Saint Peterburg State University

GULYAEV Nikita Alekseevich

Master's thesis

Game-theoretic analysis of the Restricted Play principle

Specialization 01.04.02
Applied Mathematics and Informatics
Master's Program Game Theory and Operation Research

Research Advisor:
Elena Gubar,
Ph.D., Associate Professor of Department of
Mathematical Game Theory and Statistical Decisions

Reviewer:
Denis Fedyanin,
IEEE member, V. A. Trapeznikov Institute of Control
Sciences of RAS

Saint Petersburg

2023 г.

Contents

Introduction

Game balance is the fine-tuning phase of the game development, involving the slight adjustment of the game rules to make the game closer to its intended form [8]. This process may often be costly, but it is crucial for how the game is perceived by its target audience. Therefore, game balance is a key for the game's success [7]. Moreover, as shown in [9], the applications are not limited solely to the game development. As a consequence, there is a demand for the high-efficient methods of the game balance.

This thesis is an attempt to formulate some empiric game balance criteria mathematically. It uses the concept of restricted play first introduced by A. Jaffe to define certain game imbalance criteria formally. In [2], it is described in the following way: "...to understand the balance of some dynamic, we may frame it as the fairness of a match between two players, one of whom is restricted in a way that highlights that dynamic. In other words, we hallucinate a player (realistic or otherwise) whose behavior captures that dynamic or the lack thereof".

[2], as well as [3], provides some examples on the applications of this technique: for example, to evaluate the fairness of starting conditions, a fairness of a match between a restricted player always choosing a specific starting condition and an unrestricted player may be assessed. Similarly, the importance of playing unpredictably may be derived from the observations of the match between a normal player and a player who implements the low-entropy mixed strategies. The same logic is applied to a diversity of other balance features.

It should be noted that the original formulation assumes the (at least theoretical) possibility of holding a match between two players. Therefore, in this form, this principle cannot be directly applied to, for instance, single-player games, for which the problem of obtaining the effective balancing techniques (for example, difficulty evaluation, as in [1]) remains relevant. On the other hand, the notion of *fairness* is not well-defined and thus, before mathematical reasoning may be applied, it is necessary to introduce the formal definition for this concept.

Due to the reasons outlined above, for the purposes of this thesis, we introduce

a more formal, albeit less general, formulation of the restricted play principle. The main difference between this and the original version is that, in our case, we compare the performance of two imaginary versions of the **same** player, one of which is restricted in its choices. The performance is defined in two different ways, emerging from the game theory and probability theory correspondingly: through *worst-case* and *average-case* payoff values. While it is natural to apply the first estimate of a payoff when a player has no prior information about his/her possible opponent, the second estimate is more suited to the cases when such information is widely available, for example, when there is some statistical data on the preferences of the game's audience.

Using the game-theoretic concept of a normal-form game as the modelling framework, we provide formal definitions for some game balance criteria through the notions of *aspect set* and *aspect mapping*. Defined arbitrarily by a game designer according to his or her goals, the aspects can be any kind of mathematical objects corresponding to the important qualities of the player strategies, which, for any concrete strategy of a concrete player, may be either implemented by this strategy or not. This correspondence is strictly defined by the introduction of the aspect mapping, which maps any element of the aspect set to the set of strategies implementing it. Then, the formulations of the considered criteria emerge naturally from the single question, namely, whether there exists a rational strategy that doesn't implement a certain aspect. There, *rational* strategy is defined as a strategy yielding a non-negative (worst-case or average-case) payoff of a player implementing it.

The analogous questions regarding the existence of a rational strategy that does implement an aspect or about whether all the rational strategies do/don't implement an aspect can be easily reduced to that form by introducing the complementary aspect, which is implemented by a strategy if and only if this strategy doesn't implement the given one. This makes the corresponding criteria applicable to both the situations where we want to give a player the freedom to avoid the aspect without sacrificing some payoff and the cases when we implicitly try to force the players to use only the strategies implementing a certain aspect, crucial for delivering the right gameplay experience.

Alternatively, instead of a rational strategy one may use a *sub-optimal* strategy, which yields a payoff lying within the given range of a maximum possible value. For this modification, a separate set of criteria is also constructed.

The whole proposed approach heavily relies on the assumption that players tend to play the game as rationally as possible given their knowledge and skills. This assumption doesn't hold for every real-world gameplay scenario. Normally, the goal of each player is to have a fun time playing. This doesn't always correspond to getting the non-negative or maximum (depending on the rationality definition) payoff. Such discrepancy often leads to the behaviour of the players becoming more uncontrollable than it is expected to be. A so-called *griefing*, i.e., a deliberate act of irritating and harassing other players, may serve as an example of such disruptive behaviour. As such problems frustrate the players and make them leave the game eventually, it is considered a good practice to align the goals of the game with the most entertaining and satisfying aspects of the game. The concept of *fun* is very individual, but the game designer's experience is typically enough to be able to guide most of the players towards playing for the win using just the game's rules. For that reason, we can only consider the games where players try to play rationally. Alternatively, the estimated amount of fun a player has may itself be taken as this player's payoff.

The criteria proposed in this work are illustrated on one concrete example: the continuous version of the Colonel Blotto game, where each of the aspects corresponds to a combination of a player and a battlefield and is implemented by a strategy if and only if this strategy belongs to a corresponding player and involves sending a positive number of units to the corresponding battlefield. The criteria are checked for the three variations of this game, each with its own payoff function.

Literature review

In [8], Becker and Görlich examine a number of game balance definitions concluding that further research is necessary before a commonly agreeable definition may be derived, but nevertheless "it seems coherent that game balancing supports the design goal "fun"". In this thesis, we put the burden of separating the strategies

that may prevent some players from *having fun* on the game designer. The criteria derived, in turn, provide a way to ensure that the players will be motivated to avoid (or, conversely, to stick with) a certain class of strategies.

As was mentioned in the introduction, the first version of the approach used for assessing the player motivation in regards to choosing a strategy was proposed in [3] and then described more thoroughly in [2]. In addition, with the usage of MCTS, these papers demonstrated the feasibility of automated playtesting.

The automation of game balancing is a growing area of research [23]. Dynamic game balancing, a process in which a game adapts itself to the player's performance as a result sustaining an optimal level of difficulty throughout the whole playthrough, is widely studied. [7] explores a variety of AI-based approaches to game balancing. [18] demonstrates how reinforcement learning can be applied to determine when and how to spawn enemy units to keep the score as close to zero as possible effectively ensuring that the win probability is always around 50%. The usage of the NEAT and rtNEAT methods for the similar purpose is explored in [19]. [25] uses particle swarm optimization to ensure diversity while also adapting to the change in the player's characteristic. Overall, the problems of dynamic game balancing in the most cases are relevant to the single-player games only, though the automation of game balancing may also be applied to multi-player games [24].

Actionable conclusions regarding are usually derived through the analysis of the automatically collected data. In [11], a set of AlphaZero self-play games was generated and then used to gather the statistical data, in particular on the percentage of draws, for the different chess variants, provided near-optimal play. [10] measures the impact of cooperation in *Halo: Reach*, the data is extracted from public API. [14] and [15] analyze the data collected via telemetry.

The formal analysis is often performed using PCA, for example, in [10] and [13]. In the latter it is used to assess the diversity of the player's strategies. The importance of this quality serves as a motivation for the aspect inclusion criteria discussed in the later chapters. [12] uses PCA alongside a number of other methods, such as survival analysis, binary classification and k-means to predict customer churn after the content update is released. Similarly, [14] makes use of k-means and simplex

volume maximization to cluster the player behaviour. There were attempts to model the player behaviour with the usage of the deep learning as well [22]. Machine learning is also applied in [16] to predict the game's outcome based on a given set of the starting conditions.

Despite the significant amount of work in the field of game balance, there were a few attempts to provide some formal criteria. One attempt was made in [17]: in this paper, the hypothesis that the orthogonal unit differentiation is necessary for the game to be balanced was tested on the example of StarCraft II. However, the results were negative: the game that was assumed to be balanced had units falling below the expected rank.

The purpose of this thesis is to fill the gap in the study of the formal approaches to game balance. One of the possible applications is the partial automation of the parameter fine-tuning (one of the different approaches would be to use active learning technique, as in [26]). Another one is procedural content generation, which is currently one of the most active fields of research within AI in games research motivated by a real need within the industry [20]. Much of the research is devoted to studying the level generation, however, even generation of the complete games was also discussed [21]. Formal game balance criteria may be used as an additional filter for the generated content.

Chapter 1. Problem formulation

1.1 Game-theoretic description

We will formulate the subsequent results within the framework of normal-form games [4].

Definition 1. *An n-player normal-form game is a tuple $\Gamma = (N, \mathcal{X}, \mathbf{u})$, where:*

- $N = \{1, ..., n\}$ *is the set of n players;*
- $\mathcal{X} = X_1 \times ... \times X_n$ *is the strategy space, and X_i is the set of pure strategies of player i;*

- $\mathbf{u} = (u_1, \ldots, u_n)$ *is the vector-valued payoff function, whose* i*th component* $u_i : \mathcal{X} \mapsto \mathbb{R}$ *is the payoff function of player* i*.*

Note that the individual strategies of a player can be vectors, whose components correspond to different choices made by the player. Thus, we assume, in general, that $X_i \subset \mathbb{R}^{n_i}$ (or $X_i \subset \mathbb{Z}^{n_i}$), where n_i is the number of choices available to the ith player. An element of $\mathcal{X}$, denoted by $\mathbf{x} = (x_1, \ldots, x_n)$, is called the *strategy profile*. Hence, the payoff of any specific player depends not only on the strategy that this player chooses, but also on the strategies of other players.

In practice, the payoff of a player represents some measure of how much this player succeeded in reaching his or her goals. Often, it can be easily quantified as the number of points scored in the game. It is assumed that the payoff function can take on both positive and negative values, whereas the positive values correspond to a positive outcome of the game, while negative values correspond to a certain degree of loss.

Below, we introduce two measures of the outcome of a strategy: the *worst-case payoff* and the *average-case payoff*.

Given $I \subset N$, define $X_I = \bigtimes_{i \in I} X_i$ and $X_{-i} = X_{N \setminus \{i\}}$. We will use the short notation $u_i(\hat{x}_i, \mathbf{x}_{-i})$ to denote $u_i(x_1, ..., x_{i-1}, \hat{x}_i, x_{i+1}, ..., x_n)$, where $\hat{x}_i \in X_i$, and $\mathbf{x}_{-i} = (x_1, ..., x_{i-1}, x_{i+1}, ..., x_n) \in X_{-i}$.

Definition 2 (Worst-case payoff)**.** *The worst-case payoff of the* i*th player corresponding to the strategy* $x_i \in X_i$ *is denoted by* $\underline{u}_i(x_i)$ *and defined as*

$$\underline{u}_i(x_i) = \min_{\mathbf{x}_{-i} \in X_{-i}} u_i(x_i, \mathbf{x}_{-i}).$$

Let $\underline{u}_i$ denote the maximal possible worst-case payoff that the player can guarantee by a proper choice of the strategy. In game theory, this is referred to as the *lower value* of the game for player i [5]:

$$\underline{u}_i = \max_{x_i \in X_i} \underline{u}_i(x_i) = \max_{x_i} \min_{\mathbf{x}_{-i}} u_i(x_i, \mathbf{x}_{-i}).$$

To introduce the notion of *average-case payoff*, the described framework has to be extended by considering *mixed strategies*. Assume that $\mathbf{x}_{-i}$ is a random vector with discrete probability distribution $f_{-i}(\bar{\mathbf{x}}_{-i}) = \mathbb{P}\{\mathbf{x}_{-i} = \bar{\mathbf{x}}_{-i}\}$.

Definition 3 (Average-case payoff)**.** *Given the discrete probability distribution $f_{-i}(\bar{\mathbf{x}}_{-i})$, the average-case payoff of the ith player corresponding to the strategy $x_i \in X_i$ is denoted by $\mathbb{E}u_i(x_i)$ and defined as*

$$\mathbb{E}u_i(x_i) = \sum_{\mathbf{x}_{-i} \in X_{-i}} u_i(x_i, \mathbf{x}_{-i}) f_{-i}(\mathbf{x}_{-i}).$$

Similarly, if x_{-i} is a random vector with the continuous probability density function $f_{-i}(\bar{\mathbf{x}}_{-i})$ (implying that the strategy sets are uncountable), the above definition is modified as follows:

Definition 4 (Average-case payoff)**.** *Given the probability density function $f_{-i}(\bar{\mathbf{x}}_{-i})$, the average-case payoff of the ith player corresponding to the strategy $x_i \in X_i$ is denoted by $\mathbb{E}u_i(x_i)$ and defined as*

$$\mathbb{E}u_i(x_i) = \int_{\mathbf{x}_{-i} \in X_{-i}} u_i(x_i, \mathbf{x}_{-i}) f_{-i}(\mathbf{x}_{-i}) d\mathbf{x}_{-i}.$$

We will use $\mathbb{E}u_i$ to denote the best average payoff that the ith player can guarantee by a proper choice of pure strategy when the rival players employ mixed strategies:

$$\mathbb{E}u_i = \max_{x_i \in X_i} \mathbb{E}u_i(x_i).$$

One can readily observe that $\mathbb{E}u_i(x_i) \geq \underline{u}_i(x_i)$ for any x_i (and, consequently, $\mathbb{E}u_i \geq \underline{u}_i$). Note that the average-case payoff is computed with respect to a particular probability distribution function. In the following, we will assume that the probability function f_i is specified, either explicitly or implicitly.

Since optimal strategies are typically not or hardly achievable in a well-designed game, the player aims at playing *as good as possible*. To formalize this, we introduce two classes of player's strategies that correspond to two types of reasonable

behaviour of a player.

Definition 5 (Rational strategy)**.** *The strategy $x_i \in X_i$ of a player i is said to be worst-case rational if $\underline{u}_i(x_i) \geq 0$. Further, the strategy x_i is said to be average-case rational if $\mathbb{E}u_i(x_i) \geq 0$.*

Definition 6 (ε-suboptimal strategy)**.** *The strategy $x_i \in X_i$ of player i is said to be worst-case ε-suboptimal if*

$$\underline{u}_i - \underline{u}_i(x_i) \leq \varepsilon.$$

The strategy x_i of player i is said to be average-case ε-suboptimal if $\mathbb{E}u_i - \mathbb{E}u_i(x_i) \leq \varepsilon$.

A worst-case (average-case) rational strategy has the property that a player may restrict him- or herself to using only this strategy and still warrant (expect) a non-negative total payoff when playing the game repeatedly. Therefore, it is important to ensure that no such strategy has certain undesirable qualities. In other words, preferring a strategy with such qualities over the other possible ones should be punished by the means of using the negative payoff values. A formal definition along with an example of the qualities being discussed is presented later in Sec. 1.2. It should be noted, though, that this approach should be used thriftily to provide the players with sufficient degree of freedom.

While the rational strategies correspond to the secure, risk-adverse behaviour of the players who attempt at least not to lose, the suboptimal strategies are better suited to describe the behaviour of players who tend to maximize their guaranteed or expected profit. Therefore, it is natural for a game designer to define the payoff functions in such a way that the payoff of the strategies possessing the desired qualities will be closer to the optimal value than the payoff of the strategies that do not possess these qualities. In contrast to the rational strategies, analysing the suboptimal strategies is, hence, more useful for determining whether the desired choice of strategies is rewarded sufficiently. Conversely, the analysis of the rational strategies, or rather vice versa, the strategies *that are not rational*, may help to check whether the undesired behaviour is penalized.

Let us denote by $X_i^{\min[+]} \subset X_i$ and $X_i^{\min[\varepsilon]} \subset X_i$ the set of all worst-case rational

strategies and the set of all worst-case ε-suboptimal strategies of the ith player. Similarly, $X_i^{\mathrm{avg}[+]} \subset X_i$ and $X_i^{\mathrm{avg}[\varepsilon]} \subset X_i$ denote the sets of average-case rational and average-case ε-suboptimal strategies. We will refer to each of these four sets as a *reasonable strategy set* $X_i^{\mathfrak{b}}$ for the corresponding *reasonable behaviour scheme* $\mathfrak{b} \in \{\min[+], \min[\varepsilon], \mathrm{avg}[+], \mathrm{avg}[\varepsilon]\}$.

1.2 Aspects of a game

Although the ultimate goal of any player is to win, the overall experience of the player also depends on the gameplay, its richness and variability. To address the player's experience during the game, we introduce the notion of an *aspect*.

Let $A^i = \{a_1^i, \ldots, a_{k_i}^i\}$ be the set of aspects specified for the ith player. We define a map that associates with each aspect a subset of the player's strategies: $\alpha_i : A_i \to S_i$, where $S = 2^{X_i}$.

We say that the strategy profile $\mathbf{x} = (x_1, ..., x_n)$ *implements* the aspect a_j^i if

$$x_i \in \alpha_i(a_j^i).$$

Each aspect corresponds to a certain experience the player encounters when playing the game. To determine whether the choice of the strategy containing certain aspect by a (realistic) player is encouraged (or, conversely, discouraged), the following questions have to be answered:

- does there exist a rational strategy implementing this aspect?
- does there exist an ε-suboptimal strategy implementing this aspect?

Take, for example, the shooting game genre. Dealing damage to the members of the same team is usually considered bad behaviour and is therefore impossible in a wide range of games. However, some games, such as *Rainbow Six: Siege*, allow the friendly fire feature [6]. A logical step in motivating the players to avoid harming their teammates is to ensure that no rational strategy will involve such actions. In this example, the presence of the intentional friendly fire is taken as an aspect and the first of the questions mentioned above is posed.

1.3 Criteria definitions

Given aspect a_k^i defined for the ith player's strategies, a designer may wish to construct the gameplay in a way that the player inescapably chooses the strategies that either strictly avoid or completely include this aspect when employing a particular type of behaviour. On the other hand, the preceding requirements can be relaxed by asking that the player always has an option of selecting a strategy that either avoids or includes the considered aspect when following a particular behavioural scheme. In this way, we can formulate 4 classes of conditions: *strict aspect inclusion*, *strict aspect exclusion* as well as *weak aspect inclusion*, and *weak aspect exclusion*. Below, we will formally describe conditions for checking the second class of conditions (i.e., strict aspect exclusion) and show that the remaining 3 classes can be derived from this one.

Each condition is associated with a Boolean function that returns 1 (True) if the condition is satisfied, and 0 (False) otherwise. In the following, we will use these functions to show equivalence between different criteria.

The strict aspect exclusion criterion characterizes the conditions such that a player who acts reasonably (in some sense) would be compelled to avoid the strategies implementing a certain aspect. To formalize the above said we introduce a map $SE : (i, a_k^i, \mathfrak{b}) \to \{0, 1\}$, which is defined for the ith player, reasonable behaviour scheme $\mathfrak{b}$ associated with the reasonable strategy set $X_i^{\mathfrak{b}}$ and the aspect $a_k^i \in A^i$. We say that the strict aspect exclusion is satisfied, i.e., $SE(i, a_k^i, \mathfrak{b}) = 1$ if the following condition holds true:

$$X_i^{\mathfrak{b}} \cap \alpha_i(a_k^i) = \varnothing.$$

For each of the four reasonable behaviour schemes $\mathfrak{b}$, the strict aspect exclusion criterion can be formulated as an optimization problem as stated in the following lemma.

Lemma 1 (Strict aspect exclusion)**.** *For each reasonable behaviour scheme $\mathfrak{b}$, the satisfaction of the strict aspect exclusion criterion is equivalent to a particular optimization problem:*

- *Worst-case rationality,* $\mathfrak{b} = \min[+]$,

$$SE(i, a_k^i, \min[+]): \quad X_i^{min[+]} \cap \alpha_i(a_k^i) = \varnothing \iff \max_{x \in \alpha(a_k^i)} \underline{u}_i(x) < 0. \quad (1)$$

- *Average-case rationality,* $\mathfrak{b} = \text{avg}[+]$,

$$SE(i, a_k^i, \text{avg}[+]): \quad X_i^{avg[+]} \cap \alpha_i(a_k^i) = \varnothing \iff \max_{x \in \alpha_i(a_k^i)} \mathbb{E}u_i(x) < 0.$$

- *Worst-case sub-optimality,* $\mathfrak{b} = \min[\varepsilon]$,

$$SE(i, a_k^i, \min[\varepsilon]): \quad X_i^{min[\varepsilon]} \cap \alpha_i(a_k^i) = \varnothing \iff \max_{x \in \alpha_i(a_k^i)} \underline{u}_i(x) < \underline{u}_i - \varepsilon.$$

- *Average-case sub-optimality,* $\mathfrak{b} = \text{avg}[\varepsilon]$,

$$SE(i, a_k^i, \text{avg}[\varepsilon]): \quad X_i^{avg[\varepsilon]} \cap \alpha_i(a_k^i) = \varnothing \iff \max_{x \in \alpha_i(a_k^i)} \mathbb{E}u_i(x) < \mathbb{E}u_i - \varepsilon. \quad (2)$$

Proof. We prove the equivalences (1) and (2).

By definition,

$$X_i^{min[+]} = \big\{x \in X_i \big| \underline{u}_i(x) \geq 0\big\}.$$

Therefore,

$$\begin{aligned} X_i^{min[+]} \cap \alpha_i(a_k^i) = \varnothing &\iff \forall x \in \alpha_i(a_k^i): \ \ x \notin X_i^{min[+]} \\ &\iff \forall x \in \alpha_i(a_k^i): \ \ \underline{u}_i(x) < 0 \\ &\iff \max_{x_i \in \alpha(a_k^i)} \underline{u}_i(x_i) < 0. \end{aligned}$$

As for (2), by definition we have

$$X_i^{avg[\varepsilon]} = \big\{x \in X_i \big| \mathbb{E}u_i - \mathbb{E}u_i(x) \leq \varepsilon\big\},$$

which implies the following chain:

$$\begin{aligned} X_i^{avg[\varepsilon]} \cap \alpha_i(a_k^i) = \varnothing &\iff \forall x \in \alpha_i(a_k^i): \quad x \notin X_i^{avg[\varepsilon]} \\ &\iff \forall x \in \alpha_i(a_k^i): \quad (\mathbb{E}u_i - \mathbb{E}u_i(x)) > \varepsilon \\ &\iff \min_{x_i \in \alpha(a_k^i)} (\mathbb{E}u_i - \mathbb{E}u_i(x)) > \varepsilon \\ &\iff \mathbb{E}u_i - \varepsilon > \max_{x_i \in \alpha(a_k^i)} \mathbb{E}u_i(x). \end{aligned}$$

The remaining implications are shown in the same way. □

In contrast to the strict aspect exclusion case described above, one may often be more interested in providing the player with the freedom to choose whether to implement a certain aspect by ensuring that there exists at least one reasonable strategy for implementing such an aspect. To formalize this situation, we introduce the weak inclusion function $WI : (i, a_k^i, \mathfrak{b}) \to \{0, 1\}$ in the same way as the function SE above. We have $WI(i, a_k^i, \mathfrak{b}) = 1$ if the following condition holds:

$$X_i^{\mathfrak{b}} \cap \alpha_i(a_k^i) \neq \varnothing.$$

From the formula presented above, it follows that the weak aspect inclusion criterion can, in fact, be obtained as the negation of the strict aspect exclusion criterion:

$$WI(i, a_k^i, \mathfrak{b}) \equiv \overline{SE(i, a_k^i, \mathfrak{b})}.$$

The remaining two criteria are constructed analogically by considering whether a player has the freedom to *avoid* implementing a certain aspect or not.

The strict aspect inclusion criterion, which is fulfilled when the player i is forced to include a certain aspect a_k^i while following the behavioural scheme $\mathfrak{b}$, is associated with the function $SI(i, a_k^i, b)$ that takes the value 1 if

$$(X_i \setminus X_i^b) \cap \alpha_i(a_k^i) = \varnothing.$$

At the same time, weak aspect exclusion corresponds to the cases when the player has the option to avoid using the strategies implementing a particular aspect. The fulfilment of this criterion is described by the function $WE(i, a_k^i, \mathfrak{b})$, which takes the value 1 (true) if

$$(X_i \setminus X_i^{\mathfrak{b}}) \cap \alpha_i(a_k^i) \neq \varnothing.$$

Analogically to the first two criteria, SI and WE are connected through the boolean negation as well:

$$WE(i, a_k^i, \mathfrak{b}) \equiv \overline{SI(i, a_k^i, \mathfrak{b})}.$$

To establish a relation between the two pairs of criteria, we introduce the notion of the aspect complement $\overline{a_k^i}$, which is defined as follows:

Definition 7. *The complement of the aspect a_k^i is defined as*

$$\alpha_i(\overline{a_k^i}) = X_i \setminus \alpha_i(a_k^i),$$

This allows us to formulate the relation between all the four criteria defined above:

$$SI(i, a_k^i, \mathfrak{b}) \equiv \overline{WE(i, a_k^i, \mathfrak{b})} \equiv \overline{WI(i, \overline{a_k^i}, \mathfrak{b})} \equiv SE(i, \overline{a_k^i}, \mathfrak{b}). \tag{3}$$

In (3), the first and the third equivalences have already been shown, while the second one is valid because

$$X_i \setminus (X_i \setminus X_i^{\mathfrak{b}}) = X_i^{\mathfrak{b}}.$$

It is worth noting that the four criteria outlined above define the approach that is, in fact, the modified version of the restricted play principle. Indeed, by considering only the strategies implementing a certain aspect, we assess the efficiency of a player who is "restricted in some way" that distinguishes some quality of a strategy expressed by this aspect. Moreover, by making use of the notion of the sub-optimal strategies, we compare this efficiency to one of an unrestricted player.

On the other hand, the outlined approach bears several important distinctions from the original formulation of restricted play in [2].

First, instead of comparing the efficiency of the two players in a match between them, this approach relies on the comparison of the efficiencies of two instances of the **same** player, making this approach applicable to a wider range of cases, including single-player games and games with asymmetric conditions, where it is impossible to organize or simulate a match between a restricted player and an unrestricted player. In addition, the proposed approach offers another way to judge about the efficiency of a restricted player, which does not involve an unrestricted player at all: by testing for the existence of a rational strategy implementing an aspect.

Second, in contrast to the more abstract original formulation of the principle, two concrete measures of a strategy's efficiency are proposed: the worst-case payoff and the average-case payoff. This distinction allows the formal definition of the above-mentioned criteria for the evaluation of game feature balance.

Chapter 2. Example: Continuous Colonel Blotto

To illustrate the application of the formulated criteria, we will first consider a simple two-player normal form game $\Gamma_B = (N, X_1, X_2, u_1, u_2)$, where $N = \{1, 2\}$, the strategy sets are defined as

$$\begin{aligned} X_1 &= \left\{ x = (x_1, ..., x_m) \in \mathbb{R}^m_{\geq 0} \,\middle|\, \sum_{j=1}^{m} x_j c_j^1 \leq 1 \right\}, \\ X_2 &= \left\{ y = (y_1, ..., y_m) \in \mathbb{R}^m_{\geq 0} \,\middle|\, \sum_{j=1}^{m} y_j c_j^2 \leq 1 \right\}, \end{aligned} \tag{4}$$

and the payoff functions $u_i : X_1 \times X_2 \mapsto \mathbb{R}$, $i \in N$ will be defined below. Here, the dimension of the strategy space is larger than 1, i.e., $m > 1$, and c_j^i are positive weights.

The kth component of a player's strategy corresponds to the amount of resource invested into the battle on battlefield k, which has a certain value $R_k > 0$. Fur-

thermore, the coefficients c_j^i model the costs of transporting the jth unit of the ith player to the battlefield. The rules for determining which player claims the battlefield as well as how it affects the players' payoffs are defined by the concrete form of the payoff functions. In this thesis we will consider three cases:

- Utilization
$$\begin{cases} u_1(x,y) = \sum_{j=1}^m R_j(x_j - y_j), \\ u_2(x,y) = \sum_{j=1}^m R_j(y_j - x_j), \end{cases} \tag{5}$$
where $R = (R_1,\ R_2, \ldots, R_m)$;

- Modified utilization
$$\begin{cases} u_1(x,y) = \sum_{j=1}^m R_j \max(x_j - y_j; 0), \\ u_2(x,y) = \sum_{j=1}^m R_j \max(y_j - x_j; 0); \end{cases} \tag{6}$$

- Capture
$$\begin{cases} u_1(x,y) = \sum_{j=1}^m R_j \cdot \mathbb{1}_{x_j > y_j} - p, \\ u_2(x,y) = \sum_{j=1}^m R_j \cdot \mathbb{1}_{y_j > x_j} - p. \end{cases} \tag{7}$$

Note that the utilization and capture cases are formulated as zero-sum games, i.e., $u_1 \equiv -u_2$, while the modified utilization is not.

Suppose each battlefield has its own unique terrain and requires a different approach to battle. Consequently, a designer does not want the players to skip any battlefield, but rather invest at least a bit of resource into each of them.

We will attempt to reach this goal by adjusting the values of the following parameters for all $i \in N$ and $j = 1, \ldots, m$:

- $R_j > 0$ - battlefield valuations;
- $c_j^i > 0$ - resource transportation costs.

This can be reformulated as the following strict aspect exclusion problem:

$$\begin{cases} A^i = \{a^i_j | j \in \overline{1,m}\}, & i \in N, \\ \alpha_i(a^i_j) = \{v | v \in X_i \wedge v_j = 0\}, & i \in N \wedge j \in \overline{1,m}, \\ SE(i, a^i_j, \mathfrak{b}) \equiv 1, & i \in N \wedge j \in \overline{1,m}. \end{cases} \tag{8}$$

There, as we don't have any prior information about players' preferences, the scheme $\mathfrak{b}$ may be either $\min[+]$ or $\min[\varepsilon]$ depending on the concrete problem statement. From there, we will consider both cases separately.

2.1 Utilization case

This case, having the linear payoff function $R(x-y)$ (or $R(y-x)$), corresponds to a situation where the battlefields do not yield immediate reward, but instead provide the winner an opportunity to make use of their resources (such as oil or other fossil fuels). After the battlefield was captured, the remainder of the troop gets involved in resource extraction. Hence, the reward is proportional to the number of units left and, in the simplest case, it is a linear function with no bias, whose slope is determined by the battlefield's valuation.

Recall that here we deal with a zero-sum game. When the battle for a certain battlefield ends with a decisive outcome, rather than just dealing with the consequences of the money being wasted for nothing, the losing side also bears the cost equal to the corresponding increase in the winner's payoff.

2.1.1 Rational behaviour

Below, we consider the $\min[+]$ behaviour scheme. First, note that

$$\underline{u}_1(x) = \min_{y \in Y} u_1(x,y) = \min_{y \in Y} R(x-y) = \min_{y \in Y} \Big[Rx - Ry \Big] = Rx - \max_{y \in Y} Ry.$$

The minimum of u_1 with respect to y is thus determined by the maximum of Ry and thus doesn't depend on x. This means that the formula for the maximin value

of u_1 can be decoupled:

$$\max_{x\in\alpha(a_j^1)} \underline{u}_1(x) = \max_{x\in\alpha(a_j^1)} \Big[Rx - \max_{y\in Y} Ry\Big] = \max_{x\in\alpha(a_j^1)} Rx - \max_{y\in Y} Ry. \tag{9}$$

The same logic may be used to obtain the maximin payoff for the second player:

$$\max_{y\in\alpha(a_j^2)} \underline{u}_2(y) = \max_{y\in\alpha(a_j^2)} Ry - \max_{x\in X} Rx.$$

The following proposition holds, for which the proof is presented in the appendix:

Proposition 1. *Each optimal strategy assigns positive values to the battlefields with the maximum reward-to-transportation-cost ratio only. Those values, multiplied by the respective transportation costs, sum up to 1. This means that, for optimal strategies, the budget of player i is fully distributed between battlefields j_k maximizing $R_{j_k}/c^i_{j_k}$.*

As a direct corollary of the proposition 1, there is always a strategy $v \in X_i$ maximizing Rv such that only one of its components is greater than zero (this can be any of the components, for which the ratio of the reward to the transportation cost is maximal). Therefore,

$$\max_{v\in\alpha(a_j^i)} Rv = \max_{v\in X_i} Rv = \max_j (R_j/c^i_j). \tag{10}$$

And thus

$$SE(i, a^i_j, \min[+]) \Leftrightarrow \max_j (R_j/c^i_j) < \max_j (R_j/c^{i'}_j),\ i' \neq i.$$

As a consequence, $SE(1, a^1_j, \min[+]) \wedge SE(2, a^2_j, \min[+]) \equiv 0$, implying that the problem (4), (5), (8) has no solution for worst-case rational behaviour scheme $\mathfrak{b} = \min[+]$.

2.1.2 Sub-optimal behaviour

Directly from (9) and (10) it follows that

$$\max_{x \in \alpha(a_j^1)} \underline{u}_1(x) = \max_{x \in X_1} \underline{u}_1(x).$$

And therefore,

$$SE(i, a_j^i, min[\varepsilon]) \equiv \max_{x \in \alpha(a_j^1)} \underline{u}_1(x) < \max_{x \in X_1} \underline{u}_1(x) - \varepsilon \equiv \varepsilon < 0 \equiv 0.$$

Consequently, for worst-case sub-optimal behaviour scheme b problem (4), (5), (8) doesn't have any solution as well.

2.2 Modified utilization case

The results obtained above lead to the question of whether the payoff function can be slightly altered to make it impossible to represent its maximin value in the decoupled form, as in (9). One logical way to construct such a function is to make the game non-antagonistic. Modified utilization payoff (6), considered in this section, achieves this by making the players care only about their rewards, but not the ones of their opponent.

Proposition 2. *For every strategy of the first player, there will be a second player's strategy delivering a minimum value to the first player's payoff which possesses the following property: for each battlefield, the amount of resources assigned is less than or equal to the amount assigned by the first player*

From this proposition, it follows that

$$\min_{y \in X_2} u_1(x, y) = \min_{y \in X_2} \sum_{j=1}^{m} \max(0; R_j(x_j - y_j)) = \min_{y \in X_2} \sum_{j=1}^{m} R_j(x_j - y_j) = \min_{y \in X_2} R(x - y).$$

From there, by following the same steps as in the previous section, it can easily be shown that problem (4), (6), (8) has no solution for both worst-case behaviour

schemes.

2.3 Capture case

The results that were achieved in the previous section imply that the payoff function needs more substantial modifications in case we want the strict aspect exclusion to be satisfied for both players simultaneously. We will abandon the previous interpretation of the payoff function in favour of the new one. Suppose the battlefields do not contain any useful resources, instead, they are the important vantage points granting the side which manages to capture one an immediate advantage that doesn't depend on the number of units left alive. This case is described by the payoff functions defined as in (7).

Let us define some supplementary sets. First, for convenience, a set of battlefields (referred to by their indexes) is introduced:

$$J = \{1, ..., m\}.$$

Then, we define a one-parametric family of sets. Each set $\mathcal{D}_a$ in this family will consist of all sets of battlefields D, such that control over these battlefields guarantees the payoff not less than a regardless of the outcome on the remaining ones.

$$\mathcal{D}_a = \Big\{ D \subset J \;\Big|\; \sum_{j \in D} R_j - p \geq a \Big\}.$$

These sets $\mathcal{D}_a$ possess a significant quality. If a player has a strategy that guarantees that, for any response of an opponent, the battlefields he or she manages to capture will constitute at least one of the sets in $\mathcal{D}_a$, this player will guarantee the payoff not less than a. An important point (and the main roadblock) here is that the strategy guaranteeing such payoff may alternate between those sets of captured battlefields depending on what strategy the opponent will use. The idea is that even if the opponent succeeds in preventing the player from capturing one of the sets in $\mathcal{D}_a$, there will be another set in $\mathcal{D}_a$ consisting of battlefields captured by the player.

One can also note that this family consists of sets nested inside each other. Indeed,

$$\sum_{j \in D} R_j - p \geq a \implies \forall a' < a \sum_{j \in D} R_j - p \geq a \geq a'.$$

From there,

$$\forall a, a' \colon a > a' \; D \in \mathcal{D}_a \Rightarrow D \in \mathcal{D}_{a'}.$$

In other words,

$$\forall a, \epsilon > 0 \; \mathcal{D}_{a-\epsilon} \supset \mathcal{D}_a \supset \mathcal{D}_{a+\epsilon}.$$

As the sets $\mathcal{D}_a$ demonstrate the possible combinations of the battlefields a player needs to capture to guarantee a certain payoff, the logical question arises: over which combinations of the battlefields an opponent needs to establish control to prevent a player from receiving the payoff not less than a? The next supplementary set is constructed to serve as an answer to this question:

$$\mathcal{T}'_a = \Big\{ T \subset J \;\Big|\; \forall D \in \mathcal{D}_a \quad T \cap D \neq \varnothing \Big\}.$$

Indeed, if, for any set $D \in \mathcal{D}_a$, there exists a battlefield in it that is not captured by the player, this player will not capture the battlefields guaranteeing at least a payoff. Rigorous proof for this fact is presented in the appendix.

It can be noted, though, that $\mathcal{T}'_a$, in a way it is constructed, is a bit of an overkill. If some set of battlefields belongs to it, then all of its supersets will also be among the elements of $\mathcal{T}'_a$. These supersets bear no additional relevant information. Therefore, *the core* of $\mathcal{T}'_a$, denoted as $\mathcal{T}_a$ may also be used:

$$\mathcal{T}_a = \Big\{ T \in \mathcal{T}'_a \;\Big|\; \nexists \hat{T} \in \mathcal{T}'_a \colon \hat{T} \neq T \wedge \hat{T} \subset T \Big\}. \tag{11}$$

Now we will present the important result, which allows us to compute the maximin values necessary for checking the strict aspect exclusion criteria.

Proposition 3.

$$\max_{x \in X} \min_{y \in Y} u_1(x, y) = \max(B_0),$$

$$\max_{x \in \alpha(a_j^1)} \min_{y \in Y} u_1(x, y) = \max(B_j),$$

*where B_0 is a set of all values of **a** for which the following system has a solution:*

$$\begin{cases} \sum\limits_{j \in T} c_j^y x_j > 1, \quad T \in \mathcal{T}_a, \\ x_j \geq 0, \quad j \in J, \\ \sum\limits_{j \in J} c_j^x x_j \leq 1, \end{cases} \tag{12}$$

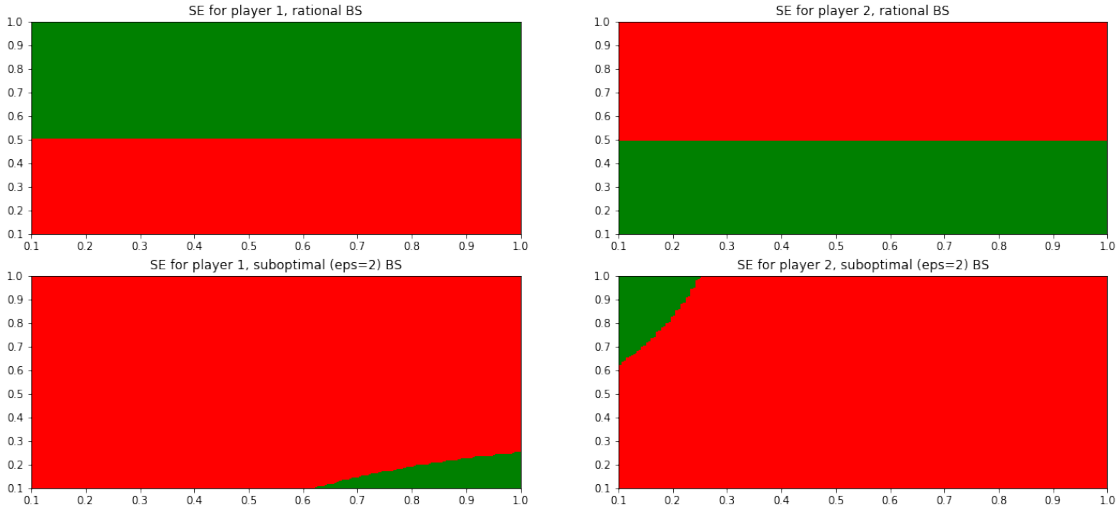
*and B_j is a set of all values of **a** for which there exists a solution of a system:*

$$\begin{cases} \sum\limits_{j \in T \setminus \{k\}} c_j^y x_j > 1, \quad T \in \mathcal{T}_a, \\ x_j \geq 0, \quad j \in J \setminus \{k\}, \\ \sum\limits_{j \in J \setminus \{k\}} c_j^x x_j \leq 1. \end{cases} \tag{13}$$

The algorithm for computing $SE(i, a_j^i, min[+])$ and $SE(i, a_j^i, min[\varepsilon])$ consists of several stages. First, as the number of possible payoff values is finite, every possible value of $\mathcal{T}_a$ is obtained and stored along with the maximum of the corresponding values of a. Next, the $m + 1$ maximin values are calculated by solving the linear problems outlined in the proposition. Finally, those maximin values are substituted into the formulas of the strict aspect exclusion criteria.

Both the algorithm (written in Python 3.9) and the proof of proposition 3 are presented in the appendix.

In the concrete case of two battlefields with valuations 3 and 5, epsilon threshold of suboptimal behaviour schemes equal to 2 and the transportation cost matrix with values on the main diagonal equal to 0.5 and the other two varying from 0.1 to 1, the following results were obtained with the usage of the constructed algorithm:



The possible values of c_2^x are plotted on the X-axes and the possible values of c_1^y are plotted on the Y-axes. The red areas correspond to the cost combinations for which the respective criterion doesn't hold. Conversely, in the green areas, the respective criterion is valid.

Conclusion

We have shown how some of the game balance problems can be reformulated as rigorous mathematical criteria. To do this, we introduced an improved restricted play principle, applicable for games with any number of players and not requiring simulation of the game instances. After that, using the flexible framework of normal-form games, we proposed two measures of strategy effectiveness: the worst-case and the average-case payoffs. To be able to work with the realistic behaviour of the players, who do not always play optimally, we introduced two reasonable behaviour schemes: rational and suboptimal behaviour. After that, we provided formal criteria that can be used to assess the balance of a game. These criteria were then illustrated using the concrete example: a continuous version of the Colonel Blotto game.

To the best of our knowledge, this thesis is novel in the application of formal methods to game balance problems. Other papers focused mainly on two subjects. The first one is performing manual statistical analysis on either real data collected using various methods (telemetry, public APIs, etc.) or data derived from the

matches between AI players. The second subject is automatic adjustment of the difficulty in single-player games. On the contrary, this thesis proposes an approach that doesn't require manual analysis and is applicable to multiplayer games.

We believe that further work on the formalization of game balance problems may lead to significant improvements in the field of playtesting. The criteria proposed in this thesis allow for the development of automatic testing systems, which will both reduce the amount of manual playtesting needed and provide a way to monitor matches between real players to be able to react in time in case some design flaws went unnoticed. This will allow designers to focus on high-level goals while parameter tweaking is being done automatically.

Appendix

Proof of Proposition 1. We prove that

$$\forall i \in N \ \operatorname*{argmax}_{v \in X_i} Rv = \Big\{ v^* \in X_i \Big| \sum_{j \in \operatorname{argmax}(R_j / c_j^i)} c_j^i v_j^* = 1 \Big\}.$$

We will prove the proposition above for the strategies of the second player. The proof for the first player's strategies can be derived analogically.

Let

$$Y^* = \Big\{ y^* \in X_2 \Big| \sum_{j \in \operatorname{argmax}(R_j / c_j^y)} c_j^y y_j^* = 1 \Big\}.$$

Then, for $y \in X_2$

$$y \notin Y^* \implies \sum_{j \in \operatorname{argmax}(R_j / c_j^y)} c_j^y y_j \neq 1 \overset{y \in X_2}{\Longrightarrow} \sum_{j \in \operatorname{argmax}(R_j / c_j^y)} c_j^y y_j < 1 \implies$$

$$\implies \left[\begin{array}{l} \displaystyle\sum_{j=1}^{m} c_j^y y_j < 1, \\ \displaystyle\sum_{j \notin \operatorname{argmax}(R_j/c_j^y)} c_j^y y_j > 0. \end{array}\right.$$

In the first case,

$$\sum_{j=1}^{m} c_j^y y_j < 1 \implies y \notin \operatorname*{argmax}_{y \in X_2} Ry,$$

since all valuations are positive ($R_j > 0$) and thus an increase in any of the components y_s by $\frac{1-\sum_{j=1}^{m} c_j^y y_j}{c_s^y}$ will yield strictly greater payoff while the whole strategy y will still be inside X_2.

In the second case,

$$\sum_{j \notin \operatorname{argmax}(R_j/c_j^y)} c_j^y y_j > 0 \implies \exists j_1 \notin \operatorname{argmax} \frac{R_j}{c_j^y} \colon y_{j_1} > 0 \implies$$

$$\implies \exists j_1, j_2 \colon y_{j_1} > 0 \wedge \frac{R_{j_2}}{c_{j_2}^y} > \frac{R_{j_1}}{c_{j_1}^y}.$$

This, once again, implies that $y \notin \operatorname{argmax}_{y \in X_2} Ry$ because there exists a strategy y' defined as

$$\begin{cases} y'_j = 0, & j = j_1 \\ y'_j = y_{j_1} + y_{j_2}, & j = j_2 \\ y'_j = y_j & \text{otherwise,} \end{cases}$$

which guarantees strictly greater payoff than y does.

To sum up,

$$y \notin Y^* \implies y \notin \operatorname*{argmax}_{y \in X_2} Ry.$$

In other words,

$$\operatorname*{argmax}_{y \in X_2} Ry \subset Y^*.$$

On the other hand,

$$\forall y^* \in Y^* \quad Ry^* = \sum_{j \in \operatorname{argmax}(R_j/c_j^y)} R_j y_j^* = \sum_{j \in \operatorname{argmax}(R_j/c_j^y)} R_j y_j^* \frac{c_j^y}{c_j^y} =$$

$$= (\max_j \frac{R_j}{c_j^y}) \cdot (\sum_{j \in \operatorname{argmax}(R_j/c_j^y)} c_j^y y_j^*) = \max_j \frac{R_j}{c_j^y} = const.$$

Therefore,

$$\operatorname*{argmax}_{y \in X_2} Ry = Y^*.$$

Proof of Proposition 2. We prove that

$$\forall x \in X_1 \; \exists y \in \operatorname*{argmin}_{y \in X_2} u_1(x, y) \colon \forall j \; y_j \leq x_j.$$

Since X_2 is compact,

$$\forall x \in X_1 \; \exists y \in \operatorname*{argmin}_{y \in X_2} u_1(x, y).$$

Assume $\exists i \colon y_i > x_i$. Let

$$\hat{y} = (\min(x_1; y_1), ..., \min(x_m; y_m)).$$

By construction,

$$\forall j \quad \hat{y}_j \leq x_j.$$

Obviously,

$$\hat{y} \in X_2.$$

And

$$u_1(x, \hat{y}) = \sum_{j=1}^{m} \max(0; R_j(x_j - \min(x_i; y_i))) = \sum_{j=1}^{m} \max(0; R_j \max(0; x_i - y_i))) =$$

$$= \sum_{j=1}^{m} \max(0; R_j(x_j - y_j)) = u_1(x, y) \implies \hat{y} \in \operatorname*{argmin}_{y \in X_2} u_1(x, y).$$

Proof of Proposition 3. Before presenting the proof, we need to formulate several auxiliary lemmas. Recall that $\mathcal{T}'_a$ is a one-parametric family of the sets of battlefields such that capturing every battlefield in such set guarantees that the opponent's payoff will not exceed a, and $\mathcal{T}_a$ is the core of $\mathcal{T}'_a$.

Lemma 2. *For every set of battlefields in $\mathcal{T}'_a$ there exists at least one subset in $\mathcal{T}_a$*

Proof. The lemma's statement may be written as

$$\forall T' \in \mathcal{T}'_a \quad \exists T \in \mathcal{T}_a : \quad T \subset T'.$$

We carry out the proof by contradiction. Suppose

$$\exists T^1 \in \mathcal{T}'_a : \quad \forall T \in \mathcal{T}_a \quad T \not\subset T^1.$$

As $T^1 \in \mathcal{T}'_a$, $T_1 \subset J$ and thus $|T^1| \leq |J| = m$. On the other hand, as $T^1 \subset T^1$,

$$T^1 \notin \mathcal{T}_a \quad \overset{(11)}{\Longrightarrow} \quad \exists T^2 \in \mathcal{T}'_a : \quad T^2 \neq T^1 \wedge T^2 \subset T^1.$$

From $T^2 \neq T^1 \wedge T^2 \subset T^1$ it follows that

$$|T^2| < |T^1| \leq m \Rightarrow |T^2| \leq m - 1.$$

$T^2 \in \mathcal{T}_a$ contradicts the initial assumption as $T^2 \subset T^1$, therefore

$$T^2 \notin \mathcal{T}_a \quad \overset{(11)}{\Longrightarrow} \quad \exists T^3 \in \mathcal{T}'_a : \quad T^3 \neq T^2 \wedge T^3 \subset T^2.$$

Here, by the same logic, $|T^3| \leq m - 2$ and

$$T^3 \subset T^2 \subset T^1 \Rightarrow T^3 \notin \mathcal{T}_a.$$

After $m+1$ steps one obtains

$$\exists T^{m+1} \in \mathcal{T}'_a \colon |T^{m+1}| \leq 0 \Rightarrow T^{m+1} = \varnothing.$$

But then

$$T^{m+1} = \varnothing \Rightarrow \forall D \in \mathcal{D}_a \quad T^{m+1} \cap D = \varnothing \Rightarrow T^{m+1} \notin \mathcal{T}'_a ?!$$

This contradiction refutes the initial assumption, thus

$$\forall T' \in \mathcal{T}'_a \quad \exists T \in \mathcal{T}_a \colon \quad T \subset T'.$$

□

Lemma 3. *If for every combination of battlefields in $\mathcal{D}_a$ there exists a battlefield not captured by the first player, then there exists a combination in $\mathcal{T}_a$ consisting solely of the battlefields player 1 failed to capture. The reverse is true as well.*

Proof. We can rewrite the lemma's statement as

$$\forall D \in \mathcal{D}_a \quad \exists j \in D \colon \; y_j \geq x_j \quad \Leftrightarrow \quad \exists T \in \mathcal{T}_a \colon \; \forall j \in T \quad y_j \geq x_j.$$

First, let us prove that

$$\forall D \in \mathcal{D}_a \quad \exists j_D \in D \colon \; y_j \geq x_j \quad \Rightarrow \quad \exists T \in \mathcal{T}_a \colon \; \forall j \in T \quad y_j \geq x_j.$$

Assume

$$\forall D \in \mathcal{D}_a \quad \exists j_D \in D \colon \; y_j \geq x_j.$$

Let $T' = \{j_D \mid D \in \mathcal{D}_a\}$. Then

$$\forall D \in \mathcal{D}_a \quad j_D \in T' \cap D \Rightarrow T' \cap D \neq \varnothing.$$

Consequently, by construction

$$T' \in \mathcal{T}'_a.$$

As a result,

$$\exists T' \in \mathcal{T}'_a\colon \quad \forall j \in T' \quad y_j \geq x_j.$$

But due to Lemma 2,

$$\exists T \in \mathcal{T}_a\colon \quad T \subset T'.$$

That is,

$$\exists T \in \mathcal{T}_a\colon \quad \forall j \in T \quad y_j \geq x_j.$$

Now, let us prove that

$$\exists T \in \mathcal{T}_a\colon\ \forall j \in T \quad y_j \geq x_j \quad \Rightarrow \quad \forall D \in \mathcal{D}_a \quad \exists j \in D\colon\ y_j \geq x_j.$$

If

$$\exists T \in \mathcal{T}_a\colon\ \forall j \in T \quad y_j \geq x_j.$$

Then

$$\exists T\colon\ \forall D \in \mathcal{D}_a \quad T \cap D \neq \varnothing\ \wedge\ \forall j \in T \quad y_j \geq x_j.$$

In particular,

$$\exists T\colon\ \forall D \in \mathcal{D}_a \quad T \cap D \neq \varnothing\ \wedge\ \forall j \in T \cap D \quad y_j \geq x_j.$$

And so

$$\forall D \in \mathcal{D}_a \quad \exists j \in D\colon\ y_j \geq x_j,$$

Which proves the lemma's statement in both directions. □

Lemma 4. *For any combination of battlefields T the necessary and sufficient condition for the existence of the second player's strategy that will make the first player surrender every battlefield in T is $\sum_{j \in T} c_j^y x_j \leq 1$*

Proof. We prove that

$$\exists y \in X_2\colon\ \forall j \in T \quad y_j \geq x_j \quad \Longleftrightarrow \quad \sum_{j \in T} c_j^y x_j \leq 1.$$

Suppose

$$\sum_{j \in T} c_j^y x_j \leq 1.$$

Let y be such that

$$\forall j \in J \quad y_j = \begin{cases} x_j, & j \in T, \\ 0, & j \notin T. \end{cases}$$

Then

$$c^y y \leq 1 \quad \wedge \quad \forall j \in T \quad y_j = x_j \geq x_j.$$

This means that the desired strategy y was found and the sufficiency is proven.

Now, suppose

$$\exists y \in X_2 \colon \ \forall j \in T \quad y_j \geq x_j.$$

Then, from the definition of X_2,

$$1 \geq \sum_{j \in J} c_j^y y_j \geq \sum_{j \in T} c_j^y y_j \geq \sum_{j \in J} c_j^y x_j.$$

This immediately proves the necessity. □

Now we are ready to present a proof of Proposition 3.

Proof of Proposition 3.

$$\max_{x \in A} \min_{y \in X_2} u_1(x, y) = M \Rightarrow \exists x \in A \colon \forall y \in X_2 \ u_1(x, y) \geq M \Rightarrow$$

$$\Rightarrow \exists x \in A \colon \forall y \in X_2 \ \sum_{j=1}^{m} R_j \cdot \mathbb{1}_{x_j > y_j} - p \geq M \Rightarrow$$

$$\Rightarrow \exists x \in A \colon \forall y \in X_2 \ \sum_{x_j > y_j} R_j - p \geq M \Rightarrow$$

$$\Rightarrow \exists x \in A \colon \forall y \in X_2 \; \exists D \in \mathcal{D}_M \colon \bigwedge_{j \in D} x_j > y_j.$$

On the other hand,

$$\max_{x \in A} \min_{y \in X_2} u_1(x, y) = M \Rightarrow \nexists x \in A \colon \forall y \in X_2 \; u_1(x, y) > M \Rightarrow$$

$$\Rightarrow \nexists M' > M : \exists x \in A \colon \forall y \in X_2 \; u_1(x, y) \geq M' \Rightarrow$$

$$\Rightarrow \nexists M' > M : \exists x \in A \colon \forall y \in X_2 \; \exists D \in \mathcal{D}_{M'} \colon \bigwedge_{j \in D} x_j > y_j.$$

Thus,

$$\max_{x \in A} \min_{y \in X_2} u_1(x, y) = \max \left(\left\{ a \middle| \exists x \in A \colon \forall y \in X_2 \; \exists D \in \mathcal{D}_a \colon \bigwedge_{j \in D} x_j > y_j \right\} \right). \tag{14}$$

There,

$$\exists x \in A \colon \quad \forall y \in X_2 \quad \exists D \in \mathcal{D}_a \colon \bigwedge_{j \in D} x_j > y_j \Leftrightarrow$$

$$\Leftrightarrow \neg \Big(\forall x \in A \quad \exists y \in X_2 \colon \forall D \in \mathcal{D}_a \quad \exists j \in D \colon y_j \geq x_j \Big) \overset{Lemma\ 3}{\Longleftrightarrow}$$

$$\overset{Lemma\ 3}{\Longleftrightarrow} \neg \Big(\forall x \in A \quad \exists y \in X_2 \colon \; \exists T \in \mathcal{T}_a \colon \; \forall j \in T \quad y_j \geq x_j \Big) \Leftrightarrow$$

$$\Leftrightarrow \neg \Big(\forall x \in A \quad \exists T \in \mathcal{T}_a \colon \; \exists y \in X_2 \colon \; \forall j \in T \quad y_j \geq x_j \Big) \overset{Lemma\ 4}{\Longleftrightarrow}$$

$$\overset{Lemma\ 4}{\Longleftrightarrow} \neg \Big(\forall x \in A \quad \exists T \in \mathcal{T}_a \colon \; \sum_{j \in T} c_j^y x_j \leq 1 \Big) \Leftrightarrow$$

$$\Leftrightarrow \exists x \in A \colon \; \forall T \in \mathcal{T}_a \quad \sum_{j \in T} c_j^y x_j > 1 \Leftrightarrow$$

$$\Leftrightarrow \exists x \in A\colon \bigwedge_{T\in\mathcal{T}_a} \sum_{j\in T} c_j^y x_j > 1 \Leftrightarrow$$

$$\Leftrightarrow \left(x \in A \wedge \bigwedge_{T\in\mathcal{T}_a} \sum_{j\in T} c_j^y x_j > 1\right) \text{ has a solution.}$$

For $A = X$,

$$x \in A \wedge \bigwedge_{T\in\mathcal{T}_a} \sum_{j\in T} c_j^y x_j > 1 \Leftrightarrow \begin{cases} \sum_{j\in T} c_j^y x_j > 1, & T \in \mathcal{T}_a, \\ x_j \geq 0, & j \in J, \\ \sum_{j\in J} c_j^x x_j \leq 1. \end{cases}$$

For $A = \alpha(a_j^1)$,

$$x \in A \wedge \bigwedge_{T\in\mathcal{T}_a} \sum_{j\in T} c_j^y x_j > 1 \Leftrightarrow \begin{cases} \sum_{j\in T\setminus\{k\}} c_j^y x_j > 1, & T \in \mathcal{T}_a, \\ x_j \geq 0, & j \in J \setminus \{k\}, \\ \sum_{j\in J\setminus\{k\}} c_j^x x_j \leq 1. \end{cases}$$

Combined with (14), this leads directly to the formulation of the proposition.

Algorithm for checking SE criteria in Continuous Colonel Blotto, capture case
The algorithm is written in Python 3.9 and makes use of several libraries. *numpy* and *scipy* are used to solve the linear programming problems. Some data structures, such as SortedSet and SortedDict come from the *sortedcontainers* library. Lastly, *matplotlib* is used to plot the result for the concrete case.

```
import matplotlib.pyplot as plt
import math
from itertools import chain, combinations
from sortedcontainers import SortedSet, SortedDict
from collections import defaultdict
import numpy as np
from scipy.optimize import linprog
```

First, a utility function *powerset* computing the power set of a set is defined.

```
def powerset(orig_list):
    return chain.from_iterable(combinations(orig_list, r) for r in␣
 ↪range(len(orig_list)+1))
```

As was stated before, the set of all possible payoff values u_i is finite. *payoff_to_combinations_mapping* function computes these values. Moreover, for every such value, it also finds all of the battlefield combinations which, when captured, guarantee that the payoff will not be less than this value (but do not guarantee any higher possible value). The results are stored in the dictionary *mapping* with its keys being the possible payoff values and the values being the respective sets of battlefield combinations. This dictionary is sorted in descending order of its keys.

The function, as well as some of the following ones, takes two arguments: a list of real numbers R and a single real number p, which are the parameters defining the explored payoff function.

```
def payoff_to_combinations_mapping(R, p):
    mapping = SortedDict(lambda x: -x)

    for combination in powerset(list(range(len(R)))):
        payoff = sum([R[k] for k in combination]) - p
        if payoff not in mapping:
            mapping[payoff] = []
        mapping[payoff].append(combination)

    return mapping
```

Likewise, *a_to_Da_mapping* function computes all possible values of $\mathcal{D}_a$ and maps them to the corresponding values of a such that for any key a in the resulting mapping and the sufficiently small ϵ, $\mathcal{D}_a = \mathcal{D}_{a-\epsilon}$.

```
def a_to_Da_mapping(R, p):
    mapping = SortedDict()
    accumulated_combinations = []

    for a, combinations in payoff_to_combinations_mapping(R, p).items():
        accumulated_combinations += combinations
        mapping[a] = accumulated_combinations.copy()

    return mapping
```

By analogy, the mappings for $\mathcal{T}'_a$ and then $\mathcal{T}_a$ are calculated by their definitions in the functions *a_to_Taprime_mapping* and *a_to_Ta_mapping* respectively.

```
def a_to_Taprime_mapping(R, p):
    mapping = SortedDict()

    for a, Da in a_to_Da_mapping(R, p).items():
        mapping[a] = []
        for p2_combination in powerset(list(range(len(R)))):
            belongs_to_Ta = True
            p2_comb_set = set(p2_combination)
            for p1_combination in Da:
                if not (set(p1_combination) & p2_comb_set):
                    belongs_to_Ta = False
                    break
            if belongs_to_Ta:
                mapping[a].append(p2_comb_set)

    return mapping
```

```
def a_to_Ta_mapping(R, p):
    a_to_Taprime = a_to_Taprime_mapping(R, p)
    mapping = SortedDict()

    for a, Taprime in a_to_Taprime.items():
        mapping[a] = []
        for set1 in Taprime:
            included = True
            for set2 in Taprime:
                if set2 < set1:
                    included = False
                    break
            if included:
                mapping[a].append(set1)

    return mapping
```

Finally, *lp_has_solution* attempts to solve the linear programming problem equivalent for the feasibility problems 12 and 13. The first argument is the value of $\mathcal{T}_a$. The second and the third argument are the lists of all transportation costs for the first and the second player respectively (it is assumed that the lengths of these lists are equal to the total number of battlefields and for each index, the corresponding list elements will be equal to the transportation costs to the **same** battlefield). The last argument defines the index of the battlefield to be excluded (and equals *None* in the case of problem 12).

```
def lp_has_solution(Ta, cx, cy, k):
    c=np.append(np.zeros_like(cx), [-1])
    A = np.array([
        [-cy[j] if j in combination and (k is None or j != k) else 0 for j in␣
 ↪range(len(cx))] + [0] for combination in Ta
    ] + [
        [cx[j] if k is None or j != k else 0 for j in range(len(cx))] + [0]
    ])
    b = np.array([-1 for comb in Ta] + [1])
    bounds = np.array([(0, None) for j in range(len(cx))] + [(0, 1)])
    res = linprog(c=c, A_ub=A, b_ub=b, bounds=bounds)
    return res.status == 0
```

Function *calc_maxmins* calculates maximin values used in the expressions of the strict aspect exclusion criteria given the mapping for values of $\mathcal{T}_a$ computed using *a_to_Ta_mapping* function.

```
def calc_maxmins(a_to_Ta, cx, cy):
    max_unrestricted_a = a_to_Ta.keys()[0]
    max_restricted_a = max_unrestricted_a

    unrestricted_infeasible = False
    restricted_infeasible = [False for k in range(len(cx))]

    for a, Ta in a_to_Ta.items()[1:]:
        if not unrestricted_infeasible:
            if lp_has_solution(Ta, cx, cy, None):
                max_unrestricted_a = a
            else:
                unrestricted_infeasible = True
        for k in range(len(cx)):
            if not restricted_infeasible[k]:
                if lp_has_solution(Ta, cx, cy, k):
                    max_restricted_a = a
                else:
                    restricted_infeasible[k] = True

    return (max_unrestricted_a, max_restricted_a)
```

Lastly, *compute_se_values* checks the validity of the strict aspect exclusion criteria for both rational and suboptimal reasonable behaviour schemes, for each player. Concrete suboptimal behaviour scheme to be used is defined using the value ϵ passed as an argument.

```python
def compute_se_values(R, p, cx, cy, eps):
    a_to_Ta = a_to_Ta_mapping(R, p)

    unrestricted_1p_maxmin, restricted_1p_maxmin = calc_maxmins(a_to_Ta, cx, cy)
    unrestricted_2p_maxmin, restricted_2p_maxmin = calc_maxmins(a_to_Ta, cy, cx)

    se_1p_rational = restricted_1p_maxmin < 0
    se_1p_suboptimal = restricted_1p_maxmin < unrestricted_1p_maxmin - eps
    se_2p_rational = restricted_2p_maxmin < 0
    se_2p_suboptimal = restricted_2p_maxmin < unrestricted_2p_maxmin - eps

    return {
        "se_1p_rational": se_1p_rational,
        "se_1p_suboptimal": se_1p_suboptimal,
        "se_2p_rational": se_2p_rational,
        "se_2p_suboptimal": se_2p_suboptimal
    }
```

To plot the areas of validity of the abovementioned criteria in the concrete case of two battlefields with valuations $R_1 = 3$ and $R_2 = 5$, payoff penalty $p = 4$ and transportation costs $c_1^x = c_2^y = 0.5$, $c_2^x, c_1^y \in [0.1, 1]$, the following code is used. For suboptimal behaviours schemes ϵ is set to 2.

```
R = [3, 5]
p = 4
eps = 2

values1p = []
values2p = []
values1pso = []
values2pso = []

x_values = np.linspace(0.1, 1, 100)
y_values = np.linspace(0.1, 1, 100)
M1, M2 = np.meshgrid(x_values, y_values)

for cx2 in x_values:
    row1p = []
    row2p = []
    row1pso = []
    row2pso = []
    for cy1 in y_values:
        result = compute_se_values(R, p, [0.5, cx2], [cy1, 0.5], eps)
        row1p.append(result["se_1p_rational"])
        row2p.append(result["se_2p_rational"])
        row1pso.append(result["se_1p_suboptimal"])
        row2pso.append(result["se_2p_suboptimal"])
    values1p.append(row1p)
    values2p.append(row2p)
    values1pso.append(row1pso)
    values2pso.append(row2pso)

plt.figure(1, [18, 8])

plt.subplot(221)
plt.gca().set_title(f'SE for player 1, rational BS')
plt.contourf(M1, M2, values1p, 1, colors=['red', 'green'])

plt.subplot(222)
plt.gca().set_title(f'SE for player 2, rational BS')
plt.contourf(M1, M2, values2p, 1, colors=['red', 'green'])

plt.subplot(223)
plt.gca().set_title(f'SE for player 1, suboptimal (eps={eps}) BS')
plt.contourf(M1, M2, values1pso, 1, colors=['red', 'green'])

plt.subplot(224)
plt.gca().set_title(f'SE for player 2, suboptimal (eps={eps}) BS')
plt.contourf(M1, M2, values2pso, 1, colors=['red', 'green'])

plt.show()
```

References

[1] Aponte, Maria-Virginia and Levieux, Guillaume and Natkin, Stephane. "Measuring the level of difficulty in single player video games." *Entertainment Computing* 2.4 (2011): 205-213.

[2] Jaffe, Alexander Benjamin. *Understanding game balance with quantitative methods*. PhD diss., 2013.

[3] Jaffe, Alexander and Miller, Alex and Andersen, Erik and Liu, Yun-En and Karlin, Anna and Popovic, Zoran. "Evaluating competitive game balance with restricted play." *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment.* 8.1 (2012): 26-31.

[4] Leyton-Brown, Kevin and Shoham, Yoav. "Essentials of game theory: A concise multidisciplinary introduction." *Synthesis lectures on artificial intelligence and machine learning* 2.1 (2008): 1-88.

[5] Petrosyan, Leon A and Zenkevich, Nikolay A. *Game theory*. World Scientific, 1996.

[6] Obreja, Dragoş M. "Postphenomenology, Kill Cams and Shooters: Exploring the Code of Replay Sequences." *Games and Culture* 18.3 (2022): 267–282.

[7] Andrade, Gustavo and Ramalho, Geber and Gomes, Alex and Corruble, Vincent. "Dynamic game balancing: An evaluation of user satisfaction." *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* 2.1 (2006): 3-8.

[8] Becker, Alexander and Görlich, Daniel. "What is game balancing?-an examination of concepts." *ParadigmPlus* 1.1 (2020): 22-41.

[9] Pusey, Portia and Tobey Sr, David and Soule, Ralph. "An argument for game balance: Improving student engagement by matching difficulty level with learner readiness." *2014 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 14)*. 2014.

[10] Ashton, Martin and Verbrugge, Clark. "Measuring the Impact of Cooperation in Halo: Reach." (2012).

[11] Tomašev, Nenad and Paquet, Ulrich and Hassabis, Demis and Kramnik, Vladimir. "Assessing game balance with AlphaZero: Exploring alternative rule sets in chess." *arXiv preprint arXiv:2009.04374* (2020).

[12] Khan, Sulman. "Predicting Customer Churn in World of Warcraft." *arXiv preprint arXiv:2006.15735* (2020).

[13] Breuer, Kelvin. *Competitive Balance Through Diversity and Technological Ambidexterity: A Case Study of Magic: The Gathering*. Master's Thesis, 2020.

[14] Drachen, Anders and Sifa, Rafet and Bauckhage, Christian and Thurau, Christian. "Guns, swords and data: Clustering of player behavior in computer games in the wild." *2012 IEEE conference on Computational Intelligence and Games (CIG)*. IEEE, 2012.

[15] Traish, Jason and Tulip, James and Moore, Wayne. "Data Collection with Screen Capture." *Artifical Intelligence and Simulation of Behaviour Convention* (2015): 1–4.

[16] Semenov, Aleksandr and Romov, Peter and Korolev, Sergey and Yashkov, Daniil and Neklyudov, Kirill. "Performance of machine learning algorithms in predicting game outcome from drafts in dota 2." *Analysis of Images, Social Networks and Texts: 5th International Conference, AIST 2016, Yekaterinburg, Russia, April 7-9, 2016, Revised Selected Papers 5* (2017): 26–37.

[17] Makin, Owen and Bangay, Shaun. "Orthogonal analysis of StarCraft II for game balance." *Proceedings of the Australasian Computer Science Week Multiconference* (2017): 1–4.

[18] Ludwig, Jeremy and Farley, Art. "A learning infrastructure for improving agent performance and game balance." *Proceedings of the AIIDE 7* (2007): 7–12.

[19] Olesen, Jacob Kaae and Yannakakis, Georgios N and Hallam, John. "Real-

time challenge balance in an RTS game using rtNEAT." *2008 IEEE Symposium On Computational Intelligence and Games* (2008): 87–94.

[20] Shaker, Noor and Shaker, Mohammad and Togelius, Julian. "Evolving playable content for cut the rope through a simulation-based approach." *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* 9.1 (2013): 72-78.

[21] Togelius, Julian and Nelson, Mark J and Liapis, Antonios. "Characteristics of generatable games." *Workshop on Procedural Content Generation for Games*. ACM, 2014.

[22] Pfau, Johannes and Smeddinck, Jan David and Malaka, Rainer. "Towards deep player behavior models in mmorpgs." *Proceedings of the 2018 Annual Symposium on Computer-Human Interaction in Play* (2018): 381-392.

[23] Volz, Vanessa and Rudolph, Günter and Naujoks, Boris. "Demonstrating the feasibility of automatic game balancing." *Proceedings of the Genetic and Evolutionary Computation Conference 2016* (2016): 269-276.

[24] Hernandez, Daniel and Gbadamosi, Charles Takashi Toyin and Goodman, James and Walker, James Alfred. "Metagame Autobalancing for Competitive Multiplayer Games." *2020 IEEE Conference on Games (CoG)* (2020): 275-282.

[25] Xia, Wen and Anand, Bhojan. "Game balancing with ecosystem mechanism." *2016 international conference on data mining and advanced computing (SAPIENCE)* (2016): 317-324.

[26] Zook, Alexander and Fruchter, Eric and Riedl, Mark O. "Automatic playtesting for game parameter tuning via active learning." *arXiv preprint arXiv:1908.01417* (2019).