Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia

COPPE UFRJ

PROFIT MODEL AND OPTIMAL BREEDING STRATEGY APPLIED TO THE AXIE INFINITY GAME

Henrique José dos Santos Ferreira Júnior

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientador: Daniel Ratton Figueiredo

Rio de Janeiro
Fevereiro de 2024

PROFIT MODEL AND OPTIMAL BREEDING STRATEGY APPLIED TO THE AXIE INFINITY GAME

Henrique José dos Santos Ferreira Júnior

Orientador: Daniel Ratton Figueiredo

RIO DE JANEIRO, RJ – BRASIL
FEVEREIRO DE 2024

dos Santos Ferreira Júnior, Henrique José

Profit Model and Optimal Breeding Strategy Applied to the Axie Infinity Game/Henrique José dos Santos Ferreira Júnior. – Rio de Janeiro: UFRJ/COPPE, 2024.

XI, 59 p.: il.; $29,7$cm.

Orientador: Daniel Ratton Figueiredo

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2024.

Referências Bibliográficas: p. 55 – 58.

1. Optimization. 2. Profitability. 3. Breeding. I. Ratton Figueiredo, Daniel. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

A minha esposa Jennifer por todo apoio prestado.

Agradecimentos

Ao atingir esse ponto, de galgar o título de mestre, é possível perceber, novamente, as grandes coisas que Deus faz em nossa vida. O agradecimento é primeiramente a Ele. Agradeço também à minha esposa Jennifer, que apoiou as várias mudanças que ocorreram nesse caminho e me deu as forças para terminar esse mestrado. Meus pais e irmãos, sempre presentes dando o incentivo para voar cada vez mais alto.

Esse mestrado só foi possível se iniciar e terminar graças ao meu orientador Daniel, que me deu a oportunidade e toda a orientação necessária pra o sucesso desse trabalho. Da mesma forma, um agradecimento especial a professora Janaína, ela me ensinou a programar, me encorajou a trilhar o caminho da computação e me apresentou ao professor Daniel. Desde então tenho buscado lagos cada vez maiores...

A todos que vocês que me fizeram chegar até aqui, minha mais profunda gratidão. Obrigado!

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

PROFIT MODEL AND OPTIMAL BREEDING STRATEGY APPLIED TO THE AXIE INFINITY GAME

Henrique José dos Santos Ferreira Júnior

Fevereiro/2024

Orientador: Daniel Ratton Figueiredo

Programa: Engenharia de Sistemas e Computação

Os jogos online do tipo "play-to-earn"ganharam grande popularidade, oferecendo aos jogadores a oportunidade de gerar lucros reais. Entre esses jogos, Axie Infinity se destaca como um excelente exemplo, permitindo aos jogadores se envolverem na procriação de criaturas virtuais, representadas como Tokens Não Fungíveis (NFTs) únicos. Os jogadores podem comprar criaturas em um mercado, procriar criaturas pagando uma taxa de procriação e vender as crias no mesmo mercado, com potencial de obter lucro. No entanto, um desafio fundamental neste cenário é a seleção de pares reprodutores, uma vez que cada par pode produzir milhares de crias diferentes com preços de venda potencialmente diferentes. Além disso, as combinações de pares reprodutores crescem de forma quadrática com o número de criaturas disponíveis para reprodução. Esta tese concentra-se na otimização de lucros em jogos baseados em NFT, particularmente Axie Infinity, propondo um modelo matemático para o problema de reprodução como um novo Problema Não Linear Inteiro Misto (em inglês, MINLP). Exploramos dois cenários: utilização de NFTs disponíveis ao jogador para criação; comprando NFTs no mercado. Nossa abordagem de solução em dois estágios envolve a metaheurística Adaptive Large Neighborhood Search (ALNS) para identificar mais rapidamente pares reprodutores lucrativos, seguida pelo solucionador MINLP para determinar estratégias ideais de reprodução. Até onde sabemos, este trabalho apresenta a primeira formulação MINLP e uma solução eficiente para otimização de lucros em jogos online envolvendo procriação de NFT. A validação usando dados reais do Axie Infinity demonstra a eficiência da abordagem proposta na obtenção de lucros quase ótimos em baixo tempo computacional. Por último, esta pesquisa fornece ferramentas práticas para procriadores de NFT que buscam aumentar a lucratividade por meio de estratégias de procriação otimizadas.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

PROFIT MODEL AND OPTIMAL BREEDING STRATEGY APPLIED TO THE AXIE INFINITY GAME

Henrique José dos Santos Ferreira Júnior

February/2024

Advisor: Daniel Ratton Figueiredo

Department: Systems Engineering and Computer Science

Play-to-earn online games have gained widespread popularity, offering players the opportunity to generate profits within and outside the game. Among these games, Axie Infinity stands out as a prime example, allowing players to engage in the breeding of virtual creatures, represented as unique Non-Fungible Tokens (NFTs). Players can purchase creatures from a marketplace, breed creatures by paying a breeding fee, and sell the offspring in the same marketplace, with the potential of making a profit. However, a key challenge in this scenario is selecting breeding pairs, as each pair can yield thousands of different offspring with potentially different selling prices. Moreover, pairwise combinations of breeding pairs grows quadratic with the number of creatures available for breeding. This thesis focuses on profit optimization within NFT-based games, particularly Axie Infinity, proposing a mathematical model for the breeding problem as a novel Mixed-Integer Nonlinear Problem (MINLP). We explore two scenarios: using NFTs available to the player for breeding; buying NFTs from the marketplace. Our two-stage solution approach involves the Adaptive Large Neighborhood Search (ALNS) metaheuristic to more quickly identify lucrative breeding pairs, followed by the MINLP solver to determine optimal breeding strategies. To the best of our knowledge, this work introduces the first MINLP formulation and an efficient solution for profit optimization in online games involving NFT breeding. Validation using real data from the Axie Infinity marketplace demonstrates the efficiency of the proposed approach in achieving near-optimal profits within reduced computational time. Last, this research provides practical tools for NFT-based online game breeders seeking to enhance profitability through optimized breeding strategies.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Humans have been engaged in animal husbandry (breeding animals) for over 15000 years. In the beginning, the breeding aimed at the growth of meat supply, and to generate animals with bigger sizes and soft meat in a process known as selective breeding [4]. Then, since the introduction of currency about 2600 years ago, selective breeding had a new objective which is to maximize some form of profit [5]. In this context, the farmer has costs associated with the breeding, to keep and feed the animals in addition to special care needed for the offspring. Thus, the selling price of the offspring has to exceed these breeding costs to be profitable.

In the context of maximizing profits, many recent works use mathematical models and optimization to find the optimal breeding strategies to achieve an objective. This objective usually maximizes the occurrence of some specific traits chosen by the farmer, which are commonly chosen according to the market demand in order to have larger profits [6, 7]. Some works have designed and developed a software product that aims in maximizing profit directly in the optimization model through the estimation of breeding values (EBVs) [8–10].

Some modern computer games simulate farms and animal husbandry like the Farming Simulator. Although players make money inside these games they cannot withdraw money out of the game. However, the blockchain and its non-fungible tokens (NFT) opened an opportunity to make real money by breeding digital creatures. As with the real world, offspring in the digital world are random and their traits depend on the traits of their parents as well as on the breeding rules of the game. Thus, when breeding two creatures we have a probability distribution over the set of possible offsprings. In the blockchain, a digital creature is an NFT that is unique and may battle, be collected, and be used in breedings inside a game. For example, CryptoKitties - one of the first games to use NFT - is a game "where players purchase, breed, and trade virtual cats that have different visual features of varying levels of rarity"[11]. This trade uses real money.

Moreover, a new economy emerged from the Blockchain and NFTs technologies where assets can be negotiated worldwide without government interference. In this context, tokens may increase or decrease in value over time. Thus, many works have described the NFT markets from the portfolio management (PM) point of view [12, 13]. The goal of any PM is to analyze a market in order to determine strategies to achieve the financial objectives and risk tolerance. Thus, breeding an NFT would be described by PM as selecting the creatures (NFT) to invest in (buy) having profit as the main objective according to the breeder's risk tolerance.

In this context, Axie Infinity is an NFT-based online game that focuses on digital pets (Axies) that can battle and be bred. This game introduced the concept of "play-to-win", where players and breeders can make a profit by playing and breeding Axies. NFT-based games that have a breeding mechanism are similar to animal husbandry, as it is possible to perform selective breeding of the NFT creatures in order to improve some traits of the offspring and to maximize profit.

In this work, the main objective is to define an optimal breeding strategy with a profit objective for the Axie Infinity game. The idea is to estimate the expected breeding profit (EBP) with data available from Axie Infinity Marketplace, similar to the EBVs from animal husbandry. However, calculating the EBP for all the possible pairs of creatures available in the Marketplace would take months, requiring faster approaches to determine which creatures to buy and pairs to breed.

Thus, we formulated a Mixed-Integer Nonlinear Problem (MINLP) with a maximum profit objective to represent different optimization problems for the Axie Infinity game with the goal of determining optimal strategies in terms of profit. In this model, a breeder may buy the creatures from the marketplace, breed them, and then sell all the creatures in the Marketplace in order to make a profit. This model is applied to the Axie Infinity scenario in a two-stage approach: First, the Adaptive Large Neighborhood Search (ALNS) is designed to find the most profitable pairs by computing the EBP within a limited time; Then, the MINLP is solved using as input the most profitable pairs obtained by the ALNS. We evaluate the proposed approach by creating a dataset from the Axie Infinity Marketplace and also a ground truth to compare with the solutions of the proposed approach. Experiments validated the proposed approach since results observed for different scenarios were similar to results obtained using the ground truth but with a much smaller (feasible) running time.

1.1 Contributions

This work's main contributions are listed below:

- **Mathematical modeling of the breeding problem:** The proposed NFT optimal breeding strategy problem was modeled as a Mixed-Integer Nonlinear

Problem (MINLP) with a maximum profit objective for two different use-cases. In the first problem, the breeder may breed only its initial set of NFTs (no cost to aquire the NFTs), and, in the second problem, NFTs have to be purchased from the marketplace to breed and then sold back to the Marketplace in order to generate a profit. The second problem is the focus of this work. As far as we are concerned, this is the first work to model and optimize the profit of the NFT breeding policy in the context of online games.

- **A proposed ALNS-MP framework to solve the problem:** We proposed a two-stage approach to solve the NFT optimal breeding strategy. The first stage comprises the ALNS metaheuristic (with our proposed operators) to find the most profitable pairs in the marketplace. The second stage uses the individuals in the most profitable pairs as input to the proposed MINLP formulation to provide the optimal solution that is, which pairs to buy, breed, and sell.

- **ALNS operators designed for the Axie application:** We designed the destroy and repair operator specific for the Axie Infinity context. These operators were essential to achieve the desired performance.

- **An evaluation of the solution based on a real dataset collected for this dissertation:** We collected real data from the Axie Infinity Marketplace in order to evaluate the solution obtained by our proposed approach against the solution given by ground truth. Results revealed that our approach achieved near-optimum profit values in a much lower running time than the ground truth.

1.2 Organization

This work is organized into five additional chapters. Chapter 2 describes the necessary background and related works that are the foundation of this dissertation. This chapter describes the blockchain and NFTs technologies, the Axie Infinity main aspects, animal husbandry, portfolio management, and the metaheuristic basis with the ALNS definition.

Chapter 3 presents mathematical definitions for the NFT breeding problem. Then, two Optimal Breeding Policy MINLP is defined: Commercial Animal Husbandry, which is more similar to animal husbandry in farms where farmer (breeder) animals' (NFT) are bred to obtain profit. And Active Portfolio Management (APM) where a breeder may buy, breed, and sell NFT creatures to obtain profit. At the end of the chapter, the APM is applied to Axie Infinity.

In order to solve the problem presented in Chapter 3 in a feasible time, Chapter 4 presents the two-stage approach using the ALNS metaheuristic (in the first stage) and Mathematical Programming (MP) (in the second stage) for the Axie Infinity. Some implementation details and modifications in the ALNS framework are explained. Besides, the proposed ALNS operators specifically for Axie Infinity are described. The last stage, where the final solution is obtained from the MINLP formulation, is also exposed in this chapter.

Chapter 5 describes the experiments and results obtained to evaluate the proposed approach. First, the implementation details with the dataset and ground truth are provided. Then, the ALNS (first stage) results are evaluated and discussed. Last, the final solution is evaluated against the ground truth for different model parameters.

Finally, Chapter 6 has the concluding remarks, reviewing this work's key aspects and proposing future work directions.

Chapter 2

Background and Related Work

This chapter provides the necessary background for this dissertation and the fundamental theory upon which this work is based. Besides, some related work is mentioned highlighting similarities and differences with this work. To the best of our knowledge, this work is the first to propose a mathematical formulation and provide an efficient solution to the optimal breeding problem for NFT games. However, the basis of this formulation relies on two well-known problems in academia described in this Chapter: Optimal Breeding Strategy (applied to animals) and Portfolio Management. This chapter also describes the basis of Blockchain and NFT technologies. In addition, an overview of the Axie Infinity is provided in order to provide the context of our application. Finally, the ALNS metaheuristic - which is fundamental to our proposed approach - is also described.

2.1 Blockchain and NFT

From the Blockchain Research Institute: "A blockchain is a distributed software network that functions both as a digital ledger and a mechanism enabling the secure transfer of assets without an intermediary. Just as the internet is a technology that facilitates the digital flow of information, blockchain is a technology that facilitates the digital exchange of units of value. Anything from currencies to land titles to votes can be tokenized, stored, and exchanged on a blockchain network" [14].

The Bitcoin blockchain network introduced this technology in 2009 and applied it to currency. Bitcoin (BTC) is still the most popular and valuable cryptocurrency i.e. the blockchain network native token. However, the Bitcoin network was designed to be solid and stable without many updates to its rules. Ethereum is another popular blockchain network, its cryptocurrency Ether (ETH) is the second most valuable. Unlike Bitcoin, the Ethereum network has been updated frequently, for example, on 15 September 2022 an update in its network cut Ethereum's energy usage by 99%.

The Ethereum network also introduced the concept of a smart contract which is a self-executing program that automates the actions in a contract. To implement a new smart contract in Ethereum one may follow the protocols, for example, ERC-20 is the protocol for normal tokens and ERC-721 is the protocol for the non-fungible tokens (NFT). The NFT is a "special" token in a blockchain that is unique and indivisible, which allows it to represent any data like collectibles or files. One popular usage of the NFT is in games, where NFTs representing various objects are used by players of the game in different ways. Besides, some games have NFTs that represent virtual creatures that behave similarly to real animals, where it is also possible to breed them.

In this context, the NFT application and technologies are exhaustively studied in the literature. [15] was the first systematic study on the NFT ecosystems, which allows newcomers to understand the fundamentals of this technology and its possible applications. [16] provides a summary of the challenges associated with NFT transaction data. However, to the best of our knowledge, no work has studied the NFT breeding policy although [12] has studied the economic impacts of breedings on its ecosystem, and [13] has analyzed the CryptoKitties breeding data.

2.2 Axie Infinity Application

From the Axie Infinity (AI) definition provided on its website: "Axie Infinity is a virtual world filled with cute, formidable creatures known as Axies. Axies can be battled, bred, collected, and even used to earn resources & collectibles that can be traded on an open marketplace. Axie was designed to introduce the world to an exciting new technology called Blockchain, through a fun, nostalgic, & charming game." [17] Their definition is kind of modest to the ecosystem created by the Vietnamese studio Sky Mavis. Many details about this ecosystem make it attractive, but this section focuses on giving an overview of AI with only the details necessary for this work.

The term "Axie Infinity" is usually used to refer to the popular NFT-based online game that had almost 3 million monthly players by Jan 2022 [18]. This game now is referred to as "Axie Classic" since in April 2022 a new game was launched named "Axie Infinity: Origins". Nowadays there are 6 games listed, 4 from Sky Mavis and 2 from the community. We will focus on the Axie Classic since it was the only game when this work started and data was collected from this game. We will simply refer to the Axie Classic as a "game".

AI started its game on the Ethereum blockchain, but soon Sky Mavis created its side-chain Ronin due to scalability problems. There are 3 main tokens on this chain:

- Axie Infinity Shards (AXS): The governance token, from the AI website "AXS

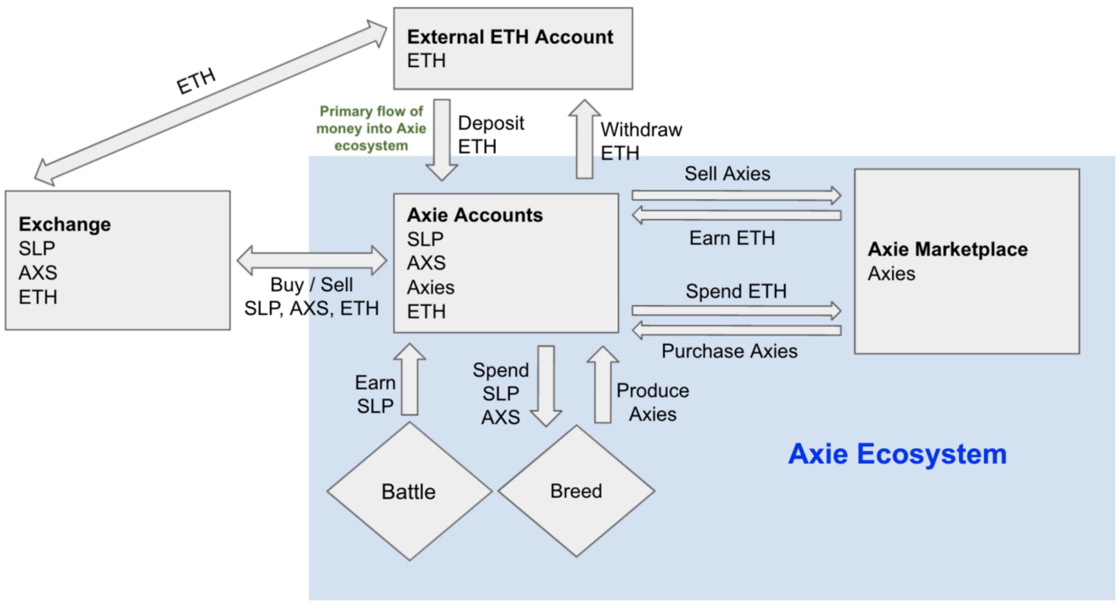


Figure 2.1: A flow chart of the Axie Ecosystem. Source: [1]

holders will be able to claim rewards if they stake their tokens, play the game, and participate in key governance votes. Players will also be able to earn AXS when they play various games within the Axie Infinity Universe and through user-generated content initiatives."

- Smooth Love Potion (SLP): The token necessary to breed the Axies, due to financial reasons is also used inside some of the AI games. This token is the reward when winning battles in the game.

- Ronin (RON): The chain token (currency) used to pay the gas fee when performing transactions on the Ronin blockchain. Note that the RON token was created after Figure 2.1 was made, before its creation the transactions were free.

The central aspect of AI is the Axie, a Non-Fungible Token (NFT) creature that can be collected, used in the games, and bred to generate new Axies. The Axies are traded on a marketplace with a fee (to the game owners) of 4.25% discounted on the selling price. To have a sustainable economy it is necessary to balance the mint and burn of the tokens and Axies. To achieve it, the Axie Economy Ecosystem, illustrated in Figure 2.1, works as follows: The winning players receive SLP (minted) as rewards, which can be used to buy better Axies from breeders; and breeders intending profit breeds good Axies burning (destroying) the SLP.

Like any NFT, an Axie is uniquely defined by its token and smart contract on the blockchain. The Axie NFT stores the following data attributes:

- Id: The Axie's ID number.

- sireId: The Axie's sire ID number.
- matronId: The Axie's matron ID number.
- birthDate: The Axie's birth date.
- genes: The Axie's information, stored in a 512-bit array.
- breedCount: The number of breedings this Axie has made.
- level: The Axie's current level.

The Axie's attributes are straightforward, except for the genes, which need to be decoded. The genes hold the characteristics used by the games and the visual featres. Figure 2.2 shows the block division of the Axie gene structure. The game uses only the class and the parts: eyes, mouth, ears, horn, back, and tail. As a consequence, the Axies prices are mostly determined by these components of the genes. Thus, in this work, we will refer to genes considering only the class and the parts.

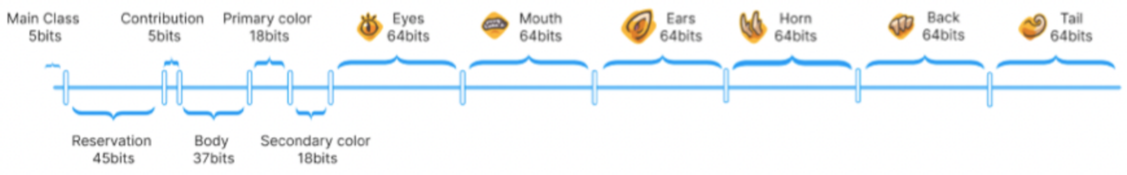


Figure 2.2: The Axie 512-bit gene structure. Source: Sky Mavis Developer Docs [2].

The Axie genetics has dominance, with one dominant gene D and two recessives $R1$ and $R2$. Only the dominant gene appears and affects the gaming performance. However, the recessives will influence the breeding, determining the genetic probability distribution of the offspring. Still, the breeding probability distribution (BPD) for each gene D in the offspring for each parent is given by:

- Receives the parent D gene with a chance of 37.5%.
- Receives the parent $R1$ gene with a chance of 9.375%
- Receives the parent $R2$ gene with a chance of 3.125%

Despite the BPD, there are some rules to perform breeding in AI, otherwise, players would do unlimited breedings. An Axie may breed at most 7 times and incest is forbidden, in the sense that an Axie cannot breed an offspring with its parents or brothers. Besides, there is a fee paid to the game in order to breed composed of a fixed part of 1/2 AXS, and the SLP cost which depends on the

Breed Count	**Breed Number**	**SLP Cost Per Parent**
(0/7)	1	900
(1/7)	2	1350
(2/7)	3	2250
(3/7)	4	3600
(4/7)	5	5850
(5/7)	6	9450
(6/7)	7	15300

Table 2.1: The breeding fee in Axie Infinity. Source: Axie Infinity Whitepaper [3].

number of breeding for each parent as shown in Table 2.1. Note that the SLP cost increases exponentially with the breed count.

Although the BPD is well-known by the community with tools to calculate the offsprings for a given pair, breeders still get confused when evaluating if breeding will be profitable or not. Most breeding tutorials on the internet indicate to consult the probability distribution in order to see if the desired parts have a high chance of outcome. However, this strategy does not consider the combinations that are not the most likely. Imagine, for example, a breeding in which each desired part (to make a profit) is in both D genes of the parents, one $R1$ and both $R2$ having a chance of 90.625% to outcome in the offspring. In this example, the breeder may think that the pair will have a high chance of being profitable. However, the probability of having offspring with the 6 desired parts (and not just one part) is $(90.625\%)^6 = 55.4\%$. Thus, this breeding profitability depends almost a half on the other possibilities, and if those are cheap offsprings this pair may be not profitable. Note that when all genes of the parents are different, this Axie pair may have $6^6 * 2 = 93312$ offspring possibilities.

An Axie has a class and their parts belong to classes. An Axie may have its parts belonging to different classes regardless of Axie class. There are 9 different classes, however, only 6 of them have parts. Table 2.2 shows the number of different parts per Axie class. The combination of the classes and parts results in 8.7 billion possible Axies (calculated as $(36^4 * 24^2) * 9$). Still, today as of January 2024, there is a population of almost 12 million Axies with only 440 thousand available to buy in the marketplace. Therefore, selecting Axies uniformly will almost always result in a unique Axie due to the sparse genetic possibilities within the Axie population.

Class	**Back**	**Ears**	**Eyes**	**Horn**	**Tail**	**Mouth**
Aquatic	6	6	4	6	6	4
Beast	6	6	4	6	6	4
Bird	6	6	4	6	6	4
Bug	6	6	4	6	6	4
Plant	6	6	4	6	6	4
Reptile	6	6	4	6	6	4
Dawn	0	0	0	0	0	0
Dusk	0	0	0	0	0	0
Mech	0	0	0	0	0	0
Total	36	36	24	36	36	24

Table 2.2: The number of parts per Axie class.

2.3 Animal Husbandry

In animal husbandry, the farmer needs to breed its animals to maintain its herd. From bee honey production [6] to milk production in cattle [7], farmers want to improve their profit. Thus, it is important to have a good breeding strategy in order to keep good traits in the offspring such as good health and fertility. Farmers have traditionally been doing selective breeding to increase economically important traits based on phenotypic recordings. A popular selective breeding method is the Selection Index (SI) in which the breeder defines weights for important traits to compute a score for each individual in a population based on the animal's pedigree. Animals with the best scores are then selected to breed.

With the scientific research in this field, the Estimated Breeding Values (EBVs) provides a more robust quantitative approach to predict the traits of an offspring. The EBVs may be calculated in many different ways including the use of modern technologies based on genomic selection [7]. The Best Linear Unbiased Prediction (BLUP)[19] is a popular method to compute the EBVs, and its variations provide support for multi-traits and genomic information. BLUP is a statistical method that relies on a mixed model which separates the observed trait (e.g., milk production in cows) into various effects:

- Fixed effects, which influences all individuals equally.
- Random effects, which is defined for each individual and can be further divided into:
  - Genetic effects, which represent the inherited trait variation due to genes.

    – Residual effects, which represents all other, random, unmeasured factors.

In short, BLUP uses a weighted linear combination of all available information in order to minimize both the error variance and the expected difference between the predicted and true breeding value (unbiased).

An example of the practical application of these theories is the BREEDPLAN software. Its history started in 1968 with the foundation of the Agricultural Business Research Institute (ABRI) from the University of New England (UNE) in Australia. "In 1982 the first set of Estimated Breeding Values (EBVs) were produced by ABRI for the Simmental breed" [20] and later, in 1985, the first version of BREEDPLAN was launched. Nowadays, BREEDPLAN has evolved to incorporate many genotype analyses to compute the EBVs [8].

This work is directly related to animal breeding software, especially with software products like BREEDPLAN [8], ZPLAN [9], and AlphaSim [10] that calculates the Estimated Breeding Values (EBVs) based on the genotypes to optimize the profit within a given animal population. While animal husbandry efforts are mainly on estimating the genome based on the traits record, in our work the genome is explicitly provided by the NFT data. Besides, the real marketplace provides a way of estimating the selling price according to the genes directly. Thus, our work differs by applying breeding optimization to NFTs that require different mathematical models and methods to solve for the optimal breeding strategy.

2.4 Portfolio Management

From Investopedia's definition: "Portfolio management is the art and science of selecting and overseeing a group of investments that meet the long-term financial objectives and risk tolerance of a client, a company, or an institution"[21]. So, portfolio management refers to the choices a manager has to make about asset allocation, diversification, and rebalancing in the stock market. Usually, these choices involve the trade-off between risk and profit (lower expected risk providing lower expected profit). The portfolio management is mainly divided into two types:

- **Passive management:** This case follows an arbitrary market index. In this scenario, "managers buy the same stocks that are listed on the index, using the same weighting that they represent in the index"[21]. This case is less exposed to market variations (risk).

- **Active management:** This case aims to overcome an arbitrary market index. In this scenario, managers need to actively buy and sell stock and assets in order to obtain the largest possible profit. However, this case is more exposed to market variations (risk).

The portfolio management studies include many strategies to achieve profit with many metrics that leverage risk and profit. One popular metric is Value at Risk (VaR), which measures the potential loss within a given probability for a period of time [22]. This metric was exhaustively studied with many variations or propositions to improve it [23–25]. Some recent works also use this metric to perform portfolio management in the blockchain markets [26, 27].

Our proposed optimal breeding strategy model is an active portfolio management with the addition of risk management. However, the VaR metric was not used due to our computation time requirements. Instead, we preferred to constrain the probability of having a profit (e.g., consider breeding if the probability of profit is at least some specified value). Thus, this work is not directly related to portfolio management in its scientific studies and methods, but in general, our objective is to manage a portfolio in the NFT breeding context. The models and objective functions will be described in Chapter 3.

2.5 Adaptive Large Neighborhood Search (ALNS)

The Adaptive Large Neighborhood Search (ALNS) is an extension of the original Large Neighborhood Search (LNS) optimization metaheuristic first proposed by Shaw [28]. The LNS explores a neighborhood of a current point in the optimization space and is classified as a Very Large-Scale Neighborhood Search (VLSN) algorithm [29]. The search in LNS occurs by destroying and repairing the solution candidate to achieve an improved candidate solution. The neighborhood is implicitly defined by the possible solutions obtained after the application of the destroy operator and the repair operator. The destruction is commonly more random than the repair operator which tends to be more greedy [29].

Originally, the LNS only accepted a solution candidate if it had an improvement from the current candidate, however, later it became popular to use the Metropolis criteria the same way used in Simulated Annealing [30]. Using that criteria, a worst solution can be accepted with a probability of $e^{-(r(x)-r(x'))/T}$. Here $r(x)$ is the current solution value, $r(x')$ is the same for the candidate solution, and $T > 0$ is the current temperature. The temperature is initialized at $T_0 > 0$ and is decreased gradually, for example by performing the update $T_{\text{new}} = \alpha T_{\text{old}}$ at each iteration, where $0 < \alpha < 1$ is a parameter. This allows the algorithm to widely explore the solution space in early iterations (high temperature) and to focus on finding good-quality solutions in the later iterations (low temperatures).

However, that was not enough to give the necessary diversification to achieve better solutions in the variants of the Vehicle Routing Problems, for example [31]. To achieve better performance the ALNS incorporates a set of destruction and repair operators, denoted by $\Omega^- = \{\Omega_1^-, ..., \Omega_{|\Omega^-|}^-\}$ and $\Omega^+ = \{\Omega_1^+, ..., \Omega_{|\Omega^+|}^+\}$ respectively, with an adaptive layer that gives more weight to well-performing operators. Thus, operators are chosen adaptively during the execution of the method.

For each iteration of the ALNS, a reward ψ is given for the selected operators, where larger values of ψ mean better performance for the operators, according to the formula:

$$\psi = \max \begin{cases} \omega_1 & \text{if the new solution is a new global best,} \\ \omega_2 & \text{if the new solution is better than the current one,} \\ \omega_3 & \text{if the new solution is accepted,} \\ \omega_4 & \text{if the new solution is rejected.} \end{cases} \tag{2.1}$$

Note that ω_i are parameters of the method. The weight (ρ) for selecting an operator (s) is updated accordingly with a sensitive factor λ as follows:

$$\rho_s' = \lambda\rho_s + (1-\lambda)\psi, \; 0 < \lambda < 1$$

The selection of the operator in the adaptive mechanism is usually the roulette wheel (random choice of operators), where the probability of selecting an operator s, ϕ_s is defined as

$$\begin{cases} \phi_s^- = \frac{\rho_s}{\sum_{k=1}^{|\Omega^-|} \rho_k} & \text{for the destruction operators,} \\ \phi_s^+ = \frac{\rho_s}{\sum_{k=1}^{|\Omega^+|} \rho_k} & \text{for the repair operators.} \end{cases} \tag{2.2}$$

Note that the method adaptively selects both kinds of operators according to the improvements observed when using the operators. Thus, operators that frequently provide improvements are likely to be chosen more frequently.

Chapter 3

Mathematical Modeling

This chapter presents the mathematical definitions involved in the NFT breeding problem. Consider two creatures η_a and η_b that generate a new one η_c. Those creatures have, respectively, a set of attributes (genes) a, b and c. The creatures usually have a breed counting number as a meta-attribute. A creature with genes a that made 2 breedings during its lifetime, as an example, will be referred to as $\eta_{a,2}$ when necessary.

The genes set G, is a subset of the set of attributes that influence the breeding. Some of them may also influence the gaming performance and selling price. The creature (instance) may have other kind of attributes like ID, birth date, breed counting, belonging to a special edition, etc. In general, the set of attributes A of a creature influences the gaming skills and selling price, however, only the genes a and b influence the genes of the offspring, c.

Every creature is unique, which is a basic principle in an NFT. However, the genes from two different creatures may be identical or similar, and the same may happen to the gaming skills. So, let's define the selling price as s_a, where a is the genes of a creature. The price estimation s_a is a tough task, and there are many different ways to estimate this value as discussed in subsection 3.0.1.

Before breeding can occur, it is necessary to check if the pair is breedable. In some contexts there may exist gender restrictions, or any other rule like forbidding incest or limiting the number of breeds of a creature. To verify if a pair is breedable let's define the function $b_v(\eta_a, \eta_b)$ that returns a binary value with 1 indicating that is breedable and 0 otherwise. In addition, let's consider L_{η_a} as the limit number of breedings that η_a can make.

Note that a breeding between a breedable pair η_a, η_b generates a random creature η_c. Thus, taking the genes, let's define a breeding set-valued function $c = B(a, b)$ that generates a random set of genes c based on the genes of a and b. The discrete probability distribution of the B function has many possible offspring $\eta_c^{a,b}$ each one with probability $p_c^{a,b}$. Note that the number of possible genes c (number of different offsprings) depends on the genes a and b.

The costs associated with breeding are usually associated only with the breeding fee and not the genes of the creatures, which is deterministic and paid on the breeding act. However, there are other costs associated when breeding the creatures that are listed below, in function of the number of breeding q:

- Breeding fee: The fee paid to perform the breeding at the moment you select a pair to breed depends on the η_a and η_b. This cost is known and documented in the project's whitepaper. Usually, it depends only on the breed counting number. Denoted by $C_b(\eta_a, \eta_b, q)$ function.
- Parent devaluation: The breeding may change some meta-attributes from the parents η_a and η_b besides increasing their breeding counting number. The virgin creatures are usually more expensive. So, let's denote $C_d(\eta_a, \eta_b, q)$ by:

  $$C_d(\eta_a, \eta_b, q) = s_{\eta_{a,\alpha}} - s_{\eta_{a,\alpha+q}} + s_{\eta_{b,\beta}} - s_{\eta_{b,\beta+q}} \geq 0$$

  where α and β represents, respectively, the breeding count of η_a and η_b before the breeding.
- Operational costs: The transactions in a blockchain have costs, namely gas fees. Besides, there are some marketplace fees when selling. In addition, in some countries, one may have to pay taxes to the government when selling as well. Let $C_o(\eta_a, \eta_b, q)$ function denote the operational (fixed) cost of breeding.

Note that only the Breeding Fee is an expense and others we will consider that only appear discounting the return when selling. The total cost of the breed between η_a and η_b is $C(\eta_a, \eta_b, q) = C_b(\eta_a, \eta_b, q) + C_d(\eta_a, \eta_b, q) + C_o(\eta_a, \eta_b, q)$. However, for simplicity, we assume that for each cost is possible to dissociate the pair cost to each individual, having $C(\eta_a, \eta_b, q) = C(\eta_a, q) + C(\eta_b, q)$. The number of breeding q is necessary due to the nonlinearities present in this cost. The offsprings may take time to mature into adults (Δt), before that they may not be able to breed, play in the game, or even have their attributes defined.

Thus, it is possible to decide if one breeding is profitable based on the expected value and variance of the offspring selling price defined as:

$$\mathrm{E}[s_{B(a,b)}] = \mu_{a,b} = \sum_{c\ \in B(a,b)} p_c^{a,b}\ s_c$$

$$\mathrm{Var}[s_{B(a,b)}] = \sigma_{a,b}^2 = \sum_{c\ \in B(a,b)} p_c^{a,b}\ (s_c - \mu_{a,b})^2$$

If the expected value of the offspring selling price is greater than the breeding cost, is expected to be profitable (on average), else, it is not. However, one may also determine the probability of having profit which is:

$$\mathrm{P}[q\ s_{B(a,b)} > C_(\eta_a, \eta_b, q)] = \sum_{c\ \in B(a,b)} p_c^{a,b}\ I(q\ s_c > C(\eta_a, \eta_b, q)$$

Thus, we can now say a pair is profitable only if $\mathrm{P}[q\ s_{B(a,b)} > C_(\eta_a, \eta_b, q)]$ is large enough, say greater than 1/2. This is important since the skewness in the distribution of offsprings may produce high $\mu_{a,b}$ with a low probability of having profit. Thus, we can define the profit when doing q breedings between η_a and η_b as:

$$q\ \mu_{a,b} - C(\eta_a, q) - C(\eta_b, q)$$

3.0.1 Pricing Estimation Function (PEF)

The profit from breeding depends on the selling price of the creatures involved, as stated previously. However, it is a tough task to define the selling price of a creature η_a or estimate this price with respect to a creature with genes a (s_a). This difficulty comes from the auctions made in the marketplace, where prices are very noisy.

A seller wanting to sell one creature fast may announce it for a low price with the buyer reselling it later at a higher price, making a profit. This turnaround also introduces the liquidity of a NFT, that is the time it takes to sell an asset. In addition, we may have two similar creatures (with the same genes) η_a and η'_a with different orders of magnitude in their auction prices. Thus, it is necessary to have a price estimation function to handle these noisy selling prices and provide more accurate selling price estimates.

A simple strategy to estimate the selling price is to consider the average or median between the last X sold and the top Y cheapest with genes a. However, this strategy is not enough to estimate the price if the genes a are not present in an auction in the marketplace or if there are few creatures with genes a with very noisy prices. In that case, an algorithm should provide this price estimation and may simply assume the price of the cheapest creature, use some notion of similarity to perform averages, or even a complex approach using a Deep Neural Network.

The selling price is also noisy in time, however, this time-varying behavior was ignored in this model. Thus, the price is fixed for a snapshot from the marketplace. Regardless of the price estimating function, the selling price of an individual in the marketplace η_a is considered s_{η_a} with $s_{\eta_a} \neq s_a$ because the estimation s_a will consider all the market.

Adopting a specific price estimation function is enough for our formulation, despite the fact that different pricing estimates may lead to a different breeding policy. Since the pricing estimation is not the focus of this work, simpler strategies will be adopted. More details about the pricing estimation function used in this work will be presented in Section 4.1.

3.1 Optimal Breeding Strategy Problem

This section introduces the mathematical formulations for the Optimal Breeding Policy as a Mixed-Integer Nonlinear Problem (MINLP). Two kinds of problems will be formulated: Commercial Animal Husbandry is closer to the reality practiced by farms and is more intuitive to most people, and Active Portfolio Management considers the creatures more like assets to invest (buy) perform breeding operations, and sell. The Commercial Animal Husbandry will be exposed first in Subsection 3.1.1 and the Active Portfolio Management - the focus of this work - in Subsection 3.1.2.

The list below summarizes the notation used in the formulations presented in this section that are better explained along the text:

- η_a or η_b: Axie that has a set of genes a ou b.
- q_{η_a}: How many times the Axie η_a was chosen to breed.
- H: The Axie set from the Marketplace.
- x: Decision variable which represents a chosen pair.
- y: Decision variable which represents an Axie bought from the marketplace.
- L_{η_a}: The number of times the Axie η_a can still breed.
- $b_v(\eta_a, \eta_b)$: Indicates if the breeding between η_a e η_b is allowed.
- s_{η_a}: The buying price of Axie η_a.
- $\mu_{a,b}$: The expected selling price of an offspring from η_a e η_b.
- $s_{a,+q_{\eta_a}}$: The estimated selling price Axie η_a after breeding q times.
- L_c: The capital limit, budget.
- $C(\eta_a, q_{\eta_a})$: The cost of q_{η_a} breedings with the Axie η_a.

3.1.1 Commercial Animal Husbandry

In this scenario, a person has a fixed set of creatures (H) and would like to breed them in order to gain some profit. Thus, one must determine the set of pairs to breed in order to yield the maximum profit (z). This problem is closely related to commercial breeding, in which the farmer wants to maximize his or her profit by following a breeding policy with the animals available to him. First, we will consider the case in which it is possible to breed only the initial set, as presented in Equations 3.1 to 3.4. Then, we will show a formulation that takes sequential decisions in breeding rounds, making it possible to breed an offspring after maturing it, presented in Equations 3.5 to 3.9.

Short-term objective

The non-negative integer decision variable x determines the number of times that a given pair is used for breeding (0 indicating the pair is not used in breeding). However, a pair can only be selected if it is breedable (Equation 3.2). The objective function (Equation 3.1) is the total profit having as income the expected offspring and parent estimated prices with the breeding fee as an expense. The restriction on Equation 3.4 denotes the limitation in the capital available (L_c) to pay for the breeding fees. Besides, each creature η_a can breed at most L_{η_a} times as expressed in Equation 3.3 having q_{η_a} as the number of times η_a was selected for breeding. Note that, if $q_{\eta_a} = 0$ the associated breed fee will be zero $C_b(\eta_a, q_{\eta_a}) = 0$.

$$\max_{x \in \mathbb{N}^{\binom{|H|}{2}}} \sum_{\eta_a, \eta_b \in H} x_{\eta_a,\eta_b} \; \mu_{a,b} \; + \sum_{\eta_a \in H} s_{a,+q_{\eta_a}} \; - C_\Sigma \tag{3.1}$$

$$\text{s.t.} \qquad x_{\eta_a,\eta_b} \leq L_{\eta_a} \; b_v(\eta_a, \eta_b) \; \forall \; \eta_a, \eta_b \in H \tag{3.2}$$

$$q_{\eta_a} = \sum_{\eta_b \in H} x_{\eta_a,\eta_b} \leq L_{\eta_a} \; \forall \; \eta_a \in H \tag{3.3}$$

$$C_\Sigma = \sum_{\eta_a \in H} C_b(\eta_a, q_{\eta_a}) \leq L_c \tag{3.4}$$

Note that this formulation assumes the person has a fixed budget L_c available to pay for the breeding fees. This will in turn determine which pairs will be selected for breeding and consequently determine the maximum profit.

Long-term objective

The long-term objective differs by having as its objective the maximum profit in a future time $t = T$. To achieve this, we consider a fixed number T of breeding rounds that generate the offspring that are added to the H set in the next round (described in Equation 3.6). The previous, short-term problem, is the particular case of $T = 1$. On the other hand, for $T > 1$ the choice of which to breed in $t = 1$ may be different from the short-term problem due to restrictions in the capital and the breeding rules. However, the formulation is very similar, by including a dynamic set of available creatures H_t set of creatures in round t to accommodate the offspring. Similarly, the variable x^t now denotes the pairs (number of times) that are bred in round t.

Note that the number of possible offsprings grows exponentially with the number of rounds since the offsprings of one round can be used for breeding in the next round. Thus, computing the expected value of the offspring selling price becomes intractable when the number of rounds grows.

$$\max_{x^t \in \mathbb{N}^{\binom{|H_t|}{2}}} \sum_{\eta_a \in H_T} s_{a,+q_{\eta_a}} - C_\Sigma \tag{3.5}$$

$$\text{s.t. } H_{t+1} = H_t \cup \{x^t_{\eta_a,\eta_b} \cdot B(a,b) \; \forall \; \eta_a, \eta_b \in H_t\} \tag{3.6}$$

$$\sum_{t=1}^{T} x^t_{\eta_a,\eta_b} \leq L_{\eta_a} \; b_v(\eta_a,\eta_b) \; \forall \; \eta_a, \eta_b \in H_T \tag{3.7}$$

$$q_{\eta_a} = \sum_{\eta_b \in H_T} \sum_{t=1}^{T} x^t_{\eta_a,\eta_b} \leq L_{\eta_a} \; \forall \; \eta_a \in H_T \tag{3.8}$$

$$C_\Sigma = \sum_{\eta_a \in H_T} C_b(\eta_a, q_{\eta_a}) \leq L_c \tag{3.9}$$

Note that in this formulation, the creatures are sold only at the end of the T breeding rounds.

3.1.2 Active Portfolio Management

This problem differs from the previous one due to interactions with the marketplace and the short-term objective. Instead of breeding an initial set of creatures, one may use the capital available (L_c) to buy creatures from the marketplace to breed and then sell all of them back to the marketplace. Thus, the set H is now all the creatures available to buy from the marketplace. This problem is closely related to portfolio management, with the risk being the offspring randomness since we consider a fixed price for the NFTs creatures (assets). The mathematical formulation is presented in Equations 3.10 to 3.15.

This formulation differs from the previous in Equations 3.13 to 3.15. A new binary decision variable y is added to select which creatures to buy from the marketplace. The budget to buy the new creatures is incorporated in the capital limit (Equation 3.13). The objective function is almost the same since it maximizes the profit, but it also considers the income of selling the parents selected for breeding. This approach inherits the parent devaluation in the difference between s_{η_a} and s_a, as discussed in section 3.0.1. Note that, the $s_{a,+q_{\eta_a}}$ is a simplification where given a creature $\eta_{a,\alpha}$, $s_{a,+q_{\eta_a}} = s_{a,\alpha+q_{\eta_a}}$, so it means the estimated price for a creature with genes a considering a creature that bred q times more than η_a.

The Equation 3.14 adds the profitability restriction, but is simplified to ensure that a pair is profitable in its first breeding since it is not possible to define the profitability by the pairs because one creature may belong in a different pair. Instead, the ideal would be to define the profit probability of the overall solution, but that increase in the complexity of both the formulation and the problem is not worth it. Last, Equation 3.15 defines the mechanism to ensure that the creatures in a pair had been bought from the marketplace.

$$\max_{x\in\mathbb{N}^{\binom{|H|}{2}}, y\in\mathbb{B}^{|H|}} \sum_{\eta_a,\eta_b\in H} x_{\eta_a,\eta_b}\ \mu_{a,b} + \sum_{\eta_a\in H} y_{\eta_a} s_{a,+q_{\eta_a}} - \sum_{\eta_a\in H} C(\eta_a, q_{\eta_a}) \tag{3.10}$$

$$\text{s.t.} \quad x_{\eta_a,\eta_b} \leq L_{\eta_a}\ b_v(\eta_a,\eta_b)\ \forall\ \eta_a,\eta_b \in H \tag{3.11}$$

$$q_{\eta_a} = \sum_{\eta_b\in H} x_{\eta_a,\eta_b} \leq L_{\eta_a}\ \forall\ \eta_a \in H \tag{3.12}$$

$$\sum_{\eta_a\in H} C_b(\eta_a, q_{\eta_a}) + \sum_{\eta_a\in H} y_{\eta_a} s_{\eta_a} \leq L_c \tag{3.13}$$

$$P[s_{B(a,b)} > C(\eta_a,1) + C(\eta_b,1)] \geq 0.5\ \forall\ \eta_a,\eta_b \in H \tag{3.14}$$

$$\operatorname{sgn}(x_{\eta_a,\eta_b}) \leq \frac{y_{\eta_a}+y_{\eta_b}}{2}\ \forall\ \eta_a,\eta_b \in H \tag{3.15}$$

The optimal policy is immediate, the breeder may get the solution and then buy the new creatures and breed them. As an active strategy, it is necessary to sell some creatures to acquire the budget to repeat the process iteratively. The advantage of this method compared to the long term is the possibility of larger profit and less exposure to time-varying prices since the creatures are bought and sold constantly. On the other hand, this strategy is exposed to the availability of creatures in the marketplace, because during the time to acquire the data and compute the solution some creatures may be sold compromising the optimality of the solution.

Theoretically, the problem above can be defined as a Long-term Active Portfolio Management similar to the extension of Commercial Animal Husbandry. The prices and availability of creatures are subject to change based on market conditions. Thus, it would be necessary to perform prediction on what might be available which can be infeasible as the number of creatures available in the marketplace is large.

3.1.3 Active Portfolio Management applied to Axie Infinity

In order to apply the Active Portfolio Management model (presented in Equations 3.10 to 3.15) to Axie Infinity it necessary to define its parameters. We have:

- The number of breedings an Axie $\eta_{a,Q}$ (that bred Q times) can make, defined as L_{η_a}, is $7 - Q$ since 7 is the absolute breeding limit for all the Axies.
- The breedable function b_v for an Axie, should return 0 if the pair parents intersection is null (not brothers) and if them are not father and son.
- The costs in breeding is divides as:
  - Breeding fee: According to Table 2.1 plus a half AXS.
  - Parent devaluation: From Axie Infinity Marketplace data, was estimated a devaluation of 5% per breeding.
  - Operational costs: 4.25% from the marketplace fee when selling plus the Ronin gas fee, paid in RON, to process transactions. However, the Ronin gas fee was ignored since it is small.
- The PEF used in s and used to calculate the estimated breeding profit (EBP) μ is described further on 4.1.
- The capital (budget) available L_c will depend on the breeder and not on the Axie Infinity.

That parameters are enough to solve the MINLP defined as the Active Portfolio Management. Note that the breeding probability distribution (BPD) is according to the one presented in Section 2.2.

Chapter 4

Proposed ALNS-MP Framework

This Chapter describes an approach to solving the breeding problem formulated in the previous Chapter applied to the Axie Infinity scenario. Starting from the Optimal Breeding Policy Problems presented in Section 3.1, we developed a solution only for the Active Portfolio Management Optimal Breeding problem (the focus of this work).

Since the problem was modeled as a Mixed-Integer Nonlinear Problem (MINLP), a simple approach is to use a commercial solver to find the optimal solution. In fact, this approach would be enough to solve the Commercial Animal Husbandry case (with few Axies). On the other hand, this approach would take too much time to find a solution in the Active Portfolio Management case, making it infeasible. The time limit for a solution to be considered feasible was set as 4 hours because the prices and Axies auction vary constantly in the marketplace.

The difficulty in using a solver comes from the combinatorial explosion when considering all the possible pairs on the computation time in the Estimation of the Breeding Profit (EBP) to each pair. For example, a set with 100000 Axies would lead to almost 5 billion pairs, and the EBP - which takes 4.5ms for each pair - would take about 8 months of computation time.

Therefore, the ALNS (metaheuristic) is an appropriate approach that can avoid the combinatorial explosion. A direct application of the method in the formulation presented in Subsection 3.1.2 must define as a neighborhood a set of Axie pairs and the search might navigate removing and adding pairs to neighborhood sets. However, the main difficulty is not the permutation of the pairs to compose a solution, but that is to find which pairs are profitable avoiding computing the EBP on the pairs not profitable.

Thus, our proposed framework is composed of two stages:

- **Find the profitable pairs:** First, the ALNS executes the search looking for the most profitable pair. All the profitable pairs are saved and ranked.

- **Solve the MINLP (Mathematical Programming) using the most profitable pairs:** Then, the N most profitable pairs (usually, $N = 100$) are given to a solver that will find the optimal solution considering only these pairs.

When looking for profitable pairs, preliminary analysis with a random set of 10 thousand Axies showed that all profitable pairs had the same class. So, instead of searching for profitable pairs in the entire set of Axies, it is more efficient to divide the set and group by class. This also allows us to run multiple problem instances in parallel with each instance running the ALNS for a different group (class).

Figure 4.1 represents the proposed ALNS-MP framework in a flow chart: The set of Axies are separated by classes and fed into the ALNS; Each ALNS runs in a separated process, however, the profitable pairs found during the search are grouped before sending it to the solver; and then, the solver analyses the profitable pairs to give the best solution, which may contain pairs from different classes.

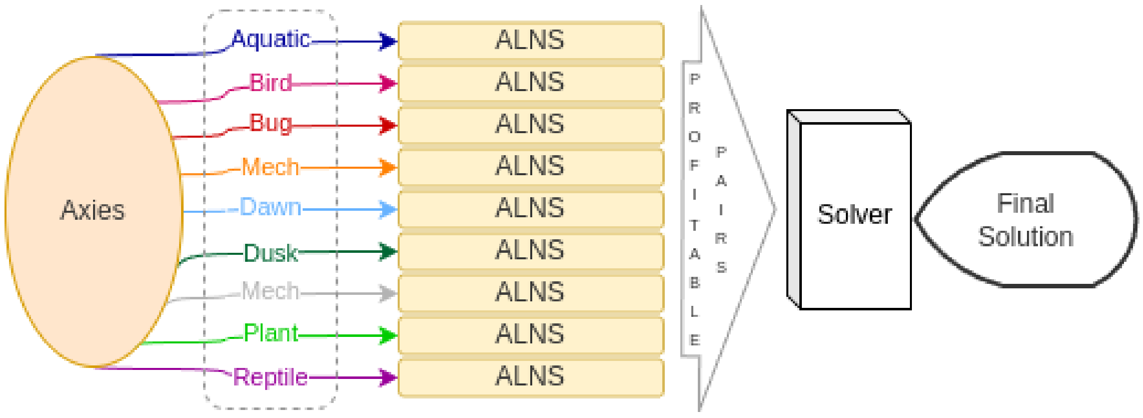


Figure 4.1: A flow chart of the proposed ALNS-MP framework.

Using our approach the EBP is computed only inside the ALNS for a small subset of all possible pairs of Axies. Besides, the combinatorial explosion in the solver by exploring the valid solutions is reduced because the input has only N pairs (the most profitable). So, this approach becomes feasible in the computation time. More details about the EBP, the ALNS, and solver implementations will be described in the next sections.

4.1 Estimation of the breeding profit

The estimation of the breeding profit (EBP) has a central aspect in the search for a solution due to two aspects: First, it will directly influence the objective function when deciding which pair is more profitable. Second, for each candidate pair, the breeding profit must be estimated, and the time spent for that estimation is the major time in the computation time during the ALNS execution.

As defined in Chapter 3 EBP is defined as the mean of the expected selling price of each possible offspring. Besides, for each possible offspring the Pricing Estimation Function (PEF) as defined in Section 3.0.1 must be evaluated, having an impact on the computation time.

Thus, we assumed for a given Axie η_a the PEF is computed as the simplest possible: Take the price of the cheapest available Axie with the same subset of genes. The subset considered is composed of the Tail, Horn, Back, and Mouth; because in the Axie Infinity Classic, they are more important than the Eyes and Ears. In the case where there is no Axie with the same subset of genes available in the marketplace, the price of the cheapest Axie in the marketplace is adopted. The quality/accuracy of the PEF does not affect our proposed approach, but it will affect directly the results and our analysis.

This simple approach for PEF was taken because any other filter would decrease the number of Axies available on the marketplace that would be used to compute an estimate for the price of the offspring. In addition, this filter would require the PEF to define a smarter policy to compensate for the lack of Axies, and this would likely increase computation time.

4.2 ALNS Implementation

As previously stated, the goal of the proposed ALNS is to find the most profitable pair. So, naturally, a solution for the ALNS is a pair with the Destroy Operators removing one of the Axies from the pair and then the Repair Operators adding one Axie to the solution. The initialization was performed by randomly searching for a pair with a profit probability greater than 1/2. The stop criterion for the ALNS was the time limit of 1 hour.

As discussed in Subsection 2.5 the modern ALNS implementations incorporate some of the other metaheuristics. We found it to be useful to include the Tabu Search, to avoid local maxima, with a short-term memory of length 50 pairs. That means the last 50 explored pairs will be excluded from the neighborhood set, allowing the ALNS to explore new candidates. Besides, there are two other modifications made to the ALNS implementation. The first modification was enforcing a minimum probability of 5% when selecting the operators to avoid operator exclusion. The other was a reset method based on the number of pairs found, i.e. the neighborhood during the execution. In our case, if the number of pairs found did not increase for 1500 iterations the ALNS is reset. The reset method sets the temperature and the operators' weights to their initial values and clears the Tabu Table. This modification helped the algorithm explore a broader neighborhood inducing a cycle of intensification with temperature decrease and better operator selection; and diversification with the reset method when no novelty was observed.

The parameterization of the algorithm, according to the definitions presented in Section 4.2, is the following:

- Initial temperature (T_0): 5,000,000,
- Cooling factor (α): 0.995,
- ALNS rewards:
  - The candidate solution is a new global best (ω_1): 5
  - The candidate solution is better than the current solution (ω_2): 2
  - The candidate solution is accepted (ω_3): 1
  - The candidate solution is rejected (ω_4): 0.5
- ALNS sensitive factor (λ): 0.8

4.2.1 Destroy Operators

The proposed approach uses four different destroy operators, with the goal of capturing different insights for each operator.

Random

Removes one Axie from the pair uniformly random. It is the destroy operator that provides the most diversification since it uses no criteria to remove a pair.

Most Expensive Breeding (MEB)

Removes the Axie that costs the most to breed, in our case is the same as removing the Axie that bred the most. The idea behind this operator comes from the Knapsack Problem heuristic of discarding the heavier items to make room for more items, in our case the goal is to save the budget to allow to breed more pairs.

Lowest Partial Profit (LPP)

Removes the Axie that has the lowest partial profit, where the partial profit for a given Axie η_a is $s_a - C(\eta_a, 1)$. The heuristic behind this operator assumes that the offspring sell price will be close to its parents, thus, a pair with a higher partial profit will likely have a higher profit.

Lowest Purity (LP)

Removes the Axie with the lowest purity, where the purity is calculated by evaluating how many recessive genes are equal to the dominant one (D) (for each part) pondered by the probability of the recessive 1 ($R1$) and recessive 2 ($R2$). A pure Axie, with $D = R1 = R2$ for all parts, will have a purity of 1, besides an impure Axie with $D \neq R1 \ \ \& \ \ D \neq R2$ for all the parts will have a purity of 0.75. The heuristic behind this operator tries to avoid mixing Axies with a big variety of genes that will likely result in many "strange" offspring possibilities that tend to have a low selling price.

4.2.2 Repair Operators

The repair operators can be divided into two major groups: One that finds the pair complement based on a gene similarity and the other is based on individual characteristics. The gene-based similarity tends to be less diversifier, however, the individual one that uses a fixed set of Axies tends to be more biased.

Four different gene-based similarity repair operators are considered: Gene Random Keep (GRK), Gene Random Change (GRC), Gene Greedy Keep (GGK), Gene Greedy Change (GGC).

All gene-based similarities use the same similarity function. The Keep and Change operators differ in the Axie used in the similarity function. Keep refers to similarities with the Axie that was not removed after using the destroy operator, while Change refers to similarities with the removed Axie. The difference between Greedy and Random is that the Random operators choose uniformly random within the set of similar pairs found and the Greedy chooses the pair within this set that has the largest profit.

The similarity function defines an Axie pair as similar if they have 3 specific parts in common, the specific part that will be considered is chosen uniformly at random from the set of combinations of the 6 parts chosen three at a time. Thus, even the Greedy operators have some randomness helping to increase the diversification of the operators.

The heuristic behind the Change operators is an attempt to make a new pair closer to the previous one, thus if the current state in ALNS is a good one we would like to keep around good pairs. The Keep operators is based on the heuristic that having a pair where the individuals are similar will decrease the variance of its offspring and possibly increase the probability of having a profitable pair.

Three different repair operators based on individual characteristics are considered, as follows:

Individual Random (IR)

Selects the first Axie that makes the pair profitable (if there exists one) from the shuffled population of Axies. This repair operator provides the most diversification.

Individual Top Partial Profit (ITPP)

Selects the first Axie that makes the pair profitable (if there exists one) from the shuffled list of the Axies with only the 100 most profitable Axies according to their partial profit. The partial profit is calculated as with the LPP operator. The idea behind it is also analogous to the LPP operator, adding an individual with a higher partial profit will likely increase the pair profit.

Individual Top Expensive Breeding (ITEB)

Selects the first Axie that makes the pair profitable (if there exists one) from the shuffled list of the Axies with only the 100 Axies that have the lowest breed cost. The idea behind this operator is analogous to the MEB operator, adding an individual with low cost will save the budget to breed more pairs.

4.3 Obtaining the Final Solution

The second (and final) stage is to use a solver to obtain the solution i.e. the set of Axie pairs to buy and breed from the marketplace. This can be done directly using an off-the-shelf numerical solver. As will be shown in Chapter 5 the ALNS was able to find many profitable pairs, however, using more than 5000 pairs would take the numerical solver too long to find a solution. Hopefully, our results showed that using the top 100 most profitable pairs is enough to have a good solution from the solver in most cases. Otherwise, another metaheuristic would have to be used to select the pairs to be included in the final solution.

Besides, the formulation of this MNILP is slightly different from the one presented in Subsection 3.1.2. The theoretical model considers the input set all the available Axies in the marketplace H, but now the input set is the profitable pairs found by the ALNS. In order to use that model we transformed the top N most profitable pairs into a set of individual Axies (each Axie appears at least once in the top N), this is the set H given as input to the solver. Then, the breedable function b_v used in equation 3.11 is modified to return 0 for pairs that are not profitable. The expected value of the offspring $\mu_{a,b}$ will be used from a table acquired from the ALNS execution, thus, no PEF is calculated during the execution of the solver.

These adaptations would be enough, except that having the decision variable $x \in \mathbb{N}$ increase would make the combinatorial explosion too large to solve in a feasible time. Thus, it was necessary to make a pair breed only once, having $x \in \mathbb{B}$. This modification forces the solution to include more variety of pairs, and consequently, more Axies, although one Axie can still be present in more than one pair, thus being used to breed with different pairs.

Chapter 5

Experiments and Results

This Chapter describes the experiments to evaluate the proposed ALNS approach (PA) and the results obtained. Since the PA is composed of two stages, there are experiments to evaluate each of the stages separately, in Section 5.2 the ALNS is evaluated, and in Section 5.3 the final solution is evaluated. Besides, Section 5.1 describes some technical aspects of the implementation with Subsection 5.1.1 describing the dataset constructed from real data for these experiments.

5.1 Implementation Details

In order to design and develop the technical solution, it was necessary to build an infrastructure with the following softwares:

- A database to store Axie's data once extracted from the real marketplace.
- A script to access Axie's marketplace API and store information in the database.
- The programs to manipulate the Axies and run the ALNS software.
- A MINLP solver and its code interface.

During the development of this work, there was no public library to manipulate the Axies. Thus, it was necessary to design and develop the programs to obtain the data from the marketplace using the API, and then manipulate, transform, and store it in the database. To run the ALNS there are good libraries available, like python-alns [32], however, they tend to give away the efficiency in order to be more general-purpose. Therefore, the ALNS algorithm was fully implemented in the scope of this work with the necessary adaptations as discussed in Section 4.2. The software used:

- Programming Language: Python 3.10

- MINLP Solver: SCIP 7 (wrapped through Pyomo 6.3 in Python)
- Database: MySQL 8

The hardware used:

- CPU: i7 6700K (4 cores, 8M Cache, 4.20 GHz)
- Memory: 32GB (DDR4, 2133MHz)

As discussed before, the computation time is critical to be able to find an adequate solution. Solutions that take days to obtain are not adequate given the price dynamics in the marketplace. So, the implementation was carefully developed profiling the computation time of each function to avoid bottlenecks. The estimation of the breeding profit (EBP), which takes the most time and is called many times, was implemented using Cython which allowed the compilation and fast execution of that part of the code. Thus, the solution implemented was computationally efficient although using Python.

5.1.1 Dataset

The dataset built for this work used real data acquired from the Axie Infinity Marketplace on 2022-03-07 15:37:29. At that time, there was no official documentation about using the Marketplace API and there were many obstacles to download large amounts of data since that API was designed to be used by the marketplace website front-end only and no other purpose. Nowadays there is an open official API to the Axie Marketplace that allows downloading all the data after paying for it. However, it would be necessary to change the current code to this new API and also spend some money to obtain the data. Besides, the Axie Infinity has been changing a lot and has faced a devaluation in the last two years. Thus, we use only the dataset downloaded on 2022-03-07.

The dataset is composed of the active auctions available in the marketplace. In theory, the dataset should have all the Axies available to buy at that time, however, we were able to obtain information on 65091 Axies from a total of 300000 (as estimation) present in the marketplace. Table 5.1 shows the number of Axies per class with its combination of possible pairs.

Class	**Number of Axies**	**Possible pairs**
Aquatic	25653	329M
Beast	0	0
Bird	40	780
Bug	8910	40M
Dawn	683	233K
Dusk	7566	28M
Mech	7300	27M
Plant	12991	84M
Reptile	1948	2M
Total	65091	510M

Table 5.1: Number of Axies and possible pairs per class in the dataset.

Despite some special Axies from limited editions, no other Axies were filtered out from the dataset. The dataset with the sales ad has many Axies announced with exaggerated prices, for example:

Axie ID	**Sell price ($)**
9675684	25638325
4155	25638325
773332	14538132
7606111	10570260
4081812	10485330
25769	7874274
277772	7309660
991	6539906
2626831	6208652
7012707	5515065

Those Axies would never be purchased at these selling prices in the real world and also by the ALNS. However, if an Axie with an exaggerated price is unique to the subset of genes considered by the PEF, the PEF price will be the exaggerated price. Then, the most profitable pairs would come from Axies that have some of those genes that breeding may result in an Axie with a very large (exaggerated) price. Although for practical situations that may not be good (since such Axies are likely not to be sold at these prices), for this work that helps to evaluate the PA.

5.1.2 Ground Truth

To evaluate the PA it was necessary to build a Ground Truth (GT) for comparison. The GT was built from the dataset considering all possible pairs separated by class according to Table 5.1. It took about 2 weeks to compute the EBP for the GT. With the GT, the sparsity of the search space became clear: there are only a few profitable pairs in comparison with all possible pairs.

The GT dataset should be interpreted as the ground truth for this dataset with the considered PEF under consideration, only. The same dataset with different PEFs would result in a different GT set and vice versa. Since the dataset does not exclude the exaggerated prices the GT does not represent the profit one may obtain from following this breeding strategy in the real world.

5.2 Evaluation of the ALNS Solution

As explained in Chapter 4, the ALNS execution was divided into Axie classes leading to one problem instance for each class. Thus, the results and analysis will be made for each class separately. The ALNS is analyzed during its execution and with the final results.

During the ALNS execution, we would like to know how the algorithm is evolving in finding good candidate pairs, the explored, and how often the operators are being chosen and succeeding. To perform this analysis we consider a separate figure for each ALNS execution (i.e. Axie class) from 5.1 until 5.7. All the plots have as the horizontal axis the ALNS iteration, from the beginning of the execution until the stop criteria are matched. The vertical dotted lines inside the plot mark the iteration where an ALNS reset has occurred. In each figure, we have the following plots:

- Subfigure (a):
  - At left: The most profitable pair found and the profit of the current pair, the vertical axis is the profit value.
  - At right: The number of pairs found (visited), the vertical axis is the number of pairs considered by the ALNS in the neighborhood without duplicates (includes non-profitable pairs).
- Subfigure (b) for the destroy operators and Subfigure (c) for the repair operators:
  - At left: The result of the applied operator (status), where each dot is a different operator.

– At right: The probability of selecting a given operator as computed from the weights associated with each operator. A minimum of 5% is enforced for all operators.

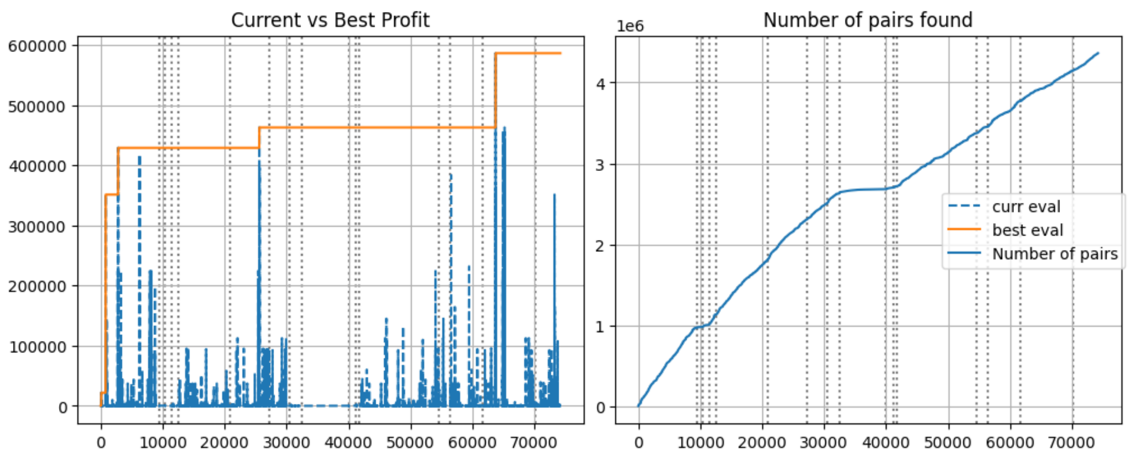
The ALNS was designed to find the most profitable pair, so its evaluation should be on that goal. This analysis was done considering we took the top 10 profitable pairs from the ground truth (GT) to see which of them the ALNS was able to find. We will also analyze the number of profitable pairs and the ground truth found by the ALNS, although the quality (top 10 analysis) is much more important than the quantity. For each class, tables from 5.2 to 5.7 display the top 10 pairs from the GT with the pairs found by the ALNS highlighted in bold. Since the pairs found by the ALNS are a subset of the GT the Jaccard similarity (JS) between these two sets between these two sets will be expressed as a percentage.

5.2.1 Aquatic class results

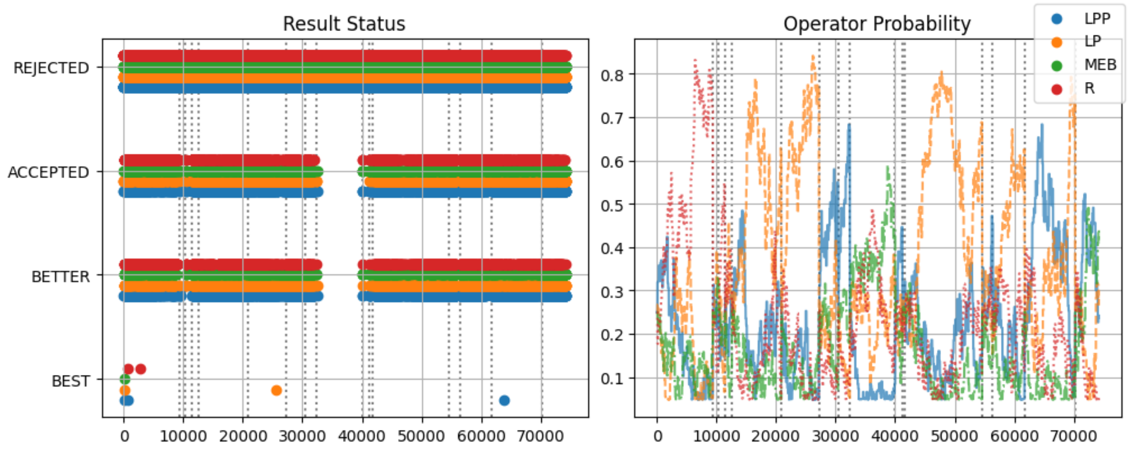
For the Aquatic class, Figure 5.1a shows a fast increase in the most profitable pair at the beginning of the execution. The profits found during the search vary a lot, however, most of them have small values when compared to the most profitable. The ALNS succeeded in exploring well the neighborhood as the number of pairs found grew almost linearly with the number of iterations achieving more than 4 million Axie pairs in 70000 iterations. Concerning the resets, the ALNS did a reset sometimes, but got stuck between iterations 30000 and 40000, and a little around iteration 10000 due to repeated rejections.

The destroy operators' performances in Figure 5.1b didn't show an exceptional operator. Although the Lowest Purity (LP) was the most likely operator for most of the time, the Lowest Partial Profit (LPP) gave the best pair found and the other operators also found profitable pairs sometimes.

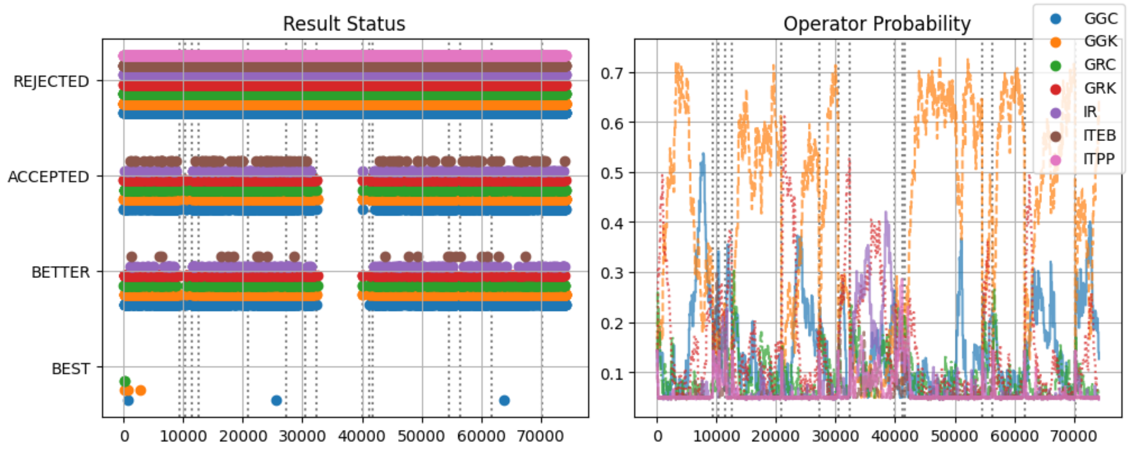
The repair operators' performances during the execution as shown in Figure 5.1c are very different from the performance of the destroy operators. The Individual Top Partial Profit (ITPP) was completely useless because it was never even accepted. The Individual Top Expensive Breeding (ITEB) was a little better than the ITPP but it had difficulty finding a candidate better than the current solution. The gene-based similarities clearly had better performance, with the Gene Greedy Change (GCC) finding the best and the Gene Greedy Keep (GGK) being the more likely operator most of the time.



(a) Solution evolution.



(b) Destroy operator performance.



(c) Repair operator performance.

Figure 5.1: ALNS execution plots for the **Aquatic** class.

The ALNS performance when compared to the GT for the Aquatic class was very good since it found 8 pairs from the top 10 most profitable pairs of the GT, as presented in Table 5.2. Besides, the ALNS was able to find 12914 profitable pairs from the 25011 in the GT, however, ALNS visited around 4.5 million pairs (see

Pair	Profit
8783058, 9162367	**586205.56**
8783058, 9201410	**544024.31**
8783058, 9360641	**512377.56**
8775387, 8783058	**476075.01**
8783058, 9740716	**472234.96**
7228397, 9136614	463872.32
7180703, 8783058	**462779.16**
6647251, 9136614	**456628.75**
8664734, 8783058	**446258.96**
8783058, 9136614	444100.90

Table 5.2: Top 10 results for the **Aquatic** class, JS of 80%.

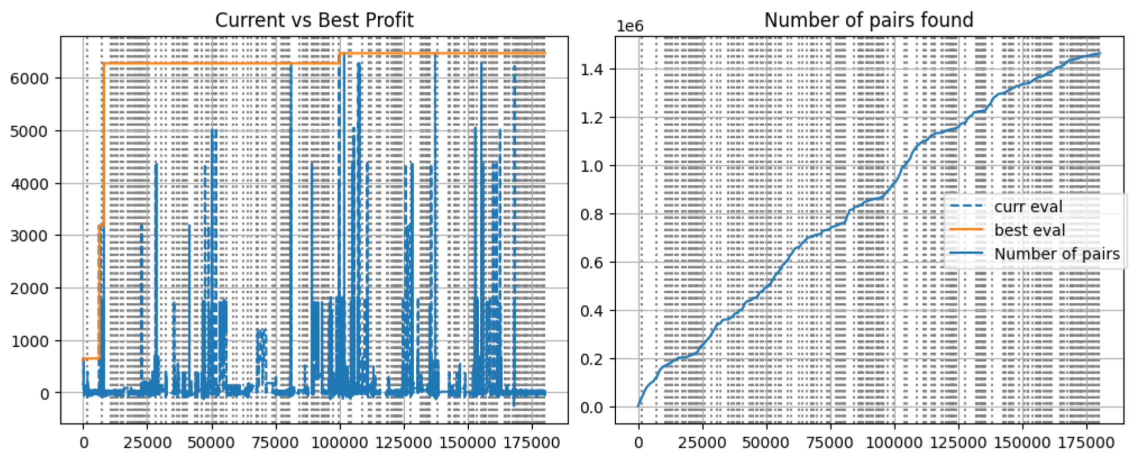
Figure 5.1a) while the GT calculated all the 329 million pairs in this class. Thus, considering only 1.4% of the pairs the proposed ALNS found 52% of the profitable pairs and 80% of the top 10.

5.2.2 Bug class results

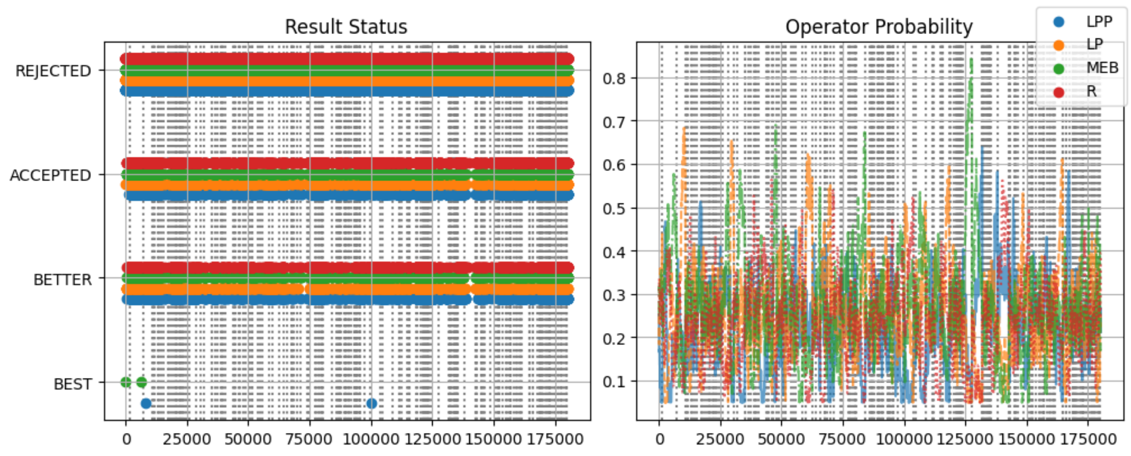
For the Bug class, Figure 5.2a shows a fast increase in the most profitable pair at the beginning of the execution. The profits found during the search vary a lot, however, most of them have small values when compared to the largest profit. The ALNS succeeded in exploring well the neighborhood as the number of pairs found grew almost linearly with the number of iterations achieving more than 1.4 million Axie pairs in 175000 iterations. Concerning the resets, the ALNS resets were very often and the execution did not get stuck by constant rejections.

The destroy operators' performances in Figure 5.2b do not show an exceptional operator. Since the resets were very frequent, the probability distribution across the operators was nearly uniform most of the time. The Lowest Partial Profit (LPP) gave the best pair found and the other operators also found the best pair sometimes.

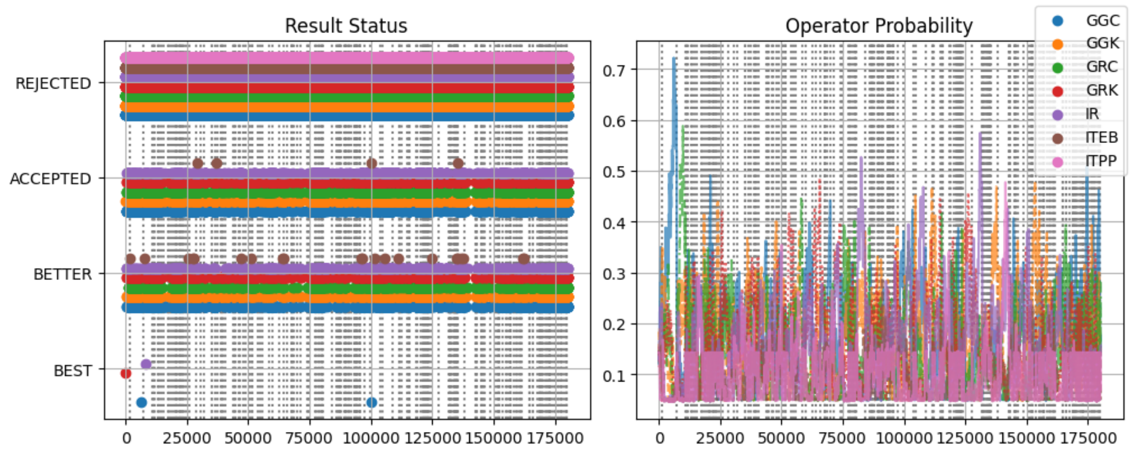
The repair operators' performances during the execution as shown in Figure 5.2c are very different from the destroy performance. The Individual Top Partial Profit (ITPP) was completely useless because it was never even accepted. The Individual Top Expensive Breeding (ITEB) was a little better than the ITPP but had difficulty finding a candidate better than the current solution. The gene-based similarities had better performance, with the Gene Greedy Change (GCC) finding the best. Note that the Individual Random found the top 2 pairs.



(a) Solution evolution.



(b) Destroy operator performance.



(c) Repair operator performance.

Figure 5.2: ALNS execution plots for the **Bug** class.

The ALNS performance when compared to the GT for the Bug class was very good since it found 8 pairs from the top 10 most profitable pairs of the GT, as presented in Table 5.3. Besides, the ALNS was able to find 582 profitable pairs from the 679 in the GT, however, ALNS visited around 1.4 million pairs (see Figure 5.2a)

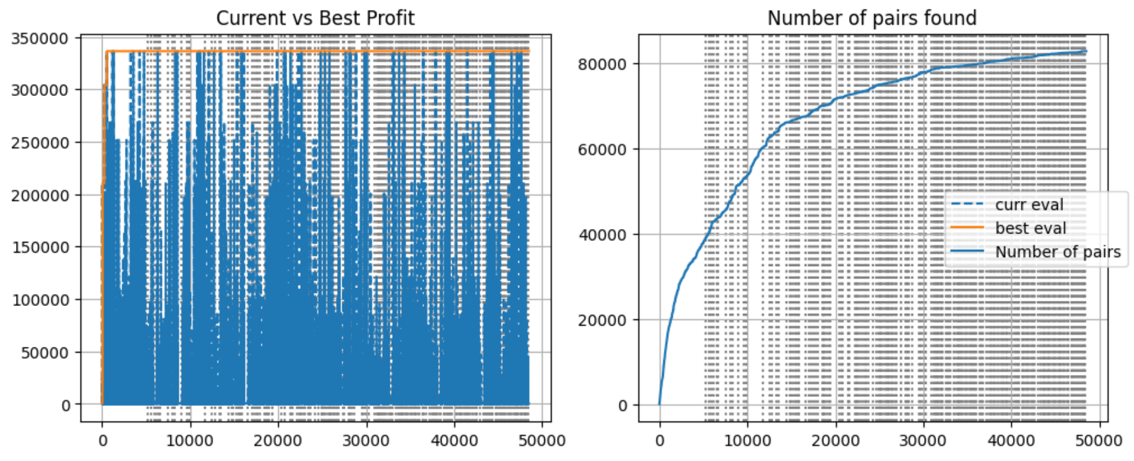
Pair	Profit
9727581, 9804604	13884.54
9723749, 9908727	13005.58
9375804, 9838205	**6464.22**
9324481, 9838205	**6271.40**
9352406, 9838205	**6220.87**
8586864, 9838205	**5043.81**
8695984, 9838205	**5039.69**
9360060, 9838205	**4909.65**
8695851, 9838205	**4513.35**
9838205, 10066000	**4410.35**

Table 5.3: Top 10 results for the **Bug** class, JS of 80%.

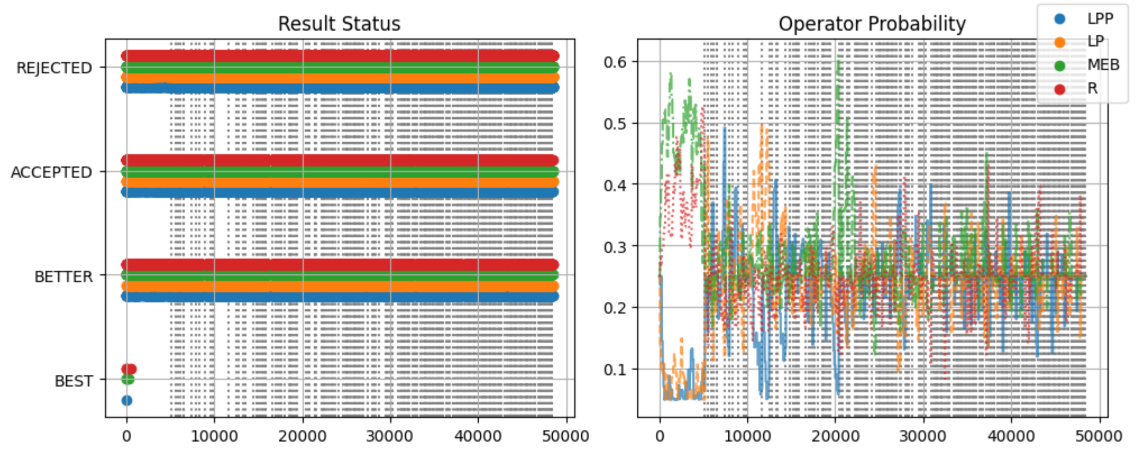
while the GT calculated all the 40 million pairs in this class. Thus, considering only 3.5% of the pairs the proposed ALNS found 86% of the profitable pairs and 80% of the top 10.

5.2.3 Dawn class results

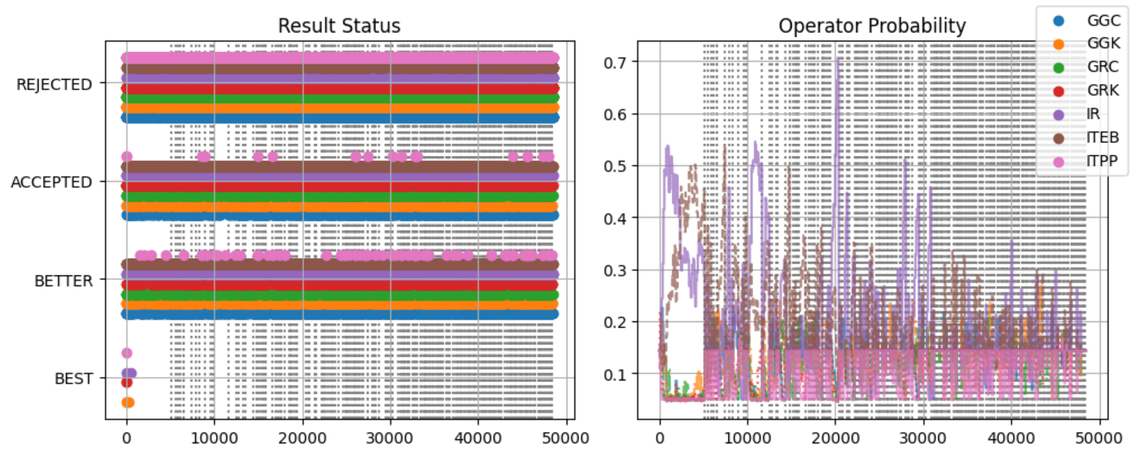
For the Dawn class, Figure 5.3a shows that the most profitable pair is encountered in the first iterations. After that, the algorithm resets very often keeping the search around the most profitable found. Although the number of pairs increases sub-linearly with the number of iterations, the exploration was considered satisfactory for such a small set (683 pairs). Concerning the operators, the resets hindered the analysis, but the ITPP and ITEB were also the worst operators.



(a) Solution evolution.



(b) Destroy operator performance.



(c) Repair operator performance.

Figure 5.3: ALNS execution plots for the **Dawn** class.

The ALNS performance when compared to the GT for the GC for the Dawn class was outstanding finding all the top 10 most profitable pairs of the GT, as presented in Table 5.4. Besides, the ALNS was able to find 457 profitable pairs from the 462 in the GT, however, ALNS visited around 80 thousand pairs (see Figure 5.3a) while

Pair	**Profit**
2962436, 4802390	**336330.68**
2322249, 3020218	**304224.72**
817251, 3662343	**268467.08**
2322249, 4802390	**258769.86**
3020218, 3366355	**251671.89**
2770637, 3020218	**212634.47**
3366355, 4802390	**210288.05**
3020218, 9282053	**200520.85**
4194848, 6440766	**197127.08**
817251, 4802390	**164447.47**

Table 5.4: Top 10 results for the **Dawn** class, JS of 100%.

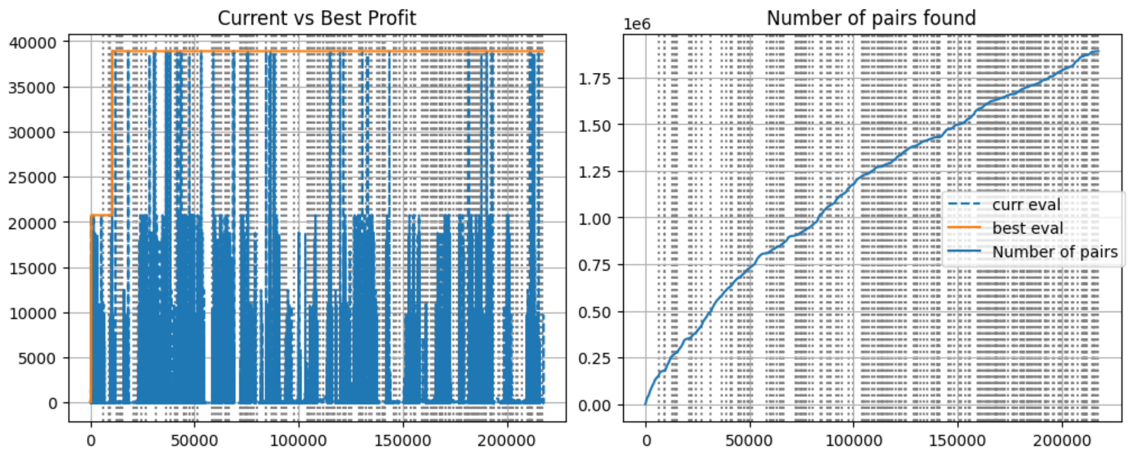
the GT calculated all the 233 thousand pairs in this class. Thus, considering only 34% of the pairs the proposed ALNS found 99% of the profitable pairs and 100% of the top 10. This is a strong indication that the proposed ALNS can be very effective in identifying profitable pairs efficiently (in a relatively small computation time).

5.2.4 Mech class results

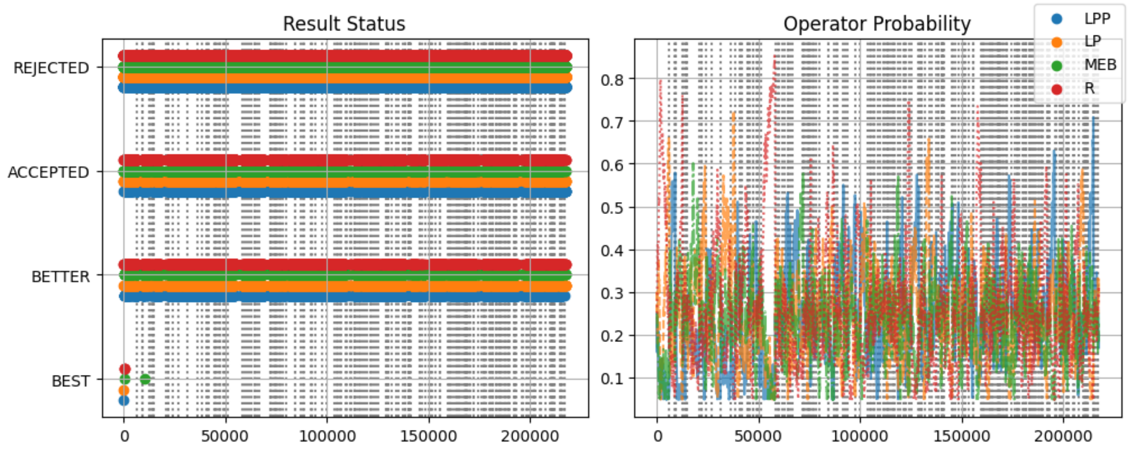
For the Mech class, Figure 5.4a shows that the most profitable pair is encountered in the first iterations. The profits found during the search vary a lot, however, most of them have small values when compared to the most profitable. The ALNS succeeded in exploring well the neighborhood as the number of pairs found grew almost linearly with the number of iterations achieving more than 1.75 million Axie pairs in 200000 iterations. Concerning the resets, the ALNS reset very often but did not get stuck by constant rejections.

The destroy operators' performances in Figure 5.4b didn't show an exceptional operator. Since the resets were very often the probability distribution was nearly uniform most of the time. The Lowest Partial Profit (LPP) gave the best pair found and the other operators also found profitable pairs sometimes.

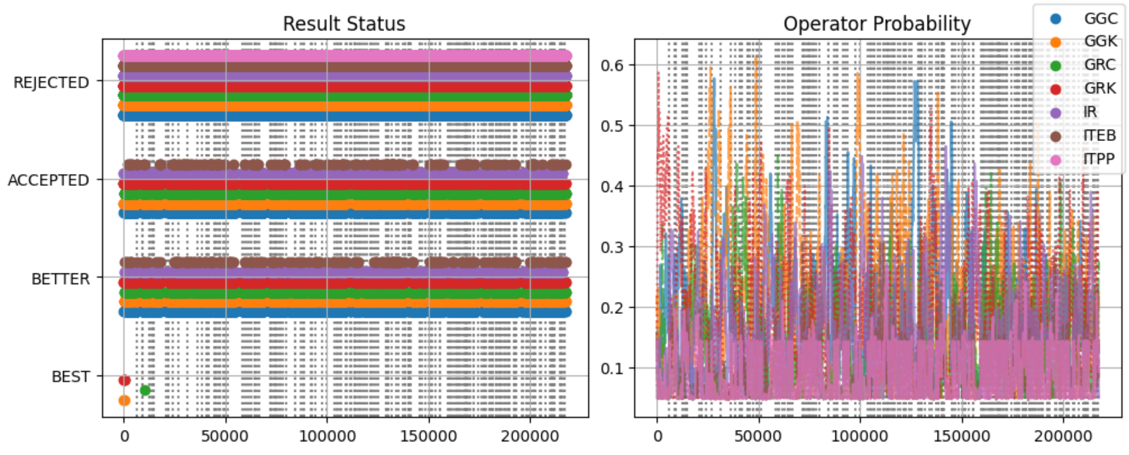
The repair operators' performances during the execution as shown in Figure 5.4c are very different from the performance of the destroy. The Individual Top Partial Profit (ITPP) was completely useless because it was never even accepted. The Individual Top Expensive Breeding (ITEB) was a little better than the ITPP but had difficulty finding a candidate better than the current solution. The gene-based similarities had better performance, with the Gene Random Change (GRC) finding the best.



(a) Solution evolution.



(b) Destroy operator performance.



(c) Repair operator performance.

Figure 5.4: ALNS execution plots for the **Mech** class.

The ALNS performance when compared to the GT for the Mech class was very good since it found 9 pairs from the top 10 most profitable pairs of the GT, as presented in Table 5.5. Besides, the ALNS was able to find 994 profitable pairs from the 1097 in the GT, however, ALNS visited around 1.8 million pairs (see Figure 5.4a)

Pair	Profit
428861, 3996801	**38884.29**
6525748, 8726822	**38498.77**
9772551, 9781149	31075.16
860053, 3203418	**20736.52**
860053, 2962339	**18770.41**
2962339, 9605881	**18039.36**
2962339, 9520456	**16909.69**
1607628, 2962339	**16711.24**
2962339, 3808357	**16042.60**
4424758, 7797142	**12453.30**

Table 5.5: Top 10 results for the **Mech** class, JS of 90%.

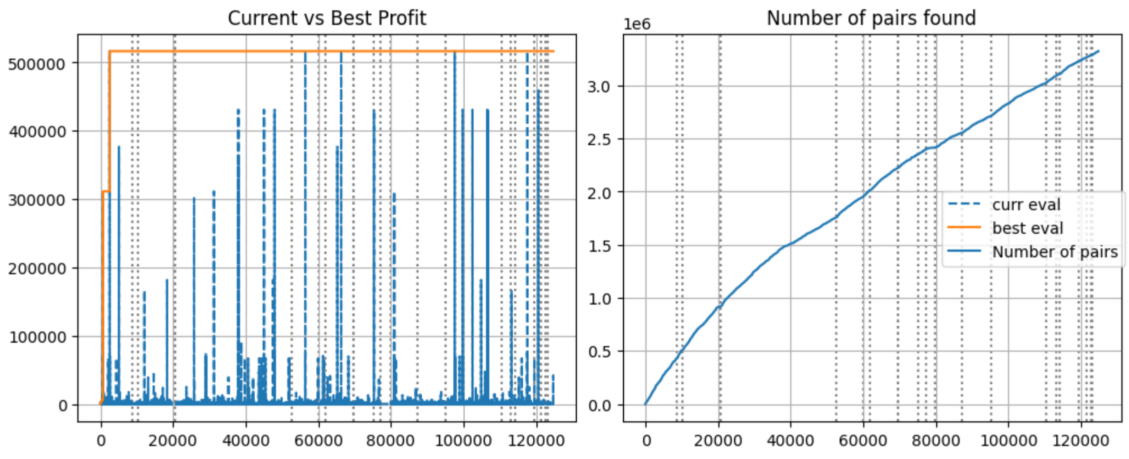
while the GT calculated all the 27 million pairs in this class. Thus, considering only 6.7% of the pairs the proposed ALNS found 91% of the profitable pairs and 90% of the top 10.

5.2.5 Plant class results

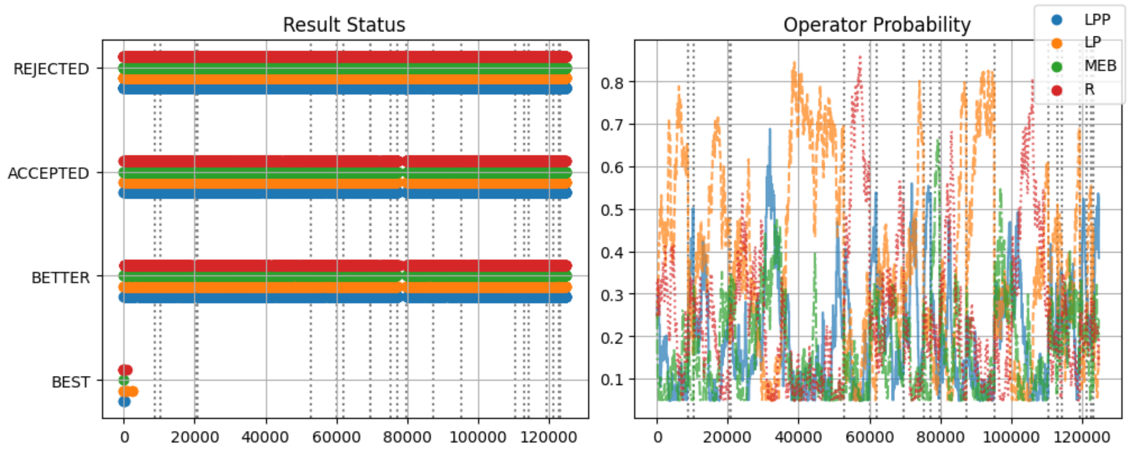
For the Plant class, Figure 5.5a shows that the most profitable pair is encountered in the first iterations. The profits found during the search vary a lot, however, most of them have small values when compared to the most profitable. The ALNS succeeded in exploring well the neighborhood as the number of pairs found grew almost linearly with the number of iterations achieving more than 3 million Axie pairs in 120000 iterations. Concerning the resets, the ALNS did a reset a few times without getting stuck with many rejections.

The destroy operators' performances in Figure 5.5b highlight the Lowest Purity as the best destroy operator insofar as it was the most likely operator for most of the time and gave the best pair. Despite that, all the destroy operators had a good performance.

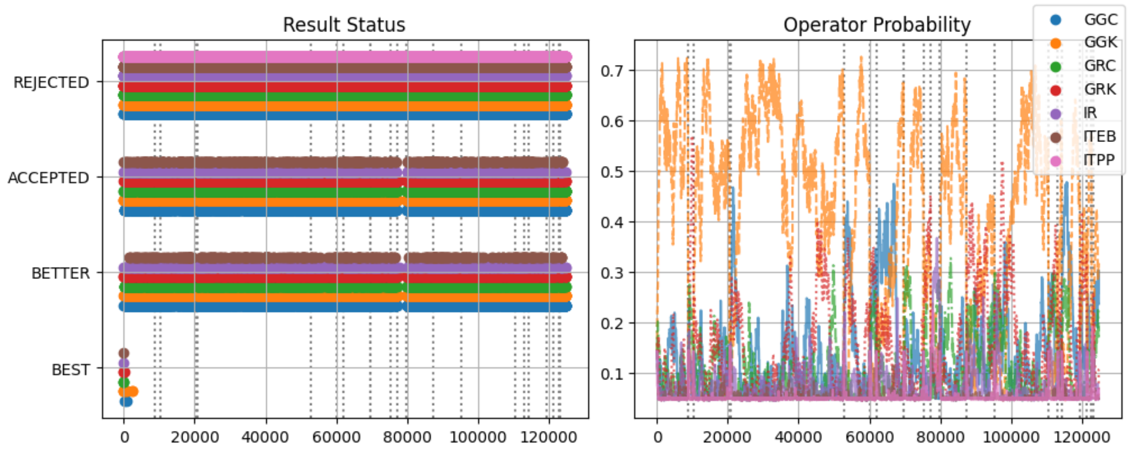
The repair operators' performances during the execution as shown in Figure 5.5c highlight the Gene Greed Keep (GGK) as the best destroy operator insofar as it was the most likely most of the time and gave the best pair. The Individual Top Partial Profit (ITPP) was completely useless because it was never even accepted. The Individual Top Expensive Breeding (ITEB) was a little better than the ITPP but had difficulty finding a candidate better than the current solution. The gene-based similarities clearly had better performance.



(a) Solution evolution.



(b) Destroy operator performance.



(c) Repair operator performance.

Figure 5.5: ALNS execution plots for the **Plant** class.

The ALNS performance when compared to the GT for the Plant class was very good since it found 7 pairs from the top 10 most profitable pairs of the GT, as presented in Table 5.6. Besides, the ALNS was able to find 19004 profitable pairs from the 31394 in the GT, however, ALNS visited around 3.5 million pairs (see

Pair	**Profit**
7760504, 8769136	**516584.89**
7760504, 8650871	467087.25
3246696, 7760504	**459175.02**
8769136, 9235265	**430862.22**
7760504, 8783630	429530.47
496782, 7760504	**403084.20**
7760504, 7790308	**376664.58**
4626134, 8769136	**363132.47**
3827279, 8769136	**361609.89**
3246696, 3827279	334798.25

Table 5.6: Top 10 results for the **Plant** class, JS of 70%.

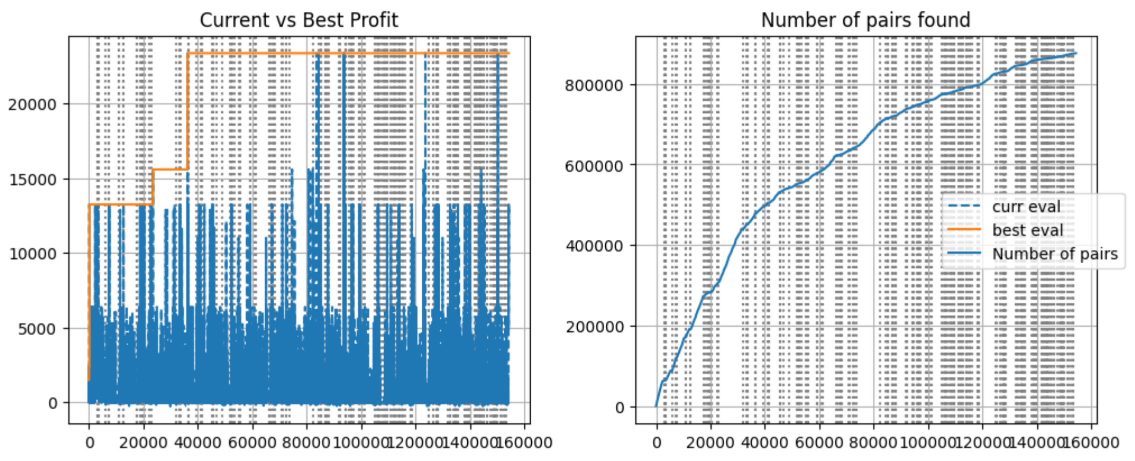
Figure 5.5a) while the GT calculated all the 84 million pairs in this class. Thus, considering only 4.2% of the pairs the proposed ALNS found 60.5% of the profitable pairs and 70% of the top 10.

5.2.6 Reptile class results

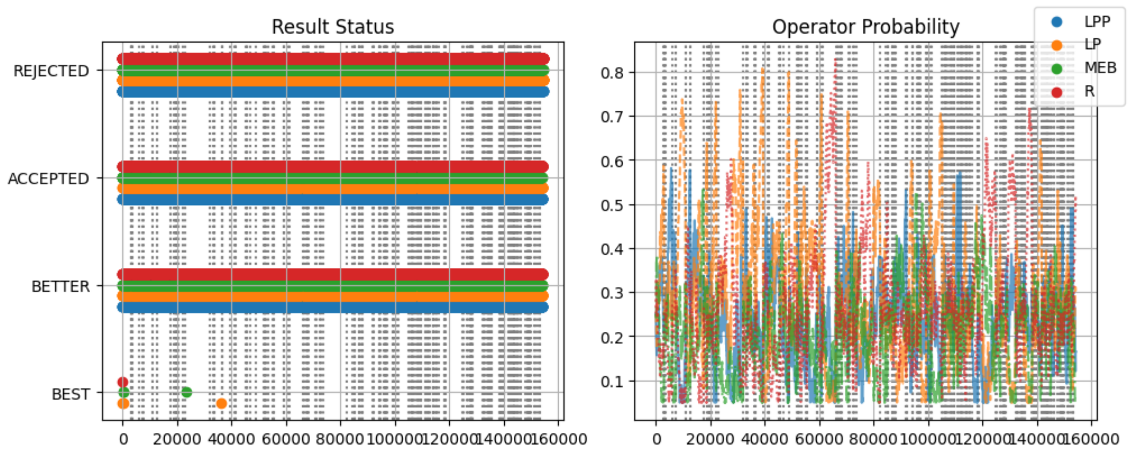
For the Reptile class, Figure 5.6a shows a fast increase in the most profitable pair in the beginning. The ALNS succeeded in exploring well the neighborhood as the number of pairs found grew almost linearly with the number of iterations achieving about 900 thousand Axie pairs in 150000 iterations. Concerning the resets, the ALNS did a reset often but did not get stuck by constant rejections.

The destroy operators' performances in Figure 5.6b didn't show an exceptional operator. Since the resets were very often the probability distribution was nearly uniform most of the time. The Lowest Purity (LP) gave the best pair found and the other operators also found profitable pairs sometimes.

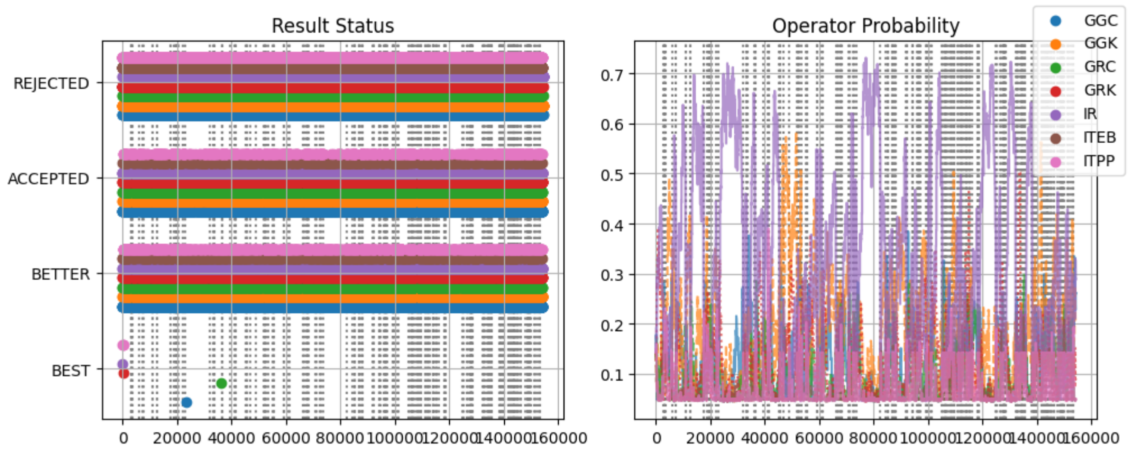
The repair operators' performances during the execution as shown in Figure 5.6c were similar to the destroy performance. The Gene Random Keep (GRK) found the best pair, but the Individual Random was more likely most of the time.



(a) Solution evolution.



(b) Destroy operator performance.



(c) Repair operator performance.

Figure 5.6: ALNS execution plots for the **Reptile** class.

The ALNS performance when compared to the GT for the Reptile class was very good since it found 9 pairs from the top 10 most profitable pairs of the GT, as presented in Table 5.7. Besides, the ALNS was able to find 5685 profitable pairs from the 5728 in the GT, however, ALNS visited around 900 thousand pairs (see

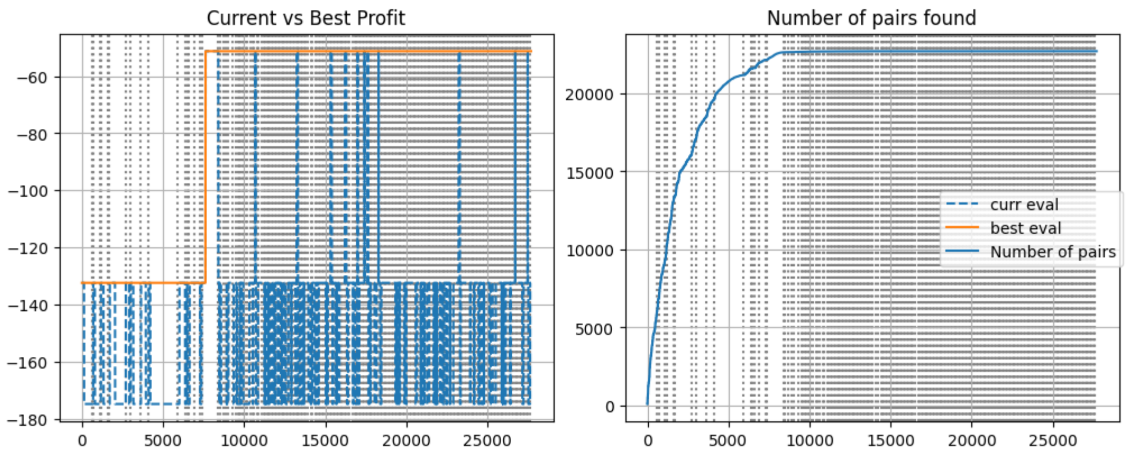
Pair	**Profit**
4964610, 6734380	**23366.46**
4917682, 5601017	**15582.32**
3672157, 6734380	**12962.65**
5601017, 6058786	**12123.75**
3363517, 3363700	11696.85
6734380, 9589589	**11695.00**
6734380, 7501716	**10998.93**
3925285, 9229322	**6371.90**
4648134, 5601017	**6231.76**
4914314, 5601017	**6031.33**

Table 5.7: Top 10 results for the **Reptile** class, JS of 90%.

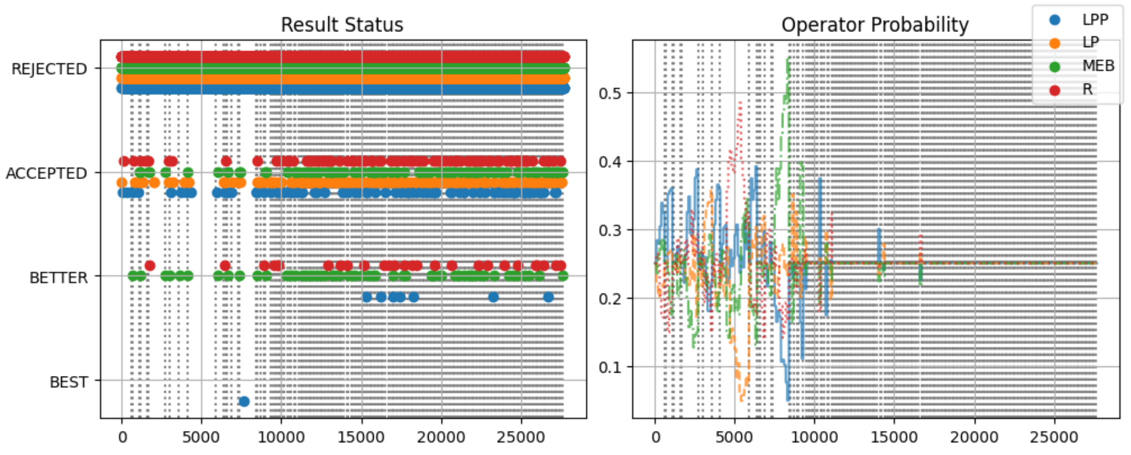
Figure 5.6a) while the GT calculated all the 1.9 million pairs in this class. Thus, considering only 47% of the pairs the proposed ALNS found 99% of the profitable pairs and 90% of the top 10.

5.2.7 Dusk class results

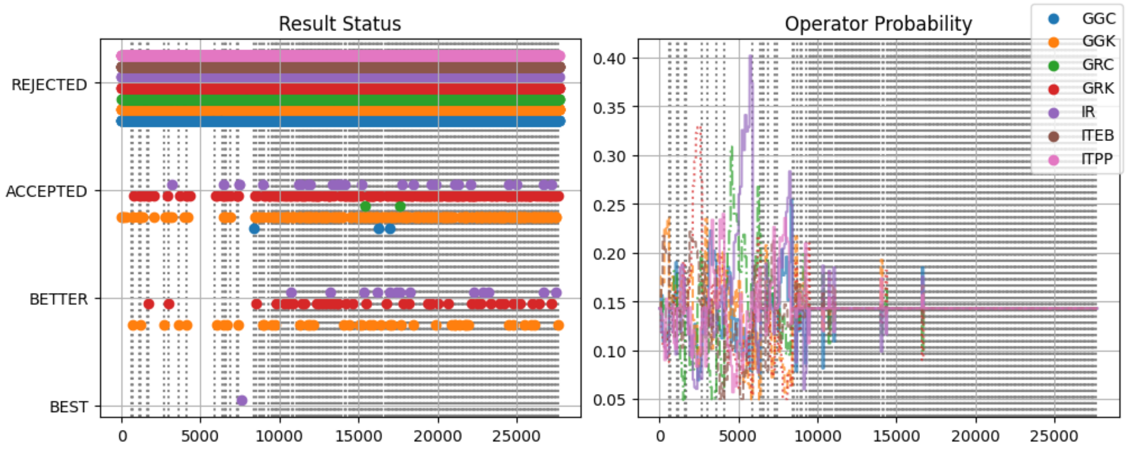
Although the Dusk class had 7566 Axies in the dataset, only 30 pairs were profitable in the GT and the ALNS did not find any of them because it got stuck after some iterations, despite the resets. The search space was very sparse in considering profitable pairs, so ALNS was unable to explore the neighborhood achieving less than 30000 pairs visited. Figure 5.7 shows the execution results, indicating that after around 10000 iterations the method becomes stuck, despite the frequent reset operations.



(a) Solution evolution.



(b) Destroy operator performance.



(c) Repair operator performance.

Figure 5.7: ALNS execution plots for the **Dusk** class.

5.2.8 Final remarks

Remarkably, the proposed ALNS found most of the profitable pairs even though, in general, a small subset of the search space was explored. Besides, the top-10 analysis shows excellent performance since almost all of the best profitable pairs were found by the proposed ALNS.

Between the destruction operators, none of them exhibited a prominent role. On the other hand, between the repair operators, the gene-based showed better results with ITPP being useless and ITEB very poor, indicating that gene-based similarity operators are very important. In the end, the ALNS successfully accomplished the objective of finding the best profitable pairs.

		GT		**ALNS**	
Class	**Number of Axies**	**Visited pairs**	**Profitable pairs**	**Visited pairs**	**Profitable pairs**
Aquatic	25653	329M	25011	4.5M	12914
Beast	0	0	0	0	0
Bird	40	780	0	0	0
Bug	8910	40M	679	1.4M	582
Dawn	683	233K	462	80K	457
Dusk	7566	28M	30	23K	0
Mech	7300	27M	1097	1.8M	994
Plant	12991	84M	31394	3.5M	19004
Reptile	1948	1.9M	5728	900K	5685
Total	65091	510M	64401	19.4M	39636

Table 5.8: Summarized results of the number of profitable pairs found from GT compared to ALNS.

Table 5.8 summarizes the results of the number of profitable pairs discussed in each class section. Looking at the total results, the ALNS was able to find 39636 profitable pairs from the 64401 in the GT, however, ALNS visited around 19.4 million pairs while the GT calculated EBP for all the 510 million possible pairs. Thus, considering only 4% of the pairs the proposed ALNS found 61.5% of the profitable pairs. In addition, the ALNS computing time was 1 hour (since it is the ALNS time limit) while GT spent over two weeks. So, ALNS completely outperforms the "brute force" approach of computing every possibility in GT.

5.3 Evaluation of the final solution

Once a set of profitable pairs has been identified, we need to determine which individuals should be purchased within a budget to obtain the maximum profit, namely, solve the MINLP given in Subsection 3.1.2. There a two important parameters in this problem: N, the number of profitable pairs that should be given as input; B, the budget for purchasing Axies. The solution of the MINLP depends on these parameters and the following evaluation considers different values. The budget value depends on the investor and may be a large, medium, or small value. The N influences the running time of the MILP: larger values will take longer to process with the risk of being infeasible and low values for N would generate a lower profit. To analyze the possible cases the experiments used N with values $100, 1000, 10000$ and budget with values $500, 1000, 2500, 5000, 7500, 10000, 25000, 50000, 100000$. The results are presented in Table 5.9.

Budget ($)	**Top-N (pairs)**	**GT ($)**	**Time (min)**	**ALNS ($)**	**Time (min)**	**DIF (%)**
	100	878299	0.01	878299	0.01	0.00%
500	1000	878299	0.39	878299	0.28	0.00%
	10000	878299	7.48	878299	6.08	0.00%
	100	1858473	0.05	1768412	0.04	4.85%
1000	1000	1858473	1.17	1768412	1.03	4.85%
	10000	1858473	27.35	1768412	23.55	4.85%
	100	4628870	0.19	4302032	0.10	7.06%
2500	1000	4628870	4.97	4302032	2.75	7.06%
	10000	4516948	>240	4302032	168.96	4.76%
	100	8474520	0.62	7922102	0.15	6.52%
5000	1000	8474520	16.37	7922102	3.57	6.52%
	10000	8474520	>240	7922102	>240	6.52%
	100	11736000	0.48	10668540	0.11	9.10%
7500	1000	11736000	14.54	10668540	2.66	9.10%
	10000	11736000	>240	Timed out	>240	
	100	14438661	0.37	12777580	0.12	11.50%
10000	1000	14438661	13.02	12777580	3.40	11.50%
	10000	14438661	>240	Timed out	>240	
	100	23685934	0.01	19022066	0.02	19.69%
25000	1000	24299523	35.81	19022066	23.74	21.72%
	10000	24232097	>240	19022066	>240	21.50%
	100	24554616	0.00	19977287	0.00	18.64%
50000	1000	29528647	>60	23422033	>60	20.68%
	10000	Timed out	>240	Timed out	>240	
	100	24554616	0.00	19977287	0.00	18.64%
100000	1000	34885870	>60	26287393	>60	24.65%
	10000	Timed out	>240	Timed out	>240	

Table 5.9: Profit values for the MINLP using top-N pairs from the GT and from ANLS for different parameters.

The first look on Table 5.9 shows that N does not influence the profit results until the budget becomes 10000. This occurs because the high budget makes a hundred pairs not enough to spend the total budget. However, the $N = 100$ is not the best value to use with small budget values. Let's take as an example the case with budgets 1000 and 2500. Besides, these results depend on the market prices (also on the PEF). In the end, since we tested until top-10000, it is remarkable that 100 pairs from the top are enough to get excellent.

About the computation time, Table 5.9 revealed that the time increases with the budget and number of pairs in different ways. If we consider only $N = 100$ the time will increase with the budget until a budget of 5000 where achieves its maximum time and decreases. For $N = 1000$ it appears that the same would happen, the budget of 5000 is a local maximum, however with a budget of 25000 the computation time achieves the maximum. On the other hand, for higher budgets like 50000 the computation for $N = 100$ was smaller than a budget of 500. This happens because, with a small N and a high budget, the solution includes almost all the pairs. However, when N increases or in a medium budget (like 5000 for $N = 100$) there are more feasible combinations of pairs to consider, thus spending more time. Surprisingly, the time spent to solve to ALNS pairs was smaller than GT, although the number of pairs is the same. This may happen due to the lack of compatible pairs to compose the reuse of the same Axies in different pairs (as we will discuss further). Thus, there would be fewer feasible solutions in ALNS pairs than in GT.

When inspecting the set of pairs and Axies in the solutions of the MINLP, shown in Table 5.10, it is possible to note the reuse of the same Axies to compose many different pairs. It is more profitable to spend more on breeding (by reusing an Axie) than on buying new "virgin" (Axies with 0 breed count) pairs to breed. Note that in all scenarios, more than half the budget is used to buy axies rather than to breed, for both GT and ANLS. Moreover, in all scenarios, ANLS used more of the budget than GT to buy Axies. Besides, most of the Axies in the solutions were "virgin". As an example, for a 10000 budget GT used 39 pairs with just 37 Axies while ALNS used 35 pairs with 32 Axies.

A better look at the percentage difference (labeled as DIF on Table 5.9) shows an increase in the relative difference with the increase in the budget. With a larger budget, more Axies are bought, and the solution increases the number of pairs by adding (buying) a few Axies (due to the reuse of Axies). Thus, the difference between ALNS and GT occurs mainly because the Axies that would be added (bought) to compose pairs with the ones already in solution are missing in the ALNS list.

Budget	GT solution				ALNS solution			
	Number of		Spent to		Number of		Spent to	
	pairs	Axies	breed	buy	pairs	Axies	breed	buy
500	2	3	29.27%	70.47%	2	3	29.27%	70.47%
1000	5	5	41.39%	57.62%	4	5	32.35%	66.12%
2500	12	11	45.55%	54.06%	11	13	37.41%	61.97%
5000	21	21	40.66%	58.69%	20	20	38.70%	61.10%
7500	31	28	40.96%	59.01%	27	26	39.69%	60.16%
10000	39	37	42.60%	57.36%	35	32	40.94%	58.96%
25000	84	72	46.74%	53.17%	77	68	43.90%	56.09%
50000	156	142	45.53%	54.43%	146	129	43.33%	56.66%
100000	306	267	46.39%	53.61%	251	229	38.92%	61.06%

Table 5.10: Overview of the final solutions found using different budget values (for $N = 1000$).

Chapter 6

Conclusion

The optimization of breeding strategies with profit objectives has been applied to animal husbandry for over decades. New technologies like blockchain and NFT created a market where digital assets can be traded to obtain profits. Since some NFT systems allow for breeding, generating new NFTs, breeding started to be performed with the goal of obtaining profit. Although many works are studying animal husbandry breeding strategies and portfolio management applied to blockchain and the NFTs economy, this work is the first (as far as we are concerned) to propose a mathematical model to capture the NFT breeding mechanism in a real application with a profit objective.

In this work, we presented the Axie Infinity breeding mechanism and proposed a MINLP model with a profit objective and risk management. Besides, we proposed a two-stage approach to solve the MINLP using the ALNS metaheuristic and SCIP Solver, since finding the optimal solution directly would take months to conclude. In order to evaluate the proposed approach, real data was collected from the Axie Infinity Marketplace and fed into our designed ALNS. The results and comparison with the ground truth validated our proposed approach, ALNS successfully found the most profitable pairs for breeding and the solver selected the set of Axie to achieve total profit values near the ground truth data (using the best profitable pairs).

The results revealed that, within a couple of hours, we can achieve good results for the Active Portfolio Management problem formulation with a percentual difference of less than 10% for a limited budget ($7500). However, a higher budget increases the percentual difference and the running time in the last stage to achieve the solution.

In this breeding problem, it is desirable to spend more on breeding than on buying new creatures. As a consequence, all the solutions for different budgets have the Axie reusing in different breeding pairs as a common characteristic. Note that reuse implies avoiding buying one more. However, as the breeding cost for

Axies increases exponentially, the results revealed higher expenses in buying than in breeding. In addition, for the ALNS profitable pairs, the expense of buying Axies was higher than for the GT since the lack of some pairs resulted in worse solutions.

It is remarkable the sparsity (in many senses) present in Axies. The Axie gene encodes many more possibilities (about 8.7 billion) than the number of existing Axies (about 12 million). Besides, an Axie breeding may generate a diversity of 90000 thousands possible offsprings. Finally, from the pair combination in the GT only 0.01% was profitable. Thus, finding an approach that navigates through profitable pairs avoiding unnecessary calculations is difficult. In this context, the proposed approach which uses a customized "guided" random choice of pairs (the ALNS paradigm) is seen to be remarkably successful.

The proposed ALNS mechanism designed to find profitable pairs included some essential modifications to achieve good performance like the resets, Tabu Table, and a minimum probability for choosing an operator. These modifications were only possible with our implementation of the ALNS, which achieved good running time performance (searching through millions of pairs within 1 hour). The operators were designed exclusively for the Axie Infinity application, and contributed to the overall success of the ALNS in this scenario. The gene-based similarity repair operators performed much better than the individual top repair operators, but these individual operators did not affect the success of the ALNS since the adaptive layer reduces their probabilities.

6.1 Future work

Although this work presented some good preliminary results, there are many improvements that can be performed in order to achieve real profit in Axie Infinity. First, the dataset should be updated and a new PEF should be designed since the Marketplace has changed significantly. With the official marketplace API available one may obtain recently sold Axies, the new Axie auctions, and data about existing Axies and auctions, even in real-time. So, this allows for a better PEF which will impact the profitable pairs found and the final profit. With updated data and a better PEF, we strongly believe that one following our approach will obtain real and significant profit.

Besides, the overall proposed architecture can also be improved. The actual architecture performs the stages sequentially: Download the data, run ALNS for different classes, and solve the MINLP with ALNS results. This architecture may be improved by running each stage in parallel, and having the stages running online at the same time with a distributed architecture. In this new architecture, multiple processes would download the data and separate it by class while there would be multiple ALNS processes for the same class updating its data while running, generating profitable pairs. Still, the MINLP solution would be also computed in parallel receiving the profitable pairs online. The ALNS multiprocessing implementation has been commonly used in literature with recent work implementing it with GPU processing [33].

Finally, the last stage of our proposed approach, i.e. solving the MINLP given the profitable pairs may be improved with a custom solution. Using the SCIP solver gave us the optimal solution for a subset of the profitable pairs. However, a custom solution would fit better the online distributed architecture by adding the pairs iteratively, while the solver uses a predefined set of pairs. In addition, one limitation of our proposed approach was the excessively long running time in the last stage when solving for a large number of pairs with a high budget. Thus, a custom solution for this last step would be more efficient and overcome this limitation.

References

[1] ALAM, O. "Understanding the economies of blockchain games: An empirical analysis of Axie Infinity". 2022.

[2] SKY MAVIS. "Sky Mavis Developer Docs". https://docs.skymavis.com/, . Accessed:14/01/2024.

[3] SKY MAVIS. "Axie Infinity Whitepaper". https://whitepaper.axieinfinity.com/, . Accessed:14/01/2024.

[4] WIKIPEDIA. "Animal husbandry". https://en.wikipedia.org/wiki/Animal_husbandry#:~:text=Husbandry%20has%20a%20long%20history,were%20being%20raised%20on%20farms. Accessed:14/01/2024.

[5] BEATTIE, A., CLARKE, C. "The History of Money: Bartering to Banknotes to Bitcoin". https://www.investopedia.com/articles/07/roots_of_money.asp. Accessed:14/01/2024.

[6] BERNSTEIN, R., DU, M., HOPPE, A., et al. "Simulation studies to optimize genomic selection in honey bees", Genetics Selection Evolution, v. 53, n. 1, pp. 1–17, 2021.

[7] FLEMING, A., ABDALLA, E. A., MALTECCA, C., et al. "Invited review: Reproductive and genomic technologies to optimize breeding strategies for genetic progress in dairy cattle", Archives Animal Breeding, v. 61, n. 1, pp. 43–57, 2018. doi: 10.5194/aab-61-43-2018. Disponível em: <https://aab.copernicus.org/articles/61/43/2018/>.

[8] ALLEN, J., OTHERS. "An overview of BREEDPLAN", Development Strategies for Genetic Evaluation for Beef Production in Developing Countries, 2002.

[9] TÄUBERT, H., REINHARDT, F., SIMIANER, H. "ZPLAN+, a new software to evaluate and optimize animal breeding programs". In: Proceedings of the 9th world congress on genetics applied to livestock production, v. 1, 2010.

[10] FAUX, A.-M., GORJANC, G., GAYNOR, R. C., et al. “AlphaSim: software for breeding program simulation”, The plant genome, v. 9, n. 3, pp. plantgenome2016–02, 2016.

[11] SCHROEDER, S. “How to play CryptoKitties, the insanely popular crypto game”. https://mashable.com/article/how-to-play-cryptokitties. Accessed:14/01/2024.

[12] MEISTER, B. K., PRICE, H. C. “NFTs: The Game is Afoot”. 2022.

[13] KIREYEV, P., LIN, R. “Infinite but rare: valuation and pricing in marketplaces for blockchain-based nonfungible tokens”, 2021.

[14] INSTITUTE, B. R. “AN INTRO TO BLOCKCHAIN AND NFTS”. https://www.blockchainresearchinstitute.org/an-intro-to-blockchain-and-nfts/. Accessed:14/01/2024.

[15] WANG, Q., LI, R., WANG, Q., et al. “Non-Fungible Token (NFT): Overview, Evaluation, Opportunities and Challenges”. 2021.

[16] JASON B. CHO, S. S., MATTESON, D. S. “Non-Fungible Token Transactions: Data and Challenges”, Data Science in Science, v. 2, n. 1, pp. 2151950, 2023. doi: 10.1080/26941899.2022.2151950. Disponível em: <https://doi.org/10.1080/26941899.2022.2151950>.

[17] SKY MAVIS. “Axie Infinity Website”. https://axieinfinity.com/, . Accessed:14/01/2024.

[18] ACTIVEPLAYER. “Axie Infinity Live Player Count and Statistics”. https://activeplayer.io/axie-infinity/. Accessed:14/01/2024.

[19] HENDERSON, C. R. “Best linear unbiased estimation and prediction under a selection model”, Biometrics, pp. 423–447, 1975.

[20] HISTORY. “BREEDPLAN History”. https://breedplan.une.edu.au/about/history/. Accessed:14/01/2024.

[21] HAYES, A., BROCK, T. “Portfolio Management: Definition, Types, and Strategies”. https://www.investopedia.com/terms/p/portfoliomanagement.asp. Accessed:14/01/2024.

[22] DUFFIE, D., PAN, J. “An overview of value at risk”, Journal of derivatives, v. 4, n. 3, pp. 7–49, 1997.

[23] PFLUG, G. C. “Some remarks on the value-at-risk and the conditional value-at-risk”, Probabilistic constrained optimization: Methodology and applications, pp. 272–281, 2000.

[24] MENG, X., TAYLOR, J. W. “Estimating value-at-risk and expected shortfall using the intraday low and range data”, European Journal of Operational Research, v. 280, n. 1, pp. 191–202, 2020.

[25] SOMA, T., YOSHIDA, Y. “Statistical learning with conditional value at risk”, arXiv preprint arXiv:2002.05826, 2020.

[26] TRUCÍOS, C., TIWARI, A. K., ALQAHTANI, F. “Value-at-risk and expected shortfall in cryptocurrencies’ portfolio: A vine copula–based approach”, Applied Economics, v. 52, n. 24, pp. 2580–2593, 2020.

[27] ZHANG, W., LI, Y., XIONG, X., et al. “Downside risk and the cross-section of cryptocurrency returns”, Journal of Banking & Finance, v. 133, pp. 106246, 2021.

[28] SHAW, P. “Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems”. In: Maher, M., Puget, J.-F. (Eds.), Principles and Practice of Constraint Programming — CP98, pp. 417–431, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg. ISBN: 978-3-540-49481-2.

[29] PISINGER, D., ROPKE, S. “Large Neighborhood Search”. In: Gendreau, M., Potvin, J.-Y. (Eds.), Handbook of Metaheuristics, pp. 399–419, Boston, MA, Springer US, 2010. ISBN: 978-1-4419-1665-5. doi: 10.1007/978-1-4419-1665-5_13. Disponível em: <https://doi.org/10.1007/978-1-4419-1665-5_13>.

[30] WINDRAS MARA, S. T., NORCAHYO, R., JODIAWAN, P., et al. “A survey of adaptive large neighborhood search algorithms and applications”, Computers Operations Research, v. 146, pp. 105903, 2022. ISSN: 0305-0548. doi: https://doi.org/10.1016/j.cor.2022.105903. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0305054822001654>.

[31] PISINGER, D., ROPKE, S. “A general heuristic for vehicle routing problems”, Computers Operations Research, v. 34, n. 8, pp. 2403–2435, 2007. ISSN: 0305-0548. doi: https://doi.org/10.1016/j.cor.2005.09.012. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0305054805003023>.

[32] WOUDA, N. A., LAN, L. "ALNS: a Python implementation of the adaptive large neighbourhood search metaheuristic", Journal of Open Source Software, v. 8, n. 81, pp. 5028, 2023. doi: 10.21105/joss.05028. Disponível em: <`https://doi.org/10.21105/joss.05028`>.

[33] BACH, L., HASLE, G., SCHULZ, C. "Adaptive Large Neighborhood Search on the Graphics Processing Unit", European Journal of Operational Research, v. 275, n. 1, pp. 53–66, 2019. ISSN: 0377-2217. doi: https://doi.org/10.1016/j.ejor.2018.11.035. Disponível em: <`https://www.sciencedirect.com/science/article/pii/S0377221718309767`>.

Appendix A

Algumas Demonstrações