Genetic Algorithms in Combinatorial Optimization: An Evolutionary Approach for Pokémon team generating

Marcelo de M. Fonseca Jr.[1]

Universidade Presbiteriana Mackenzie
[1] Pós-Graduação em Engenharia Elétrica e Computação, São Paulo, SP – Brasil
72356741@mackenzista.com.br

Abstract. Genetic Algorithms (GAs), inspired by the principles of natural evolution, have emerged as a potent tool for tackling complex challenges in diverse domains, particularly in combinatorial optimization. This article delves into the practical application of GAs in solving a specific instance of a complex combinatorial problem—forming the optimal Pokémon team based solely on the types of Pokémon in both the player's and opponents' teams.

Throughout this exploration, we comprehensively delve into the theoretical underpinnings of Genetic Algorithms, dissecting their core components and operators. The article highlights their functionality within the context of the Pokémon team optimization problem, a novel application of this evolutionary approach. The study encompasses real-world case studies, providing concrete examples of the successful deployment of GAs.

Furthermore, we unveil the methodology used to address this unique combinatorial problem and share the code and resultant findings, fostering a deeper understanding of both Genetic Algorithms and their adeptness in tackling combinatorial optimization. The research endeavors to present and collect insights and outcomes in a scenario that deviates from the conventional, thereby equipping readers with a robust appreciation of the capabilities and adaptability of Genetic Algorithms.

Keywords: Genetic Algorithm, Pokémon, Evolution, Crossover, Mutation, Population, Natural Selection, Combinatory, Optimization.

1 Introduction

In the search for efficient solutions to complex challenges in areas ranging from production engineering to logistics, Genetic Algorithms have emerged as a powerful and versatile tool. Inspired by the process of natural evolution, these algorithms have been widely applied to solving combinatorial optimization problems, in which the search for the best solutions must consider a vast space of possible combinations.

This article explores the application of Genetic Algorithms in solving a reduced space of a complex combinatorial problem. By combining concepts from evolutionary biology with computing techniques, we are going to apply this type of algorithm to solve the best Pokémon team combination looking only by the types of the Pokémon's

of both teams in which we already know what Pokémon's our opponents are going to use.

Throughout this article, we will examine the theoretical basis of Genetic Algorithms, their main components and operators, as well as their operation in the problem. Additionally, we will present case studies that highlight the successful application of these algorithms, the methodology to solve our combinatorial problem and share our code and results.

Our goal is to apply and collects results of a combinatorial problem using Genetic Algorithms in such a scenario and thought this implementation provide the readers with a solid understanding of Genetic Algorithms and their capabilities in combinatorial optimization.

2 Genetic Algorithm

Genetic algorithms (GAs) trace their origins to the pioneering work of John Holland, who is often regarded as the father of genetic algorithms. In the early 1960s, Holland, a computer scientist and psychologist, developed the concept of adaptation by Charles Darwin's theory of natural selection, outlined in his seminal work "On the Origin of Species" (Darwin, 1859), is one of the cornerstones of evolutionary science, in the context of computer-based problem-solving. His groundbreaking book, "Adaptation in Natural and Artificial Systems," published in 1975, introduced the fundamental principles of GAs.

Holland's work laid the foundation for GAs, which mimic the process of natural selection to solve optimization and search problems. These algorithms have since evolved and found application in diverse fields, including engineering, economics, and machine learning, making them a powerful tool for complex problem-solving (Holland, 1975).

2.1 Use Cases

- **Optimization Problems.** Genetic algorithms are frequently used to find optimal or near-optimal solutions for complex optimization problems. This can include problems in engineering, logistics, finance, and scheduling.
- **Robotics.** Genetic algorithms can help in evolving control strategies and behaviors for robots in various environments.
- **Scheduling.** Genetic algorithms can optimize scheduling problems in manufacturing line, transportation, and project management.
- **Vehicle Routing.** Optimizing delivery routes for transportation companies to minimize costs and delivery times.

2.2 Advantages of Genetic Algorithms:

- **Global Search.** GAs are good at finding solutions in a large search space. They can explore a wide range of potential solutions.

- **Parallelism.** Genetic algorithms are inherently parallel, which makes them suitable for parallel and distributed computing environments.
- **Versatility.** GAs are versatile and can be applied to a wide range of problems, including optimization, machine learning, and rule discovery, among others.
- **Adaptability.** GAs can adapt to changing environments or problem spaces. They can continue to search for solutions even as the problem evolves.

2.3 Disadvantages of Genetic Algorithms

- **Computational Intensity.** Genetic algorithms can be computationally expensive, especially when dealing with large populations or high-dimensional search spaces.
- **Premature Convergence (Local Optima).** GAs may converge to suboptimal solutions prematurely if the parameters are not properly set, or if the selection pressure is too high. Not providing guarantees of finding the global optimum, and the quality of the solutions found can vary.
- **Representation Issues.** Choosing an appropriate representation for the problem can be challenging, and the choice can greatly affect the performance of the algorithm.
- **Time-Consuming for Large Search Spaces.** GAs may require a significant amount of time to explore large search spaces thoroughly.

In summary, genetic algorithms are a powerful optimization technique, but they are not a one-size-fits-all solution. They excel in global optimization and can handle complex problems, but their performance depends on parameter tuning and problem representation as well the space of being analyzed.

2.4 Crossover Techniques:

- **Single-Point Crossover.** In this technique, a single crossover point is selected in the parent chromosomes, and the genetic material is swapped between them at that point.
- **Uniform Crossover.** This method involves randomly selecting genes from each parent with equal probability, creating a child chromosome by combining these genes.
- **Arithmetic Crossover.** In this continuous-variable GA, a weighted average of genes from the parents is taken to create a child chromosome.
- **Edge Recombination Crossover (ERX).** ERX is used for the traveling salesman problem (TSP). It constructs the child by considering the edges common to the parents.

2.5 Mutation Techniques:

- **Bit Flip Mutation.** This technique is used for binary-encoded chromosomes. It involves randomly flipping the bits of a chromosome with a certain probability.
- **Swap Mutation.** Commonly used for permutation problems, swap mutation selects two positions in a chromosome and swaps the elements at those positions.

- **Gaussian Mutation.** This mutation technique is used for continuous-variable GAs. It adds a small amount of Gaussian noise to the gene values.
- **Insertion Mutation.** For permutation problems, insertion mutation selects a position and inserts an element from one location to another within the chromosome.

2.6 Selection

Selection is a critical component of genetic algorithms (GAs) that determines which individuals in the current population will be chosen as parents for creating the next generation. The goal of selection is to favor individuals with better fitness values, allowing them to pass their genetic material to the next generation. Here are some common selection techniques used in genetic algorithms:

- **Roulette Wheel Selection (Fitness Proportional Selection)** In this method, individuals are selected with a probability proportional to their fitness. The higher an individual's fitness, the more likely it is to be chosen.
- **Tournament Selection.** Tournament selection involves selecting a random subset of individuals from the population and then choosing the best (highest fitness) individual from that subset. The size of the tournament group is a parameter that can be adjusted.
- **Rank-Based Selection.** In rank-based selection, individuals are ranked by their fitness, and selection is based on the rank rather than the actual fitness values. This method can help maintain diversity in the population.
- **Stochastic Universal Sampling.** Stochastic universal sampling is a variation of roulette wheel selection that selects multiple individuals simultaneously by evenly spacing "pointers" around the wheel. This technique can be more efficient and less biased than traditional roulette wheel selection.
- **Elitist Selection.** Elitist selection ensures that the best-performing individuals from the current generation are preserved and directly copied to the next generation without any modification.
- **Fitness Sharing.** Fitness sharing is another technique used in multi-objective optimization. It adjusts an individual's fitness based on the number of other individuals in its proximity.

The choice of selection technique depends on the specific problem and the goals of the genetic algorithm. Different selection methods offer different trade-offs between exploration and exploitation, diversity preservation, and convergence speed.

3 Methodology

Now on we are going to describe the implementation of the previously described algorithm and what techniques were used. How the parameters were chosen and in addition to other details relevant to the reproducibility of what was done.

We could have developed a reinforcement learning approach in which we would generate the team and simulate the battle (sequentially, randomly, or based on attribute affinity) to obtain results. This would have been a other valid implementation approach for the problem. However, I opted for the genetic approach because I wanted to simulate a natural evolution process by exercising the skill, and it will be a type of algorithm that I will apply in future applications.

3.1 Experiment

The experiment consist in finding the best Pokémon team combination looking only by the types of the Pokémon's of both teams in which we already know what Pokémon's our opponents are going to use.

The premises used for the reduction of the space complexity is that:

- the Pokémon's only use attack of their type's;
- Only their stats (hp, total attack, total defense, speed) and type affinity were used to consider their effeteness against their opponents;
- Special attack and attack were summed to total attack;
- Special defense and defense were summed to total defense;
- Default team of 6 Pokémon's were used in the simulations;

3.2 Data Used

Pokedex.
A list of Pokémon's for their characteristics, types and stats.

- https://gist.github.com/armgilles/194bcff35001e7eb53a2a8b441e8b2c6
- https://www.kaggle.com/datasets/rounakbanik/pokemon

Matrix multipliers types.
A matrix of advantages of each of the types to account the attack multiplier for a type of Pokémon based on his opponent.

- https://bulbapedia.bulbagarden.net/wiki/Type#Type-affected_game_mechanics

3.3 Characterization of the problem

Genotype.

Considered a genotype as a team of six individuals. Each of the individuals can be one of the 800+ Pokémon's form the database used.

Fig. 1. Representation of the genotype

Phenotype.

Considered the phenotype a single Pokémon.

Fig. 2. Representation of the phenotype

The individuals have the following characteristics varying from Pokémon to Pokémon:

- pokedex_number: Numeric identifier of a Pokémon on the national pokedex;
- name: string identifier of a Pokémon;
- type1: the first type of the Pokémon;
- type2: the second type of the Pokémon if exists;
- hp: the base hp of the Pokémon on their max level;
- attack_total: the base sum of attack and special attack of the Pokémon on their max level;
- defense_total: the base sum of defense and special defense of the Pokémon on their max level;
- speed: the base speed of the Pokémon on their max level;
- capture_rate: the capture rate of the Pokémon on the game;

Fitness.

$$fitness = \frac{hp}{hpOponente} + \left(\frac{Attack}{DefenseOponente} \cdot AttackMultiplier\right) \cdot \frac{Speed}{SpeedOponente}$$

That way each Pokémon of our team is evaluated throughout each Pokémon of our opponent. In which we would have the sum of 36 fitnesses comparing the effeteness of our team against our opponent's.

3.4 Steps of the algorithm

- **Initialization.** Start by creating an initial population of candidate solutions (teams of individuals). These solutions are generated randomly.
- **Evaluation.** Evaluate the fitness of each team in the population. The fitness function quantifies how well each solution performs with respect to the optimization objective. Higher fitness indicates better solutions.
- **Selection.** Select teams from the current population to serve as parents for the next generation. The selection process were based on the technique of the default roulette wheel, for simplify the complexity. In which with higher fitness values of an individual have higher probability of being selected.
- **Crossover (Recombination).** Take pairs of selected parents and perform crossover or recombination to create new individuals (offspring). Crossover combines genetic information from the parents, creating one children. Used the Single-Point Crossover for simplify the complexity.
- **Mutation.** Introduce genetic diversity by applying mutation to some of the offspring. Mutation involves making small random changes to an team's genetic code. Mutation helps the algorithm explore the search space more thoroughly with a given mutation rate. If activated a new random Pokémon will be allocated on the team.
- **Replacement.** Create the next generation by combining the offspring generated through crossover and mutation with the existing population. Various replacement strategies can be used. In our case was used generational replacement for simplify the complexity.
- **Termination Criteria.** Check termination conditions. The algorithm Termination Criteria, chosen by simplicity, will terminate when a maximum number of generations is reached.

4 Results

This topic we are going to share and describe the results obtained, relating this to the parameters used and making a quantitative and qualitative assessment of these results.

4.1 Experimental Procedure:

There were three steps so we could conclude the experiment.

1. Manipulate the dataset
2. Implement the classes and the methods for the genetic algorithm
3. Run the code, debugging and collecting the results

The dataset's we manipulated se that in executing the algorithm all was already prepared for we didn't have the need for implementing methods for working the data, optimizing the execution.

The implementation of the classes resulted on three of them:

– Individuals;
– TeamIndividuals;
– GeneticAlgorithm;

That way every part of the code have responsibility well defined. The same line of thinking were used for implementing the methods.

At least running the code and debugging as erros ocurrs and adaptation need was implemented.

4.2 Data Analysis:

For sharing the data in this article, we executed the algorithm fixating the size of the population on 20 and the number of generation on 100, varying only the mutation rate. We chosen this number of generations so do not extend the processing and we could clearly have data point on the timeline. The number for size of the population was defined so do not have a time consuming execution. So as those parameters were the more critical of time execution we decide to vary the mutation rate.

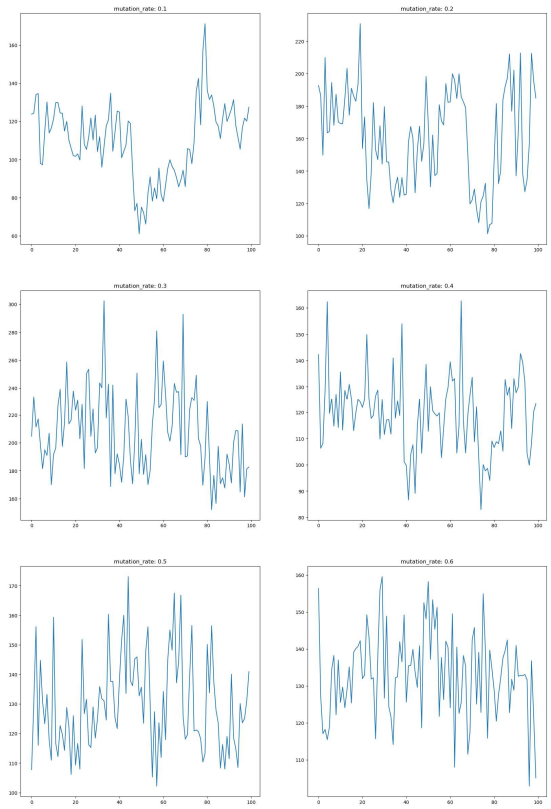

Fig. 3. Execution of the algorithm varying the mutation rate

Fig. 4. Algorithm best fitted team

As we can see in fig. 4 indeed, the team suggested by the algorithm have advantage against our opponent's team

4.3 Replicability:

To allow other researchers to replicate your study and verify your results, follows the source code:

https://github.com/FroschFT/NaturalComputing_artigo

4.4 Future endeavors

Here we are going to discuss what colud and can be done to complement the experiment.

Other's types of selections

We colud use what we discussed on subsection 2.6 Selection to observe the behavior of other types of selection other then the roulette selection implemented.

Faithfully replicate the pokemon battle system

In this subsection, discuss the possibilities of expanding the current simulation of the Pokémon battle system. Explore how it can be enhanced by replicating various aspects, such as the battle mechanics, attacks, items, and abilities.

Delve into the potential for developing more advanced attack strategies within the simulation. Explore how genetic algorithms can be used to optimize the selection of attacks, considering factors like type advantages, move sets, and opponent strategies.

Discuss the application of genetic algorithms in optimizing item usage during battles. Explain how these algorithms can be used to determine the most effective timing and choices for items like Potions, Berries, and status-affecting items.

Poisson Distribution

Explore the potential for incorporating probability models like the Poisson Distribution to create a more realistic and dynamic Pokémon capture simulation. Discuss how the capture_rate, a fundamental aspect of the Pokémon games, can be integrated into

the genetic algorithm to influence the likelihood of capturing Pokémon during generating the teams or during the mutation all based on the capture rate of the Pokémon.

5 Conclusões

In the ever-evolving landscape of problem-solving, Genetic Algorithms have proven to be a remarkable and versatile tool, offering innovative solutions to multifaceted challenges. Drawing inspiration from the intricate processes of natural evolution, these algorithms have been successfully applied to address complex combinatorial optimization problems across diverse domains, from production engineering to logistics. In this article, we embarked on an exciting journey, merging concepts from evolutionary biology with computational techniques, to explore the application of Genetic Algorithms in solving a distinctive challenge.

Our mission centered on discussing and implementing a genetic algorithm in a problem was successful on the team optimization problem. Through a meticulous examination of the theoretical underpinnings, core components, and fundamental operators of Genetic Algorithms, we strived to uncover the inner workings of this powerful tool.

By presenting case studies and methodology details, we illuminated the successful application of these algorithms in a real-world context. Moreover, our commitment to transparency was underscored as we openly shared our code and results, inviting readers into the intricacies of our process.

Our ultimate aspiration has been to offer a unique vantage point, not only in showcasing the application of Genetic Algorithms in a distinctive scenario but also in providing readers with a deeper comprehension of these algorithms and their prowess in the realm of combinatorial optimization. As we venture forward, the possibilities are limitless, and Genetic Algorithms stand ready to unlock innovative solutions to the intricate challenges that await us on this exciting journey of problem-solving and discovery.

References

1. Bulbapedia article, https://bulbapedia.bulbagarden.net/wiki/Damage, last accessed 2023/10/20
2. de Castro, L.N. (2006). Fundamentals of Natural Computing: Basic Concepts, Algorithms, and Applications (1st ed.). Chapman and Hall/CRC.
3. Holland, J. H. (1975). Adaptation in Natural and Artificial Systems. University of Michigan Press.
4. Darwin, Charles, and Leonard Kebler. On the origin of species by means of natural selection, or, The preservation of favoured races in the struggle for life . London: J. Murray, 1859.
5. Weisstein, Eric W. “Poisson Distribution.” From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/PoissonDistribution.html