JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 37 (2025) 500–507
Received: 11 September 2025
Accepted: 12 October 2025

Analysis of methods for simulating character encounters in a game with RPG elements

Analiza metod symulowania potyczek postaci w grze z elementami RPG

Michał Zdybel*, Jakub Smołka

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract
This paper investigates algorithms that predict the outcome of a duel in a game with RPG elements and determine the losses incurred. The aim is to evaluate the effectiveness of the following approaches: based on Lanchester's laws and stochastic, using the Monte Carlo method. Verification was carried out through manual gameplay and comparison of the obtained results with those predicted by the algorithms, measuring their accuracy with the MAPE. The analysis showed greater efficiency and stability of the Monte Carlo algorithm, while the Lanchester model turned out to be less reliable in one of the cases.

Keywords: algorithms; Lanchester's laws; Monte Carlo; games with RPG elements; quick combat

Streszczenie
W niniejszej pracy badane są algorytmy przewidujące wynik pojedynku w grze z elementami RPG i wyznaczające poniesione straty. Celem jest ocena skuteczności podejść: opartego na prawach Lanchestera oraz stochastycznego, wykorzystującego metodę Monte Carlo. Przeprowadzono weryfikację poprzez rozgrywki manualne i porównanie uzyskanych wyników z przewidzianymi przez algorytmy, mierząc ich dokładność wskaźnikiem MAPE. Analiza wykazała większą skuteczność i stabilność algorytmu Monte Carlo, podczas gdy model Lanchestera okazał się być mniej wiarygodny w jednym z przypadków.

Słowa kluczowe: algorytmy; Prawa Lanchestera; Monte Carlo; gra z elementami RPG; szybki pojedynek

*Corresponding author
Email address: s95620@pollub.edu.pl (M. Zdybel)

Published under Creative Common License (CC BY 4.0 Int.)

1. Introduction

The subject of the study is a game, one of whose elements is a duel between two players - each of them commands between one and ten units. The gameplay takes place in turns. Based on the initiative statistic of each unit, an action order is created for every turn. Based on this list, the right to make a move is given to one of the players. The active player chooses one of own units and then decides, based on unit position on the hexagonal grid, what to do. Unit can move or attack hostile unit. The range of each unit is limited to adjacent tiles (Figure 1).

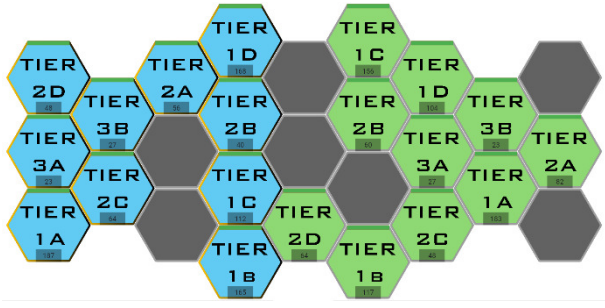

Figure 1: Representation of the battlefield in the game - a duel between the blue and green player.

The exception is ranged units - they can attack any enemy unit, provided that no enemy unit is located on an adjacent hexagon. Otherwise, the unit can only perform a melee attack. Ranged attacks avoid counterattacks. Each unit can only perform one action per turn.

Each unit can be described by the following characteristics:

- creature (its statistics are described below),
- stack number (also called count) (determining how many creatures are in a unit; the greater the stack number, the more damage it can deal at a moment).

Each creature is described by its own statistics. Statistics include:

- attack (appears in two forms as two separate stats -for physical and magical sources, it determines the average base damage range for each source. The attack stat value is only indicative for players; thresholds are used in calculations).
- defense (similar to attack, there are two forms of this statistic - for physical and magical sources. Defense refers to the reduction of taken damage - it is expressed as a percentage),
- chance of critical damage (expressed as a percentage),
- range (range for ranged creatures, which, in addition to melee attacks, can attack from a distance as long as they are not adjacent to an enemy unit),
- hit points (determine the health of a creature in a unit),
- tier (from the weakest level 1 to the strongest level 3, it indicates the rarity of the creature - higher levels are characterized by higher statistics values. The tier

value itself has no effect on calculations, but it is synonymous with the height of the statistics).

There is a significant randomness factor in the game, so the results of a duel can vary greatly, even with identical starting data. Each player initially has their own side of the board, consisting of thirteen adjacent hexagons (Figure 1). The player's units are randomly placed on their side of the board. Unused hexagons remain empty but are available for movement by both sides. The central field is neutral and cannot be occupied during the starting placing.

Each creature has hidden statistics (invisible to players) that determine physical attack value, magical attack value, and damage dealt. These statistics define the upper and lower thresholds of possible damage. For physical sources, the minimum physical damage F_{min} and maximum physical damage F_{max} are used. Similarly, for magical sources, there are the minimum magical damage M_{min} and maximum magical damage M_{max}.

While performing an attack, the ranges for damage from physical sources (minimum physical damage PF_{min} and maximum physical damage PF_{max} (formula 1) and magical sources (the formulas are analogous) are calculated.

$$PF_{min} = L \cdot F_{min} \cdot m$$
$$PF_{max} = L \cdot F_{max} \cdot m \quad (1)$$

Where L is the stack number and m is the modifier. The modifier means how the current situation of the unit affects its strength - a value of 1 occurs when the unit attacks in melee or at full range. In the case of a counterattack, the modifier takes the value 0.7, and in the case of not full range, the value is 0.5. Using these calculations, the system obtains the range of possible damage from a certain source. Analogous formulas are used for magical sources.

Then, system calculates thresholds for possible damage. The formula is determined for minimum modified physical damage OF_{min}, taking into account the defense D_F of the defending unit (formula 2).

$$OF_{min} = PF_{min} \cdot (\frac{100 - D_F}{100}) \quad (2)$$

In this way, the system obtains thresholds adapted to the current situation in the game. A similar procedure is performed for the upper threshold of physical damage and for the magical source.

After performing these calculations, the system has the following variables at its disposal: lower threshold for physical damage OF_{min}, upper threshold for physical damage OF_{max}, lower threshold for magic damage OM_{min}, upper threshold for magic damage OM_{max}. Based on these, the system is able to calculate the minimum damage O_{min} and maximum damage O_{max}, without dividing them by source (formula 3).

$$\boldsymbol{O_{min} = OF_{min} + OM_{min}}$$
$$\boldsymbol{O_{max} = OF_{max} + OM_{max}} \quad (3)$$

These values determine the range of possible damage. A value is randomly selected and subtracted from the defending unit's hit points pool. A possible critical hit (the chance of which is determined by the unit's statistics) increases the damage by 25%. After a melee attack, the units switch places. The duel ends when the number of all units of one player drops to zero.

The hypothesis of this paper says that one of the developed algorithms is able to predict the outcome of a duel between two players.

2. Literature review

Lanchester's laws, which use systems of differential equations, describe how to calculate the outcome of a battle between units in armed conflict. These laws refer to the existence of two units of a specific size and strength, and how these parameters affect the outcome of the battle [1].

R. Hoffmann and T. Protasowicki propose a new perspective on the application of Lanchester's laws in the context of contemporary conflicts [1]. The aim of their work was to build a dynamic combat model based on Lanchester's equations, to identify and propose a new perspective on the laws. The goal was to enable many other researchers to develop this topic. The authors mention and indicate directions that could be subject for future research, including the influence of random factors on the course of battles taking place today. Hoffmann and Protasowicki analyze the already known Lanchester's laws, taking into account both the square variant and two linear variants [1]. These laws are still being analyzed and modified today in order to achieve desirable and realistic results.

An analysis of these laws in relation to contemporary armed conflicts was also carried out in [2] by M. Kress. The author takes into account the types of battles. The author also reviews current achievements and developments in Lanchester modeling. The author looks at contemporary conflicts around the world. Presented models take into account irregular actions, considering the importance of the target in given clashes. The author also discusses situations involving more than two sides to the conflict.

In [3] N. J. MacKay analyzes and presents the basics of Lanchester's models, their application with potential. It is a review of models and themes in the context of Lanchester's laws.

K. Y. Lin and N. J. MacKay used Lanchester's laws to analyze the course of combat in situations with diverse forces [4]. In their work, the authors simulate a battle between homogeneous units and a side with diverse forces. The goal is to find and define the most optimal method of fire distribution. The authors consider a situation in which a “one-against-many” clash occurs. The authors aim to analyze tactical dynamics.

In another paper by Kyle Lin, “New results on a stochastic duel game with each force consisting of heterogeneous units”, an iterative algorithm is presented that is capable of determining strategy in optimal way [5]. The author draws attention to a situation in which one side

operates with only one unit. The firepower of this unit should be distributed appropriately to achieve the best results. The paper describes the topic of dynamic decision-making regarding the target of an attack in order to maximize the chance of defeating a given enemy force before the allied unit is defeated.

In another paper by N. J. MacKay, three standard Lanchester models were analyzed, namely aimed-fire, unaimed-fire and asymmetric, taking into account the mixed forces of the conflicting sides [6]. Cases of random distribution in the context of target allocation are discussed. Further on in the paper, a more general model of target allocation to units is analyzed, which leads to the conclusion that an effective approach is to eliminate enemy units only after the complete elimination of another unit - there is a sequence of target units.

In [7], S. G. Coulson extends Lanchester's laws to include the impact of intelligence on the course of a battle. The author defines how the superiority in intelligence can affect the balance of forces between units and how it can determine the final outcome of a battle. The author points out that intelligence influences the course of clashes in today's conflicts, yet this topic has not been sufficiently developed in terms of its benefits in the context of Lanchester's laws. The research determines the impact of intelligence as a force multiplier and the physical force that intelligence compensates.

M. J. Kearney and R. J. Martin extend Lanchester's combat model to include stochastic elements to take into account random modifiers, emphasizing the importance of randomness [8]. In their paper the authors emphasize that the standard form of Lanchester's laws is not ideal - it does not take into account random factors that sometimes have a great impact on the course of events. The authors analyze this issue in their work.

M. Kostić and A. Jovanović draw attention to the importance of making optimal decisions in situations of uncertainty, emphasizing the significant impact of these variables on the course of events [9]. In their paper the authors develop a model based on Lanchester's laws for heterogeneous forces, including air and land forces. The model allows simplified analysis in the decision-making process.

O. Batarseh and D. Singham discuss interval-based simulation as well as uncertainty modeling in [10]. The authors determine how the IBS approach can be used for models related to Lanchester's laws to take into account parameter uncertainty. This approach was compared with simulation using Monte Carlo principles.

D. P. Kroese and R. Y. Rubinstein, in [11] take a look at the methods of the Monte Carlo algorithm. They note that many problems in various fields of science are solved through sampling. The authors analyze the possibilities of using Monte Carlo methods, identifying three main scenarios: generating random objects as well as observing their behavior, estimating numerical quantities, and solving optimization problems.

M. McCartney, in [12] addresses the issue of reinforcements between battles. Battles were modeled using "aimed fire". The author used three reinforcement strategies: constant, linearly varying and quadratically varying.

Simulations or the use of artificial intelligence in games is still being researched. In their work [13], Ł. Gałka, P. Karczmarek, and D. Czerwiński focus on creating artificial intelligence algorithms based on neural networks as well as algorithms based on the Monte Carlo method. The task of the algorithms is to control the player in a card game. The authors have proven that there are no significant differences between these two approaches.

Monte Carlo in the context of games was also used by G. E. M. Long, D. Perez-Liebana, and S. Samothrakis. In [14], the authors look at a game in which players have at their disposal an army consisting of several units of different types, strengths, and costs. The authors propose an automated method of calculating the cost of a unit using linear regression. Monte Carlo Tree Search was used to simulate the players.

In [15], the authors used a simple game to conduct research using Lanchester's Laws. They demonstrate the importance of fighting strength in predicting the outcome. The authors address the topic of calculating the percentage of losses and the duration of the battle.

Many aspects of predicting the outcome of battles have been researched and developed. Various authors have focused on specific factors influencing the course of battles that took place in reality. However, the use of Monte Carlo and Lanchester's laws to predict the outcomes of battles in strategic turn-based games has received much less attention. This leaves an unexplored area concerning the conduct of battles with specific courses. The use of these approaches requires adaptation to the specifics of the game. Current scientific achievements greatly facilitate understanding of this topic. An important theme is the approach to the sides of the conflict as heterogeneous forces, which is often used in the cited works. An equally important issue is the influence of random factors. It is present in many games. However, the cited works have their limitations in terms of the topic addressed in this paper – they mainly concern real-life clashes. Games, depending on their mechanics, simulate battles in different ways. They add various possibilities or impose additional restrictions. The area for further analysis and research can still be expanded through the creation of new games with new rules.

3. Algorithms and laws

To predict the outcome of a battle, existing rules and algorithms can be used. They, when adapted to the current system, could bring the expected results. Two algorithms based on Lanchester's Laws and Monte Carlo were developed for the purposes of the study.

3.1. Lanchester's laws

Lanchester's laws were developed by Frederick William Lanchester in 1915-1916. They were intended to be used to calculate the results of the duels. They became an effective tool, developed and researched over subsequent decades. Many scientists analyzed these laws for their accuracy with the actual course of historical events. Thay

also improved and diversified them in order to achieve better results. The author designed differential equations for combat situations - ancient and modern battles.

The first form of the equations is known as Lanchester's linear law (also called unaimed fire (formula 4)) [12]. The attrition rate of each side in this model is proportional to the number of its units as well as to number of the hostile units.

$$\frac{dA}{dt} = -\beta AB$$
$$\frac{dB}{dt} = -\alpha AB \quad (4)$$

Where a means the size of the first unit, and *dA/dt* means the rate at which its size changes. The strength of the unit is denoted as α. Analogous symbols apply to group *B*.

Lanchester's square law (also known as targeted fire (formula 5)) [12] describes how a unit can attack several other units at a moment, while also exposing itself to damage from several sources.

$$\frac{dA}{dt} = -\beta B$$
$$\frac{dB}{dt} = -\alpha A \quad (5)$$

Lanchester's laws describe the relationship between the strength and size of two forces. They describe how two opposing forces will interact during combat and which will win. They describe the rate of mutual losses over time dt. According to Lanchester's Laws, both forces attack each other at the same time.

3.2. Monte Carlo

Monte Carlo method works different - it is a method used in situations that are too complex for a purely analytical approach to deliver the expected results. Compared to such an analytical approach, the method is enriched in its structure and operation by random samples - random values from a given range.

The Monte Carlo principle is to perform multiple calculations for a given process and determine the result based on a series of trials. The algorithm is characterized by randomness - it operates on ranges and a random value generator. This process begins with defining the appropriate value space - this is the basis for the subsequent generation of random samples. The space is a set of all possible inputs in the studied situation, which is processed by the algorithm. Random sampling involves generating values from the space. The algorithm should then process the random samples appropriately - each of the generated values is subject to the same processes. Finally, the algorithm aggregates the results to obtain an estimate.

3.3. Implementation

For the study purposes, two algorithms were developed, based on existing laws, rules, and algorithms. The first is based on Lanchester's Laws. The second one on the principles of the Monte Carlo algorithm. Both algorithms were adapted to the needs and rules of the research game - they had to be modified to reflect as closely as possible the results obtained by players during manual gameplay.

One of the common stages of both algorithms is the creation of an initiative list based on the statistics of the same name for each unit (Figure 2).

```
void playTurn() {
  List<QuickCombat_Lanchester_Unit> initiativeList = [];
  var heroes = [hero1, hero2];
  for(int hero = 0; hero < heroes.length; hero++){
    for(int creature = 0; creature
      < heroes[hero].units.length; creature++){
      if(heroes[hero].units[creature].count > 0){
        initiativeList.add(heroes[hero].units[creature]);
      }
    }
  }
  initiativeList.sort(
      (a, b) => b.initiative.compareTo(a.initiative));
```

Figure 2: The code creating list of units by their initiative.

Units are sorted in descending order according to their initiative statistics value. In manual gameplay, the player decides which unit to move. In both algorithms, the units performing the action are the successive units from the initiative list. The algorithms select the target in different ways. The Lanchester's Law algorithm selects the first enemy unit from the initiative list, while the Monte Carlo algorithm does it randomly.

For further data processing in the algorithm based on Lanchester's laws, the required data must first be determined. In their original form, Lanchester's laws take into account the size of the unit, the strength of the unit, and the time step. The size of each unit is known from the beginning - it is stack number. However, the strength *P* of unit is unknown. It can be calculated using the formula (formula 6).

$$P = 10(A_f + A_m) + 4(D_f + D_m) + 10H_p \quad (6)$$

The variables used in the formula are the statistics values of each unit under the players' command. It is important to take weights into account, as some of the basic statistics are much more important during gameplay. The most important statistics are magical attack A_m and physical attack A_f. Other statistics taken into account are magical defense D_m, physical defense D_f, and hit points H_p. When determining the strength of a unit, ranged skills increase its combat potential - strength is multiplied by a value of 1.1. The final strength modifier is an additional divided by a constant value of 20 to avoid very large final values. The algorithm assumes a time step of 0.005 in abstract game time units.

The duel between the two selected units is based on modified rules described as Lanchester's laws. The losses of unit B (the defending unit) are determined as B_{Loss} (formula 7).

$$B_{Loss} = dt \cdot A_{Power} \cdot A_{StackNumber} \quad (7)$$

Where A_{Power} means the strength of unit A, determined by formula 6. $A_{StackNumber}$ means the number of creatures in unit. The obtained value of B_{Loss} is subtracted from the current value of the stack number of unit

B - if it falls to a negative value, the stack number is set to 0, which excludes the unit from further processes (Figure 3).

```
double lossesB =
    dt * attacker.power * (attacker.count );
defender.count -= lossesB;
if(defender.count < 0) {defender.count = 0;}

if(defender.count > 0){
  double lossesA =
    dt * (defender.power * 0.5) * (defender.count + 1);
  attacker.count -= lossesA;
  if(attacker.count < 0) {attacker.count = 0;}
}
```

Figure 3: The code simulating performing of an attack by unit.

The defending unit may launch a counterattack if it is able to do so. Unit can perform it only after receiving damage and updating its strength. There is a modifier with a value of 0.5 - its task is to reduce the dealt damage (formula 8).

$$A_{Loss} = dt \cdot B_{Power} \cdot 0.5 \cdot B_{StackNumber} \quad (8)$$

The above calculations are repeated for subsequent units from the initiative list. Only the value of strength P remains unchanged. A_{Loss} and B_{Loss} are calculated based on the units selected from the initiative list. Their current stack number, which changes after taking damage, is also taken into account. If neither of the players have been defeated, the initiative list is recreated until one of the players loses their entire army.

The algorithm based on Monte Carlo principles operates on different rules for calculating losses. This process takes basic statistics into account as spaces for further processing. Four hidden unit statistics are used to define the range of possible damage: minimum physical damage, minimum magical damage, maximum physical damage, and maximum magical damage (Figure 4).

```
QuickCombat_MC_Unit(this.creature, this.count){
  initiative = creature.Initiative;
  damage_Min = creature.Attack_Magic_Minimum +
      creature.Attack_Might_Minimum;
  damage_Max = creature.Attack_Magic_Maximum +
      creature.Attack_Might_Maximum;
  currentHealthPoints = creature.HealthPoints * count;
  defense = ((creature.Defense_Might +
      creature.Defense_Magic) / 2).toInt();
}
```

Figure 4: Defining unit data for Monte Carlo algorithm.

The algorithm defines *damageMin* and *damageMax* as the sum of the minimum and maximum damage from both sources. Unit's defense is the average of its physical and magical defense, rounded to an integer. Additional statistic is the current hit points, calculated as the stack number multiplied by the unit's hit points. (Figure 4).

After selecting the units for combat, the algorithm calculates the damage dealt. It begins by generating a value R from the damage range of the attacking unit (formula 9).

$$R \in [damageMin, damageMax] \quad (9)$$

The obtained R value is multiplied by the stack number L of the attacking unit - the resulting value is denoted as R_{Dmg}.

$$R_{Dmg} = R \cdot L \quad (10)$$

Damage dealt (formula 11) must take into account the defense $B_{Defense}$ of the defending unit. Damage dealt value is reduced by a percentage, based on the target's defense statistics.

$$Dmg = R_{Dmg} - (R_{Dmg} \cdot \frac{B_{Defense}}{100}) \quad (11)$$

The damage value obtained is subtracted from the field describing the current health points of the defending unit. The stack number of the unit is also modified on an ongoing basis (Figure 5). As in the previous algorithm, if at any point during the calculations the unit's hit points become negative, the stack number will be set to 0, which excludes the unit from further processes.

```
damage_dealt = damage_dealt -
    (damage_dealt * (defender.defense / 100)).toInt();
defender.currentHealthPoints =
    defender.currentHealthPoints - damage_dealt;
defender.count = (defender.currentHealthPoints /
    defender.creature.HealthPoints).toInt();
if(defender.currentHealthPoints < 0){
  defender.count = 0;
}
```

Figure 5: The code updating unit's data.

The above calculations are repeated for subsequent units on the initiative list. If no player has been defeated, the initiative list is being recreated until one player loses their entire army. The entire combat procedure is performed 21 times. Each time, the results are placed in one of two lists. These are lists that store the results of winning games for individual player (Figure 6).

```
bool hero_1_IsAlive =
  combat.checkIfHeroIsAlive(hero1);
if(hero_1_IsAlive){
  winningArmy_Hero1.add(getUnitsCount(hero1));
} else {
  winningArmy_Hero2.add(getUnitsCount(hero2));
}
```

Figure 6: Placing combat result in the dedicated list.

These lists are built of lists of integers representing the stack numbers of the winning player's units at the end of the game. The algorithm considers the player who has won the most games to be the winner. The final result of the combat is the stack number of each unit determined by the median.

4. Research methods

The study includes eight different scenarios that may occur in the game. Three scenarios describe situations in which both players operate a full army, i.e., each of them has ten units of various types, levels, and stack numbers at the beginning. Both players have under their command four tier 1 units, four tier 2 units and two tier 3 units - creatures have different statistics values but they are balanced through the tier. The difference between these scenarios is the advantage of one player (Table 1).

The player strength is the sum of all units' powers P (formula 6), multiplied by its stack numbers.

Table 1: Scenarios of full army

No.	Player 1 strength	Player 2 strength	Description
1	39198.1	42380.1	Balanced, tiers 1-3
2	58722.4	42380.1	Advantage, tiers 1-3
3	274386.7	42380.1	Big advantage, tiers 1-3

In the first case, both players command balanced forces - the total strength of both players is similar, while in the next two situations examined, the advantage of the first player increases. There are cases of average and enormous advantage over the second player.
Several other cases possible in the application were also considered - these are situations of incomplete armies, including extreme cases (Table 2).

Table 2: Scenarios of incomplete armies

No.	Player 1 strength	Player 2 strength	Description
4	29380	20700	Tier 1 vs tier 3, 1 unit vs 1 unit
5	27926.6	41820	Advantage, tiers 1-3 vs tier 2, 5 units vs 1 unit
6	27926.6	27940	Balanced, tiers 1-3, 5 units vs 5 units
7	27926.6	42380.1	Advantage, tiers 1-3, 5 units vs 10 units
8	167559.6	42380.1	Big advantage, tiers 1-3, 5 units vs 10 units

For each of the scenarios, manual gameplay was played repeatedly, and the results were recorded. The player who was the first to win ten times in a given scenario was considered the most likely winner. Their victory, as well as the losses during the manual gameplay, should be predicted by the algorithm. The results of all ten winning combats were recorded.

For each combat result, the total strength of the army was calculated, using the same method as before. In this way, ten probable results of the combat between the players were obtained.

Both algorithms were run for the same input data. The algorithm using Lanchester's laws was used only once for a given scenario. It is an algorithm that will always return the same result for the same input. The Monte Carlo algorithm was executed ten times - as equivalents for each of the winning battles played. This delivered ten results suggested by the algorithm, taking into account the winning player and the state of their army at the end of the game. For the results obtained from both algorithms, i.e., a total of eleven battle results, the total strength of the winning army at the end of the game was calculated. In this way, accurate data was obtained for each of the scenarios, which is used for further analysis. This data is the total strength of the winning player's army at the end of the battle in ten battles, the total strength of the victorious player's army predicted by the algorithm based on Lanchester's laws, and ten values of the total strength of the army predicted by the Monte Carlo algorithm. With this data, the mean absolute percentage error (MAPE) can be calculated (formula 12).

$$\mathrm{MAPE} = \frac{1}{n}\sum_{x=1}^{n}\left|\frac{y_x - \hat{y}_x}{y_x}\right| \cdot 100\% \qquad (12)$$

Where n is the simulation number, y_x is the actual result, and $\hat{y}_x$ is the predicted result. Each scenario was played 10 times manually. The Lanchester algorithm was run once for each scenario. As a consequence, the results of each simulation were assumed to have the same value. The Monte Carlo algorithm was executed 10 times.

Additionally, for each scenario, the time needed to obtain the result is checked. For every run of the algorithm in a given scenario, the time needed to complete the task was recorded, and then the results were averaged.

5. Study results

With the presented method, tests were carried out for each of the scenarios - these are cases of a duel between the blue and green players. The expected result (the average value for manual gameplay) was compared with the average results of both algorithms. Incorrect predictions of the winner are marked in bold (Table 3).

Table 3: Expected and predicted results for each scenario

No.	Expected result	Predicted by Lanchester	Predicted by Monte Carlo
1	10257.5	8141.2	11183.3
2	35194.8	32406.1	38118.6
3	266906	271685.3	271504.5
4	12502.8	**27029.6**	12854.7
5	25677.5	32828.7	36676.1
6	8820.1	6742.7	4830.6
7	29265.2	18619.7	16184.6
8	165307.3	156435	162078.5

Table 4: MAPE for each scenario

No.	Lanchester MAPE [%]	Monte Carlo MAPE [%]
1	23.6	28.5
2	12.2	12.2
3	1.8	1.7
4	317.1	5.7
5	30.3	45.5
6	39.5	42.9
7	36.2	44.5
8	5.3	2.4

For each scenario, the average absolute percentage error was calculated to define which algorithm was closer to the expected result in a given scenario. Approximate values to two decimal places are presented in Table 4.

Each scenario was also analyzed in terms of the time required to obtain results. Only calculations are examined (Table 5).

Table 5: Average execution time of the algorithm for each scenario

No.	Lanchester [µs]	Monte Carlo [µs]
1	619.8	15079.8
2	260	5960.2
3	160	2539.2
4	59.8	1920.2
5	300.4	7180.2
6	239.8	6580
7	320.2	8619.6
8	320.4	3739.6

6. Analysis of results

The obtained data was analyzed. In eight of the presented scenarios, the Monte Carlo algorithm was able to correctly predict the winning player each time. The algorithm based on Lanchester's laws was significantly wrong in one case, which means that its accuracy in some situations may remain uncertain (Figure 7).

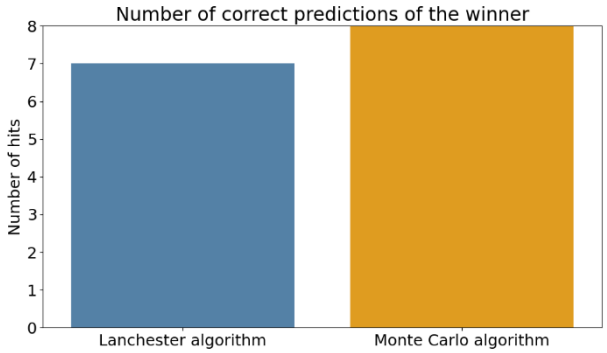

Figure 7: Number of correct predictions of the winner.

The MAPE can provide greater insight into the accuracy of the results in terms of the losses of the victorious army (Figure 8).

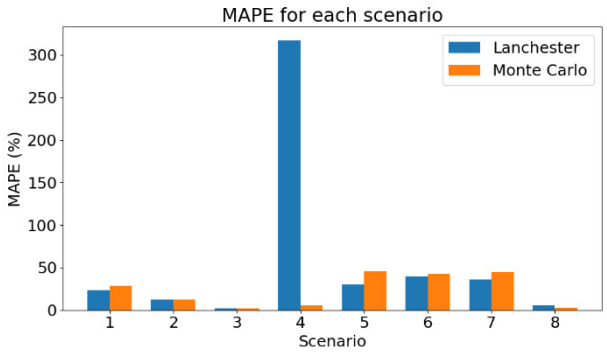

Figure 8: MAPE for each scenario.

The value of this error in the first three cases is very similar between algorithms. It can be noted that the greater the advantage of one army over the other, the smaller the error in predicting the result by the algorithms. The decline is greater on the Monte Carlo side, which is better at determining the result when there is a large advantage. The smaller the advantage, the more favorable the Lanchester algorithm is. The fourth case, the most extreme, proved to be problematic for the first algorithm. The Lanchester algorithm incorrectly determined the winner of the duel. The final army turned out to be only slightly weaker than the initial one, which would be impossible to reach in this combination during manual gameplay, regardless of the player's skill. The Monte Carlo algorithm handled this case much better, with a MAPE value of 5.68%. The next three cases - 5, 6, and 7, although the results are similar to each other, indicate greater accuracy of the Lanchester algorithm. The last case, as in the first three, proves that the greater the advantage of the army, the better Monte Carlo is (Figure 8).

In this study, the task execution time is a secondary factor when selecting the appropriate algorithm - the most important thing is its accuracy with reality. Time values exceeding a threshold of several seconds would be an obstacle, but the algorithms presented require so little time that this aspect can be ignored. However, when comparing the data obtained, the algorithm using Lanchester's laws performs its tasks faster (Figure 9).

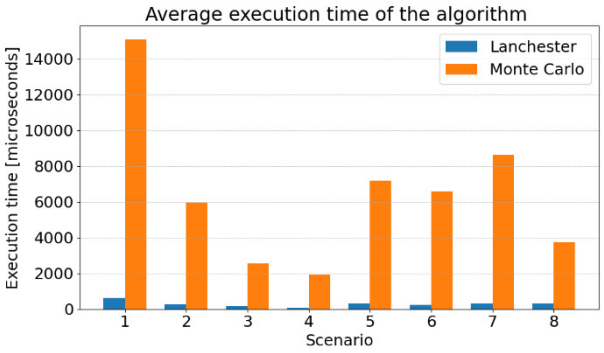

Figure 9: Average execution time of the algorithms.

7. Conclusions

The study application is a game that involves a duel between two players. They both command their own army. The army consists of up to ten units of varying stack numbers and statistics. There is an element of randomness in the game that has a significant impact on the course of the game and the final result, so it is difficult to define a specific algorithm that will be able to accurately predict the outcome of the game. The difference between the final strengths of the armies can be very large, so it is very difficult to determine a process whose average absolute percentage error is zero. Two algorithms have been developed. They are able to calculate the possible outcome of the game in a short time – one is based on Lanchester's laws and the other on Monte Carlo. The two algorithms differ in the processes and calculations they perform. The results may be more or less similar, depending on the scenario. The Monte Carlo algorithm proved to be more effective in predicting the winner - it did not make a single mistake in any of the scenarios. The algorithm inspired by Lanchester's laws proved to be ineffective in one of the cases - the most extreme one. The remaining cases were correctly predicted by both algorithms. Their effectiveness in these cases is quite similar and satisfactory. The first three cases can be classified as one group in

which both algorithms behaved stably. Depending on the advantage of one player over the another, the effectiveness of both algorithms increases and the MAPE decreases. The Monte Carlo algorithm gains significantly with increasing advantage, as confirmed by the last example. With more balanced armies, the Lanchester algorithm achieves a better result (smaller MAPE value), but these situations are usually very volatile and difficult to predict. The results of manual gameplay can vary greatly in cases of forces of similar strength. The hypothesis of this paper says that one of the developed algorithms is able to predict the outcome of a duel between two players. The hypothesis has been confirmed by research and analysis. These show that, in this situation, the better algorithm is the one based on Monte Carlo principles, whose calculated results are similar to those obtained manually. Importantly, in each scenario, the algorithm correctly predicted the most likely winning player.

References

[1] R. Hoffmann, T. Protasowicki, Modelowanie pola walki z zastosowaniem koncepcji dynamiki systemowej, Biuletyn Instytutu Systemów Informatycznych 12 (2013) 29-34.

[2] M. Kress, Lanchester models for irregular warfare, Mathematics 8(5) (2020) 737, https://doi.org/10.3390/math8050737.

[3] N. J. MacKay, Lanchester combat models, Mathematics Today 42 (2006) 170-173, https://doi.org/10.48550/arXiv.math/0606300.

[4] K. Y. Lin, N. J. MacKay, The optimal policy for the one-against-many heterogeneous Lanchester model, Operations Research Letters 42(6-7) (2014) 473-477, https://doi.org/10.1016/j.orl.2014.08.008.

[5] K. Y. Lin, New results on a stochastic duel game with each force consisting of heterogeneous units, Naval Research Logistics (NRL) 61(1) (2014) 56-65, https://doi.org/10.1002/nav.21566.

[6] N. J. MacKay, Lanchester models for mixed forces with semi-dynamical target allocation, Journal of the Operational Research Society 60(10) (2009) 1421-1427, https://doi.org/10.1057/jors.2008.97.

[7] S. G. Coulson, Lanchester modelling of intelligence in combat, IMA Journal of Management Mathematics 30(2) (2019) 149-164, https://doi.org/10.1093/imaman/dpx014.

[8] M. J. Kearney, R. J. Martin, On a stochastic version of Lanchester's model of combat, arXiv preprint arXiv:1905.03122 (2019), https://doi.org/10.48550/arXiv.1905.03122.

[9] M. Kostić, A. Jovanović, Lanchester's differential equations as operational command decision making tools, Serbian Journal of Management 18(1) (2023) 71-92, http://dx.doi.org/10.5937/sjm18-39699.

[10] O. Batarseh, D. Singham, Interval-based simulation to model input uncertainty in stochastic Lanchester models, Military Operations Research 18(1) (2013) 61–75, http://www.jstor.org/stable/24838472.

[11] D. P. Kroese, R. Y. Rubinstein, Monte Carlo methods, WIREs Computational Statistics 4(1) (2012) 48–58, https://doi.org/10.1002/wics.194.

[12] M. McCartney, The solution of Lanchester's equations with inter-battle reinforcement strategies, Physica A: Statistical Mechanics and its Applications 586 (2022) 126477, https://doi.org/10.1016/j.physa.2021.126477.

[13] Ł. Gałka, P. Karczmarek, D. Czerwiński, Application of Monte Carlo algorithms with neural network-based intermediate area to the Thousand Card Game, In: L. Rutkowski, R. Scherer, M. Korytkowski, W. Pedrycz, R. Tadeusiewicz, J. M. Zurada, (eds) Artificial Intelligence and Soft Computing. Lecture Notes in Computer Science, Springer 14125 (2023) 68-77, https://doi.org/10.1007/978-3-031-42505-9_7.

[14] G. E. M. Long, D. Perez-Liebana, S. Samothrakis, Balancing Wargames through predicting unit point costs, In 2023 IEEE Conference on Games (CoG) (2023) 1–8, https://doi.org/10.1109/CoG57401.2023.10333152.

[15] M. Flanagan, D. Lambert, T. C. Lipscombe, A. Northey, I. M. Robinson, Lanchester's fighting strength as a battle outcome predictor applied to a simple fire and manoeuvre wargame, In Recent Advances in Monte Carlo Methods, IntechOpen (2024), https://doi.org/10.5772/intechopen.1002384.