Proceedings, The Thirteenth AAAI Conference on
Artificial Intelligence and Interactive Digital Entertainment (AIIDE-17)

Balancing Zero-Sum Games with One Variable per Strategy

Albert Julius Liu and **Steve Marschner**
Department of Computer Science, Cornell University
ajul@cs.cornell.edu, srm@cs.cornell.edu

Abstract

A key challenge in game design is achieving balance between the strategies available to the players. Traditionally this has been done through playtesting, with its difficult requirements of time, labor, and interpretation of results. To make it quicker and easier to balance games, we propose a game-theoretic approach that automatically balances strategies based on a mathematical model of the game.

Specifically, we model the balance problem as modifying a zero-sum game, using one variable per strategy, so that every strategy has an incentive to be employed. We begin with a special case where these variables affect player payoffs multiplicatively, and show that the simple Sinkhorn-Knopp algorithm can be used to balance the game. We then proceed to analyze the more general case where the variables have a monotonic effect on payoffs, and show that it is amenable to standard optimization methods. We give examples inspired by well-known game series including Pokémon and *Warhammer 40,000*.

1 Introduction

"A game is a series of meaningful choices", as *Civilization* designer Sid Meier is famously quoted (Rollings and Morris 2000). A choice is not very meaningful if it is the correct choice every time, or the wrong choice every time. Overpowered choices will tend to crowd out all others, leading to stale gameplay; underpowered choices may rarely see use in practice, wasting the development resources spent on them. Game designers thus strive to balance the choices in their games so that each can be reasonably employed.

While human playtesting is the ultimate arbiter of balance and player enjoyment, dedicated human playtesting is time- and labor-intensive. Analytics techniques (Andersen et al. 2011a; 2011b; 2012; Lomas et al. 2013) can allow designers to measure the impact of decisions on balance and other design objectives using large-scale experiments with actual players. However, in this case the game's playtesters are also its audience and customers, and therefore expect to enjoy the game while they are playing it. Therefore designers cannot only be concerned with the final state of a game's balance, but also the speed at which balance is achieved and even the initial balance.

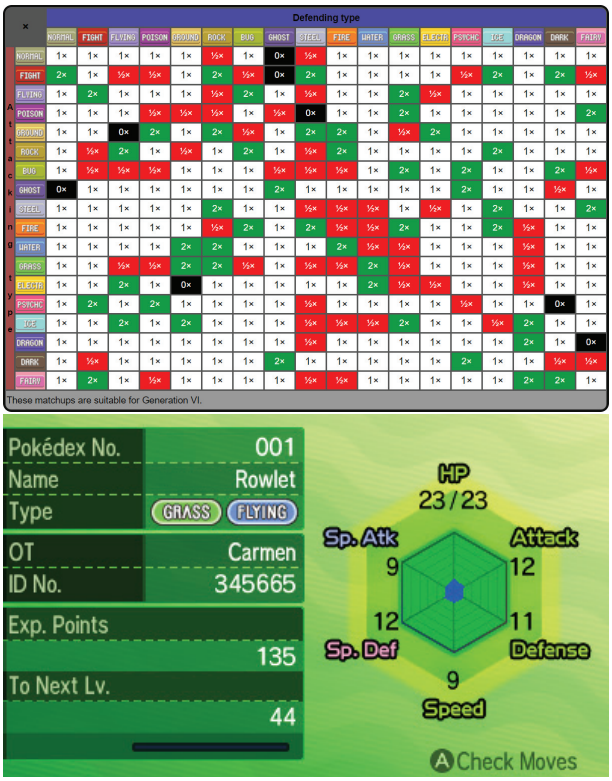

Figure 1: **Top:** The Pokémon type chart. Damage done by an attack is multiplied by a factor depending on the type of the attack and the type(s) of the defending Pokémon. **Bottom:** Example Pokémon statistics screen. In addition to the type multiplier, damage is also multiplied and divided by the (special) attack and defense values of the Pokémon. (Bulbapedia 2017a; 2017b) We consider the problem of setting balanced values for these statistics in a simplified version of the game.

An alternative to human playtesting is to use AI players. AI techniques such as Monte Carlo Tree Search (Chaslot et al. 2008; Browne et al. 2012) are capable of achieving a high level of skill and are adaptable to a wide variety of games, a recent high-profile example being AlphaGo (Silver et al. 2016). AI players have also been used to generate and bal-

ance games, often using an evolutionary algorithm as the "outer loop" (Browne 2008; Mahlmann, Togelius, and Yannakakis 2012).

Another approach, and the one we use in this paper, is to make a game-theoretic analysis using provably optimal agents. While such analysis cannot be applied to as broad a class of games as general AI techniques, it offers lower computational costs and more precisely interpretable solution definitions.

Zero-sum games (in the game theory sense) are a natural model for games (in the entertainment sense) which pit two players against each other. In some cases the entire work may be modeled by a zero-sum game, such as in the case of Rock, Paper, Scissors—though even this simple game offers deep enough dynamics to support both human and computer tournaments (Knoll, Lu, and Burdge 2017). Jaffe (2012) investigated the effect of play restrictions on optimal agents in a two-stage zero-sum game. Even more complex works may include zero-sum components; for example, Jaffe (2013) and Tavares (2016) examined the metagame of choosing characters in a fighting game and choosing strategies in a RTS game respectively.

It is well-known that the Nash equilibrium of a zero-sum game can be found via linear programming (Dantzig 1963). Here we consider an inverse problem: given an initial zero-sum game, and a Nash equilibrium and value (i.e. expected payoff at Nash equilibrium) we would like the game to have, how can we modify the game so it has that Nash equilibrium and value? In a talk Hazard (2010) gave some examples of problems of this type, but did not develop the theory further.

In this paper, we formally define this problem in terms of *handicap functions*, which determine the payoff of the game based on the strategies picked by the two players and a *handicap variable* affecting the overall strength of each strategy. We analyze the special case where the handicap function is simply the ratio of the two handicap variables times some initial value for that matchup, and show that such a game may be balanced using the Sinkhorn-Knopp algorithm, with its associated conditions on the existence and uniqueness of a solution. Then we move on the case where the handicap functions are general monotonic functions, and show that the solution set has properties favorable to numerical optimization. We demonstrate our algorithms on examples inspired by well-known games.

2 Setting

Let us now describe our formalism and how it relates to the mechanics of actual games. We consider the two-player zero-sum game, where each player chooses between a finite number of strategies. This kind of game can be represented by a matrix F, whose elements F_{ij} give the payoff if the row player picks strategy i and the column player picks strategy j. By convention, the row player is attempting to maximize this payoff, and the column player is trying to minimize it (or equivalently, maximize its negation).

Our goal is to modify this game so that it has a particular Nash equilibrium that we desire. With $n \times m$ entries in the matrix but only $n + m$ strategies, this problem is underconstrained if we are allowed to modify the entries of the matrix arbitrarily. Furthermore, the individual pairwise strategy interactions typically reflect other design goals. They may be subject to aesthetic considerations; for example, the Pokémon type chart (Figure 1) contains only four distinct values $0, \frac{1}{2}, 1, 2$, these being determined by intuitive relations between the types (e.g. fire burns grass, so Fire does double damage against Grass). Or the strategies may be different values of a numerical in-game statistic, with the payoff matrix being determined by a pre-existing mathematical equation (for example, hit probability = accuracy - evasion).

Rather, it is more common to adjust the strength of each strategy rather than the payoff matrix directly. Therefore, we assign a handicap variable h_{ri} to each strategy i of the row player (the maximizer); likewise we assign a handicap h_{cj} to each strategy j of the column player (the minimizer). A high handicap indicates that the strategy is too strong at its initial state and should be made weaker. The unit cost (in e.g. money, time...) for employing a strategy is a common balancing knob; an overperforming strategy can be made more expensive, thus allowing "less" of that strategy to be employed. Alternatively, the effectiveness of the strategy could be decreased, such as by reducing the base damage for an attacker strategy, or hit points for a defender strategy.

We then define each element of the payoff matrix of the game using handicap functions whose arguments are the corresponding row and column handicaps:

$$F_{ij} = F_{ij}\left(h_{ri}, h_{cj}\right) \tag{1}$$

This represents how changing the handicap for the strategies i and j changes the relative advantage between the two strategies. For example, if maximizer strategy i's handicap increases, each minimizer strategy j will tend to do better against it, reducing the payoff.

Problem Definition Given...

- $n \times m$ handicap functions that define a payoff matrix as in Equation 1.
- A desired Nash equilibrium with strictly positive strategy weight vectors $\mathbf{w}_r$, $\mathbf{w}_c$ with one element for each of the row and column strategies respectively. As probability distributions, each of these must sum to 1.
- A desired value v of the game (i.e. expected payoff at Nash equilibrium).

...find handicap variable vectors $\mathbf{h}_r$, $\mathbf{h}_c$—again, one element for each of the row and column strategies respectively—such that the zero-sum game defined by the resulting F has that Nash equilibrium and value. Specifically, this means at the desired Nash equilibrium all strategies have expected payoff $\mathbf{p}_r, \mathbf{p}_c$ for their player equal to the desired value of the game (negated for the column player):

$$\begin{aligned} p_{ri} &= \sum_j w_{cj} F_{ij} = v && \forall i \\ p_{cj} &= -\sum_i w_{ri} F_{ij} = -v && \forall j \end{aligned} \tag{2}$$

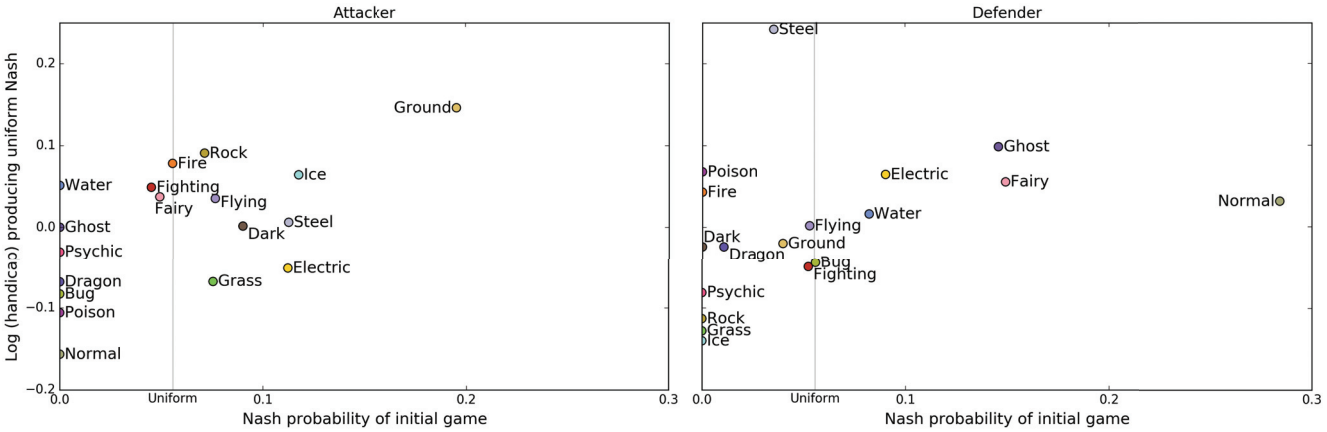

Figure 2: Handicaps that produce a uniform Nash equilibrium for our simplified Pokémon game, plotted against the Nash equilibrium of the original game. The attacker is the maximizer (left plot) and the defender is the minimizer (right plot). The handicap and the Nash equilibrium of the initial game are correlated, following the intuition that over-represented strategies should be weakened ("nerfed"). However, the correlation can be weak. For example, Steel defense gets a high handicap because it is strong against a wide variety of attack types; however, it is only rarely played in the initial Nash equilibrium because it is weak against the most dominant attack type, namely Ground.

We will occasionally concatenate vectors for the rows and columns into a single vector, which we will denote using unsubscripted vectors:

$$\mathbf{h} = \begin{bmatrix} \mathbf{h}_r \\ \mathbf{h}_c \end{bmatrix} \qquad \mathbf{p} = \begin{bmatrix} \mathbf{p}_r \\ \mathbf{p}_c \end{bmatrix} \qquad \mathbf{w} = \begin{bmatrix} \mathbf{w}_r \\ \mathbf{w}_c \end{bmatrix} \tag{3}$$

3 Multiplicative Handicaps

3.1 Example: Pokémon Types

In the *Pokémon* series of games, each attack and each defending Pokémon has an associated type. Consider a simplified version of this game where an attacker chooses the type of the attack and a defender chooses the type of the defending Pokémon, with the attacker trying to maximize the damage dealt and the defender trying to minimize it. Each type has an associated attack and defense statistic, with the damage dealt being proportional to the ratio between the attacking type's attack statistic and the defending type's defense statistic. A multiplier is applied to the damage dealt depending on the type of the attack and the type of the defending Pokémon according to the type chart shown in Figure 1. We wish to balance this game by setting the attack and defense statistics for each type so that the game's Nash equilibrium is the uniform distribution over each player's strategies.

The handicaps that produce this uniform Nash equilibrium are shown in Figure 2 and are contrasted with the Nash equilibrium of the original game. The computation took 4 ms on a single desktop computer.

3.2 Formalization and Algorithm

Let us now formalize this example and present an algorithm for finding the solution:

- Maximizer and minimizer $\Leftrightarrow$ attacker and defender.
- Pokémon types $\Leftrightarrow$ strategies. In general there are n for the maximizer and m for the minimizer.
- Handicap $\Leftrightarrow$ inverse of attack or defense statistic (for each strategy).
- Payoff $\Leftrightarrow$ damage done; we desire that the expected damage done is v.

Finally, the handicap functions are based on an "initial" $n \times m$ nonnegative matrix A, in this example the type chart (Figure 1), according to:

$$F_{ij}\left(h_{ri}, h_{cj}\right) = \frac{h_{cj}}{h_{ri}} A_{ij} \tag{4}$$

that is, the entries of the payoff matrix are the ratio of the corresponding column and row handicaps times some initial payoff.

In this case our problem of finding the handicaps $\mathbf{h}_r, \mathbf{h}_c$ that produce a desired Nash equilibrium is equivalent to finding a scaling of the rows and columns of A such that the row and column sums, weighted by the probabilities of the desired Nash equilibrium, sum to v. Conveniently, there is an existing algorithm due to Sinkhorn and Knopp (1967; 1967) that does just that. It is exceedingly simple:

1. Divide each row by its (weighted) sum.
2. Divide each column by its (weighted) sum.
3. Repeat until convergence.

Sinkhorn and Knopp (1967) showed that in the case where A is square and the weights are uniform, a unique solution exists if and only if A has total support; and that the algorithm converges to said solution. (A matrix has total support if every edge in the bipartite graph defined by its nonzero elements is part of a perfect matching.) Sinkhorn (1967) soon extended the algorithm to positive rectangular matrices with weights.[1]

[1]The same problem and algorithm has found applications as diverse as computer graphics (Solomon et al. 2015), web page ranking (Knight 2008), and voting (Smith 2005).

Chance to wound (out of 6)

Strength \ Toughness	1	2	3	4	5	6	7	8	9	10
1	3	2	1	1	0	0	0	0	0	0
2	4	3	2	1	1	0	0	0	0	0
3	5	4	3	2	1	1	0	0	0	0
4	5	5	4	3	2	1	1	0	0	0
5	5	5	5	4	3	2	1	1	0	0
6	5	5	5	5	4	3	2	1	1	0
7	5	5	5	5	5	4	3	2	1	1
8	5	5	5	5	5	5	4	3	2	1
9	5	5	5	5	5	5	5	4	3	2
10	5	5	5	5	5	5	5	5	4	3

Strength \ Toughness	1	2	3	4	5	6	7	8	9	10
1	3	1	1	1	1	1	1	1	1	1
2	5	3	2	1	1	1	1	1	1	1
3	5	4	3	2	2	1	1	1	1	1
4	5	5	4	3	2	2	2	1	1	1
5	5	5	4	4	3	2	2	2	2	1
6	5	5	5	4	4	3	2	2	2	2
7	5	5	5	4	4	4	3	2	2	2
8	5	5	5	5	4	4	4	3	2	2
9	5	5	5	5	4	4	4	4	3	2
10	5	5	5	5	5	4	4	4	4	3

Payoff matrix and handicaps after balancing

Strength \ Toughness	1	2	3	4	5	6	7	8	9	10	SH
1	3.19	2.65	1.83	2.34	0.00	0.00	0.00	0.00	0.00	0.00	1.00
2	2.34	2.20	2.02	1.29	2.15	0.00	0.00	0.00	0.00	0.00	1.81
3	1.71	1.71	1.77	1.51	1.26	2.05	0.00	0.00	0.00	0.00	3.10
4	1.05	1.31	1.44	1.38	1.54	1.25	2.03	0.00	0.00	0.00	5.07
5	0.65	0.81	1.12	1.14	1.43	1.55	1.25	2.05	0.00	0.00	8.19
6	0.40	0.50	0.68	0.87	1.17	1.43	1.54	1.26	2.15	0.00	13.36
7	0.24	0.30	0.41	0.52	0.87	1.14	1.38	1.51	1.29	2.34	22.30
8	0.19	0.23	0.32	0.41	0.68	1.12	1.44	1.77	2.02	1.83	28.52
9	0.14	0.17	0.23	0.30	0.50	0.81	1.31	1.71	2.20	2.65	39.32
10	0.11	0.14	0.19	0.24	0.40	0.65	1.05	1.71	2.34	3.19	49.09
TH	1.00	1.25	1.72	2.20	3.68	5.99	9.68	15.82	27.10	49.09	

7th Edition

Strength \ Toughness	1	2	3	4	5	6	7	8	9	10	SH
1	1.31	0.54	0.62	0.74	0.83	0.97	1.05	1.19	1.28	1.47	1.00
2	1.74	1.28	0.98	0.59	0.66	0.77	0.83	0.95	1.02	1.17	1.26
3	1.43	1.40	1.21	0.97	1.09	0.64	0.68	0.78	0.84	0.96	1.53
4	1.14	1.40	1.28	1.15	0.87	1.01	1.09	0.62	0.67	0.76	1.92
5	0.94	1.15	1.06	1.27	1.08	0.84	0.90	1.03	1.10	0.63	2.32
6	0.81	0.99	1.14	1.09	1.23	1.08	0.77	0.88	0.95	1.08	2.71
7	0.75	0.92	1.06	1.01	1.14	1.33	1.08	0.82	0.88	1.01	2.92
8	0.68	0.84	0.96	1.15	1.04	1.21	1.30	1.11	0.80	0.91	3.21
9	0.63	0.78	0.89	1.07	0.96	1.12	1.21	1.38	1.11	0.85	3.46
10	0.57	0.70	0.81	0.97	1.09	1.02	1.09	1.25	1.34	1.15	3.82
TH	1.00	1.23	1.41	1.69	1.91	2.23	2.39	2.73	2.94	3.36	

8th Edition

Figure 3: Wound tables for *Warhammer 40,000*. Depending on the Strength (S) of the attacker and the Toughness (T) of the defender, a given hit has some probability of causing a wound. Tables for both 7th and 8th edition have been included. **Top:** The chance to wound out of 6 (Games Workshop 2017; Rodriguez 2017). **Bottom:** The resulting payoff matrix after balancing with a multiplicative handicap function along with the handicaps ("SH" for Strength and "TH" for Toughness). The payoff matrix is normalized so that the game has a value of 1; the handicaps so that a S or T of 1 has handicap 1.
Additional commentary: Like many wargames, *Warhammer 40,000* uses a points system in order to balance the strength of two opposing armies. As such the handicaps could be interpreted as a cost multiplier to assign to each Strength or Toughness value. The large number of immunities (i.e. 0 chance to wound) in the 7th edition table creates a sharp escalation of handicaps as S and T increase, as well as relatively hard counters in the resulting payoff matrix (i.e. many S-T matchups having payoff much lower or much greater than the value of the game). If desired, these could be dampened by lowering the weight of strategies that have a large number of immunities; indeed, extreme values of S and T tend to be rare in the actual game. The 8th edition table changes the probabilities from being based on the difference of S and T to being based on their ratio, and removes the immunities. This results in the handicaps increasing less sharply with S and T, as well as softer counters. More subtle changes include whether it is better to choose a slightly higher or slightly lower number than one's opponent.

3.3 Example: Unit Attributes

What we model as a strategy is not limited to unordered collections of items. In the context of a roleplaying, tactics, or strategy game, we can model unit attributes by considering each possible value for an attribute to be a strategy, with a better value for such an attribute to be offset by a worse value for some other attribute (e.g. cost, damage, or hit points).

For example, consider the wound roll in *Warhammer 40,000* (Games Workshop 2017):

> **Wound Roll:** If an attack scores a hit, you will then need to roll another [6-sided] dice to see if the attack successfully wounds the target. The roll required is determined by comparing the attacking weapon's Strength characteristic with the target's Toughness characteristic, as shown on the following table:

The appropriate tables for the wound roll are shown in Figure 3 for the 7th and 8th editions of the game, along with the handicaps that produce a uniform Nash equilibrium and the resulting payoff matrix. The computation took 5 ms for each on a single desktop computer. Additional commentary can be found in the caption.

4 Monotone Handicaps

While multiplicative handicaps (Equation 4) are applicable to many cases and admit a simple algorithm, it is quite a narrow class. In particular:

- The payoffs are unbounded with respect to handicaps, and therefore cannot represent probabilities.
- They are inherently asymmetric—there is no obvious way of representing cases where both players are choosing from the same set of strategies.

Let us therefore consider a more general class of handicap functions $F_{ij}(h_{ri}, h_{cj})$, with the goal of expressing our problem as an optimization problem, i.e. one of finding the minimum or zero of some objective function. We could then apply standard nonlinear optimization methods such as Levenberg-Marquardt; see (Madsen, Nielsen, and Tingleff

2004) for a survey. However, merely expressing our problem as an optimization problem still leaves some major questions unanswered:

- What does the solution space look like? Could there be multiple solutions?
- Are there optimization methods likely—or better yet, guaranteed—to find a solution (if one exists)?

To produce satisfactory answers, we make the assumption that the handicap functions are *strictly monotone*:

$$\left.\begin{array}{l} F_{ij}\left(h_{ri}, h_{cj}\right) < F_{ij}\left(h_{ri}+x, h_{cj}\right) \\ F_{ij}\left(h_{ri}, h_{cj}\right) > F_{ij}\left(h_{ri}, h_{cj}+x\right) \end{array}\right\} \forall \quad \begin{array}{l} i, j, h_{ri}, h_{cj} \\ x > 0 \end{array} \tag{5}$$

Monotonicity is usually a reasonable assumption to make: in practical terms, it means that an increase in a strategy's handicap is always bad for the player employing it regardless of the opposing strategy.

Additionally, we will assume that $v = 0$, which we can do without loss of generality by subtracting v from all of the handicap functions F_{ij}.

4.1 Two-Parameter Monotone Handicaps

We start with the more general case where the handicap functions take the two corresponding handicap variables as independent parameters. We can express our problem as a least-squares problem in terms of the expected payoff of each strategy at the desired Nash equilibrium, weighted by their probabilities at that equilibrium. Unfortunately this problem is not convex, which rules out some powerful guarantees on the efficiency of optimization methods (Boyd and Vandenberghe 2004). However, we can show a weaker but still useful property. Suppose that the F_{ij} are differentiable and strictly monotone (Equation 5). This is enough to guarantee that the gradient of the objective function is zero if and only if it is a global minimum, a property called *invexity* (Ben-Israel and Mond 1986). Furthermore, any such point is a "perfect" solution; that is, the expected payoff of all strategies is exactly 0 at the desired Nash equilibrium. (Note that such a solution is not guaranteed to exist—for example, if all F_{ij} are bounded strictly below 0.) This at least eliminates one major failure case of nonconvex optimization, that of getting stuck in a local minimum that is not a global minimum.

Theorem 1. *Let the F_{ij} be differentiable and strictly monotone functions as defined in Equation 5. Consider the weighted sum-of-squares error in expected payoffs of strategies at the desired Nash equilibrium*

$$z\left(\mathbf{h}\right) = \mathbf{p}\left(\mathbf{h}\right) \cdot \left(\mathbf{w} \odot \mathbf{p}\left(\mathbf{h}\right)\right) \tag{6}$$

where $\odot$ denotes elementwise multiplication. Then $\nabla z = \mathbf{0}$ if and only if $\mathbf{p} = \mathbf{0}$.

Proof. Since the least possible value for the objective function is 0, the gradient must be zero at any point that achieves an objective of 0.

In the other direction, consider a single component of the gradient corresponding to a row strategy i, which is 0 if the overall gradient is zero:

$$\begin{aligned} \left(\nabla z\right)_{ri} &= \frac{dz}{dh_{ri}} = \frac{d}{dh_{ri}} \mathbf{p} \cdot \left(\mathbf{w} \odot \mathbf{p}\right) \\ &= 2\frac{d\mathbf{p}}{dh_{ri}} \cdot \left(\mathbf{w} \odot \mathbf{p}\right) \\ &= 2\sum_k w_{rk} p_{rk} \frac{dp_{rk}}{dh_{ri}} + 2\sum_j w_{cj} p_{cj} \frac{dp_{cj}}{dh_{ri}} \end{aligned} \tag{7}$$

Substituting in the derivatives of the expected payoffs (Equation 2), namely

$$\begin{aligned} \frac{dp_{rk}}{dh_{ri}} &= \begin{cases} \sum_j w_{cj} \frac{dF_{ri}}{dh_{ri}} & \text{for } k = i \\ 0 & \text{for } k \neq i \end{cases} \\ \frac{dp_{cj}}{dh_{ri}} &= -w_{ri} \frac{dF_{ij}}{dh_{ri}} \end{aligned} \tag{8}$$

we have

$$\begin{aligned} \left(\nabla z\right)_{ri} &= 2w_{ri} p_{ri} \sum_j w_{cj} \frac{dF_{ri}}{dh_{ri}} - 2\sum_j w_{cj} p_{cj} w_{ri} \frac{dF_{ij}}{dh_{ri}} \\ &= 2w_{ri} \sum_j w_{cj} \left(p_{ri} - p_{cj}\right) \frac{dF_{ij}}{dh_{ri}} \end{aligned} \tag{9}$$

This applies symmetrically to column strategies as well.

Now, select a strategy I with the largest absolute value of expected payoff $|p_{rI}|$. We will only explicitly treat the case that this strategy belongs to the row player, but a symmetric argument applies if it belongs to the column player. Given that p_{rI} has the largest absolute value of all expected payoffs, each $p_{rI} - p_{cj}$ either has the same sign as p_{rI} or is 0. Furthermore, given that all the weights w are strictly positive and the F_{ij} has strictly negative derivative, the only way that a term is 0 is if $p_{cj} = p_{rI}$, and the sum $\left(\nabla z\right)_{rI} = 0$ only if $p_{cj} = p_{rI}$ for all column strategies j.

But now we can use the symmetric argument in the other direction to prove that all $p_{ri} = p_{cj} = p_{rI}$ for all row strategies i. The expected payoff of the game if both players play the desired Nash equilibrium must be the same whether we sum the rows or columns first, that is

$$\sum_{i,j} w_{ri} w_{cj} F_{ij} = \sum_i w_{ri} p_{ri} = -\sum_j w_{cj} p_{cj} \tag{10}$$

Since all row and column weights sum to 1, we have $p_{rI} = -p_{rI}$, which means $p_{rI} = 0$, and therefore all strategies have 0 expected payoff. □

4.2 One-Parameter Monotone Handicaps

Guaranteeing that all critical points are global minima is good, but we can do better if the handicap functions can be expressed as one-parameter functions of the difference between the corresponding row and column handicaps:

$$F_{ij}\left(h_{ri}, h_{cj}\right) = F_{ij}\left(h_{cj} - h_{ri}\right) \tag{11}$$

Another way of putting this condition is that adding any global offset to all of the handicaps does not change any of the entries of the payoff matrix. The monotone assumption (Equation 5) is kept by assuming these functions are strictly

monotonically increasing. Note that we can express payoffs that depend on the ratio of the handicaps, such as the multiplicative handicaps of Equation 4, by replacing the handicap variables with their logarithms.

We can define a strictly monotone operator that is zero when we are at a solution to our problem. Finding such a zero is a special case of the variational inequality problem, which for strictly monotone operators this is guaranteed to have at most one solution (up to a global offset), and for which there exist algorithms with guaranteed convergence; see Edwards (2013) for a survey. Note that this notion of monotone *operators* is distinct from the previous notion of monotone *handicap functions*.

We begin by defining a *non*-strictly monotone operator $T : \mathbb{R}^{n+m} \to \mathbb{R}^{n+m}$:

$$T(\mathbf{h}) = -\mathbf{w} \odot \mathbf{p}(\mathbf{h}) \tag{12}$$

This operator maps each handicap to the negative of the expected payoff of the corresponding strategy at the desired Nash equilibrium, times the weight of that strategy. Since all weights are strictly positive, this is zero if and only if all strategies have an expected payoff of 0 at the desired Nash equilibrium. Therefore our problem reduces to finding a zero of T. We now show that T is a monotone operator:

Theorem 2. *Let the handicap functions F_{ij} be strictly monotone increasing. The operator T then satisfies the monotone property*

$$(\mathbf{h}' - \mathbf{h}) \cdot (T(\mathbf{h}') - T(\mathbf{h})) \geq 0 \qquad \forall \mathbf{h}', \mathbf{h} \tag{13}$$

Proof. Let the prefix Δ denote the difference of a quantity from when the handicaps are $\mathbf{h}'$ minus when the handicaps are $\mathbf{h}$. Expanding the left side of Equation 13 yields

$$\begin{aligned}
&(\mathbf{h}' - \mathbf{h}) \cdot (T(\mathbf{h}') - T(\mathbf{h})) \\
=& -\Delta\mathbf{h}_r \cdot (\mathbf{w}_r \odot \Delta\mathbf{p}_r) - \Delta\mathbf{h}_c \cdot (\mathbf{w}_c \odot \Delta\mathbf{p}_c) \\
=& -\sum_i \Delta h_{ri} w_{ri} \sum_j w_{cj} \Delta F_{ij} \\
&+ \sum_j \Delta h_{cj} w_{cj} \sum_i w_{ri} \Delta F_{ij} \\
=& \sum_{i,j} w_{ri} w_{cj} \Delta F_{ij} \Delta(h_{cj} - h_{ri})
\end{aligned} \tag{14}$$

Since the F_{ij} are strictly monotone increasing ΔF_{ij} has the same sign as the change in its argument $\Delta(h_{cj} - h_{ri})$, so the terms in this sum are always nonnegative and the sum is always nonnegative. □

Note that adding the same global offset to all handicaps does not change F, so we can fix the handicap of one strategy—say, the first row strategy—at 0 and delete it from the input of T. Likewise, by observing Equation 10 again, if the expected payoffs for all strategies except one are 0, the payoff of that last strategy must also be 0. Therefore towards finding a zero of T we may also delete the first row payoff from the output. If we perform this deletion, we have a strictly monotone operator:

Theorem 3. *Fix $h_{r0} = 0$, delete it from the input $\mathbf{h}$ of T, and delete the corresponding element $-w_{r0}p_{r0}$ from the output of T. The resulting operator $\bar{T} : \mathbb{R}^{n+m-1} \to \mathbb{R}^{n+m-1}$ then satisfies the strict monotone property*

$$(\bar{\mathbf{h}}' - \bar{\mathbf{h}}) \cdot (\bar{T}(\bar{\mathbf{h}}') - \bar{T}(\bar{\mathbf{h}})) > 0 \qquad \forall \bar{\mathbf{h}}' \neq \bar{\mathbf{h}} \tag{15}$$

Proof. Let overbars, as in $\bar{T}$, denote quantities with the first row strategy removed. Observe that

$$T\left(\begin{bmatrix} 0 \\ \bar{\mathbf{h}} \end{bmatrix}\right) = \begin{bmatrix} -w_{r0}p_{r0} \\ -\bar{\mathbf{w}} \odot \bar{\mathbf{p}} \end{bmatrix} = \begin{bmatrix} -w_{r0}p_{r0} \\ \bar{T}(\bar{\mathbf{h}}) \end{bmatrix} \tag{16}$$

Since the first row here does not contribute to the dot product of Equation 13, the inequality is saturated for T if and only if it is saturated for $\bar{T}$. When does this happen? Looking at Equation 14 we see that, since all weights are strictly positive, each term is 0 if and only if $\Delta h_{ri} = \Delta h_{cj}$. For the sum to be 0 all $\Delta h_{ri}, \Delta h_{cj}$ must be equal. Fixing $h_{r0} = 0$ implies that the change in that handicap is also $\Delta h_{r0} = 0$ for any $\mathbf{h}', \mathbf{h}$. Together, this means that the sum is 0 only if all $\Delta h_{ri}, \Delta h_{cj} = 0$, or in other words, $\mathbf{h}' = \mathbf{h}$.

Likewise, when we fix $h_{r0} = 0$, $\bar{\mathbf{h}}' = \bar{\mathbf{h}}$ if and only if $\mathbf{h}' = \mathbf{h}$. Putting this together, the inequality is saturated only when $\bar{\mathbf{h}}' = \bar{\mathbf{h}}$, so the inequality is strict for $\bar{\mathbf{h}}' \neq \bar{\mathbf{h}}$. □

This guarantees convergence for some existing algorithms (Edwards 2013). Furthermore, the solution is unique (up to the choice of global offset), if it exists, since letting $\bar{\mathbf{h}}'$ and $\bar{\mathbf{h}}$ be two distinct solutions would contradict Equation 15. The global offset does not change the resulting payoff matrix, so the difference between solutions is therefore only an aesthetic choice and not a game-mechanical one. In contrast, in the two-parameter case the solution space may be more complicated, and different solutions could even have different payoff matrices, as in the following example when all handicaps are set to the same value x:

$$F = \begin{bmatrix} 2c_0 - r_0 & c_1 - 2r_0 \\ c_0 - 2r_1 & 2c_1 - r_1 \end{bmatrix} \Rightarrow \begin{bmatrix} x & -x \\ -x & x \end{bmatrix} \tag{17}$$

4.3 Symmetric Games

Finally, we show that these properties apply analogously to symmetric games, where the strategies available to the two players are identical (and not merely sharing the same names as in the Pokémon example). Specifically:

- Both players have the same number of strategies ($n = m$), and the payoff matrix is square.
- $\mathbf{w}_s \triangleq \mathbf{w}_r = \mathbf{w}_c$; i.e. the desired Nash equilibrium weights are the same for both players.
- Likewise, $\mathbf{h}_s \triangleq \mathbf{h}_r = \mathbf{h}_c$; i.e. the handicaps are constrained to be the same for both players.
- $F_{ij}(h_{si}, h_{sj}) = -F_{ji}(h_{sj}, h_{si})$; i.e. the payoff matrix F is skew-symmetric for all $i, j, \mathbf{h}_s$. Along with the weights this ensures that $\mathbf{p}_s \triangleq \mathbf{p}_r = \mathbf{p}_c$.
- $v = 0$; i.e. the desired value of the game is 0.

Theorem 4. *In the symmetric case, let $\nabla_s z$ be the derivative of the least-squares error (Equation 6) with respect to the shared handicaps $\mathbf{h}_s$. Then $\nabla_s z = \mathbf{0}$ if and only if $\mathbf{p} = \mathbf{0}$.*

Proof. $\nabla_s z = \nabla_r z + \nabla_c z$ where ∇_r, ∇_c are the gradients with respect to the row and column handicaps $\mathbf{h}_r, \mathbf{h}_c$ only. But by symmetry $\nabla_r z = \nabla_c z$, so $\nabla_s z = \mathbf{0}$ if and only if $\nabla_r z = \nabla_c z = \mathbf{0}$. Thus this reduces to Theorem 1. □

Theorem 5. *In the symmetric case, define the operator $T_s : \mathbb{R}^n \to \mathbb{R}^n$ as follows:*

$$T_s(\mathbf{h}_s) = -\mathbf{w}_s \odot \mathbf{p}_s(\mathbf{h}_s) \tag{18}$$

Fix $h_{s0} = 0$, delete it from the input $\mathbf{h}_s$ of T_s, and delete the corresponding element $-w_{s0}p_{s0}$ from the output of T_s. Then the resulting operator $\bar{T}_s : \mathbb{R}^{n-1} \to \mathbb{R}^{n-1}$ satisfies the strict monotone property of Equation 15.

Proof. Observe that we can input $\bar{\mathbf{h}}_s$ into the operator $\bar{T}$ of Theorem 3:

$$\bar{T}\left(\begin{bmatrix} \bar{\mathbf{h}}_s \\ 0 \\ \bar{\mathbf{h}}_s \end{bmatrix}\right) = \begin{bmatrix} -\bar{\mathbf{w}}_s \odot \bar{\mathbf{p}}_s \\ -w_{s0}p_{s0} \\ -\bar{\mathbf{w}}_s \odot \bar{\mathbf{p}}_s \end{bmatrix} = \begin{bmatrix} \bar{T}_s(\bar{\mathbf{h}}_s) \\ -w_{s0}p_{s0} \\ \bar{T}_s(\bar{\mathbf{h}}_s) \end{bmatrix} \tag{19}$$

The middle 0 element, resulting from fixing $h_{s0} = 0$, does not contribute to the dot product of Equation 15, so the dot product is simply half as much for $\bar{T}_s$ as it is for $\bar{T}$. Therefore since $\bar{T}$ is strictly monotone, so is $\bar{T}_s$. □

4.4 Example: Matchup Charts

Initial matchup chart

	Dhalsim	Boxer	Claw	Ryu	Chun-Li	Guile	Dee Jay	Dictator	Sagat	E. Honda	Ken	Fei-Long	Zangief	Blanka	Cammy	T. Hawk	Mean
Dhalsim	0.5	0.55	0.35	0.7	0.5	0.85	0.8	0.65	0.9	0.65	0.75	0.65	0.75	0.8	0.65	0.65	0.669
Boxer	0.45	0.5	0.55	0.6	0.45	0.6	0.7	0.65	0.8	0.65	0.65	0.7	0.55	0.75	0.75	0.55	0.619
Claw	0.65	0.45	0.5	0.6	0.6	0.7	0.6	0.5	0.75	0.6	0.6	0.65	0.65	0.65	0.6	0.7	0.613
Ryu	0.3	0.4	0.4	0.5	0.6	0.65	0.7	0.65	0.65	0.8	0.65	0.65	0.45	0.75	0.75	0.6	0.594
Chun-Li	0.5	0.55	0.4	0.4	0.5	0.6	0.65	0.7	0.65	0.75	0.65	0.4	0.75	0.7	0.6	0.55	0.584
Guile	0.15	0.4	0.3	0.35	0.4	0.5	0.55	0.55	0.65	0.8	0.65	0.75	0.8	0.6	0.8	0.65	0.556
Dee Jay	0.2	0.3	0.4	0.3	0.35	0.45	0.5	0.65	0.55	0.85	0.5	0.75	0.45	0.75	0.8	0.7	0.531
Dictator	0.35	0.35	0.5	0.35	0.3	0.45	0.35	0.5	0.65	0.35	0.4	0.65	0.45	0.35	0.8	0.7	0.469
Sagat	0.1	0.2	0.25	0.35	0.35	0.35	0.45	0.35	0.5	0.65	0.5	0.65	0.6	0.8	0.7	0.65	0.466
E. Honda	0.35	0.35	0.4	0.2	0.25	0.2	0.15	0.65	0.35	0.5	0.3	0.7	0.85	0.75	0.8	0.65	0.466
Ken	0.25	0.35	0.4	0.35	0.35	0.35	0.5	0.6	0.5	0.7	0.5	0.45	0.4	0.5	0.5	0.55	0.453
Fei-Long	0.35	0.3	0.35	0.35	0.6	0.25	0.25	0.35	0.35	0.3	0.55	0.5	0.7	0.65	0.7	0.7	0.453
Zangief	0.25	0.45	0.35	0.55	0.25	0.2	0.55	0.55	0.4	0.15	0.6	0.3	0.5	0.4	0.35	0.55	0.400
Blanka	0.2	0.25	0.35	0.25	0.3	0.4	0.25	0.65	0.2	0.25	0.5	0.35	0.6	0.5	0.4	0.7	0.384
Cammy	0.35	0.25	0.4	0.25	0.4	0.2	0.2	0.2	0.3	0.2	0.5	0.3	0.65	0.6	0.5	0.85	0.384
T. Hawk	0.35	0.45	0.3	0.4	0.45	0.35	0.3	0.3	0.35	0.35	0.45	0.3	0.45	0.3	0.15	0.5	0.359

After balancing using a logistic handicap

	Dhalsim	Boxer	Claw	Ryu	Chun-Li	Guile	Dee Jay	Dictator	Sagat	E. Honda	Ken	Fei-Long	Zangief	Blanka	Cammy	T. Hawk	Handicap
Dhalsim	0.500	0.487	0.285	0.620	0.401	0.771	0.683	0.434	0.779	0.427	0.534	0.414	0.472	0.527	0.340	0.324	0.765
Boxer	0.513	0.500	0.538	0.575	0.414	0.535	0.617	0.497	0.668	0.490	0.477	0.533	0.319	0.519	0.517	0.289	0.513
Claw	0.715	0.462	0.500	0.586	0.575	0.652	0.521	0.358	0.612	0.448	0.436	0.488	0.428	0.411	0.360	0.448	0.466
Ryu	0.380	0.425	0.414	0.500	0.589	0.612	0.642	0.523	0.509	0.696	0.503	0.503	0.259	0.545	0.543	0.356	0.408
Chun-Li	0.599	0.586	0.425	0.411	0.500	0.571	0.598	0.590	0.520	0.643	0.514	0.275	0.572	0.493	0.383	0.320	0.364
Guile	0.229	0.465	0.348	0.388	0.429	0.500	0.525	0.459	0.549	0.730	0.544	0.658	0.667	0.413	0.651	0.446	0.246
Dee Jay	0.317	0.383	0.479	0.358	0.402	0.475	0.500	0.588	0.470	0.809	0.415	0.680	0.312	0.609	0.674	0.528	0.144
Dictator	0.566	0.503	0.642	0.477	0.410	0.541	0.412	0.500	0.637	0.343	0.381	0.631	0.371	0.266	0.729	0.593	-0.118
Sagat	0.221	0.332	0.388	0.491	0.480	0.451	0.530	0.363	0.500	0.656	0.494	0.644	0.534	0.741	0.624	0.551	-0.175
E. Honda	0.573	0.510	0.552	0.304	0.357	0.270	0.191	0.657	0.344	0.500	0.290	0.689	0.808	0.676	0.735	0.544	-0.148
Ken	0.466	0.523	0.564	0.497	0.486	0.456	0.585	0.619	0.506	0.710	0.500	0.449	0.343	0.422	0.421	0.452	-0.198
Fei-Long	0.586	0.467	0.512	0.497	0.725	0.342	0.320	0.369	0.356	0.311	0.551	0.500	0.646	0.576	0.630	0.612	-0.201
Zangief	0.528	0.681	0.572	0.741	0.428	0.333	0.688	0.629	0.466	0.192	0.657	0.354	0.500	0.384	0.334	0.514	-0.445
Blanka	0.473	0.481	0.589	0.455	0.507	0.587	0.391	0.734	0.259	0.324	0.578	0.424	0.616	0.500	0.399	0.683	-0.512
Cammy	0.660	0.483	0.640	0.457	0.617	0.349	0.326	0.271	0.376	0.265	0.579	0.370	0.666	0.601	0.500	0.840	-0.516
T. Hawk	0.676	0.711	0.552	0.644	0.680	0.554	0.472	0.407	0.449	0.456	0.548	0.388	0.486	0.317	0.160	0.500	-0.590

Figure 4: *Super Street Fighter 2 Turbo* matchup chart before and after balancing using a logistic handicap function. The values represent the expected win rate for the row player and have been color-coded with red favoring the row player and blue favoring the column player. Data from (nohoho 2008); see also commentary by Sirlin (2014).

Communities of games, particularly fighting games, often generate *matchup charts*: a matrix whose entries A_{ij} are the predicted win rate if one player picks character i and their equally-skilled opponent picks character j. Jaffe (2013) investigated matchup charts and the range of possible mixed strategies that achieve some minimum win rate.

Here we analyze matchup charts in terms of handicaps. Inspired by Elo ratings, originally created to rate chess players (see (Sismanis 2010) for recent developments), we use a logistic function as our handicap function, with the argument being the difference of the row and column handicaps plus a constant describing the specific matchup:

$$F_{ij}(h_{ri}, h_{cj}) = \frac{1}{1 + e^{-(h_{cj} - h_{ri} + \alpha_{ij})}} - \frac{1}{2} \tag{20}$$

where α_{ij} is chosen so that setting all handicaps to 0 reproduces the original matchup chart minus $\frac{1}{2}$, i.e. $F_{ij}(0,0) = A_{ij} - \frac{1}{2}$. Note that this handicap function is strictly monotone, symmetric, and one-parameter. Figure 4 shows the result of this balancing process when applied to a *Super Street Fighter 2 Turbo* matchup chart. The computation took 4 ms on a single desktop computer.

5 Conclusion

We have presented a formalization of balance in terms of zero-sum games and shown that it is amenable to optimization techniques under the assumption of monotonicity. Our open-source implementation, based on SciPy (Jones et al. 2001), is available at https://github.com/ajul/zerosum.

However, even within our setting there remain major challenges to be addressed. We discuss a couple here.

The Modeling Challenge This formalization reduces the problem of balancing $n \times m$ possible pairwise matchups between strategies to modeling the individual pairwise matchups. A strength of our formulation is that we can easily slot in alternative such pairwise models, and it is thus potentially applicable to a wide variety of games. However, even evaluating two strategies against each other can be nontrivial.

For example, our fighting game matchup chart example (Figure 4) demonstrates that our algorithm works in the mathematical and computational sense. However, it is not clear whether the logistic handicap function itself is an accurate model, nor how to interpret the resulting handicaps quantitatively in terms of what should be changed about the characters in order to achieve a balanced game. After all, the actual win rates are determined by the fighting game proper that takes place *after* character selection, the dynamics of which are difficult to model analytically.

While perfectly accurate models may often be out of reach, approximate, learning, and/or player analytics approaches could produce good enough results to be useful. For example, learning-based approaches were used by Stanescu et al. (2013; 2015) in determining the outcome when two armies battle in *Starcraft*.

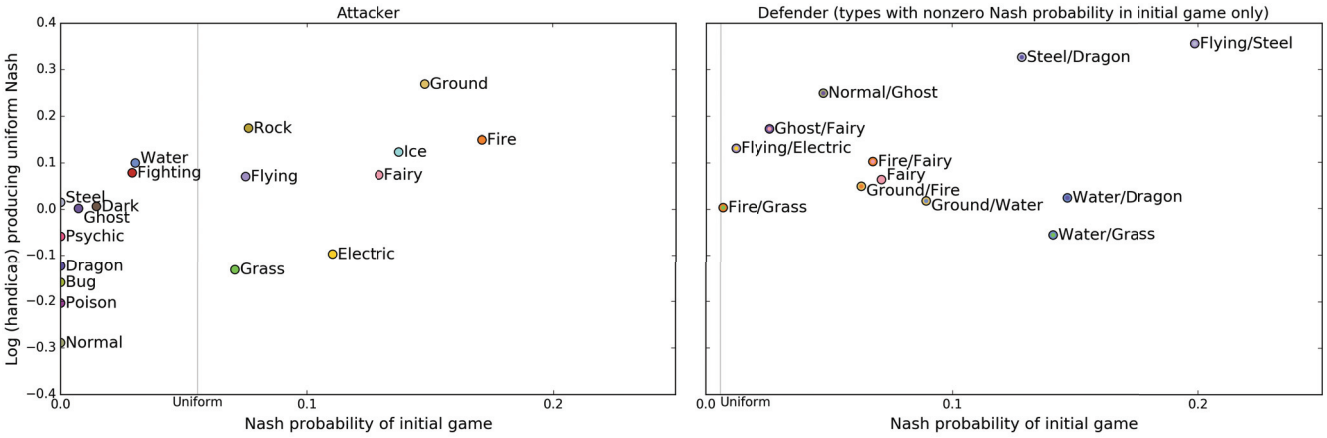

Figure 5: As Figure 2, but including dual types for the defender. To avoid excessive clutter, only defender types that have nonzero probability in the Nash equilibrium of the initial game are shown.

The Combinatoric Challenge Throughout this paper we have assumed that there is one handicap variable per strategy. This may be appropriate when all strategies are individually and explicitly specified. However, often this is not the case; rather, the strategies ultimately available to the player might be defined in a combinatorial fashion from the elements that handicaps are applied to.

In some cases balancing pairwise matchups between individual elements may still be a useful estimate; for example, in our *Warhammer 40,000* example (Figure 3) the result of the algorithm could provide a quick initial estimate for how to price individual units, even if does not consider entire armies. The quality of this estimate will of course depend on unmodeled factors such as unit synergies, targeting mechanics, and so forth.

In other cases this can be reasonably resolved by simply enumerating the possible combinations. For example, in the actual Pokémon games, defending Pokémon may have up to two types, not necessarily just one—Figure 1 shows just such an example of a Grass/Flying dual-type Pokémon. However, given that every combination of types exists in the game if and only if there is a Pokémon with that combination of types, and that each Pokémon has its own defense statistic, it is reasonable to consider every combination of types as a strategy. And while the number of strategies increases to a few hundred, the optimization is still computationally feasible (40 ms on a single desktop computer). The result is shown in Figure 5.

However, these are not always the case. For example, in the actual Pokémon games the player has a party of six Pokémon, and each Pokémon may have four moves (attacks) to choose from, increasing the number of strategies to an intractable level if we consider entire parties. Or for a conceptually simpler but also intractable scenario, consider a deck-building game: Given n card choices and a fixed deck size k, the number of possible decks grows as $\Theta\left(n^k\right)$. This poses several problems:

- n^k is a massive number for typical n, k. It is often infeasible to even enumerate all possible decks.
- n^k is also much larger than the number of cards, which are the elements that handicaps are applied to—in other words, generally the designer can modify specific cards but not specific decks. With fewer handicap variables than decks, we cannot expect that every Nash equilibrium is possible.
- Finally, it is not even clear that having a fully mixed Nash equilibrium on decks would be desirable, even if it were achievable.

Future work might define a more reasonable objective for cases like these and find an algorithm to achieve that objective.

Acknowledgments We would like to thank Erik Andersen, Jimmy Briggs, Shuo Chen, and Éva Tardos for their aid in preparing this paper.

References

Andersen, E.; Liu, Y.-E.; Snider, R.; Szeto, R.; and Popović, Z. 2011a. Placing a value on aesthetics in online casual games. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 1275–1278. ACM.

Andersen, E.; Liu, Y.-E.; Snider, R.; Szeto, R.; Cooper, S.; and Popovic, Z. 2011b. On the harmfulness of secondary game objectives. In *Proceedings of the Sixth International Conference on the Foundations of Digital Games*.

Andersen, E.; O'Rourke, E.; Liu, Y.-E.; Snider, R.; Lowdermilk, J.; Truong, D.; Cooper, S.; and Popovic, Z. 2012. The impact of tutorials on games of varying complexity. In *Proceedings of the 2012 annual conference on Human factors in computing systems*.

Ben-Israel, A., and Mond, B. 1986. What is invexity? *The ANZIAM Journal* 28(1):1–9.

Boyd, S., and Vandenberghe, L. 2004. *Convex optimization*. Cambridge University Press.

Browne, C. B.; Powley, E.; Whitehouse, D.; Lucas, S. M.; Cowling, P. I.; Rohlfshagen, P.; Tavener, S.; Perez, D.;

Samothrakis, S.; and Colton, S. 2012. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games* 4(1):1–43.

Browne, C. B. 2008. *Automatic generation and evaluation of recombination games*. Ph.D. Dissertation, Queensland University of Technology.

Bulbapedia. 2017a. Statistic — bulbapedia, the community driven pokémon encyclopedia. https://bulbapedia.bulbagarden.net/wiki/Statistic. [Online; accessed 3-July-2017].

Bulbapedia. 2017b. Type — bulbapedia, the community driven pokémon encyclopedia. http://bulbapedia.bulbagarden.net/wiki/Type. [Online; accessed 11-April-2017].

Chaslot, G.; Bakkes, S.; Szita, I.; and Spronck, P. 2008. Monte-carlo tree search: A new framework for game ai. In *AIIDE*.

Dantzig, G. B. 1963. *Linear Programming and Extensions*. Princeton University Press.

Edwards, M. R. 2013. Five classes of monotone linear relations and operators. In *Computational and Analytical Mathematics*. Springer. 375–400.

Games Workshop. 2017. Warhammer 40,000 battle primer. https://www.games-workshop.com/en-US/Warhammer-40000-Rules. [Online; accessed 21-July-2017].

Hazard, C. J. 2010. What every game designer should know about game theory. Triangle Game Conference. Raleigh, North Carolina.

Jaffe, A.; Miller, A.; Andersen, E.; Liu, Y.-E.; Karlin, A.; and Popovic, Z. 2012. Evaluating competitive game balance with restricted play. In *Eighth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*.

Jaffe, A. B. 2013. *Understanding game balance with quantitative methods*. Ph.D. Dissertation, University of Washington. https://digital.lib.washington.edu/researchworks/handle/1773/22797.

Jones, E.; Oliphant, T.; Peterson, P.; et al. 2001. SciPy: Open source scientific tools for Python.

Knight, P. A. 2008. The sinkhorn–knopp algorithm: convergence and applications. *SIAM Journal on Matrix Analysis and Applications* 30(1):261–275.

Knoll, B.; Lu, D.; and Burdge, J. 2017. Rock paper scissors programming competition. http://www.rpscontest.com/. [Online; accessed 5-April-2017].

Lomas, D.; Patel, K.; Forlizzi, J. L.; and Koedinger, K. R. 2013. Optimizing challenge in an educational game using large-scale design experiments. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 89–98. ACM.

Madsen, K.; Nielsen, H. B.; and Tingleff, O. 2004. Methods for non-linear least squares problems. Technical report, Informatics and Mathematical Modelling, Technical University of Denmark.

Mahlmann, T.; Togelius, J.; and Yannakakis, G. N. 2012. Evolving card sets towards balancing dominion. In *2012 IEEE Congress on Evolutionary Computation (CEC)*, 1–8. IEEE.

nohoho. 2008. Super turbo - new arcadia diagram. http://curryallergy.blogspot.com/2008/11/super-turbo-new-arcadia-diagram.html. [Online; accessed 28-April-2017].

Rodriguez, J. 2017. New community made 8th edition 40k wound chart. https://spikeybits.com/2017/05. [Online; accessed 21-July-2017].

Rollings, A., and Morris, D. 2000. *Game Architecture and Design*. Coriolis.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of go with deep neural networks and tree search. *Nature* 529(7587):484–489.

Sinkhorn, R., and Knopp, P. 1967. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics* 21(2):343–348.

Sinkhorn, R. 1967. Diagonal equivalence to matrices with prescribed row and column sums. *The American Mathematical Monthly* 74(4):402–405.

Sirlin, D. 2014. Game balance and yomi. http://www.sirlin.net/articles/game-balance-and-yomi. [Online; accessed 16-May-2017].

Sismanis, Y. 2010. How i won the "chess ratings-elo vs the rest of the world" competition. *arXiv preprint arXiv:1012.4571*.

Smith, W. D. 2005. Sinkhorn ratings, and new strongly polynomial time algorithms for sinkhorn balancing, perron eigenvectors, and markov chains. http://scorevoting.net/WarrenSmithPages/homepage/sinkhornv.pdf.

Solomon, J.; De Goes, F.; Peyré, G.; Cuturi, M.; Butscher, A.; Nguyen, A.; Du, T.; and Guibas, L. 2015. Convolutional wasserstein distances: Efficient optimal transportation on geometric domains. *ACM Transactions on Graphics (TOG)* 34(4):66.

Stanescu, M.; Barriga, N.; and Buro, M. 2015. Using lanchester attrition laws for combat prediction in starcraft. In *Eleventh AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*.

Stanescu, M.; Hernandez, S. P.; Erickson, G.; Greiner, R.; and Buro, M. 2013. Predicting army combat outcomes in starcraft. In *Ninth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*. Citeseer.

Tavares, A.; Azpúrua, H.; Santos, A.; and Chaimowicz, L. 2016. Rock, paper, starcraft: Strategy selection in real-time strategy games. In *Twelfth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*.