Dungeons & Replicants II: Automated Game Balancing Across Multiple Difficulty Dimensions via Deep Player Behavior Modeling

Pfau, Johannes; Liapis, Antonios; Yannakakis, Georgios N; Malaka, Rainer

IEEE Transactions on Computational Intelligence in AI and Games

Goal

Providing a more effective and efficient method for **game balancing**.

- Deep Player Behavior Modeling

- dataset of player behavior in Aion(MMORPG).

Dungeons & Replicants II

- The approach introduced in this paper.

- A method for automated game balancing across multiple difficulty dimensions using **Deep Player Behavior Modeling**.

- training AI "replicants" to reproduce the playing strategies of individual players

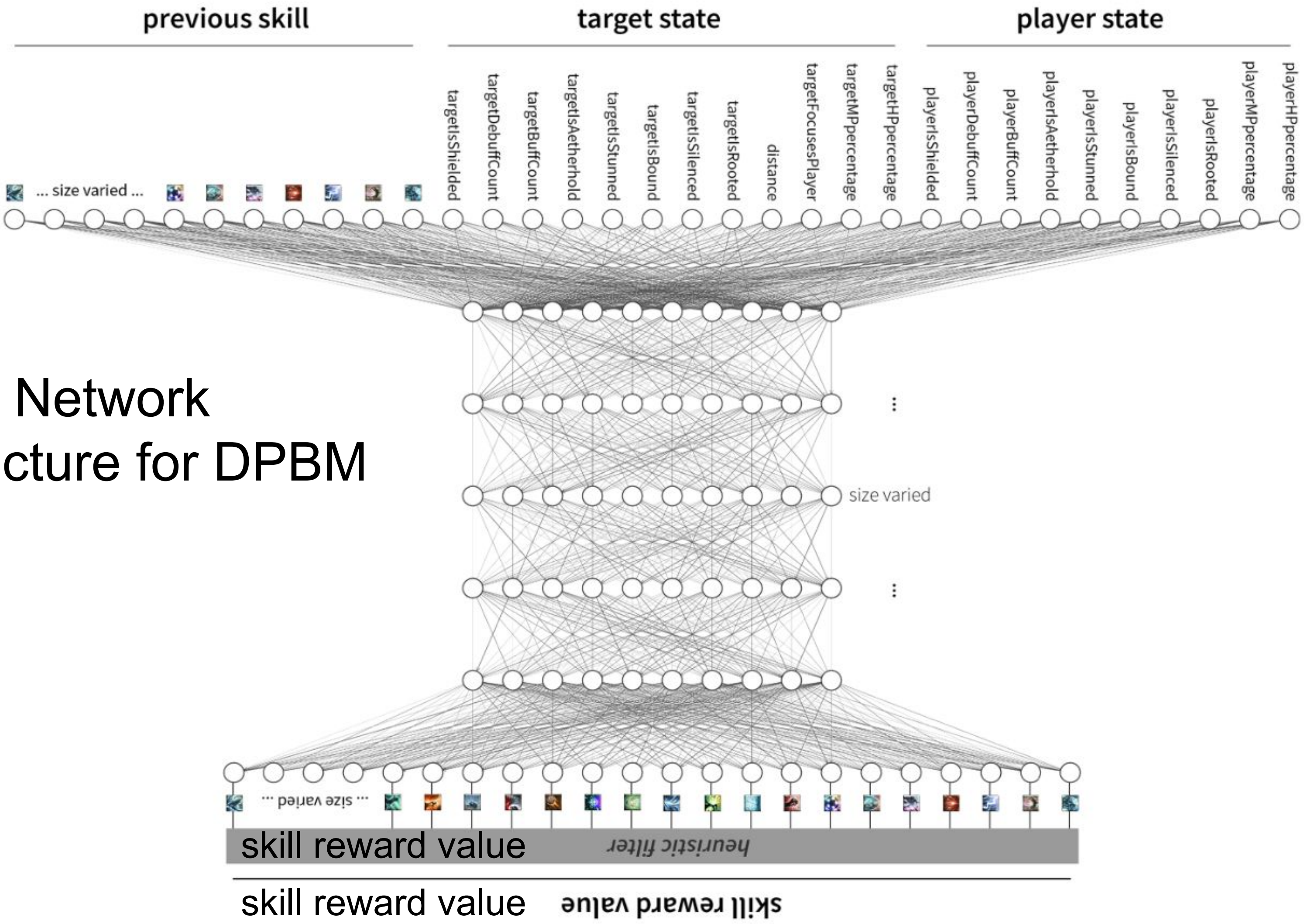
Deep Player Behavior Modeling

- It allows for individualized player behavior mapping.

- It trains AI "replicants" to automatically test and balance a game.

- It allows for the generation of agent behavior from player-constructed models, providing developers with insights on popular player strategies, parameter tuning, and the likely outcome of strategies

Neural Network architecture for DPBM



skill reward value

skill reward value

Game environment

- The MMORPG Aion (NCSoft, 2008)
    - Collected primary player data from experienced Aion players
    - Classes (melee, ranged, rogue, buffer, debuffer, healer, tank, etc.)
    - PvE settings

Proficiency Metric

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

- w: The binary value of having won against the opponent.

t: The normalized temporal duration of the fight.

hp_a: The agent’s remaining hit point (HP) percentage.

hp_o: The opponent’s remaining HP percentage.

Experiment

- 213 DPBM-driven agents as replicants of the players

- 8 scenarios

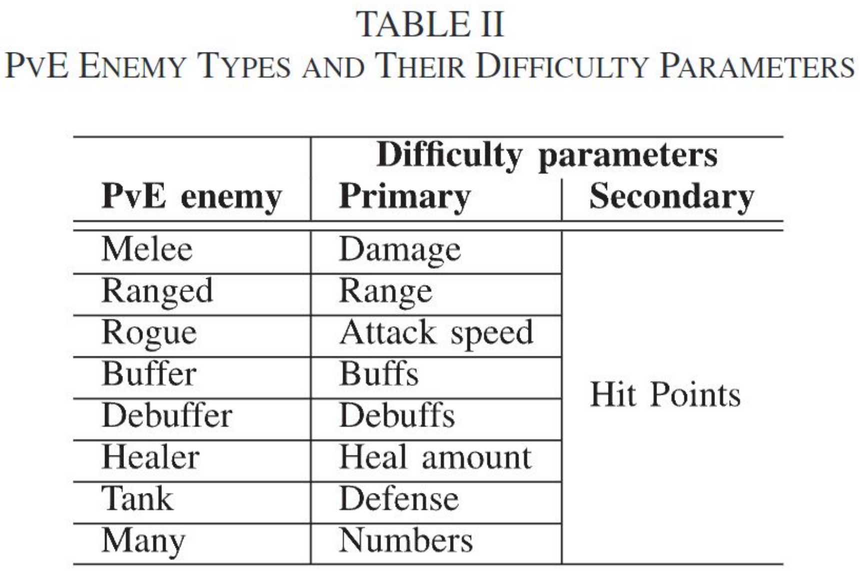


TABLE II
PvE Enemy Types and Their Difficulty Parameters

PvE enemy	Difficulty parameters	
	Primary	Secondary
Melee	Damage	Hit Points
Ranged	Range	
Rogue	Attack speed	
Buffer	Buffs	
Debuffer	Debuffs	
Healer	Heal amount	
Tank	Defense	
Many	Numbers	

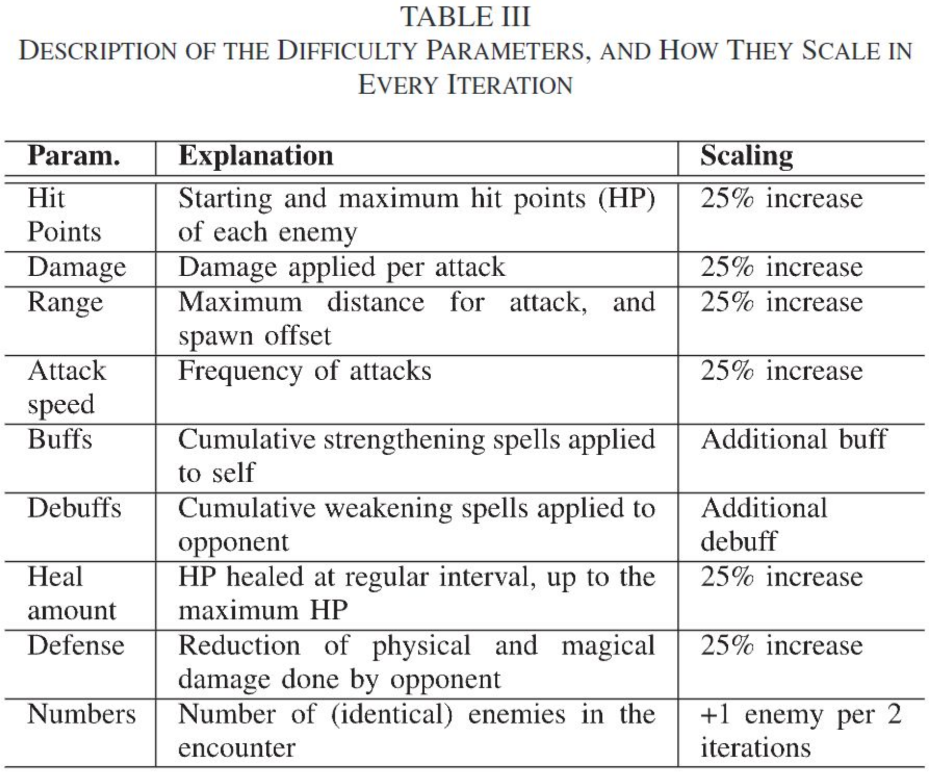


TABLE III
Description of the Difficulty Parameters, and How They Scale in Every Iteration

Param.	Explanation	Scaling
Hit Points	Starting and maximum hit points (HP) of each enemy	25% increase
Damage	Damage applied per attack	25% increase
Range	Maximum distance for attack, and spawn offset	25% increase
Attack speed	Frequency of attacks	25% increase
Buffs	Cumulative strengthening spells applied to self	Additional buff
Debuffs	Cumulative weakening spells applied to opponent	Additional debuff
Heal amount	HP healed at regular interval, up to the maximum HP	25% increase
Defense	Reduction of physical and magical damage done by opponent	25% increase
Numbers	Number of (identical) enemies in the encounter	+1 enemy per 2 iterations

Experiment

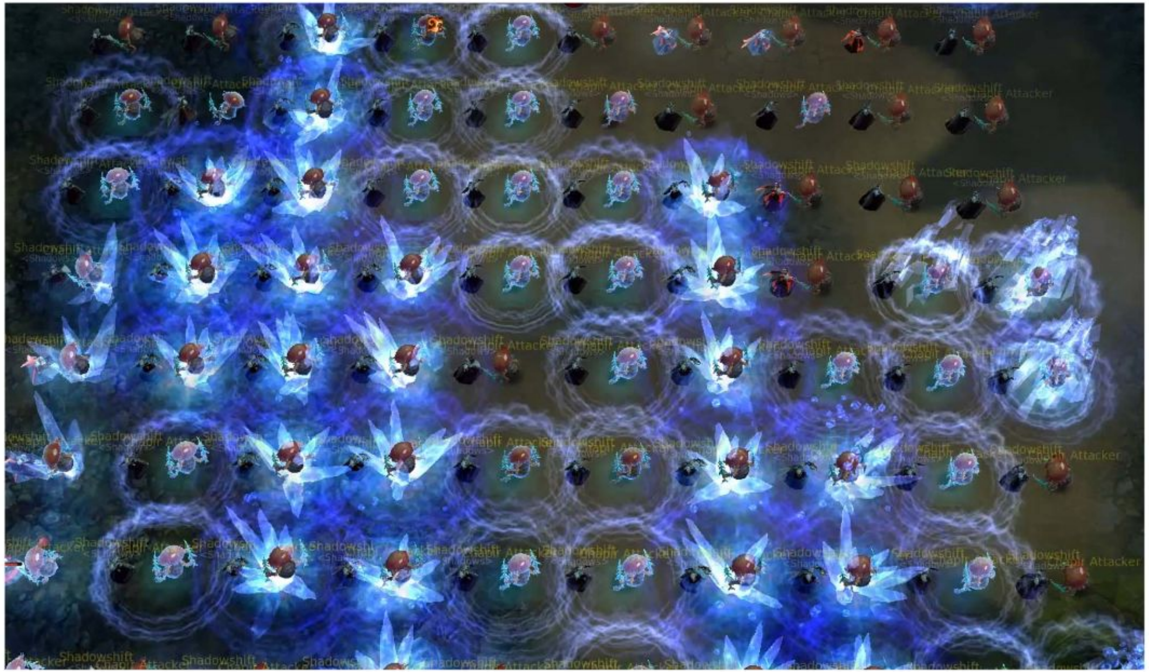


Fig. 2. In-game screenshot of the PvE benchmark in *Aion*, with a replicant fighting in 100 combat encounters against enemies with different difficulty parameters.

Result General finding

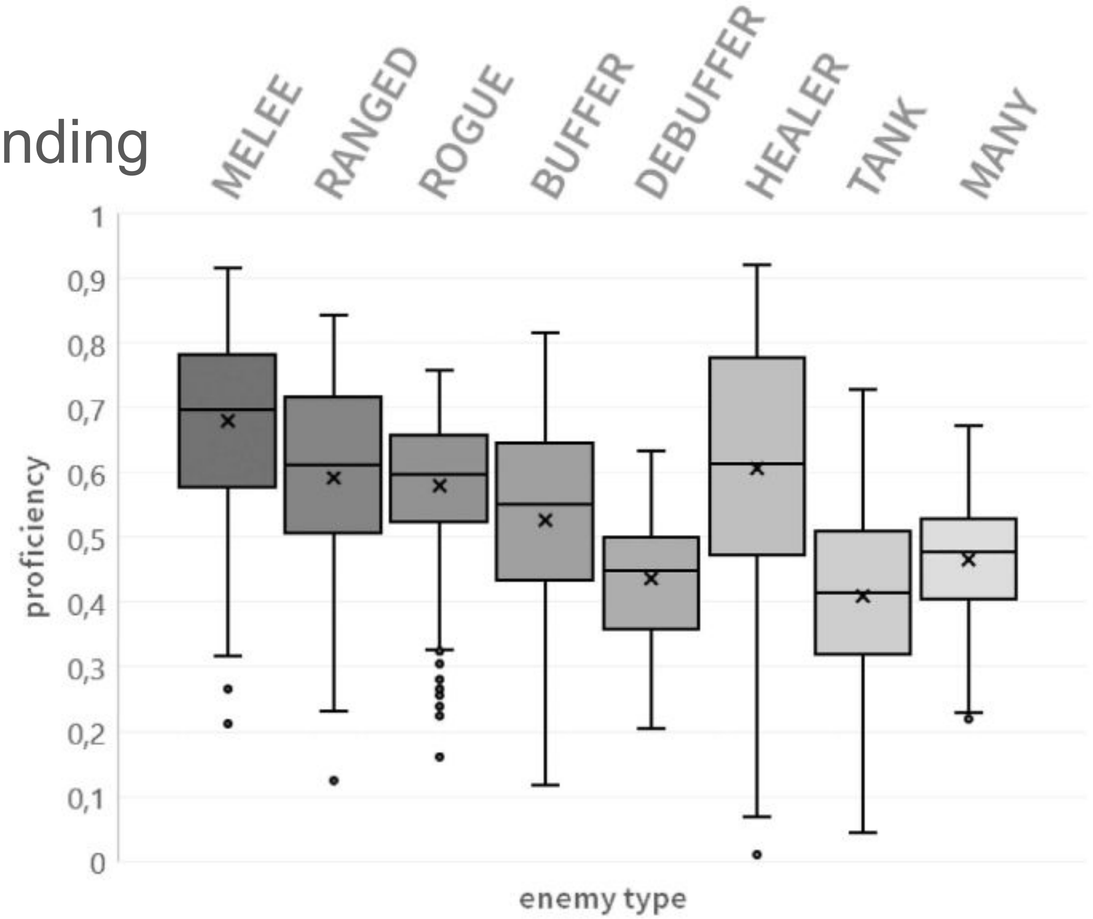


Fig. 3. Resulting ϕ proficiency of all 213 replicants against the eight different PvE encounter conditions.

Result Differences between classes

Some particular deviant cases

- Gladiators performed significantly worse against Debuffers.

- Templars performed significantly worse against Melee and Healers, but significantly better against Tanks.

- Spiritmasters performed significantly worse against Tanks.

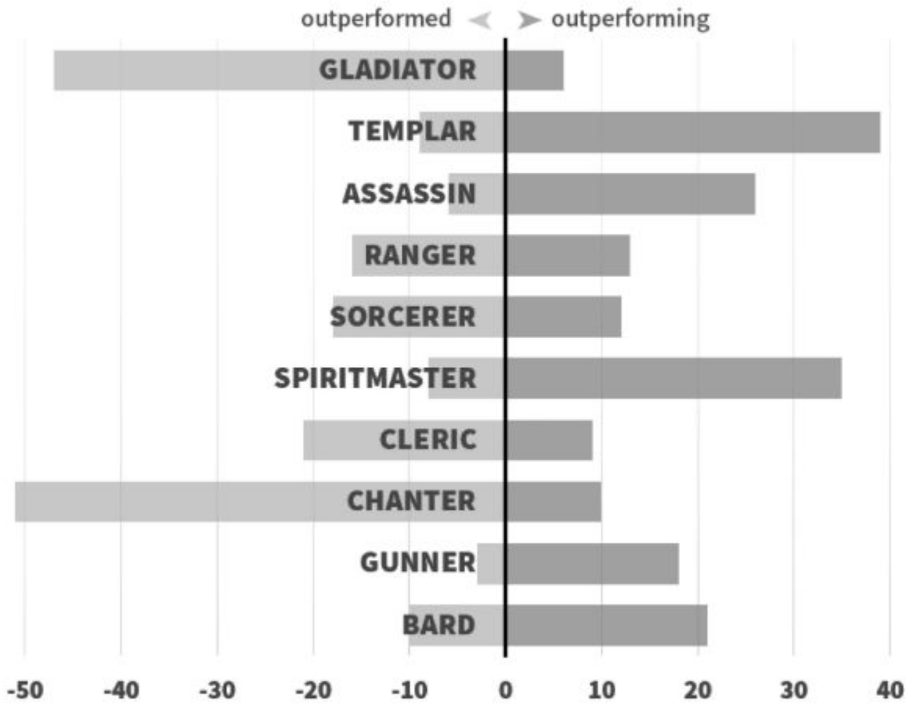


Fig. 5. Boxplot of normalized ϕ proficiencies of classes throughout the eight different encounters.

Result

Q1. Can imbalances between in-game classes be detected (through batched simulation analysis with generative player modeling)?

A1. It can be detected by generative player modeling and iterative simulations.

Q2. Does the segmentation across multiple difficulty dimensions help reveal the strengths and weaknesses of particular classes?

A2. Assessing multiple difficulty dimensions reveals class-specific strengths and weaknesses and provides overall insight.

Q3. Can the field of automated game testing harness results of generative player modeling simulations to compute balanced configurations across classes?

A3. Atomic replicants across the player population can be used to compute combination of parameter that lead to a balanced state.

Conclusion

Automatic game balancing can be achieved by having a play trace of a group of players.

This procedure can be applied to other games and genres as long as you provide
- entities to balance (such as classes),
- meaningful benchmark simulations(e.g., combat situations),
- low-level interaction data that is representative of the player population.

Thank you for listening