Team Recommendation for the Pokémon GO Game Using Optimization Approaches

Samuel da S. Oliveira
Dept. of Informatics and Applied Mathematics (DIMAp) Federal University of Rio Grande do Norte (UFRN)
Natal, Brazil
samueloliveira@ppgsc.ufrn.br

Guilherme E. P. L. Silva
Digital Metropolis Institute (IMD) Federal University of Rio Grande do Norte (UFRN)
Natal, Brazil
guigle@ufrn.edu.br

Arthur C. Gorgônio
Dept. of Informatics and Applied Mathematics (DIMAp) Federal University of Rio Grande do Norte (UFRN)
Natal, Brazil
arthurgorgonio@ppgsc.ufrn.br

Cephas A. S. Barreto
Dept. of Informatics and Applied Mathematics (DIMAp) Federal University of Rio Grande do Norte (UFRN)
Natal, Brazil
cephasax@gmail.com

Anne M. P. Canuto
Dept. of Informatics and Applied Mathematics (DIMAp) Federal University of Rio Grande do Norte (UFRN)
Natal, Brazil
anne@dimap.ufrn.br

Bruno M. Carvalho
Dept. of Informatics and Applied Mathematics (DIMAp) Federal University of Rio Grande do Norte (UFRN)
Natal, Brazil
bruno@dimap.ufrn.br

***Abstract*—Pokémon GO is one of the most popular Pokémon games. This game consists of walking around the world and collecting Pokémon characters using augmented reality. In addition, you can battle with friends, join a gym, or make attacks. These battles must happen between teams with the same size, and this poses a question that is related to the best combination for a team to beat a given opposing team. In order to solve this problem, one can use optimization algorithms. In this paper, we investigate three optimization algorithms to solve this problem: genetic algorithm (GA), memetic algorithm (MA), and iterated local search (ILS). In our experiments, we use time and fitness as evaluation metrics. Our findings indicate that the fastest algorithm is ILS with an execution time of 1.49 ± 0.11 seconds, followed by GA with an execution time of 1.51 ± 0.10 seconds, and MA with an execution time of 13.41 ± 1.00 seconds. However, when we consider the fitness metric, MA achieves the best average fitness of $50,366.27 \pm 12,055.53$, followed by GA, $43,113.00 \pm 10,482.30$, and ILS, $31,224.32 \pm 7,943.70$. All these results are statistically significant to the others according to the post-hoc Friedman test. Analyzing all the obtained results, we recommend the use of the ILS algorithm when the execution time is of utmost importance. However, if fitness is important, then we recommend the use of the memetic algorithm. Finally, if both the execution time and fitness are deemed equally important, then, we recommend the usage of the genetic algorithm because it has a runtime similar to ILS and reasonable fitness.**

***Index Terms*—Pokémon GO, Recommender Systems, Optimization algorithms**

I. Introduction

Pokemon is a big franchise, including merchandise such as games, series, books and toys [1]. For instance, there are several variations of this game. In addition, these games are famous all over the world, usually with an enormous number of players taking part of a single game.

Pokémon GO is a mobile game that was launched in 2016 [2]. In this game, the Pokémon characters are scattered around the real world and the players should go looking for the virtual monsters. The player has the objective of completing the Pokedex, which is a catalog with all the monsters present in the game. This game also has a battle system that allows two types of battles: players *vs* players and player *vs* computer.

Trainers can battle in many ways: against Team Rocket (computer), against others trainers (traditional PvP) or in gyms against a Pokémon boss (raids) or against others player's Pokémon to take control of a gym, increasing the competitiveness of the game. A victory in a battle depends on some factors, such as Health Points (HP), Combat Points (CP) and Pokemon Types (PT), among others which are considered less important ones. A player who has the team with the best combination tends to win. In a battle against another player or Team Rocket, the teams are composed of three Pokémon characters each, while the teams engaged in gym battle consist of six characters. At the moment, 651 Pokémon characters can be selected, and we need to choose the best team to win a match, given a rival team or gym battle. Hence, when a team length grows or a new Pokémon is inserted on the game, the possibilities to choose the best team grows too.

The selection of the ideal team can be laborious due to the large search space even taking into account the correct variables. The purpose of our paper is to show that team selection in the Pokémon GO game is a process that contains some restrictions. As the objective of the battles in the game is to win, we have the following problem: given a rival team, which is known, what is the best counter team that we can choose using an heuristics based on the Pokémon character

abilities. In other words, given a rival team, which counter team can beat the rival team as quick as possible. Thus, this paper proposes an approach that computes, given a rival team, the best counter team, using optimization algorithms. In the Pokémon game, this represents the selection step of composing a team. Aiming to contribute to this important issue, this paper presents an investigation of the use of optimization techniques to team recommendation. The main aim is to indicate the best team for a given rival team. In order to do that, three different techniques will be applied in order to optimize the search and to recommend a suitable team for the confrontation. The three heuristic selected techniques are:

- Iterated Local Search (ILS) [3];
- Genetic Algorithm (GA) [4];
- Memetic Algorithm (MA) [5].

The first algorithm (ILS) is a trajectory-based algorithm, while the remaining two (genetic and memetic algorithms) are population-based ones. All three aforementioned algorithms are developed in order to evaluate the Pokémon teams and to provide as result its impact factor. In this sense, it can indicate the best team for a given rival team.

This paper is divided into six sections as follows. Section II describes the theoretical aspects related to the subject of this paper, while Section III presents some related studies. The experiment methodology is described in Section IV, then the obtained results are presented in Section V. Finally, in Section VI we present the conclusions and suggestions for future works.

II. Theoretical Aspects

This section addresses the theoretical aspects of the three optimization techniques used in this paper. The next subsections will describe iterative local search, genetic algorithms and memetic algorithms, respectively.

A. Iterative Local Search (ILS)

The ILS algorithm [3] is a simple, efficient meta-heuristic. It is a trajectory-based algorithm in which it starts with an initial solution. Based on this initial solution, it moves towards a local optimum. At each iteration, a perturbation is generated and a local search is performed based on the perturbed solution, thus creating a new local optimum.

In this paper, the perturbation of the ILS algorithm occurs by exchanging Pokémon character within the current solution at random. In other words, perturbations are defined as team member changes and the new solution is testing in order to assess the efficiency of the new team.

B. Genetic Algorithm

A genetic algorithm (GA) [4], [6] is a meta-heuristic technique that is bio-inspired by the Darwin evolution theory. It is a population-based algorithm that consists of a set of solutions called population. In addition, it uses a step for the crossover of this population and a mutation step, similar to its occurrence in nature. It also has a step that evaluates the set of solutions. Each individual of a population represents a possible solution that must solve the proposed problem.

In a GA algorithm, the evolution starts from a random set of solutions and it is carried out across generations. At each generation, the adaptation of each individual of the population is evaluated and some individuals are selected for the next generation. These selected individuals are recombined and some are still mutated in order to form a new population. The new population is used as an input for the next iteration of the algorithm.

The objective function (fitness) consists of evaluating each individual, assigning a score that classifies it so that we can know which individuals are most apt to perform the recombination step in order to integrate the new population. Fig. 1 presents the GA process. This algorithm runs for a predefined number of iterations or until a minimum target value for the fitness function is achieved.

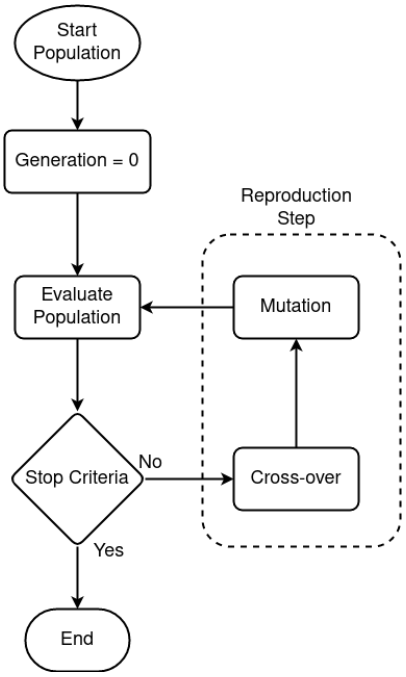


Fig. 1. Genetic Algorithm Process.

C. Memetic Algorithm

The memetic algorithm is a natural evolution of GA [5]. The main aspect of the MA approach is that a local search was included to be performed after the crossover and mutation steps. In other words, along with the operations presented in GA, the MA algorithm also performs a local search in order to improve the quality of each individual. The memetic process flow is shown on Fig. 2.

III. Related Work

In this section, we will present some important studies in the optimization area, involving heuristics techniques used by

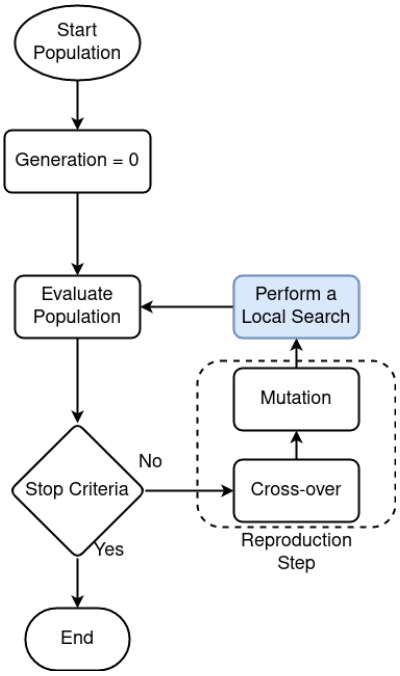


Fig. 2. Memetic Algorithm Process.

recommendation systems. The literature of optimization algorithms is vast, with different studies in different applications, such as in [7]–[11].

For instance, Bansal and Baliyan in [7] worked with web-based recommendation systems (RS). In the cited paper, the authors presented a novel MA-based Similarity Metric (MASM) for RS, leveraging the collaborative behavior of memetic algorithms (MA). It used a publicly available Movielens dataset (100K ratings) to conduct experiments. They investigated the accuracy using precision and recall with different neighborhood sizes of target users on a public available Movielens dataset. Results shows that the proposed MASM is better than conventional GA-based Similarity Metric (GASM). The precision of RS using MASM is improved by 28% over RS using GASM, resulting in an improved predictive recommendation accuracy.

In [8], the authors presented a Recommendation system based on locations and tags. In this system, they proposed a novel location and tag aware recommendation framework that uses ratings, locations, and tags to generate a recommendation. In their framework, all users are partitioned into several clusters by a newly designed MA-based clustering method. For existing users, items are recommended by applying the Latent Dirichlet Allocation (LDA) to users who are part of each cluster of the obtained partition. For cold-start (new) users, each cluster is presented as a new user. Several experiments were carried out on real-world datasets, demonstrating that, when compared to state-of-the-art location and tag aware recommendation algorithms, the proposed algorithm provided better results on making recommendations and alleviating the cold-start problems.

In [9], the authors performed a comparison of machine learning algorithms for different scenarios of Pokémon battles. In the cited analysis, the best performing algorithm was used in a testing scenario against human players, aiming to assess the capability of the selected algorithm to play at the human level. According to the authors, the most successful algorithm was a Monte Carlo search algorithm [12] which spent time, before a round, playing out the entire game from that round many times. Another evaluated technique was the linear combination of features [13], that before a battle begins, a user spends time playing several training games trying to adjust the weights related to the actions and the state of the game. In addition, the random algorithm which always picks a random action and the hard coded algorithm which picks a predefined action each round was used as benchmark algorithms for the different scenarios. In this analysis, three different battle scenarios within the Pokémon game for the Game Boy were covered. The Monte Carlo search algorithm achieved the highest performance of all evaluated algorithms, and it was then select to play the human player challenge. According to the authors, the Monte Carlo algorithm managed to win half of all battles in the the human player challenge.

In [14] a Pokémon battle environment is implemented that preserves the main elements of a Pokémon battle, allowing researchers to test learning objectives in isolation. The paper approach focuses on the advantages between the types of Pokémon and the delayed rewards through exchanges. A multi-agent environment and standard reinforcement learning are used and oriented towards high performance, reaching thousands of interactions per second in hardware merchandise. Deep competitive reinforcement learning algorithms, WPL θ and GIGA θ are able to converge to rational and effective strategies, and GIGA θ shows a faster convergence, obtaining a 100% victory rate in a disadvantaged scenario.

Unlike the aforementioned studies, our paper deals with making team recommendations in a Pokémon GO game battle. In our recommendation procedure, the user's experience and personal preferences are not taken into account, but only the characteristics that are already present in the Pokémon database are provided as input. The main aim is to provide a robust and simple team recommendation.

IV. Methodology

In order to select the best solution for the team recommendation problem, a fair comparison different aspects of each solution is needed. In this analysis, two main aspects will be evaluated, fitness and execution time. As mentioned before, in this paper, we will compare three different solutions, which are an Iterated Local Search, a Genetic Algorithm, and a Memetic Algorithm. In this sense, in our empirical analysis, these three techniques will be evaluated using those two aspects.

A. The Proposed Approach

Fig. 3 describes the flowchart of our experimental approach. In this figure, once the Pokémon GO database that contains all Pokémon features is provided, the rival team should be

selected to fight against the opposing team, in order to test the performance of the selected methods. The next step is the selection of the analysed optimization algorithms: GA, ILS or MA. Then, the next step is to select the optimal home team using the selected optimization technique.

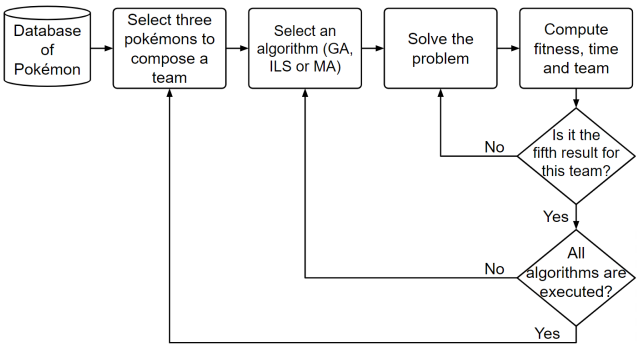


Fig. 3. Design of the Experiment.

For each rival team and optimization method selected, the selection process (solve the problem) is repeated five times since the evaluated optimization techniques are non-deterministic ones. The execution times and impact factors are stored for each run, and, after that, the best home team is selected from among all five executions. Table I presents the parameter values used for each algorithm. For GA and MA parameters, we will use the recommendations described in [15] to set crossover and mutation rate values. Additionally, the number of individuals and iterations were set after several tests in order to find the best configuration for these parameters.

TABLE I
THE TUNING OF THE ALGORITHM IN THE EXPERIMENT

Algorithm	Tunning
ILS	Iterations = 1000
GA	Individuals = 50, Iterations = 20, Crossover = 80%, Mutation = 20%, Elitism = True
MA	Individuals = 50, Iterations = 15, Crossover = 80%, Mutation = 20%, Elitism = True, local-search = ILS with 10 iterations

The number of possible solutions in the search space is huge. Given that P is the set of all Pokémon characters available in this game ($|P| = 651$), and k represents the team length ($k = 3$), (1) computes the total number of possible solutions N in the search space, i.e., of different teams.

$$N = \frac{(|P| + k - 1)!}{k!(|P| - 1)!} \quad (1)$$

This equation is a combination with replacement, because it is possible to use the same Pokémon more than once in a team. However, it is necessary to consider that, from the point of view of a game, the repetition is severely limited. This happens because the player can use only Pokémon already caught, which is very difficult, specially in the case of stronger Pokémon. In the Pokémon battle system it takes into account that each member of the home team will battle with only one member of the rival team. However this battle can occur with any permutation of that team, being necessary to calculate the fitness of a pokémon of the home team for each member of the rival team.

B. Pokémon Data

In order to define a Pokémon team, it is necessary to know the Pokémon character and its aspects. The data related to four Pokémon characters can be seen in Table II.

As it can be seen in Table II, a Pokémon character abilities can be described by the following *stats* descriptors:

- Pokedex number: ID (identification), a unique number that identifies a specific Pokémon;
- Name: The Pokémon's name;
- CP: Combat Power is a metric as defined in (2);
- HP: Pokémon's health;
- Attack: Pokémon's base attack;
- Def: Pokémon's base defense;
- Stam: Pokémon's base stamina;
- types: Pokémon's types;
- types against: how effective is a type against this Pokémon.

The CP of a Pokémon character is defined by (2) [16]:

$$\text{CP} = \frac{(\text{Attack} + \text{A}_{iv})\sqrt{(\text{Def} + \text{D}_{iv})}\sqrt{(\text{Stam} + \text{S}_{iv})}C^2}{10}, \quad (2)$$

where C is a constant defined by the Pokémon level. In this paper, we will consider only the maximum value, 0.7903, relative to the maximum level. Additionally, A_{iv}, D_{iv} and S_{iv} are variables that are relatives to Pokémon IV [17]. In this paper, we set these parameters to their maximum values, 15.

For the empirical analysis, we will use a database containing 1000 different Pokémon teams, where the Pokémon teams were randomly generated. Given a rival team, all the optimization techniques are applied to search for an acceptable counter (home team) to play in a Pokémon battle.

C. Representation of individuals

In this paper, an individual is composed of a team of three Pokémon characters. This setting was selected since it is the default team size on battles against Team Rocket and other players, where both the player's team and enemy's teams have three Pokémon. Nevertheless, all definitions and methods used in this paper can be applied to different team's sizes and situations. In this sense, a chromosome is a set of genes in which each gene represents a Pokémon character.

D. Fitness

A fitness function, also known as objective function, defines a value (grade) that is used to describe how good is an individual and this value is calculated by applying this function to each solution. For a fair comparison, the same fitness function is used to all three optimization techniques, GA, MA and ILS.

TABLE II
POKÉMON DATA EXAMPLE

Pokedex Number	Name	CP	HP	Attack	Defense	Stamina	Type	Against Electric	Against Fire
1	Bulbasaur	1 115	111	118	111	128	Grass	0.5	2
4	Charmander	980	103	116	93	118	Fire	1	0.5
7	Squirtle	946	111	94	121	127	Water	2	0.5
10	Caterpie	437	111	55	55	128	Bug	1	2

Algorithm 1 is used to calculate the fitness value. In this algorithm, the following steps describe the entire process to evaluate an individual: i) for each Pokémon character in a home team (team), it will perform a battle with each Pokémon in the opposing team (team_target); ii) Algorithm 2 presents the steps of a battle function between two Pokémon characters in order to assess which character is the best one. This battle function consists of a comparison of both Pokémon CPs affected by the adversary's types. When analysing two Pokémon teams, the winner will be the one who has the higher CP after been affected by the adversary's types. The variation's magnitude on the Pokémon's CP will be defined by its coefficient against the adversary's type. Finally, Algorithm 3 defines the main steps to compose a team that has the best fitness value within the set of solutions and returns it as a result.

Algorithm 1: fitness(team, team_target)

```
begin
    fit ← 0
    foreach home ∈ team do
        foreach opposing ∈ team_target do
            fit += battle(home, opposing)
    return fit
```

Algorithm 2: battle(poke1, poke2)

```
begin
    against1 ← best_against(poke2, poke1.types)
    against2 ← best_against(poke1, poke2.types)
    return poke1.cp * against1 - poke2.cp * against2
```

Algorithm 3: best_against(pokeA, pokeB_types)

```
begin
    best = 0
    foreach against ∈ pokeA.against(pokeB_types) do
        if against > best then
            best = against
    return best
```

In summary, given two teams, the recommended (home team) and rival (target team) ones, the fitness value of the recommended team is given by the sum of the battle values over all Pokémon character combinations, recommended and rival.

(a) Team before crossover (b) Team after crossover

Fig. 4. Example of crossover operation using Pokémon's Pokedex number.

E. Crossover

As mentioned in Section II, genetic and memetic algorithms apply a crossover technique. A crossover consists of an exchange of genes between two chromosomes (individuals or solutions). As described in Section IV-C, an individual is composed of a team of three Pokémon characters and a chromosome is a set of genes, with each gene being a Pokémon character. Thus, a crossover between two teams consists of a Pokémon trade.

During the crossover process, a recombination point is selected and it is used to cut both chromosomes, this cut separates genes into two parts, where the first part of genes stays in the corresponding chromosome, while the second group will be exchanged with the second part of the other chromosome [6]. Fig. 4 presents how a crossover operation between two teams is performed using two individuals (teams). Fig. 4a represents two teams randomly selected and Fig. 4b represents the Crossover in a cut point, that is randomly selected for each pair of teams.

F. Mutation

In nature, genetic mutations can occur due to the chemical instability of some bases as well as to errors during DNA replication [18]. Additionally, a mutation can increase or decrease an individual's fitness, and can sometimes generate an individual with a fitness that can not be generated by any crossover in a population.

In this application, a mutation is applied in all three optimization algorithms (ILS, GA and MA). Hence, given an individual and a database with all Pokémon characters, a mutation in a team (individual) is represented by the replacement of one of its member, chosen randomly, by another member from the Pokémon database, also randomly chosen.

G. Algorithm Description

In this section, we will describe the overall flow of the selected algorithms. As previously mentioned, the ILS algorithm is the only trajectory-based algorithm. The ILS made a random disturbance to generate new teams to evaluate the fitness. This strategy tries to maximize the team's evaluation changing the Pokémon teams. This process is repeated until a max number of the iteration is archived. Therefore, the final solution is the best-evaluated team over all iterations.

For the population-based algorithms (GA and MA), the same general flow will be applied, but with a difference. As previously mentioned, MA is a GA algorithm with a local search algorithm after the crossover and mutation steps. The GA main flow is described following the steps:

1) Generate the first population randomly;
2) Apply genetic operators (crossover and mutation);
3) Measure this population using the selected fitness function;
4) Select individuals using the elitism strategy;
5) Generate the next population of the genetic algorithm with the selected individuals.

The MA flow requires an extra local search step which is set to be before the calculation of the population fitness (step 3). Therefore, a step 2.5 is created in which an ILS method is used to apply a local search on all individuals of the current population.

V. Experimental Results

This section presents the results of the empirical analysis using the three different optimization techniques, ILS, GA and MA. As previously mentioned, we created a database containing 1000 different Pokémon teams that were randomly generated.

This database is given as input to our approach, where each optimization technique searches for the optimal solution for each input team. As these techniques are non-deterministic ones, each solution is executed five times. Then, the home team that possesses the best fitness value is selected as the result of the iteration. Finally, once the algorithms are non-deterministic, it is necessary to reduce the variations of these algorithms. After the five iterations of the same team, the average time and fitness are computed to describe the result of the team. Fig. 5 describes the flow to generate the results analyzed in this paper. All analysis performed in this section were made with a thousand results, thus, one result represents the average for each team in the evaluated metric.

The experiments were executed on a computer with 15GB of RAM and an octa-core AMD Ryzen 7 1800X processor. The experiments were run using the Python programming language and the pyeasyga module[1].

Tables III - IV show the results for the fitness and execution time metrics for each optimization algorithm, respectively. In these tables, we report the average values, along with standard deviation, best and worst values for each metric. The best result for each metric is highlighted in bold in each column.

[1]Avaliable in https://github.com/remiomosowon/pyeasyga

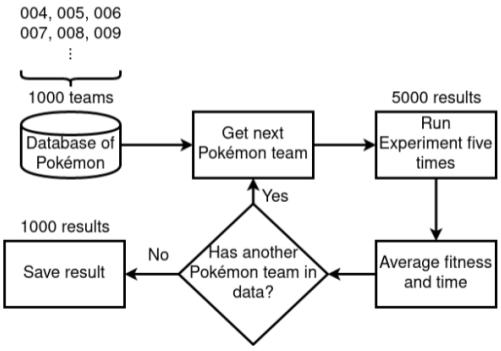


Fig. 5. Experiments Flow

For the fitness metric (Table III), it can be observed that MA obtained the best values in all cases (3 out of 3). In other words, the MA technique provided the most efficient home teams for a Pókemon game, according to the fitness function used in this analysis (described in Section IV-D). It is an expected result since it is a population-based method and it is a variation of a GA, with more elaborated search techniques.

TABLE III
Fitness Results

Alg.	Fitness ↑		
	Mean ± SD	**Best**	**Worst**
ILS	$31,224.32 \pm 7,943.70$	$72,722.20$	$12,716.70$
GA	$43,113.00 \pm 10,482.30$	$98,276.60$	$19,082.40$
MA	$\mathbf{50,366.27 \pm 12,055.53}$	$\mathbf{105,757.40}$	$\mathbf{26,337.20}$

TABLE IV
Execution Times

Alg.	Time ↓		
	Mean ± SD	**Best**	**Worst**
ILS	$\mathbf{1.49 \pm 0.11}$	1.35	$\mathbf{1.88}$
GA	1.51 ± 0.10	$\mathbf{1.18}$	1.94
MA	13.41 ± 1.00	10.60	16.67

However, when the execution time is analyzed (Table IV), MA takes 10 seconds to generate the result in the best case and it takes over 16 seconds to generate the result in the worst case. On the other hand, both GA and ILS take an average time around 1.5 seconds to generate a result, with ILS being faster than GA by 0.02 seconds on average.

In summary, we can state that MA delivers the best results and it should be used when the time requirement is not an important aspect. When comparing ILS to GA, GA achieves fitness values much larger than ILS, around $11,888.68$ points higher. In terms of execution time, both GA and ILS provide similar results. In this sense, one can state that GA provides the best overall result, taking into consideration fitness and execution time.

After evaluating the average performance of the optimization techniques, we illustrate the results in Fig. 6 and Fig. 8, that show time and fitness for all test cases individually. Fig. 6 shows the execution time results, for all compared algorithms, where we can see that MA takes the highest time when compared to other algorithms, for all test cases. The difference in execution times is a bit high. Both GA and ILS are flattened into the same line because the average difference between these algorithms was 0.02 ± 0.01. On the other hand, we expected that the MA achieves the highest execution time once it performs a local search for each individual in each generation. Fig. 7 shows a focus on Genetic and ILS methods; once these methods are flattened in Fig. 6, when both optimization methods are compared using the average time, the ILS obtains a better performance. This is justified because this method runs a fixed number of iterations, and the exchanges occur, at most three times in each iteration. On the other hand, in the case of the genetic algorithm, we define the stop criteria as being twenty iterations, or the population of the algorithm represents the same solution or permutations of this. Although we were not expecting too much similarity in the results of our experiments, both methods have the same total number of iterations. However, in the genetic algorithm, its execution is divided into 20 iterations of 50 individuals.

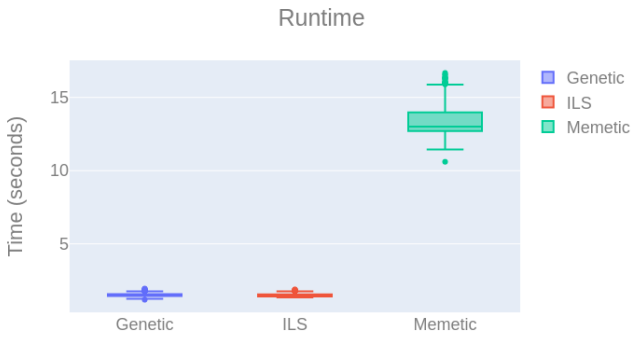


Fig. 6. The distribution of the time for all test cases.

As it can be seen in Fig. 8, MA produces the highest fitness values for almost all test cases, followed by GA and ILS. However, it can be observed an instability of all optimization techniques, being more consistent and stronger for MA. This behaviour is corroborated in Table III in which MA provides the highest standard deviations. Therefore, it can be concluded that although MA provides the more accurate values, it is an unstable method, being strongly dependent on the input case.

Table V presents the p-value of the Shapiro-Wilk normality test (first two columns) and the results of the post-hoc Friedman test (last two columns). Based on the results of this table, we can assume that the distributions delivered by the optimization techniques does not follow a normal distribution. In this case, we can use non-parametric tests to analyze the obtained results. In order to apply both the normality and

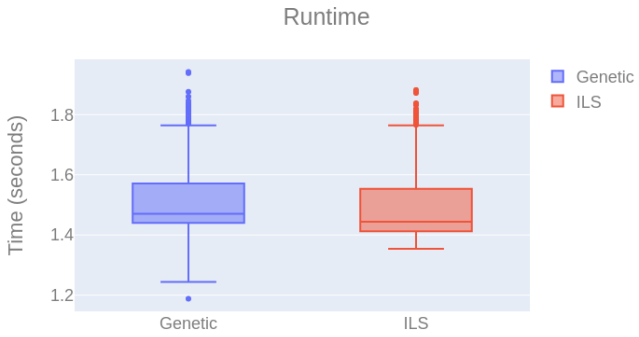


Fig. 7. The distribution of the time with focus in ILS and Genetic.

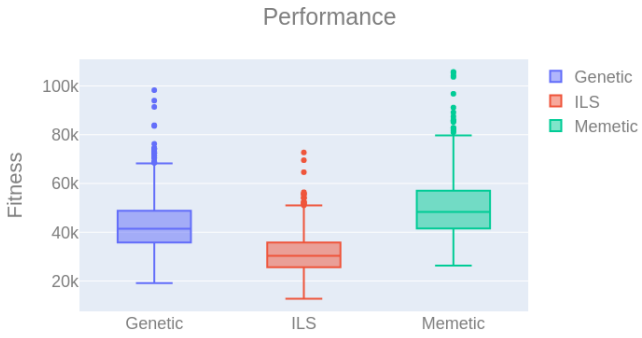


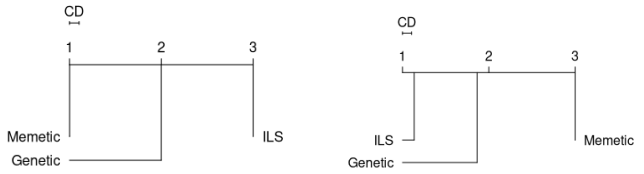
Fig. 8. The distribution of the fitness for all test cases.

post-hoc Friedman tests to evaluate the distribution of the population, we first summarize the average of the results based on same rival's team. After this step, 1000 results were summarized, and then these results are used in the statistical test to evaluate the population. Finally, one way to present the obtained result is through a critical difference diagram, which is a diagram that makes it easier to analyze and to show with clarity which algorithm is better than the others. It is applied to the result of the used post-hoc Friedman test, the Nemenyi test.

TABLE V
SHAPIRO-WILK AND POST-HOC FRIEDMAN TESTS

Metric	Shapiro-Wilk		post-hoc Friedman	
	Algorithm	p-value	Algorithms	p-value
Time	GA	$< \mathbf{0.001}$	GA × ILS	$< \mathbf{0.001}$
	ILS	$< \mathbf{0.001}$	GA × MA	$< \mathbf{0.001}$
	MA	$< \mathbf{0.001}$	ILS × MA	$< \mathbf{0.001}$
Fitness	GA	$< \mathbf{0.001}$	GA × ILS	$< \mathbf{0.001}$
	ILS	$< \mathbf{0.001}$	GA × MA	$< \mathbf{0.001}$
	MA	$< \mathbf{0.001}$	ILS × MA	$< \mathbf{0.001}$

Fig. 9 shows the critical difference diagram for both analysed metrics, the fitness and run time. The leftmost method



(a) Fitness (b) Time

Fig. 9. Critical Difference Diagram with evaluated methods.

obtained the lowest ranks, and the rightmost method obtained the highest ranks when the results are ranked. Moreover, methods that are not covered by a critical difference horizontal line are statistically different; otherwise, the null hypothesis of the Friedman test cannot be refuted. When analysing the results obtained by the post-hoc Friedman test (in both, Fig. 9 and Table V), we can see that both comparisons are statistically significant, for all analyzed two-by-two analysis and both evaluation metrics. Hence, these values show that, in relation to execution time, ILS $>$ GA $>$ MA. However, when considering fitness, MA $>$ GA $>$ ILS.

VI. Conclusion and Future Work

In this paper, we analyzed the use of three optimization algorithms, ILS, GA and MA, to generate Pokémon home teams to a specific rival team. The main aim was to provide the best team configuration to beat a rival team in a Pókemon game. In this game, the available possibilities lead to vast search space. Additionally, any variation on the length of the team configurations results in a change in the search space size.

An empirical analysis was conducted to perform this investigation. As a result of this analysis, it observed that the ILS is the less time-consuming approach of all three because of a reasonably low fixed number of iterations (1,000). Also, MA is the slowest of the tested techniques, since it performs a local search (ILS) with ten iterations for each individual in each generation. As there are 50 individuals in each generation, it requires a larger processing load. However, when the fitness values are analyzed, MA delivers the highest fitness values, since it performs a more in-depth search of the solution space. On the other hand, ILS obtains the worst fitness evaluation. We believe that is because it uses a fixed number of iterations and only one solution. Therefore, ILS may not perform well in a exhaustive search into space since it depends on many factors (e.g. seed, number of swaps).

In summary, as expected, the ILS obtained the best execution time results; on the other hand, the worst fitness results. Hence, this technique is useful in real-time situations, when fast responses are required but one does not need too much precision. In contrast, GA did not obtain the best results for both execution time and fitness metrics, but also it did not achieve the worst results. It means that GA may be chosen when a good trade-off solution is needed, being used in situations where a fast time and high fitness are both required. Finally, MA obtained the best fitness values. On the other hand, it achieved the worst execution time results, meaning that it is useful in situations where an accurate response is required, and time is not an issue.

In general, we strongly recommend the use of the GA approach if the time is primordial for the application. Due to this algorithm, executes fast when compared with the MA. Hence, if the fitness of the home team is more critical and the time is available, the best option is the MA. However, in our analysis consists of a hypothetical case when the rival's team is previously known. Besides, we just study a single instance of battle, Player versus Team Rocket (PvE battles) and this battle occur with three Pokémon in each team.

The results obtained in this paper are promising and, our future work will be focused on the investigation of other optimization algorithms, parameter tuning, cross-over and combinations for all of the above. We also plan to investigate the local search in MA with different algorithms, e.g., Tabu search.

Acknowledgment

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001.

References

[1] Koichi Iwabuchi. How 'japanese'is pokémon. *Pikachu's global adventure: The rise and fall of Pokémon*, pages 53–79, 2004.

[2] Janne Paavilainen, Hannu Korhonen, Kati Alha, Jaakko Stenros, Elina Koskinen, and Frans Mayra. The Pokémon GO experience: A location-based augmented reality mobile game goes mainstream. In *Proceedings of the 2017 CHI conference on human factors in computing systems*, pages 2493–2498, 2017.

[3] Xingye Dong, Houkuan Huang, and Ping Chen. An iterated local search algorithm for the permutation flowshop problem with total flowtime criterion. *Computers & Operations Research*, 36(5):1664–1669, 2009.

[4] Michael D Vose. *The simple genetic algorithm: foundations and theory*. MIT press, 1999.

[5] William Eugene Hart, Natalio Krasnogor, and James E Smith. Memetic evolutionary algorithms. In *Recent advances in memetic algorithms*, pages 3–27. Springer, 2005.

[6] Marco Aurélio Cavalcanti Pacheco et al. Algoritmos genéticos: princípios e aplicações. *ICA: Laboratório de Inteligência Computacional Aplicada. Departamento de Engenharia Elétrica. Pontifícia Universidade Católica do Rio de Janeiro*, page 28, 1999.

[7] Saumya Bansal and Niyati Baliyan. Memetic algorithm based similarity metric for recommender system. In *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing Companion*, pages 143–144, 2019.

[8] Shanfeng Wang, Maoguo Gong, Haoliang Li, Junwei Yang, and Yue Wu. Memetic algorithm based location and topic aware recommender system. *Knowledge-Based Systems*, 131:125–134, 2017.

[9] Linus Norström. Comparison of artificial intelligence algorithms for pokémon battles. Master's thesis, Chalmers University of Technology, Sweden, 2019.

[10] D. Huang and S. Lee. A self-play policy optimization approach to battling pokémon. In *2019 IEEE Conference on Games (CoG)*, pages 1–4, 2019.

[11] Pablo García-Sánchez, Alberto Tonda, Antonio J. Fernández-Leiva, and Carlos Cotta. Optimizing hearthstone agents using an evolutionary algorithm. *Knowledge-Based Systems*, 188:105032, 2020.

[12] Gerald Tesauro and Gregory R Galperin. On-line policy improvement using monte-carlo search. In *Advances in Neural Information Processing Systems*, pages 1068–1074, 1997.

[13] Dominik Ślezak and Jakub Wroblewski. Classification algorithms based on linear combinations of features. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 548–553. Springer, 1999.

[14] David Simões, Simão Reis, Nuno Lau, and Luís Paulo Reis. Competitive deep reinforcement learning over a pokémon battling simulator. In *2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 40–45. IEEE, 2020.

[15] Ricardo Linden. *Algoritmos genéticos (2a ediçao)*. Brasport, 2012.

[16] Pokego Complete. How CP on this web site is calculated. https://gamepress.gg/pokemongo/pokemon-stats-advanced, April 2020.

[17] Gamepress. Pokemon Go IV calculator. https://gamepress.gg/pokemongo/pokemongo-iv-calculator, April 2020.

[18] H Lodish, A Berk, SL Zipursky, P Matsudaira, D Baltimore, and J Darnell. *Molecular Cell Biology*. W. H. Freeman, New York, 4th edition, 2000. Available from: https://www.ncbi.nlm.nih.gov/books/NBK21475/.