive capabilities or leading to maintained control over the opponent.

B. Dataset and Structure

Publicly accessible datasets that comprise vast proportions of recorded real-world player information are found in several instances, yet all of these third-party data providers offer only publicly available statistical meta-data describing high-level behavioral data. Even with the information about which actions are used in which frequencies, no knowledge is contained about the contextual game state during these action decisions, which, in turn, limits the expressiveness of the eventual generative agent. In contrast, we implement a state-action architecture mapping contextual information to individual player's decision making (indicated in Fig. 2 as input and output). Over the course of 6 months, 213 players with considerable prior expertise of *Aion* were recorded within a daily single-player dungeon instance in considerably challenging one-versus-one combat situations [17]. Table I provides the number of players in the dataset for each class.

C. Deep Player Behavior Modeling

DPBM realizes individual generative player modeling by assessing atomic player behavior in a state-action architecture and establishes a mapping among these via machine learning [16]. For generating a replicative agent that is representative of a single individual, the recorded behavioral data from all relevant observations was retrieved from the underlying database and fed into a feed-forward Multilayer perceptron (MLP) with backpropagation and a logistic sigmoid activation

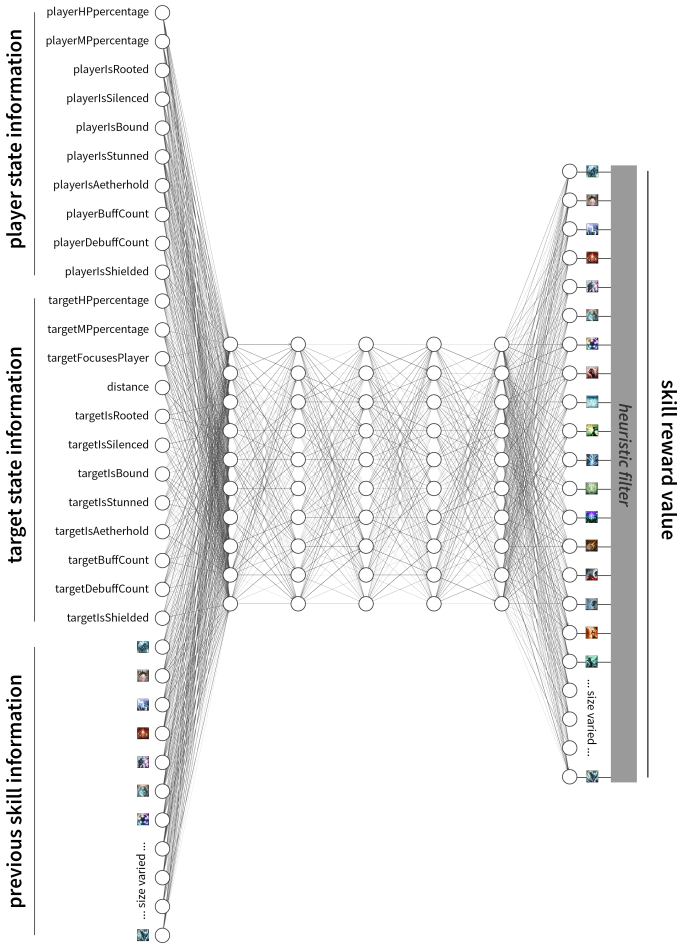

Fig. 2. The DPBM architecture mapping game state (information about player, opponent and preceding skill) to action (skill usage) probabilities. Design decisions can be found in [17]. The size of the input and output layers varied depending on the player's class, skill set and usage. The resulting action probability array is filtered heuristically by removing skills that are impossible to execute due to cool-down, MP shortage or other insufficient conditions.

TABLE I
PLAYER COUNT OF EACH *class* (AND **ARCHETYPE**) IN THE DATASET.

MELEE		**MAGIC**		**RANGED**		**HEAL**	
33	*Gladiator*	20	*Sorcerer*	13	*Ranger*	25	*Cleric*
19	*Templar*	18	*Spiritmaster*	**SUPPORT**		39	*Bard*
17	*Assassin*	10	*Gunner*	19	*Chanter*		

TABLE II
RECORDED NUMBER OF DIFFERENT SKILLS IN EACH CLASS OF *Aion*.

MELEE		**MAGIC**		**RANGED**		**HEAL**	
78	*Gladiator*	52	*Sorcerer*	53	*Ranger*	48	*Cleric*
56	*Templar*	51	*Spiritmaster*	**SUPPORT**		78	*Bard*
57	*Assassin*	42	*Gunner*	57	*Chanter*		

function. The input layer consisted of 22 nodes describing the contextual game state plus a set of nodes representing the preceding skill. Consisting of the same set of skill nodes, the output layer characterizes the probability distribution of action choices with respect to the individual player and the input situation (cf. Fig. 2). The sizes of the skill sets varied per class, as shown in Table II.

The network was initialized randomly and contained 4 hidden layers with equal size to the input layer. It was trained over 1000 epochs, based on insights from previous work [16]–[18], [42]; benchmarks prior to the study also indicated diminishing returns when further increasing the range of parameters.

When exposed to the testing environment, the trained model was applied generatively to retrieve a set of action probabilities given the occurring state description at real-time. After a weighted choice, the resulting skill was executed, followed by querying the DPBM for the next situation, effectively approximating the learned behavior from the original player's battles. Based on the player modeling taxonomy of Yannakakis et al. [15], [43], this implementation realizes a *model-free* (bottom-up) player modeling approach mapping *gameplay data* to actions via *classification*. According to the player modeling description framework of Smith et al. [44], DPBM directly utilizes *game actions* (domain) to *generate* (purpose) *individually* (scope) modeled behavior by means of *induced* (source) training of machine learning techniques.

D. Proficiency Metric

To estimate balance discrepancies between classes we construct a proficiency metric that assesses the quality of an agent's performance during evaluation. For the purpose of measuring a generalizable efficiency factor we consider four variables which are measured after a one-versus-one combat situation:

- The binary value of having won against the opponent (w)
- The normalized temporal duration of the fight (t)
- The agent's remaining health point (HP) percentage (hp_a)
- The opponent's remaining HP percentage (hp_o)

All variables lie between 0 and 1, are multiplied with their respective weight ($\alpha, \beta, \gamma, \delta$; all weights are equal for this study) and normalized over weights and the sum of observations (n), resulting in the final proficiency ϕ that ranges from worst-case (0) to optimal (1) performance:

$$\phi = \sum_{i,j=0}^{n} \frac{\alpha w + \beta(1-t) + \gamma hp_a + \delta(1-hp_o)}{(\alpha+\beta+\gamma+\delta)n^2}$$

IV. EVALUATION

In this section we outline the two evaluation environments (one-on-one PvE and PvP) used to assess the viability of each class. In addition, the section presents the regulation technique we used to mitigate balance discrepancies.

A. Player versus Environment (PvE) Evaluation

Focusing on differences between classes in one-versus-one situations, we chose to investigate performances of DPBM-driven agents encountering 100 opponents that incrementally increase in difficulty (see Fig. 3). To render the analysis visualizable and human-understandable, we only manipulate the offensive and defensive capabilities of each opponent, i.e. its *attack* and *maximal health points (maxHP)* values respectively. Since the proficiency distribution of a player population is likely to entail a great variance, the initial configuration was set to a trivial encounter, whereas the following modulations of the opponent increased *attack* and/or *maxHP* by 25% per iteration, up to a barely defeatable enemy. This led to a two-dimensional benchmark setup of continually increasing challenge with similarly decreasing expected proficiency. Figure 4 demonstrates the proficiency distributions together with the corresponding ϕ values of the best and worst DPBM-driven agent compared to the overall average.

After the evaluation of the 213 DPBM agents across 100 opponent configurations with incrementally increasing difficulty, the resulting proficiency estimations were categorized into the game-specific classes in order to compare their performance.

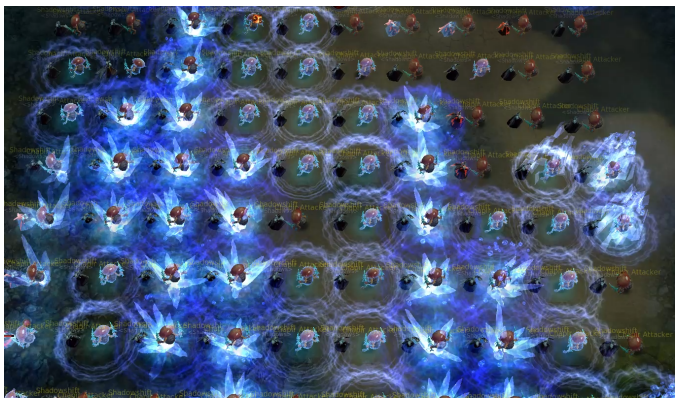

Fig. 3. In-game screenshot of the PvE benchmark (*Aion* [19]). DPBM-driven player replicas encounter 100 heuristic opponents with increasing difficulty in one-on-one situations (attack horizontally, maxHP vertically). Depending on the game state between the agent and its target, emerging behavioral patterns for action preferences and sequences can be monitored. For reasons of observability, entities are spawned with sparse distance to other confrontations. Yet, they are only able to damage and influence their respective counterpart.

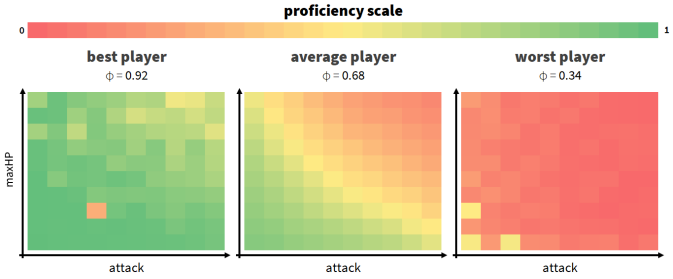

Fig. 4. Proficiency heatmaps of the best, average and worst player replication of the benchmark. The horizontal axis denotes the increasing attack value of the heuristic opponent while the vertical axis describes the increasing HP value of it (+25% per step, respectively).

As baselines, for each respective class, we observed the performance of an agent that modelled generalized (non-individual) behavior and an agent with random decision making.

B. Player versus Player (PvP) Evaluation

While the process of Section IV-A approximates the players' ability to cope with PvE encounters and therefore provides one measure of balance estimation, another dimension worth examining is the balance between the classes themselves. Thus, a subsequent evaluation pitted all player replicas against each other in one-on-one PvP confrontations, leading to $22,578$ unique combinations (including intra-class battles). The proficiency outcomes of these matches were pooled and averaged to measure systematic dominance or inferiority relationships between classes. To prevent never-ending duels (e.g. between two agents using the healer classes and mainly defensive strategies), the maximal duration was capped at five minutes.

C. Regulation

The DPBM approach primarily focuses on *informing* game development about possible imbalances within a player population; however, certain regulation techniques can follow immediately, assuming that all classes should follow a similar proficiency distribution. The most direct approach of regulation would be to tune the environmental parameters such as the offensive and defensive capabilities of opponents (similar to Section IV-A). Thus, we subsequently determine a meaningful target proficiency (in this example, the mean proficiency of all iterations) and computed the mean squared error of measured proficiency values of each player in a class. From this, we reveal the approximate parameter values to tune by calculating the center of mass of these errors per class. Eventually, the proficiency distributions for all classes can be compared, given the PvE benchmark results of the respective players and using the tuned parameters for their opponents.

V. Results

Table III outlines the testing prediction accuracies of the employed DPBMs (using a 80-20 holdout validation method) including conservative heuristic filtering within the most probable 1, 5 and 10 skill choices. In addition, the table includes the training times per player as measured on a NVIDIA GeForce RTX2080 using Keras 2.2.4 with TensorFlow 2.0.0 backend.

TABLE III
Prediction accuracy on the testing set, with corresponding training times for the various DPBMs employed.

	Testing accuracy			Training time
	Top-1	Top-5	Top-10	
M:	61.3%	75.3%	81.3%	$7.24s$
SD:	22.4%	11.8%	14.9%	$1.68s$

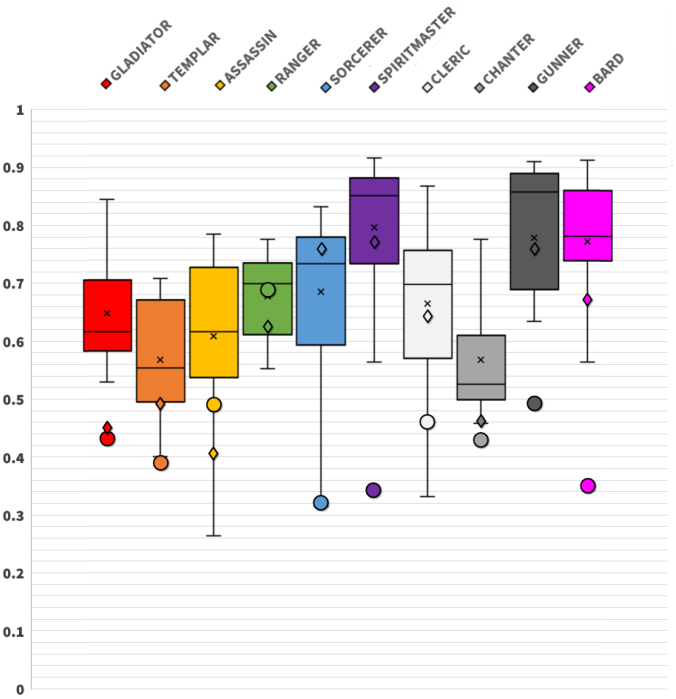

Fig. 5. Proficiency results of player replicas across different classes. The graph depicts mean values (indicated by **x**), median values (–), the proficiency of the generalized model of the class (◊) and random guessing (◯).

A. Player versus Environment (PvE) Evaluation

Using a one-way ANOVA, we find a significant difference of DPBM-agent proficiency across classes in the PvE evaluation, displaying a large effect size ($F(9, 203) = 9.63$, $p < .01$; partial $\eta^2 = 0.3$; see Fig. 5). Further we use Bonferroni-corrected two-tailed Welch's t-tests to highlight statistical differences between particular classes. Highlighting notable disparities, players of the Spiritmaster, Gunner or Bard class scored higher proficiency values than most other classes while Chanter and Templar players were almost consistently outperformed by other classes ($p < 0.05$). After further t-tests, significant proficiency differences between individual DPBM and generalized models became apparent ($p < 0.05$, Cohen's $d = 0.61$). This also holds in comparison to the random decision making agent ($p < 0.01, d = 2.45$).

B. Player versus Player (PvP) Evaluation

With respect to the PvP evaluation, Fig. 6 visualizes average proficiency values of player replicas from one class compared to all other classes. While classes within the same archetype (e.g. Gladiator and Templar both being physical Melee classes or Sorcerer and Spiritmaster both being Magical ranged

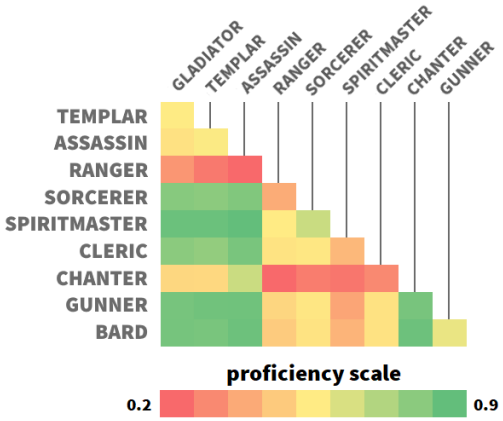

Fig. 6. Mean proficiency results of DPBM-driven player replicas in one class (vertically) when fighting replicas of other classes (horizontally).

classes) had few proficiency differences ($p > 0.05$), distinct superiority relationships emerge when different archetypes are matched up. Rangers scored significantly lower against Melee classes (Gladiator, Templar, Assassin, $p < 0.05$), yet they consistently outperformed the Magic classes (Sorcerer, Spiritmaster, Gunner, $p < 0.05$). The Magic classes were equally and consistently able to dominate Melee classes, effectively representing a rock-paper-scissors-like interaction scheme. The Heal classes (Cleric, Bard) also outperformed Melee, yet succumbed to both Rangers as well as to the Magic classes ($p < 0.05$). Being the game's primary support class, Chanters were dominated by the majority of opposing classes.

C. Regulation

Figure 7 visualizes the regulation with the tuned opponent parameter values for each class and the corresponding proficiency distribution of DPBM-driven player replicas. According to a one-way ANOVA, there are no significant differences remaining between class proficiency values after parameter tuning ($F(9, 203) = 1.42$, $p > 0.05$).

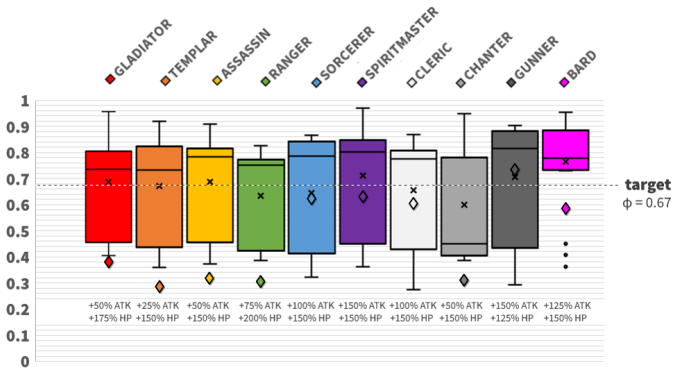

Fig. 7. Proficiency results of player replicas across different classes after parameter tuning for a balanced resulting proficiency ($\phi = 0.67$). The graph depicts mean values (indicated by **x**), medians values (–), outlier values (•) and tuned parameters.
◊ indicates the proficiency value of the generalized class model.

When compared to average adjustment values, opponents' *attack* was regulated weaker for Melee and Support classes, but notably higher for Magic and Heal classes. In contrast, Rangers faced opponents with average *attack*, but considerably increased *maxHP* after regulation.

VI. Discussion

With regards to the PvE evaluation, the ANOVA and subsequent post-hoc tests revealed significant differences in proficiency between player replicas of different classes. This might indicate imbalances of these classes, yet it should be interpreted with respect to the underlying design guidelines. For instance, the relatively low proficiency scores of the Chanter and Templar classes likely stem from their reliance on other players, either to support or to receive support from respectively. Still, under the assumption that primarily damage-dealing classes should be equally viable, certain discrepancies emerge that point to certain magical classes (such as Spiritmaster, Gunner or Bard) outperforming physical damage dealers (such as Gladiator, Assassin) significantly. When interpreting results of the PvP benchmark, it is worth noting that balanced viability does not necessarily have to result in each class having equal chances against all others. Depending on the underlying design agenda, a similarly balanced constellation is a rock-paper-scissors-like interaction scheme between classes, which this evaluation was able to demonstrate approximately (see Fig. 8). Nevertheless, the inferior performance of the Chanter in both one-on-one PvE as well as PvP situations might encourage developers to augment the versatility of this class if its role is not only meant to support other players.

The approach introduced with this paper has no access to the underlying design constellations of the original game and primarily aims to inform developers about imbalances. However, we have already indicated and tested possible procedures to adjust the viability of these classes towards a balanced configuration. The adjustment of opponents for individual classes, based on the proficiency distribution of its players, has proven to detect configurations that end up in a more balanced outcome (see Fig. 7 as compared to Fig. 5). While this

Fig. 8. Illustrated outcome of the DPBM-driven PvP simulation. On average, players of Melee classes outperform Rangers, which themselves counter Magic classes, which eventually beat Melee classes. Heal classes are dominated by Rangers and Magic classes, but can withstand Melee.

method yields promising results in terms of balanced viability for single-player games or solo dungeons, its adjustment is not trivially applicable to group PvE situations, since the opponents are only tailored to a single class and the interaction between classes further confounds the attunement. A more comprehensive regulation method would be the adjustment of in-class parameters, such as their own offensive or defensive values or particular skill values. Yet, this would significantly influence the interaction between the classes and might harm the likely intended rock-paper-scissors scheme within. If aiming for equal proficiency of all classes against each other, an iterative procedure of attunement and re-simulation would be expedient, in that the largest proficiency mismatch between classes is detected, adjusted in favor of the inferior class and affected matchups are re-simulated, reiterated up to a predefined threshold.

Based on the empirical evidence presented, our previously posed research questions can be answered as follows:

- *Significant imbalances between in-game classes can be detected through DPBM within PvE and PvP.*
- *Design specifications can be established via regulation based on DPBM-driven simulation results.*

VII. Limitations and Future Work

During the implementation of this approach, different constraints and assumptions had to be taken into account that eventually lead to a number of limitations. Perhaps most importantly, classes (especially in MMORPGs) are often designed to vary in versatility within different situations or against different classes. This includes classes that benefit greatly from party-play versus classes that are tailored for single-player situations, those who focus on dealing damage to many enemies instead of single targets or not dealing damage at all (being busy with tanking, healing or supporting otherwise). Nevertheless, the presented technique is not constrained to damage dealing, but overall one-on-one versatility. DPBM can quantify these differences to inform game developers whether their intended design aligns with the actual outcomes of a population playing it. Evidently, this requires data from a player population to employ the testing procedures, which limits its versatility before the game's launch. However, it is applicable for never-ending balance observations (and predictions) and for benchmarking novel challenges introduced with later patches or DLCs. Apart from *generalization* or *random play* in the PvE evaluation, an optimally playing agent (e.g. by self-training/reinforcement learning) would be an additional interesting candidate, to compare if this approach is closer to the real population. To filter out the influence of different attribute stats, we normalized equipment and other relevant configurations throughout all characters. A closer (yet very temporary) approximation of the overall population capability could be realized with this approach if the equipment range was taken into consideration. Eventually, player models in the PvP evaluation were driven by the same behavior, independent of their opponent, since this behavior was only trained on data stemming from battles against their own class [17]. This likely distorted the results and should be repeated when enough data of the respective situations are given; however, it does not diminish the potential of DPBM.

For future work, we primarily seek to refine behavior modeling by introducing more variables, such as global movement information (encompassing higher level goals) or the estimation of individual players' precision and their temporal cognitive computation demand. Apart from the constraint of two dimensions (attack and defense), the challenge of the PvE encounters can further be examined by altering the skill sets, decision making or movement behavior of enemies. The simulations themselves can likely be sped up by calculating battles without graphical representations. Eventually, the applicability of this approach will be investigated with respect to significantly more complex multi-player situations, such as in adjusting boss battles for a population (PvE) or simulating large-scale competitive sieges (PvP), throughout multiple player experience evaluations. Furthermore, if a mapping from mere behavioral patterns to in-game proficiency can be constructed, this prediction might augment matchmaking (for both PvP and PvE) bringing together players with approximate skill levels more accurately.

VIII. Conclusion

Balancing in-game parameters and classes to ensure diverse viable choices for players is a challenging, time-consuming and toiling expense for game developers. While traditional approaches employ expert analysis, excessive human playtesting and persistent debates with the community, this paper introduces the use of an individual generative player modeling technique (DPBM) for automating game balancing. Using a dataset of 213 players that visited a single-player dungeon of the MMORPG *Aion* over the course of six months, we we generated individual agents replicating human play behavior. Within the context of one-on-one PvE battles, we detected and sufficiently regulated significant effects between classes. For the interaction between classes, a PvP evaluation among all players revealed a rock-paper-scissors-like interaction scheme that is likely to resemble the original developers' design. The proposed approach is able to inform game development about PvE and PvP imbalances quantitatively and provide empirical evidence that player behavior entails a degree of individual proficiency.

References

[1] M. Washburn Jr, P. Sathiyanarayanan, M. Nagappan, T. Zimmermann, and C. Bird, "What went right and what went wrong: an analysis of 155 postmortems from game development," in *Proceedings of the 38th International Conference on Software Engineering Companion*, 2016, pp. 280–289.

[2] D. Lin, C.-P. Bezemer, and A. E. Hassan, "Studying the urgent updates of popular games on the steam platform," *Empirical Software Engineering*, vol. 22, no. 4, pp. 2095–2126, 2017.

[3] C. Buhl and F. Gareeboo, "Automated testing: A key factor for success in video game development. case study and lessons learned," in *proceedings of Pacific NW Software Quality Conferences*, 2012, pp. 1–15.

[4] J. Pfau, J. D. Smeddinck, and R. Malaka, "Automated game testing with icarus: Intelligent completion of adventure riddles via unsupervised solving," in *Extended Abstracts Publication of the Annual Symposium on Computer-Human Interaction in Play*. ACM, 2017, pp. 153–164.

[5] Blizzard Entertainment, "*StarCraft II*," Game [PC], July 2010, blizzard Entertainment, Irvine, CA, USA. Played 2017.

[6] Blizzard Entertainment, "*Overwatch*," Game [PC,PS4,XboxOne,Switch], May 2016.

[7] ArenaNet, "*Guild Wars*," Game [PC], April 2005, arenaNet, Bellevue, WA.

[8] D. Sirlin, "Balancing multiplayer competitive games," in *Game Developer's Conference*, 2009.

[9] K. Hullett, N. Nagappan, E. Schuh, and J. Hopson, "Empirical analysis of user data in game software development," in *Proceedings of the 2012 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*. IEEE, 2012, pp. 89–98.

[10] C. Lewis and N. Wardrip-Fruin, "Mining game statistics from web services: a world of warcraft armory case study," in *Proceedings of the Fifth International Conference on the Foundations of Digital Games*. Citeseer, 2010, pp. 100–107.

[11] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[12] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, p. 484, 2016.

[13] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel *et al.*, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," *arXiv preprint arXiv:1712.01815*, 2017.

[14] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev *et al.*, "Grandmaster level in starcraft ii using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[15] G. N. Yannakakis and J. Togelius, *Artificial Intelligence and Games*. Springer Nature, 2018.

[16] J. Pfau, J. D. Smeddinck, and R. Malaka, "Towards deep player behavior models in mmorpgs," in *Proceedings of the 2018 Annual Symposium on Computer-Human Interaction in Play*. ACM, 2018, pp. 381–392.

[17] 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*. ACM, 2020. *In press*.

[18] J. Pfau, J. D. Smeddinck, I. Bikas, and R. Malaka, "Bot or not? user perceptions of player substitution with deep player behavior models," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. ACM, 2020. *In press*.

[19] NCsoft, "*Aion*," Game [PC], Seongnam, South Korea, September 2008, nCSoft, Seongnam, South Korea. Played August 2019.

[20] S. Radomski and T. Neubacher, "Formal verification of selected game-logic specifications," *on Engineering Interactive Computer Systems with SCXML*, p. 30, 2015.

[21] S. Varvaressos, K. Lavoie, S. Gaboury, and S. Hallé, "Automated bug finding in video games: A case study for runtime monitoring," *Computers in Entertainment (CIE)*, vol. 15, no. 1, p. 1, 2017.

[22] Y. Zheng, X. Xie, T. Su, L. Ma, J. Hao, Z. Meng, Y. Liu, R. Shen, Y. Chen, and C. Fan, "Wuji: Automatic online combat game testing using evolutionary deep reinforcement learning," in *Proceedings of the 34th ACM/IEEE International Conference on Automated Software Engineering*, 2019.

[23] B. Chan, J. Denzinger, D. Gates, K. Loose, and J. Buchanan, "Evolutionary behavior testing of commercial computer games," in *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753)*, vol. 1. IEEE, 2004, pp. 125–132.

[24] J. H. Bécares, L. C. Valero, and P. P. G. Martín, "An approach to automated videogame beta testing," *Entertainment Computing*, vol. 18, pp. 79–92, 2017.

[25] S. Iftikhar, M. Z. Iqbal, M. U. Khan, and W. Mahmood, "An automated model based testing approach for platform games," in *2015 ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS)*. IEEE, 2015, pp. 426–435.

[26] C. Schaefer, H. Do, and B. M. Slator, "Crushinator: A framework towards game-independent testing," in *Proceedings of the 28th IEEE/ACM International Conference on Automated Software Engineering*. IEEE Press, 2013, pp. 726–729.

[27] E. J. Powley, S. Colton, S. Gaudl, R. Saunders, and M. J. Nelson, "Semi-automated level design via auto-playtesting for handheld casual game creation," in *2016 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 2016, pp. 1–8.

[28] M. Shaker, M. H. Sarhan, O. Al Naameh, N. Shaker, and J. Togelius, "Automatic generation and analysis of physics-based puzzle games," in *2013 IEEE Conference on Computational Inteligence in Games (CIG)*. IEEE, 2013, pp. 1–8.

[29] G. Volkmar, N. Mähmann, and R. Malaka, "Procedural content generation in competitive multiplayer platform games," in *Joint International Conference on Entertainment Computing and Serious Games*. Springer, 2019, pp. 228–234.

[30] M. Schatten, B. O. urić, I. Tomičič, and N. Ivkovič, "Automated mmorpg testing–an agent-based approach," in *International conference on practical applications of agents and multi-agent systems*. Springer, 2017, pp. 359–363.

[31] 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.

[32] F. Southey, G. Xiao, R. C. Holte, M. Trommelen, and J. W. Buchanan, "Semi-automated gameplay analysis by machine learning." in *AIIDE*, 2005, pp. 123–128.

[33] A. Jaffe, A. Miller, E. Andersen, Y.-E. Liu, A. Karlin, and Z. Popovic, "Evaluating competitive game balance with restricted play," in *Eighth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2012.

[34] P. García-Sánchez, A. Tonda, A. M. Mora, G. Squillero, and J. J. Merelo, "Automated playtesting in collectible card games using evolutionary algorithms: A case study in hearthstone," *Knowledge-Based Systems*, vol. 153, pp. 133–146, 2018.

[35] F. de Mesentier Silva, S. Lee, J. Togelius, and A. Nealen, "Ai as evaluator: Search driven playtesting of modern board games." in *AAAI Workshops*, 2017.

[36] T. Mahlmann, J. Togelius, and G. N. Yannakakis, "Evolving card sets towards balancing dominion," in *2012 IEEE Congress on Evolutionary Computation*. IEEE, 2012, pp. 1–8.

[37] P. Beau and S. Bakkes, "Automated game balancing of asymmetric video games," in *2016 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 2016, pp. 1–8.

[38] M. Morosan and R. Poli, "Automated game balancing in ms pacman and starcraft using evolutionary algorithms," in *European Conference on the Applications of Evolutionary Computation*. Springer, 2017, pp. 377–392.

[39] R. Leigh, J. Schonfeld, and S. J. Louis, "Using coevolution to understand and validate game balance in continuous games," in *Proceedings of the 10th annual conference on Genetic and evolutionary computation*. ACM, 2008, pp. 1563–1570.

[40] C. Holmgard, M. C. Green, A. Liapis, and J. Togelius, "Automated playtesting with procedural personas with evolved heuristics," *IEEE Transactions on Games*, 2018.

[41] 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.

[42] J. Pfau, J. D. Smeddinck, and R. Malaka, "Deep player behavior models: Evaluating a novel take on dynamic difficulty adjustment," in *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, 2019, p. LBW0171.

[43] G. N. Yannakakis, P. Spronck, D. Loiacono, and E. André, "Player modeling." Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2013.

[44] A. M. Smith, C. Lewis, K. Hullet, G. Smith, and A. Sullivan, "An Inclusive View of Player Modeling," in *Proceedings of the 6th International Conference on Foundations of Digital Games*, ser. FDG '11. New York, NY, USA: ACM, 2011, pp. 301–303, event-place: Bordeaux, France. [Online]. Available: http://doi.acm.org/10.1145/2159365.2159419