Balancing Wargames through Predicting Unit Point Costs

George E.M. Long
Queen Mary University of London
g.e.m.long@qmul.ac.uk

Diego Perez-Liebana
Queen Mary University of London
diego.perez@qmul.ac.uk

Spyridon Samothrakis
University of Essex
ssamot@essex.ac.uk

***Abstract*—In tactical wargames, such as Warhammer 40K, two or more players control asymmetrical armies that include multiple units of different types and strengths. In these type of games, unit are assigned point costs, which are used to ensure that all players will control armies of similar strength. Players are provided with a total budget of points they can spend to purchase units that will be part of their army lists. Calculating the point value of individual units is a tedious manual process, which often requires long play-testing sessions and iterations of adjustments. In this paper, we propose an automated way of predicting these point costs using a linear regression approach. We use a multi-unit, turn-based, non-balanced game that has three asymmetric armies. We use Monte Carlo Tree Search agents to simulate the players, using different heuristics to emulate playing strategies. We present six different variants of our unit-point prediction algorithm, and we show how our best variant is able to almost reduce the unbalanced nature of the game by half.**

***Index Terms*—wargames, automatic game balancing, game testing**

I. Introduction

Wargames are a game genre where players control armies to act out a battle or war, usually played on a detailed map under a set of rules to see if their strategies influence the game's outcome. [1] These games have different levels of scope, the most common being the tactical level and the strategy level. At the tactical level, the scenario focuses on an individual battle and players command individual units. At the strategic level, decisions are made pertaining to the entire theatre of war.

One of the most important aspects of these wargames is balance. There is no agreed upon definition for Game Balance, but in it's most basic form it can be described as "tuning a game's rules, levels, difficulty, numbers, algorithms, etc. to achieve the desired goals". [2] In strategy games such as wargames, there are several points of view on the ideal balance goal. A common goal is having a simple 50%-50% winrate, where every army would have an equal chance of winning. Another viewpoint is *intransitive superiority*, a rock-paper-scissors style in which each army counters another (while itself being countered by a different army). [3] In this paper, we have chosen the more traditional 50%-50% balance goal, without imposing any limitation to any others.

An aspect of balancing in the strategy layer is unit costs. Wargames often operate with an element of scarcity, with unit costs being one way to enforce this. For example, a common army budget in Warhammer 40k [4] is a thousand points, and each unit included in the army (and the optional equipment added to them) has a point cost. This ensures that armies fighting against each other will be of a similar strength and also adds an element of strategy to the game, as players must optimise the point costs of their armies to create the strongest army at the budget level.

Unit costs are considered one of the hardest parts of balancing in wargames. As described in Tabletop Wargames: A Designers' and Writers' Handbook: "There are essentially three things to grasp about points values - (i) they don't work, (ii) nevertheless we have to have them and (iii) even so they can't really be reduced to a mathematical formula." [5].

Some wargames, such as Song of Blade and Heroes [6], use a mathematical formula to calculate unit costs. However, this is often impractical for wargames with more complexity. In these games, these points are chosen by estimating a value after an initial playtest, and then tuning this value through many more tests. This is a lengthy and tedious process that often gives an inaccurate value that can allow overpowered units to slip through and help form dominant strategies.

Artificial intelligence has been proposed as a method to automate some of the balancing process over the years, with promising results [7]. However, to the authors' knowledge, there has not been any work on how it can be used to help estimate point costs in wargames. In this paper, we propose a method to automatically predict the point costs for units. This provides an interesting challenge, as wargames are often extremely complex, with asymmetric armies, variable terrain, and intricate unit designs creating a large problem area to explore. We see that our approach has some novel features over traditional automated balancing methods. Since it attempts to achieve balance in the game without modifying any of the rules or unit parameter values, it can be used to balance already published games. This may be useful for players and designers, as modifications can be proposed to correct potentially unbalanced aspects of the game after the game is finished. In addition, it could be used to determine alternative rule-sets for games in scenarios such as tournaments, where highly balanced rules or corrections on these are often required.

The main contribution of this paper is the a new algorithm that allows unit costs to be predicted by performing linear regression. The method uses data collected from observing

979-8-3503-2277-4/23/$31.00 ©2023 IEEE

the output of two agents playing a bespoke made wargame, simulating multiple army configurations and compositions. We validate these predicted point costs by incorporating them into an army generation process and calculate the new balance of the game. Our results show that the proposed algorithm is able to reduce the imbalance in the game by half.

Section II introduces the background of this work, with regards to wargames and AI for game balancing. Section III introduces the framework and game used in this paper, as well as the main method (Monte Carlo Tree Search) used for the testing agents. Section IV introduces our proposed method and Section V discusses our experimentation work and results. We conclude with a summary of our findings and possibilities for future work in Section VI.

II. Background

This background section will give an overview of the topics of wargames and how artificial intelligence is used in game balancing. It will briefly discuss the history of these topics, relevant aspects, and how they can be applied in our work.

A. Wargames

As hinted at in the name, modern wargames are derived from simulations of war, which were used to teach military strategy and tactics to officers. Over time these simulations became more popular in militaries around the world, and as people became fond of them, versions of these games for casual players began to emerge. These casual games such as Tactics [8] or Risk [9] evolved into two separate categories, Board Wargames and Miniature Wargames.

Our work focuses on Miniature Wargames, known for their miniature figures, which represent individual units on the battlefield. These units have aforementioned point costs that are ideally equivalent to their strength in the game. Stronger units will have more influence in the battle and will have higher point costs. These point costs are independent of any other factors, and while it is likely that a unit's usefulness will change based on the contexts of battle (e.g., a powerful unit with low health might be stronger paired with a support unit to reinforce it), its point cost will stay the same.

As mentioned in the Introduction, armies have point budgets to ensure that two armies will be of a similar power level. The collection of units that make up the army is known in the wargaming community as the army list. These army lists are often exploited to give the player an advantage, with players incorperating units into their army which are stronger than their point costs suggest.

There have been some games with calculable point costs. To elaborate on the example of A Song of Blade and Heroes from the introduction, a unit's point cost can be calculated by $\frac{(5C+S)(7-Q)}{2}$, where C is the combat power of the unit, Q is its quality, and S are the costs of any special aspects (e.g. traits). In the context of Song of Blade and Sorcery, this equation works because units only have two unique values. However, in more complex wargames such as Warhammer [10] where units have numerous attributes and traits, it quickly becomes unfeasible to have a simple equation to calculate the point costs for every unit in a wargame.

B. Artificial Intelligence in Game Balancing

The use of artificial intelligence in game balancing has been of interest for a long time in academia, with plenty of research on the topic [11]–[13]. However, this research usually focuses on using an agent to modify set parameters in the game and then evaluating how they change the win rate. For example, when describing their integrated balancing framework [14], the authors identified which parameters need to be balanced through expert knowledge gained from multiple play tests. In a wargame with lots of units and variability, it may not be feasible to identify which units are unbalanced, or the parameters that need to be tweaked, therefore, an automated process for calculating unit effectiveness is required.

Previous work has been done on evaluating the balance of alternative sets of rules for chess. [15]. One of the ways in which it achieves this is by working out the values of chess pieces and how they change depending on the variant. The results are interesting, as they show that unit costs can be estimated. However, chess is a symmetrical game with low variance, and thus calculating costs in a wargame will be a much harder task. In addition to this, there has been work on using a NEAT neural network to create weightings for units to include in enemy waves in Tower Defence games. [16] This is very interesting, however, these weights depend on which towers are currently in play, which is hard to translate to point costs as they are independent of any other factor.

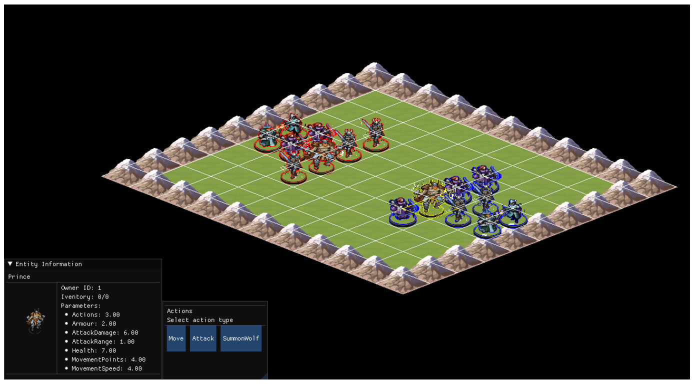
Another area of interest is M. Stanescu's work on using the Lanchester Laws to predict battle outcomes [17]. These laws serve as a way of estimating which team will win a battle based on their strength and unit numbers, with the Square Law stating that the numbers of units in an army are more influential than strength. These work well in the strategy layers of games as a factor of whether to engage in a battle; however, our work takes place in the tactical layer, where the assumptions used in the laws make it too weak to use as a factor in consideration for predicting outcome.

III. Framework

A. Stratega

Stratega[1] is a recently developed general framework for turn-based and real-time strategy games. This framework allows the user to define new strategy games in YAML format. These definitions include multiple aspects of modern strategy games, such as complex rules, different winning conditions, terrain types, customisation of units and buildings, intricate actions, technology trees, and build orders [18]. Stratega incorporates a forward model, which permits, during the decision making process of their agents, advancing the game state by supplying actions. This allows the implementation of Statistical Forward Planning agents, such as Monte Carlo Tree Search and Rolling Horizon Evolutionary Algorithms [19].

[1]https://github.com/GAIGResearch/Stratega



```
Prince
Owner ID: 1
Iventory: 0/0
Parameters:
 • Actions: 3.00
 • Armour: 2.00
 • AttackDamage: 6.00
 • AttackRange: 1.00
 • Health: 7.00
 • MovementPoints: 4.00
 • MovementSpeed: 4.00
```

Fig. 1. Wizard Wars being simulated in Stratega.

B. MCTS

Monte-Carlo Tree Search (MCTS) [20] is a highly selective best-first search method that builds an asymmetric search tree across several iterations. In each of them, the tree balances between the exploitation of the most promising actions and the exploration of moves that have been tried less often. This balancing is managed by its tree policy, for instance UCB1 [21], which balances the exploitation term $Q(s,a)$ (the average of rewards observed after applying action a in state s) and the exploration term $C \times \sqrt{\frac{\ln N}{N(s,a)}}$ ($N(s)$ is the number of times state s has been visited, $N(s,a)$ is the number of times a has been applied to s). C is a parameter, the exploration constant, which determines how much weight should be put on the exploration component.

The algorithm, in its default version, adds a node to the tree for every iteration, from which it performs a uniformly random Monte Carlo simulation until game end or a predetermined depth is reached. MCTS has been applied to multiple domains [20] [22], including strategy games [23], and has shown its use in complex and dynamic environments due to its capacity to quickly re-plan and adapt to unseen states.

C. Wizard Wars

Wizard Wars is a simple war game designed in Stratega as a test for the algorithm proposed in this paper. The game is played automatically by MCTS agents. It uses a turn-based combat system without stochastic elements. Units take actions to perform moves in the game. All units have a move and attack action; certain units have additional actions. Units also have action points, with each non-move action taken costing one of these points, which are refreshed at the beginning of each turn. Each team has one king unit. To win the game, they have to kill the opponents king, while losing if their kings dies.

Figure 1 shows a game of Wizard Wars being played. The grid layout used in the game is visible, as are the units. Grid stacking is not possible, with only one unit allowed per tile. The move action allows the unit to change position on the grid. They can move as many tiles non-diagonally as they have movement points, which are refreshed each turn.

Race and Attributes	**Unit Types**			
Dwarf	**Melee**	**Ranged**	**Support**	**Elite**
Health	4	1	4	10
Action Points	2	1	2	3
Movement Points	2	1	2	2
Attack Damage	3	5	2	4
Attack Range	1	4	1	1
Armour	2	0	2	4
Abilities			Rf	Rf
Elf	**Melee**	**Ranged**	**Support**	**Elite**
Health	4	4	3	7
Action Points	3	2	2	3
Movement Points	3	5	2	4
Attack Damage	3	3	2	6
Attack Range	2	2	2	1
Armour	1	0	0	2
Abilities	AP		Hl	AP
Orc	**Melee**	**Ranged**	**Support**	**Elite**
Health	6	4	3	8
Action Points	2	2	2	3
Movement Points	3	2	2	3
Attack Damage	4	3	2	5
Attack Range	1	2	2	1
Armour	0	0	0	2
Abilities	WC	Da	Cs	WC

TABLE I
AN ENUMERATION OF THE ATTRIBUTES OF EACH ARMY'S UNITS.

Attacking in Wizard Wars is quite simple: units use their attack action on an opposing unit. The damage dealt is then calculated as $D - A$, where D is the unit's attack damage and A is the opponent's armour. The resulting value (if positive) is then subtracted from the opponent's health. If this health drops to 0 or below, the unit is removed from play.

There are three armies in the game: Elves, Dwarves, and Orcs. Each of these armies contains four units: *Melee units*, which focus on being the front line in battle; *Ranged units*, which provide damage from a safe distance; *Support units*, who are weak in combat but provide useful abilities for the army; and *Elite units*, extra powerful entities with their stats representing their armies' style.

Table I contains the attributes and abilities for the units of each army. Each army has a specific style, which is reflected in their units. The Dwarves have slower movement, but more health and armour which allow them to employ a defensive style. The Elves have ranged units with great speed to manoeuvre around the battlefield. The Orcs have high attack damage to attack the opponent head on. These values were all chosen by hand, without any play testing done. Outlined below are the definitions of the abilities displayed in the table.

- Rf - Reinforces a chosen unit by giving them +2 armour (3 tile range) (3 turn cool down).
- AP - Unit's attacks ignore armour (passive ability)
- Hl - Heals a chosen unit, giving them +3 health (3 tile range) (3 turn cooldown).
- WC - Gives the unit +1 attack damage and movement points for 2 turns (self-only) (3 turn cooldown).
- DA - Modifies the attack damage and range of a unit by -1 for 2 turns. Units with 0 attack range cannot attack (2 tile range) (3 turn cooldown).

- Cs - Curses an enemy causing them to lose 1 health per turn for 4 turns (3 tile range) (3 turn cooldown).

These asymmetric qualities between the armies lead to diverse strategies being possible, with no clear way of seeing which armies or individual units will be particularly dominant. This provides a good test bed for our algorithm.

IV. Method

To demonstrate how these point costs can be accurately predicted to create more balanced wargames, we describe our proposed point discovery methods and how they can be integrated with the framework.

A. Point discovery methods

We define the problem of discovering the point cost as the problem of identifying the weight each unit contributes to victory, assuming that the relationship is linear over all possible opponent armies. For each individual opponent army list $m \in M$, we thus have Equation 1:

$$\begin{gathered}\left(w_0^1 x_{i0}^1 + w_1^1 x_{i1}^1 + w_2^1 x_{i2}^1 + ... + w_n^1 x_{in}^1\right) - \\ \left(w_0^2 x_{i0}^2 + w_1^2 x_{i1}^2 + w_2^2 x_{i2}^2 + ... + w_m^2 x_{im}^2\right) = \hat{y_{in}}\end{gathered} \tag{1}$$

where the weights w_l^j represent the point cost of unit l for the army j. The inputs x are the number of units of each type (up to n types of Player 1 and up to m types for Player 2). Each battle fought results in the data sample i. The sets $X = \{X_0, ..., X_{n-1}\}$ and $Y = \{y_0, .., y_{n-1}\}$ are our inputs and outcomes, respectively, and form our dataset $\mathcal{D}$. Note that game outcomes can be 1, 0, and 0.5 for win, loss and draw respectively, but we are not binding the linear equation at all.

The actual point cost is the expected value of the weights of an army against all armies M, $\mathbb{E}_M[w]$. We proceed to calculate this quantity by having multiple armies play against each other using the Wizard Wars rule set implemented in Stratega (with MCTS agents) and getting the empirical average. Note that though the elements of the set Y are all either 0, 0.5 or 1 (i.e., loss/draw/win), we do not use logistic regression, but different forms of linear regression, to do the calculations, as the coefficients of logistic regression are somewhat harder to interpret (being log-odds). Note, however, that arguably logistic regression might prove more accurate/correct for the problem we are setting to solve, but we stick with linear regression in the paper because of ease of interpretability.

B. Least squares / bounded optimisation

There are multiple ways to calculate the weights w from the data. In our first method, which we label *bounded least squares*, we take all the games played between two armies and optimise the weights in a straightforward linear optimisation setting, as implemented in SciPy (https://scipy.org/). The loss function is the standard mean squared error, in our case $mse(X,Y) = \sum_{i=1}^{|X|}(\hat{y}_i - y_i)^2$. We then solve as in Equation 2. Note that we set the optimisation bounds such that no unit from the first army can have a negative point cost and no unit in the opponent army should have a positive point cost.

		Player 1	
		army list 1	army list 2
Player 2	army list 1	$(R_1^1, -R_1^1)$	$(R_1^2, -R_1^2)$
	army list 2	$(R_2^1, -R_2^1)$	$(R_2^2, -R_2^2)$

TABLE II
An example of the defined zero sum asymmetric formal game. Each game results in rewards $(R, -R)$ for each player.

Intuitively, this binds the solution to only positive unit costs for each army.

$$\begin{gathered} minimise\ 0.5 mse(X,Y) \\ subject\ to\ w^1 \geq 0, w^2 \leq 0, \end{gathered} \tag{2}$$

In Equation 2, w^1 refers to the weights of the first army, while w^2 of the second army.

C. Elastic net CV

Our second effort involves solving a constrained linear optimisation problem, this time without setting bounds to w, but penalising high L1 and L2 norms. This is generally known as an elastic net and aims to achieve both sparsity in w and keep the w low. We discover the optimal α and $l1_{ratio}$ in Equation 3 through cross-validation, and use scikit-learn's implementation [24].

$$\begin{aligned} minimise\ 1/(2 \cdot n_{samples}) \cdot mse(X,Y) \\ +\alpha \cdot l1_{ratio} \cdot ||w||_1 \\ +0.5 \cdot \alpha \cdot (1 - l1_{ratio}) \cdot ||w||_2^2 \end{aligned} \tag{3}$$

As in *bounded least squares*, there is the danger that certain units will have negative weights w. We solve this by either artificially setting the minimum of each w to 0 for each army-list combination (which we term *bounded elastic net*) or after taking the average for all armies (which we call *elastic net*).

D. Normal form game / linear programming

The problem, as defined above, encodes the implicit idea that both players will more or less choose their army lists randomly. This is, however, likely not the case. Both players will try to select an army list that best corresponds to their opponent losing and them winning. As such, one can model the whole procedure of choosing army lists as a zero sum, normal form, asymmetric game. Each possible army list is defined as an action/strategy, with each army corresponding to each player. This formal game can be seen in Table II

The formal game is then solved using a standard linear programming solver, as implemented in OpenSpiel [25]. This results in a mixed strategy profile for each player. Given that not all possible games have necessarily been played (it is computationally too expensive to play all 65536 possible games in a reasonable amount of time), we eliminated all impossible army lists that have not been compared against. We then sampled from the mixed policy all the army lists that have not been dominated (i.e. the support of the army has not

TABLE III
THE WEIGHTING OF EACH FACTOR PER HEURISTIC (FROM 0-1)

Factor	Defensive	Balanced	Aggressive
Player Army Size	1	0.5	0.25
Player Army HP	1	0.5	0.25
Player King HP	1	0.5	0.25
Opponent Army Size	0.25	0.5	1
Opponent Army Health	0.25	0.5	1
Opponent Kings HP	0.25	0.5	1
Units in Range of Support	0.5	0.5	0.5
Mean Distance from Opp King	0.25	0.5	1

been zero) to generate hypothetical games based on army lists the players would have used. Note that given that not all army list combinations may have taken place in the actual data, our solution should be considered an approximation. We term the army lists that resulted from this process as *equilibrated*.

E. Algorithms for Unit Cost Prediction

The weights w produced were then used as the point values for the units. Six different methods were used, *least squares* (LS), *elastic net* (E), and *bounded elastic net* (BE) were the three baseline methods we used. We also used *equilibrated* data on these three methods (ELS, EE and EBE, respectively).

With these point costs calculated, new simulations were run for each method used. The army lists chosen for these simulations used a point budget system similar to ones used in aforementioned games like Warhammer. Each army gets a point budget, with any unit added to the army list taking point away from the budget based on its cost until the budget is exhausted and new units cannot be added.

V. EXPERIMENTAL RESULTS

This section will discuss how the experiments were setup, provide an analysis of the point costs predicted by each method, and how using these predicted costs to generate army-lists impacts the balance of the game by examining win-rates.

A. Heuristics

As our method uses data provided by simulations of Wizard Wars, it is important that it represents a variety of play-styles, as otherwise there is the risk that it balances towards only one strategy. To account for this, we include multiple heuristics that the MCTS agent can use.

To this end, we created a base heuristic which evaluates several factors to decide how advantageous the chosen game state is. Then, we differentiated three heuristics representing different play-styles (Aggressive, Balanced and Defensive). The results of the heuristics are normalised for use by MCTS, so they will be between 0 (worst state) and 1 (best state).

The factors included in the heuristic evaluation were:

- The current size of the player / opponent army.
- How much current HP the player's / opponent's army has.
- The current HP of the player / opponent's king.
- How many player units are in range of a support.
- The mean distance of the player's non-supportive units from the opponent's kings.

Table III shows the weights for each of these factors for each one of the heuristics. The factors considered are normalised in $[0, 1]$, and multiplied by their weights in a linear combination fashion to produce the reward for the MCTS agent.

In addition to the heuristics, the MCTS agents also used an opponent model to predict the actions of the opposing player. This model is based on the opponent trying to attack the player's lowest health unit, with it's units moving towards it and attacking if they are in range.

B. Experimental Setup

For each run of experiments, two runs were computed: the first provided training data for the methods, the second to validate the effectiveness of each linear regression method.

Each game was played on a simple rectangle grid-based map of 5×13 squares. No special terrain was included, and the units starting positions were randomly allocated within reason (e.g. melee units will be on the front line, ranged units at the back). For the training run, units were randomly chosen for inclusion in the army list, with the constraint of having 0 - 3 units of each type in the list.

For the validation run, the points predicted by each method were used to choose which units to include in the army list. Each army had a budget of 60 points; this value was chosen to keep the army sizes consistent with the size of the map. Units were added to the list by sampling with replacement. Units with a point cost of 0 or less were not able to be chosen (since it meant that the method considered them useless), and up to 5 units of each type can be included in the list. The army list generator will attempt to use all of its budget; however, there are chances where this is not possible, and instead it will try to use as much of its budget as possible.

Both players used Stratega's built-in MCTS agent, with the aforementioned heuristic and opponent model. It had a computational budget of 300 ms per move, a rollout length of 20, and an exploration constant of $\sqrt{2}$. These values were tuned through trial and error. Each turn had a time limit of 10 seconds and a total turn limit of 50 turns. After 50 turns, the game would end and count as a draw for each player.

For the results, we have defined a metric to gauge the accuracy of the predicted point costs. We refer to this as the Balance Loss, abbreviated as BL, which can be defined as the standard deviation of the win rates for each matchup. A value of 0 represents a perfectly balanced game where each match-up has 50-50 odds of winning, and increasing values increased balance inequality for our scenario (we are balancing towards 50-50, however, for future work we would like to try use an intransitive superiority balancing style).

C. Training Run Configuration

To provide training data for the methods, 1,200 (2,400 if against the same army e.g, Orc vs. Orc) army list pairs were generated for each match-up (Dwarf vs. Elf, Elf vs. Dwarf, Orc vs. Dwarf, Dwarf vs. Orc, Orc vs. Elf, Elf vs. Orc,

Race / Method	Unit Types			
Dwarf	**Melee**	**Ranged**	**Support**	**Elite**
Least Squares (LS)	7	12	5	13
Elastic-Net (E)	6	12	4	13
Bounded Elastic-Net (BE)	6	12	4	13
Equil. Least Squares (ELS)	2	7	3	10
Equil. Elastic-Net (EE)	2	5	3	7
Equil. Bounded Elastic-Net (EBE)	2	5	3	7
Elf	**Melee**	**Ranged**	**Support**	**Elite**
Least Squares (LS)	10	10	6	17
Elastic-Net (E)	10	9	6	17
Bounded Elastic-Net (BE)	10	9	6	17
Equil. Least Squares (ELS)	5	4	4	11
Equil. Elastic-Net (EE)	5	4	3	9
Equil. Bounded Elastic-Net (EBE)	5	4	3	9
Orc	**Melee**	**Ranged**	**Support**	**Elite**
Least Squares (LS)	10	12	6	16
Elastic-Net (E)	9	12	6	16
Bounded Elastic-Net (BE)	9	12	6	16
Equil. Least Squares (ELS)	4	6	6	11
Equil. Elastic-Net (EE)	3	5	4	9
Equil. Bounded Elastic-Net (EBE)	3	5	4	9

TABLE IV
THE POINT COSTS OF EACH UNIT FOR THE THREE ARMIES AS PREDICTED BY THE SIX LINEAR REGRESSION METHODS (EQUIL. = EQUILIBRATED).

Dwarf vs. Dwarf, Elf vs. Elf, Orc vs. Orc), giving in total 14,400 army lists per run. Each of these pairs was played by every heuristic combination (e.g., Aggressive vs. Balanced), resulting in 172,800 games per run. Each run took around 24 hours to complete in a High-Performance Computer.

D. Validation Run Configuration

For the validation run, the amount of games played was the same; however, this was now divided by which method's point costs were used to generate the army list pair, resulting in 200 pairs per combination of armies, heuristics, and method.

E. Point-Cost Prediction

Table IV shows the point costs predicted by each method, and we can see that some trends emerge. In terms of the overall points predicted, the values chosen by the methods make intuitive sense: the elite units have the highest point cost for each army, which is sensible as they are the most numerically powerful. The melee and ranged units have the next highest point costs, with the stronger unit varying per army. Finally, the support units have by far the lowest predicted point costs, which could suggest that their abilities are not as useful in the game compared to the raw stats of the other units.

In terms of the methods themselves, we can see that there are three groupings of points. All three of the basic non-equilibrated methods seem to converge to similar values, with only a couple of points differing between them. This may suggest that with the large amount of training data provided, the different ways these methods come to point costs have a negligible difference. This may also present a ceiling for these basic methods, with additions needed to provide better results.

This contrasts to the equilibrated methods, where the least-squares and elastic-net methods predict different point costs; this is most likely due to the equilibration process. It is also

TABLE V
THE FIRST ARMY'S WIN-RATE AND STANDARD ERROR FOR EACH METHOD AND THE BALANCE LOSS (BL).
E VS D - ELF VS DWARF E VS O - ELF VS ORC O VS D - ORC VS ELF

Method	EvD	EvO	OvD	BL
Uncosted	53% (0.003)	39% (0.003)	59% (0.003)	9.03
LS	46% (0.006)	35% (0.006)	54% (0.006)	10.13
E	44% (0.006)	34% (0.006)	52% (0.006)	10.58
BE	43% (0.006)	35% (0.006)	52% (0.006)	10.18
ELS	48% (0.006)	48% (0.006)	43% (0.005)	4.86
EE	40% (0.005)	42% (0.006)	41% (0.005)	9.65
EBE	40% (0.005)	41% (0.006)	42% (0.005)	9.84

TABLE VI
THE WIN-RATE'S AND STANDARD ERROR VS. THE SAME FACTION
D V D - DWARF VS DWARF EVE - ELF VS ELF O V O - ORC VS ORC.

Method	DvD	EvE	OvO
Uncosted	61% (0.002)	56% (0.003)	55% (0.003)
LS	62% (0.006)	60% (0.006)	54% (0.007)
E	62% (0.005)	61% (0.006)	54% (0.007)
BE	63% (0.005)	62% (0.006)	54% (0.007)
ELS	58% (0.005)	60% (0.006)	58% (0.006)
EE	55% (0.006)	57% (0.006)	57% (0.005)
EBE	54% (0.006)	59% (0.006)	57% (0.005)

interesting to see that there were no different predicted point costs between the bounded / non-bounded elastic net pairs; this suggests that none of the units had a point cost less than zero; from this we can infer that every unit was considered at least somewhat useful by the methods.

In terms of the per army point costs, there are some interesting predictions. For Dwarves, there is only a 1-point difference between its elite and ranged unit in the basic methods, compared to the other two armies which have sizeable differences between the elite and non-elite units. This suggests that the Dwarven elite unit has less influence on victory compared to the elite units of the other armies. Looking at the stats, we can see that it only has an attack damage of 4, lower than both the other elites and the Dwarven ranged unit. This could indicate that attack damage is a particularly valuable stat in Wizard Wars. It is also interesting to see that, with the exception of the equilibrated least squares (ELS) method, the Dwarves have lower point costs for every unit except the ranged compared with other armies, from this it may be inferred that these methods see the Dwarven army as overall less strong than the other counterparts.

Compared to the Dwarves, the basic methods determine similar point costs across the board for Elves and Orcs, with only minor differences between them. Comparing them directly to each other results in only a few differences in points per unit type, this could indicate that these methods see these two armies as being similar in power level.

F. Win-Rates

From Table V we can see that the balance loss of the uncosted simulations (i.e. The Training Run) is 9.03, which provides a baseline to compare our results to. Considering the fact that no play-testing was involved in the creation of Wizard Wars, this is a relatively low balance loss, and suggests that in the original implementation the game is relatively balanced.

Out of the 6 methods, Equilibrated Least Squares manages achieve a more balanced game than the baseline. It manages to reduce the balance loss by almost half to a value of 4.86. Looking at the win-rates for this method, we can see that the Elf vs. Dwarf and Elf vs. Orc match-ups are extremely close to the ideal win-rate of 50%, with only the Orc vs. Dwarf match-up being relatively unbalanced at 43% in favour of the Dwarves. However, it is still an improvement compared to the baseline of 59%. None of the other methods manage to do achieve better balance than the baseline, however they all manage to get relatively close. Looking at the win-rates of the methods and the point costs predicted, we can see some patterns which may indicate why this has occurred.

For the basic methods from the win-rates we can see that the loss of balance is most likely caused by the elves under-performing relative to the other armies. As mentioned before, these methods predicted almost identical points for the Elves and Orcs, which contrasts to the baseline win rate of the Elf vs. Orc of just 39% in favour of the Orcs. This suggests that these methods predicted the Elven units to be much stronger than they actually are, which is also reinforced by them losing against the Dwarves when it was a previously winning match-up for them. On the other hand, the Orc vs Dwarf match-up becomes more balanced with these methods, which may indicate that the points predicted for the Dwarves and Orcs are much more accurate.

The Equilibrated Elastic-Net methods perform better than the basic methods; however, they still under-perform compared to the baseline. Looking at the win-rates we can see that the Dwarves perform much better compared to the baseline, winning both their match-ups where they lost both before. This is most likely the biggest loss of balance for these methods, and looking at the point costs we can see that they have lower point costs compared to the other armies, which suggests that the methods are underestimating the performance of the Dwarven units. The Elves perform slightly better against the orcs than the baseline, with their slightly lower point costs reflecting this, however, they are still not optimal.

In addition to the win-rates of the army playing against different armies, we also collected results on the army playing against itself (with different choices of units). Table VI shows us these win rates. Immediately we can see that for all factions, Player 1 leads against Player 2, this informs us that there is most likely a first-turn bias in the game. In addition, none of the methods manage to mitigate this first turn advantage, which suggests that point costs alone might not be able to eliminate this bias, thus changes to the rules of the game might be needed to address this issue.

G. *Biased Units Experiment*

In addition to our main experiment, we also conducted a biased experiment in which we modified the values of one unit (Dwarven Melee) to be overpowered and for another (Elven Ranged) to be underpowered. This experiment is aimed at testing the ability of our methods to identify heavily unbalanced units. For this experiment, we only used the Balanced heuristic and a set of 600 / 100 (Training / Validation) army list pairs per game match-up, to total 7200 pairs per run.

Race / Method	**Unit Types**			
Dwarf	**Melee**	**Ranged**	**Support**	**Elite**
Least Squares (LS)	22	9	4	11
Elastic-Net (E)	21	8	3	10
Bounded Elastic-Net (BE)	21	8	3	10
Equil. Least Squares (ELS)	18	8	5	8
Equil. Elastic-Net (EE)	15	6	4	7
Equil. Bounded Elastic-Net (EBE)	15	6	4	7
Elf	**Melee**	**Ranged**	**Support**	**Elite**
Least Squares (LS)	10	4	5	16
Elastic-Net (E)	9	4	5	15
Bounded Elastic-Net (BE)	9	4	5	15
Equil. Least Squares (ELS)	6	3	4	8
Equil. Elastic-Net (EE)	5	3	3	8
Equil. Bounded Elastic-Net (EBE)	5	3	3	8
Orc	**Melee**	**Ranged**	**Support**	**Elite**
Least Squares (LS)	8	11	6	14
Elastic-Net (E)	8	10	5	14
Bounded Elastic-Net (BE)	8	10	5	14
Equil. Least Squares (ELS)	6	8	3	9
Equil. Elastic-Net (EE)	4	6	2	7
Equil. Bounded Elastic-Net (EBE)	4	6	2	7

TABLE VII
THE PREDICTED POINT COSTS FOR THE BIASED EXPERIMENTS.

TABLE VIII
THE WIN-RATES AND STANDARD ERRORS FOR THE BIASED EXPERIMENTS.

Method	EvD	EvO	OvD	BL
Uncosted	22% (0.008)	34% (0.008)	28% (0.008)	24.55
LS	21% (0.016)	34% (0.019)	30% (0.02)	24.51
E	18% (0.016)	30% (0.019)	24% (0.018)	28.92
BE	19% (0.017)	32% (0.02)	23% (0.018)	28.42
ELS	26% (0.018)	43% (0.02)	32% (0.02)	19.74
EE	21% (0.018)	33% (0.018)	31% (0.02)	24.28
EBE	22% (0.018)	35% (0.018)	32% (0.02)	23.17

Table VII shows us the predicted point costs per method. We can immediately see that the Dwarven Melee unit has much higher predicted point costs compared to the main experiment, gaining a higher point cost than any of the Elite units. This suggests that the methods find this unit to be the most powerful unit in the game. The Elven Ranged unit also has lower point predicted point costs compared to the main experiment, as well as having similar costs to the support units. However, its attributes are objectively less powerful than them (while still being playable), which could suggest a floor of how costs can be predicted with these methods.

Table VIII shows how these point costs have impacted the win rates compared to the baseline. ELS again performs the best, managing to reduce the Balance Loss to 19.74, with each match-up becoming more balanced. However, the game is still quite unbalanced, which could suggest that while it is able to identify these deliberately unbalanced units, deeper changes in the rules could be required to completely balance the game.

VI. Conclusions and Future work

It is our understanding that prior to this work there was no automated method of predicting unit costs in complex wargames. We have proposed a game agnostic algorithm that will automatically calculate the value of units costs using just data from game results. The initial results of our work are promising, with the Equilibrated Least Squares algorithm almost halving the balance loss compared to the baseline in the main experiment, and with the deliberately unbalanced units being detected in the biased unit experiment. It is worth framing how this method could work in more complex wargames: with their larger number of armies and units, the fact that our method does not require domain knowledge is an advantage. However, the large number of sample games required can be considered a drawback, and to mitigate this a way of estimating game results may be required.

We have identified two avenues through which we can improve our approach for predicting unit point costs. The first looks at improving the method used in the algorithm. The equilibrated methods outperformed the basic methods; however, it is worth seeing how they could be further improved, and while for these methods we assume the relationship between the win-rate and units is linear, it may be interesting to see how using a form of nonlinear regression (e.g., from simpler algorithms such as non-linear least squares to other approaches like genetic programming) may change the results. The second way we can see this is to expand on the game data we model. Right now, the algorithm only predicts independent unit costs; however, as mentioned previously, there are many factors that modify the usefulness of a unit in battle. Therefore, it might be lucrative to investigate how taking the factors (e.g. a support unit is more useful when paired with a powerful unit for it to help) into account may help to predict more accurate values.

Another possibility for future work is changing the balancing target. For this paper we were balancing towards every army having a 50% win-rate, however, it would be interesting to try an intransitive superiority approach, where choosing an army based on game factors is vital for victory.

We have also identified ways in which this algorithm could be applied in the future. The first is to use it on other types of games. Wizard Wars is a wargame, and it may be worth seeing how our algorithm performs in games of other genres (e.g. Trading Card Games). Also, the points predicted by our algorithm can serve as a proxy for the power level of a unit; therefore, it may be possible to use it as a diagnostic tool to see which units are unbalanced, and feeding this information into an automatic game balancing algorithm (e.g. NTBEA [26]) to create a balancing system that would modify the property values of unbalanced units, forgoing the requirement of a human providing expert knowledge.

VII. Acknowledgments

For the purpose of open access, the author(s) has applied a Creative Commons Attribution (CC BY) license to any Accepted Manuscript version arising. This work was supported by the EPSRC IGGI CDT (EP/S022325/1) and utilised QMUL's Apocrita HPC facility.

References

[1] J. F. Dunnigan, *The Complete Wargames Handbook*. 1997.

[2] A. Becker and D. Görlich, "What is game balancing?-an examination of concepts," *ParadigmPlus*, vol. 1, no. 1, pp. 22–41, 2020.

[3] M. Preuss, T. Pfeiffer, V. Volz, and N. Pflanzl, "Integrated balancing of an rts game: Case study and toolbox refinement," in *2018 IEEE Conference on Computational Intelligence and Games*, 2018.

[4] *Core Book*. Warhammer 40,000, Games Workshop, Limited, 2020.

[5] R. Priestley and J. Lambshead, *Tabletop Wargames: A Designers' and Writers' Handbook*. Pen & Sword Books Limited, 2016.

[6] A. Sfiligoi, J. Hartman, J. Hartman, and J. McBride, *Song of Blades and Heroes - Revised Edition*. CreateSpace Ind. Publishing Platform, 2012.

[7] V. Volz, G. Rudolph, and B. Naujoks, "Demonstrating the feasibility of automatic game balancing," in *Proc. of the Genetic and Evolutionary Computation Conference*, 2016.

[8] C. S. Roberts, *Tactics*. Avalon Game Company, 1954.

[9] A. Lamorisse, *Risk*. Parker Brothers, 1959.

[10] M. Ward, *Warhammer Rulebook. 8th Edition*. Games Workshop, 2009.

[11] H. Chen, Y. Mori, and I. Matsuba, "Solving the Balance Problem of On-Line Role-Playing Games Using Evolutionary Algorithms," *Journal of Software Eng. and Applications*, vol. 05, no. 08, pp. 574–582, 2012.

[12] M. Moroşan, *Automating Game-design and Game-agent Balancing through Computational Intelligence*. PhD thesis, Univ. of Essex, 2019.

[13] F. de Mesentier Silva, R. Canaan, S. Lee, M. C. Fontaine, J. Togelius, and A. K. Hoover, "Evolving the hearthstone meta," *CoRR*, vol. abs/1907.01623, 2019.

[14] M. Beyer, A. Agureikin, A. Anokhin, C. Laenger, F. Nolte, J. Winterberg, M. Renka, M. Rieger, N. Pflanzl, M. Preuss, and V. Volz, "An integrated process for game balancing," in *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, pp. 1–8, Sept. 2016.

[15] N. Tomašev, U. Paquet, D. Hassabis, and V. Kramnik, "Assessing game balance with alphazero: Exploring alternative rule sets in chess," *arXiv preprint arXiv:2009.04374*, 2020.

[16] D. Hind and C. Harvey, "A neat approach to wave generation in tower defense games," in *2022 International Conference on Interactive Media, Smart Systems and Emerging Technologies (IMET)*, pp. 1–8, 2022.

[17] M. Stanescu, N. Barriga, and M. Buro, "Using lanchester attrition laws for combat prediction in starcraft," 2015.

[18] A. Dockhorn, J. H. Grueso, D. Jeurissen, and D. P. Liebana, "Stratega: A general strategy games framework.," in *AIIDE Workshops*, 2020.

[19] A. Dockhorn, D. Perez-Liebana, *et al.*, "Game state and action abstracting monte carlo tree search for general strategy game-playing," in *2021 IEEE Conference on Games (CoG)*, pp. 1–8, IEEE, 2021.

[20] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A Survey of Monte Carlo Tree Search Methods," *IEEE Transactions on Comp. Intelligence and AI in Games*, vol. 4, no. 1, pp. 1–43, 2012.

[21] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *European Conf. on machine learning*, pp. 282–293, Springer, 2006.

[22] M. Swiechowski, K. Godlewski, B. Sawicki, and J. Mandziuk, "Monte carlo tree search: a review of recent modifications and applications," *Artificial Intelligence Review*, 2022.

[23] L. Xu, J. Hurtado-Grueso, D. Jeurissen, D. P. Liebana, and A. Dockhorn, "Elastic monte carlo tree search with state abstraction for strategy game playing," in *IEEE Conference on Games (CoG)*, 2022.

[24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[25] M. Lanctot, E. Lockhart, J.-B. Lespiau, V. Zambaldi, S. Upadhyay, J. Pérolat, S. Srinivasan, F. Timbers, K. Tuyls, S. Omidshafiei, *et al.*, "Openspiel: A framework for reinforcement learning in games," *arXiv preprint arXiv:1908.09453*, 2019.

[26] K. Kunanusont, R. D. Gaina, J. Liu, D. Perez-Liebana, and S. M. Lucas, "The n-tuple bandit evolutionary algorithm for automatic game improvement," in *IEEE Congress on Evolutionary Computation*, 2017.