

REINFORCEMENT LEARNING FOR MINING PRODUCTION PLANNING

Taylon Higor Pinheiro Costa

Departamento de Ciência da Computação (DCC), Universidade Federal de Minas Gerais - UFMG
Av. Pres. Antônio Carlos, 6627 - Pampulha, Belo Horizonte - MG
tayloncosta@dcc.ufmg.br

Thiago Henrique Nogueira

Departamento de Engenharia de Produção (DEP), Universidade Federal de Viçosa - UFV
Rodovia BR 230 KM 7, Rio Paranaíba - MG
thiagoh.nogueira@ufv.br

Martín Gómez Ravetti

Departamento de Ciência da Computação (DCC), Universidade Federal de Minas Gerais - UFMG
Av. Pres. Antônio Carlos, 6627 - Pampulha, Belo Horizonte - MG
martin@dcc.ufmg.br

ABSTRACT

Machine learning (ML) and artificial intelligence (AI) significantly impact society on the last years, including, in solving complex Mixed-Integer Linear Programming (MILP) problems. This paper investigates the use of Proximal Policy Optimization (PPO), a reinforcement learning algorithm, for assisting the solution of MILP in mining production planning. Efficient planning is crucial for optimizing resources and ensuring economic feasibility in mining, where traditional methods struggle with large datasets and dynamic conditions. We compare PPO's performance with open-source and commercial solvers using a real-world MILP case study. Our results demonstrate that PPO improves the process of finding feasible and optimal solutions in complex problem instances, highlighting its potential for more efficient and scalable solutions.

KEYWORDS. Reinforcement Learning. Machine Learning. AI-Assisted Optimization. Mixed-Integer Linear Programming. Mining Production Planning.

POI - OR in Industry, IC - Computational Intelligence

1. Introduction

The field of machine learning (ML) and artificial intelligence (AI) has experienced remarkable advancements and widespread usage in recent years, revolutionizing various sectors of our society including healthcare, finance, and transportation. This rapid progress underscores the necessity for ongoing research to uncover innovative applications and methodologies that can harness the full potential of AI. One area of significant interest is the application of AI in solving Mixed-Integer Linear Programming (MILP) problems, which are fundamental in fields such as operations research and optimization. MILP problems are notoriously challenging due to their combinatorial nature and the computational complexity involved in finding optimal solutions.

As Caccetta [2007] demonstrates, careful production planning and management are a crucial factor for the economic feasibility of a mine. The author also highlights the utility of optimization techniques in addressing critical challenges within mining planning, including production sequencing, mine design, and optimal layouts, equipment maintenance, transportation, among others. The Mining operations are complex, involving multiple interdependent activities. Efficiently planning these activities requires solving MILP problems to optimize resource allocation and ensure sustainable practices. The high stakes and intricate nature of production planning in mining demand robust solutions that can handle large datasets and dynamic conditions.

Proximal Policy Optimization (PPO) is an algorithm in the realm of reinforcement learning, maintained by researchers on OpenAI and presented by Schulman et al. [2017], PPO presents an equilibrium between simplicity and performance. The algorithm works by iteratively refining policies to optimize the cumulative reward, ensuring stable and reliable learning. Its design allows for the effective handling of high-dimensional spaces and complex decision-making scenarios, which are essential characteristics when tackling MILP problems.

In the work of Esteso et al. [2023] they show the application of reinforcement learning on a myriad of problems of production planning. Recent research by Zhu et al. [2020] has also shown the efficacy of PPO in constructing solutions to MILP problems compared with traditional methods such as genetic algorithms. These studies have shown that PPO can significantly enhance the process of finding feasible and optimal solutions, particularly in large and complex problem instances. These findings suggest that leveraging PPO in the context of large-scale MILP can lead to more efficient and scalable solutions, thus opening new avenues for practical applications and theoretical advancements.

In light of these developments, this paper aims to explore the potential of Proximal Policy Optimization helping in constructing solutions for complex MILP formulations, utilizing as a case study a real-world-based problem of the production planning of a mining company, however, due to its complexity, for this paper we propose a summarized version of the problem. We compare the results obtained by free open-source and commercial solvers, both before and after assisted by a trained PPO agent. Our goal is to contribute to the ongoing discourse on AI-driven optimization techniques and to propose novel strategies that can further improve the efficiency and effectiveness of solving MILP problems using ML.

2. Problem description

As mentioned, this study addresses a problem based on the real-world operation of the production planning of the Brazilian mining company, with the main objective point of planning the production in an integrated approach. We take into consideration the relations between the many stages of production in order to maximize the production output of the line. Given the substantial scale, intricacies, and complexity inherent to the operations, an abstraction became essential to retain solely its fundamental attributes for analysis. For a comprehensive understanding, we can dissect the scenario into four stages:

- **Mining:** This step includes the extraction of base ore, here, we must take into consideration factors such as the rate at which the ore is extracted and the start and end of base ore campaigns, which are specified as parameters. This base is stored in a buffer pile. In order to guarantee a continuous supply and to maintain the physical integrity of the pile, we are considering the minimum and maximum capacity levels.
- **Concentration** In this step, the stored ore from the buffer pile is pulled into the factory to undergo processing in the concentrator, this pulling rate is calculated utilizing daily values of physical availability, the operational percentage of the concentrator, and its humidity, resulting in the creation of a pulp with specified concentrations of chemical elements, the same base ore can be turned into different types of pulp, we control this via a relation table. It is also important to note that each type of base ore must undergo this process just one at a time. Based on this pulling rate, the total production is determined. This resulting pulp is stored in a tank, in which we also consider minimum and maximum capacity levels.

Besides this, we must measure the volume of sludge and the sandy tailing generated by the factory, all these factors are given as input and derived by the pulling value.

- **Pipeline** The pulp stored in the tank is then transported via pipeline, connecting the concentrator to the storage of the final phase of the production line. The pipeline must adhere to strict rules for the maintenance of its structural integrity. The fundamental rule is that it has to alternate the pumping of pulp with the pumping of water, *i.e.*, after a specific number of pulp batches, the pipeline must switch its pumping to water, and vice versa. Another critical rule is that the size of each batch sequence, whether pulp or water, must fall within a specified time-duration window. Additionally, the batch sequence starts with a type of pulp, it must follow a minimum size sequence before changing the pulp type pumped.
- **Pelletizing Plant** The final phase where it undergoes further processing to attain the desired final product. Considering that different types of pulp can be turned into the same final product, we also have to control this via a relation table. The total volume of production on the pelletizing plant is derived from the amount of pulp fed to it multiplied by a conversion factor given on input. During this phase, we have as well a yard used to store the pulp.

One particularly challenging phase of the process is managing the constant pulp flow throughout the pipelines, which represents the most significant bottleneck in production. Ensuring the efficiency of this phase is crucial to maintain continuous production output and preventing outages or material overload in all storage facilities. The difficulty is imposed by the strict constraints regarding the physical integrity of the pipelines and the rigorous operational regulations. These factors collectively pose substantial challenges in modeling and solving this phase using traditional methods implemented on MILP solvers. To address these complexities, we developed an approach that combines machine-learning techniques to assist traditional operational research approaches thereby enhancing overall production efficiency and reliability.

3. Mathematical formulation

The objective function in 1 aims to maximize the total production at the pelletizing plant.

$$\text{Max } Z = \sum_{c \in C''', dh \in H'} \text{Prod}'_{c,dh} \quad (1)$$

3.1. Crushing constraints

Constraints (2) limit the crushing production rate based on the mine's delivery schedule, setting the rate to zero if not delivered. Constraints (3) and (4) define the concentrator's buffer stock, with initial stock (EP_0) provided as input. Constraints (5) link feed rates in the concentrator to buffer stock from the mine.

$$RO_{c,dh} \leq B_{max_{day_{dh}}}(PM_{c,dh}), \forall c \in C', dh \in H', \quad (2)$$

$$SB_{c,H'_i} = EP_{c,H'_{i-1}} + RO_{c,H'_i} - \sum_{p \in C''} FRC_{c,p,H'_i}, \forall c \in C', i \in \{1, |H'| - 1\}, \quad (3)$$

$$SB_{c,H'_0} = EP_c^0 + RO_{c,H'_0} - \sum_{p \in C''} FRC_{c,p,H'_0}, \forall c \in C', \quad (4)$$

$$RO_{c,dh} = \sum_{p \in C'} FRC_{p,c,dh}, \forall c \in C'', dh \in H', \quad (5)$$

3.2. Concentrator constraints

Constraint (6) ensures only one product is processed per hour. Constraints (7)-(9) manage the feed rate and production relationship in the concentrator using the relation matrix. Constraints (10) calculate the pulled variable Pux_C_{dh} using various inputs, determining production volumes ($Pr_C_{c,dh}$) via equations (11). Parameters Sol and $Dens$ convert production to m^3 using (12).

$$\sum_{c \in C', p \in C''} Prod_C_{c,p,dh} \leq 1, \forall dh \in H', \quad (6)$$

$$Prod_C_{c,p,dh} \leq 1, \forall c \in C', p \in C'', dh \in H' : W_{c,p} = 1, \quad (7)$$

$$Prod_C_{c,p,dh} = 0, \forall c \in C', p \in C'', dh \in H' : W_{c,p} \neq 1, \quad (8)$$

$$FeedRateC_{c,p,dh} \leq M(Prod_C_{c,p,dh}), \forall c \in C', p \in C'', dh \in H', \quad (9)$$

$$PlC_{dh} = \left(\sum_{c \in C''} Rate_C_{c,dh} \right) (DF_{dia_{dh}}) (UF_{dia_{dh}}) (1 - Umid_{dia_{dh}}), \forall dh \in H', \quad (10)$$

$$PC_{c,dh} = (RC_{c,dh})(1 - Umid_{dia_{dh}}) * (RP_{c,dia_{dh}}/100) (DF_{dia_{dh}})(1 - Dif_{dia_{dh}}), \forall c \in C'', dh \in H' \quad (11)$$

$$Vol_C_{c,dh} = Pr_C_{c,dh}(1/Sol)(1/Dens), \forall c \in C'', dh \in H', \quad (12)$$

Equations (13) and (14) calculate sludge (L_C_{dh}) and sandy tailings (R_C_{dh}) generation based on Pux_C_{dh} and $Pr_C_{c,dh}$.

$$L_C_{dh} = Pux_C_{dh}(1 - GSluge)\forall dh \in H', \quad (13)$$

$$R_C_{dh} = Pux_C_{dh} - L_C_{dh} - \sum_{c \in C''} Pr_C_{c,dh}, \forall dh \in H', \quad (14)$$

3.3. Storage tanks constraints

Constraints (15)-(17) specify the concentrator tank's stock limits, with initial and daily flow rates given as inputs.

$$\sum_{c \in C''} E_EB06_{c,dh} \leq Cap_EB06, \forall dh \in H', \quad (15)$$

$$E_EB06_{c,H'_i} = E_EB06_{c,H'_{i-1}} + Vol_C_{c,H'_i} - (BP_EB06_{c,H'_i}) * (V_EB06_{dia_{H'_i}}) \forall c \in C'', i \in \{1, |H'| - 1\}, \quad (16)$$

$$E_EB06_{c,H'_0} = E_EB06^0_c + Vol_C_{c,H'_0} - (BP_EB06_{c,H'_0}) * (V_EB06_{dia_{H'_0}}) + Env_EB04_EB06_{c,0} - Env_EB06_EB04_{c,0} \forall c \in C'', \quad (17)$$

3.4. Pipelines constraints

The variable $BP_EB06_{c,kh}$ tracks pulp pumping each hour, with constraint (18) ensuring only one product is pumped per hour. Constraints (19)-(22) calculate the amount of accumulated pulp pumped. Utilizing parameter BP_{max} to indicate the maximum amount of pulp that can be pumped continuously, the equations (23)-(27) are used to calculate the final number of hours in which pulp pumping occurred.

$$\sum_{c \in C''} BP_EB06_{c,dh} \leq 1, \forall dh \in H', \quad (18)$$

$$Xac_{H'_i} \leq Bac_i + 1 + M(1 - \sum_{c \in C''} BP_EB06_{c,H'_i}), \forall i \in \{0, |H'| - 1\}, \quad (19)$$

$$Xac_{H'_i} \geq Bac_i + 1 - M(1 - \sum_{c \in C''} BP_EB06_{c,H'_i}), \forall i \in \{0, |H'| - 1\}, \quad (20)$$

$$Xac_{H'_i} \leq M(\sum_{c \in C''} BP_EB06_{c,H'_i}), \forall i \in \{0, |H'| - 1\}, \quad (21)$$

$$Xac_{H'_i} \leq BP_{max}, \forall i \in \{0, |H'| - 1\}, \quad (22)$$

$$X_{H'_i} \leq Bac_i + M(1 - \sum_{c \in C''} Bant_{c,i} + \sum_{c \in C''} BP_EB06_{c,H'_i}), \forall i \in \{0, |H'| - 1\}, \quad (23)$$

$$X_{H'_i} \geq Bac_i - M(1 - \sum_{c \in C''} Bant_{c,i} + \sum_{c \in C''} BP_EB06_{c,H'_i}), \forall i \in \{0, |H'| - 1\}, \quad (24)$$

$$X_{H'_i} \leq M(\sum_{c \in C''} Bant_{c,i}), \forall i \in \{0, |H'| - 1\}, \quad (25)$$

$$X_{H'_i} \leq M(1 - \sum_{c \in C''} BP_EB06_{c,H'_i}), \forall i \in \{0, |H'| - 1\}, \quad (26)$$

$$X_{H'_i} \leq BP_{max}, \forall i \in \{0, |H'| - 1\}, \quad (27)$$

Similarly, the equations (28)-(31) refer to water pumping, calculating the accumulated amount of water pumping. With BA_{max} that indicates the maximum amount of water that can be pumped intermittently (32)-(36) calculates the final accumulated quantities of water pumping.

$$X_{ac_A_{H'_i}} \leq B_{ac_A_i} + 1 + M(1 - (1 - \sum_{c \in C''} BP_EB06_{c,H'_i})), \forall i \in \{0, |H'| - 1\}, \quad (28)$$

$$X_{ac_A_{H'_i}} \geq B_{ac_A_i} + 1 - M(1 - (1 - \sum_{c \in C''} BP_EB06_{c,H'_i})), \forall i \in \{0, |H'| - 1\}, \quad (29)$$

$$X_{ac_A_{H'_i}} \leq M(1 - \sum_{c \in C''} BP_EB06_{c,H'_i}), \forall i \in \{0, |H'| - 1\}, \quad (30)$$

$$X_{ac_A_{H'_i}} \leq BA_{max}, \forall i \in \{0, |H'| - 1\}, \quad (31)$$

$$X_{_A_{H'_i}} \leq B_{ac_A_i} + M(1 - (1 - \sum_{c \in C''} B_{ant_{c,i}}) + (1 - \sum_{c \in C''} BP_EB06_{c,H'_i})), \forall i \in \{0, |H'| - 1\}, \quad (32)$$

$$X_{_A_{H'_i}} \geq B_{ac_A_i} - M(1 - (1 - \sum_{c \in C''} B_{ant_{c,i}}) + (1 - \sum_{c \in C''} BP_EB06_{c,H'_i})), \forall i \in \{0, |H'| - 1\}, \quad (33)$$

$$X_{_A_{H'_i}} \leq M(1 - \sum_{c \in C''} B_{ant_{c,i}}), \forall i \in \{0, |H'| - 1\}, \quad (34)$$

$$X_{_A_{H'_i}} \leq M(1 - (1 - \sum_{c \in C''} BP_EB06_{c,H'_i})), \forall i \in \{0, |H'| - 1\}, \quad (35)$$

$$X_{_A_{H'_i}} \leq BA_{max}, \forall i \in \{0, |H'| - 1\}, \quad (36)$$

3.5. Pelletizing constraints

Constraints (37)-(39) manage pulp influx and pelletization, ensuring production follows the relation matrix $Y_{c,p}$. Constraints (43) ensure only one product is produced at a time in pelletization. Equations (40) and (41) define Ubu production ($Prod_Ubu_{c,p,dh}$), incorporating conversion factors. Constraints (44)-(46) calculate pre-pelletization stock, constrained by E_Ubu_{max} .

$$Arriv_P_Ubu_{c,H'_i} = Bomb_EB06_{c,H'_i-(TM-TGM)}, \forall c \in C'', i \in \{|H'| - 1\}, \quad (37)$$

$$ProdPelot_{c,p,dh} \leq 1, \forall c \in C'', p \in C''', dh \in H' : Y_{c,p} = 1, \quad (38)$$

$$ProdPelot_{c,p,dh} = 0, \forall c \in C'', p \in C''', dh \in H' : Y_{c,p} \neq 1, \quad (39)$$

$$Prod_Ubu_{c,p,dh} = Prod'_{p,dh}(1 + Conv_{dia_{dh}}), \forall c \in C'', p \in C''', dh \in H' : Y_{c,p} = 1, \quad (40)$$

$$Prod_Ubu_{c,p,dh} = 0, \forall c \in C'', p \in C''', dh \in H' : Y_{c,p} \neq 1, \quad (41)$$

$$Prod_Ubu_{c,p,dh} \leq M(ProdPelot_{c,p,dh}), \forall c \in C'', p \in C''', dh \in H', \quad (42)$$

$$\sum_{c \in C'', p \in C'''} ProdPelot_{c,p,dh} \leq 1, \forall dh \in H', \quad (43)$$

$$\begin{aligned}
 EUbu_{c,H'_i} &= EUbu_{c,H'_{i-1}} + (APUbu_{c,H'_i})(Sol)(Dens) \\
 &- \sum_{p \in C'''} PUbuc_{p,H'_i} - VY_{c,H'_i} + RY_{c,H'_i}, \forall c \in C'', i \in \{1, |H'| - 1\},
 \end{aligned} \quad (44)$$

$$\begin{aligned}
 EUbu_{c,H'_0} &= EUbu_c^0 + (APUbu_{c,H'_0})(Sol)(Dens) \\
 &- \sum_{p \in C'''} PUbuc_{p,H'_0} - VY_{c,H'_0} + RY_{c,H'_0}, \forall c \in C'',
 \end{aligned} \quad (45)$$

$$\sum_{c \in C''} EUbu_{c,dh} \leq EUbu_{max}, \forall dh \in H', \quad (46)$$

Constraints (47)-(48) determine yard stock in Ubu, with initial stock ($E_Praca_c^0$) as input. Constraint (49) limits the rate of material return from yard to pelletization (Ret_{max} as input). Constraints (50)-(52) manage stock transfer to the yard, ensuring transfers only if pre-pelletization stock meets a minimum capacity (F_Exc_yard and E_Ubu_{max} as inputs). Finally, constraint 53 sets the minimum production level using $Prod_Ubu_{c,p,dh}$, with a slack variable utilized to prevent infeasibility while training the PPO agent.

$$EY_{c,H'_i} = EY_{c,H'_{i-1}} + VY_{c,H'_i} - RY_{c,H'_i}, \forall c \in C'', i \in \{1, |H'| - 1\}, \quad (47)$$

$$EY_{c,H'_0} = EP_c^0 + VY_{c,H'_0} - RY_{c,H'_0}, \forall c \in C'', \quad (48)$$

$$\sum_{c \in C''} RY_{c,dh} \leq Ret_{max}, \forall dh \in H', \quad (49)$$

$$\sum_{c \in C''} VY_{c,H'_i} \leq TY(LF_{H'_i}), \forall i \in \{0, |H'| - 1\}, \quad (50)$$

$$\sum_{c \in C''} E_Ubu_{c,H'_i} - F_Exc_Yard(E_Ubu_{max}) \leq M(Lib_Filt_{H'_i}), \forall i \in \{0, |H'| - 1\}, \quad (51)$$

$$F_Exc_Yard(E_Ubu_{max}) - \sum_{c \in C''} E_Ubu_{c,H'_i} \leq M(1 - Lib_Filt_{H'_i}), \forall i \in \{0, |H'| - 1\}, \quad (52)$$

$$\sum_{c \in C''} Prod_Ubu_{c,p,dh} + slack \geq min_prod_ubu, \forall c \in C'', i \in \{|H''| - 1\} \quad (53)$$

4. Methodology

In this work, we propose to enhance the solution process for this complex problem by applying a reinforcement learning technique to tackle its most challenging phase. As demonstrated by van Hezewijk et al. [2023]; Wang et al. [2021]; Zhu et al. [2020]; Zhang et al. [2022], reinforcement learning methods have been successfully used to solve various types of optimization problems.

Specifically, we implement the Proximal Policy Optimization (PPO) algorithm to manage the pipeline stage. The constraints described in section 3.4 are incorporated into the policy framework, to guide the agent's decision towards a solution that does not violate any operational constraint. As the objective function is to maximize the overall production output, the quality of

each solution determined by the agent is evaluated by how much this output increased in contrast with the best previous solutions, the agent's is defined by the difference between these values.

Upon completing the training phase, the agent's policy can be utilized to pre-process the model's input, providing the mathematical model solver with near-feasible batch pumping values for the pipelines. This feedback improves the overall efficiency of the production process. A basic implementation of PPO is presented in 1.

Algorithm 1 Proximal Policy Optimization

```

1: while Training Episodes not done do
2:   Initialize variables
3:   for  $h \in horizon$  do
4:     Initialize validation masks
5:     Call predict method to get action
6:     Call step function and collect reward
7:     Store rewards and objectives functions values

```

4.1. Policy training implementation

Our method begins by building a policy grounded in the constraints of the operational rules of the pipelines, which have been described in detail in the earlier section. This policy serves as a guide for the agent, enabling it to make decisions and update the solution state accordingly [Zhu et al., 2020]. The agent's actions are based on the interactions with elements presented in the environment, such as the total production volume at the concentrator and pelletizing plant, the storage levels, and the objective function value.

The observation space is constituted of a binary array of options, in our case, each position of the array determines whether the agent can choose to pump that type of pulp or to decide to pump water. To optimize the agent's learning and prevent actions that lead to model infeasibility, we make use of a masked environment, which restricts the agent's pool of choices. For each action, we consider the current volume stored in the concentration tank and determine whether, with this volume plus the anticipated concentrator production for the minimum batch sequence size, it is feasible to pump the pulp without causing an outage. If this condition is met, the option to pump pulp is presented to the agent. Otherwise, the only available option for the agent is to pump water. This strategy ensures that the agent consistently makes feasible decisions, streamlining the learning process and enhancing the overall efficiency of the reinforcement learning model in managing the pipeline stage of the production process.

The algorithm presented in 2 implements the operational rules that the pipeline must follow, instructing the agent to manage the pulp pumping process of batches. It determines the actions based on the current state and conditions. The algorithm begins by checking if no pulp batches are being processed ($nBatchesP == 0$) and if the action is not water. If these conditions are met, it processes pulp batches by iterating from the current step up to a limit, updating the pumping array with the specified action, and incrementing both the step and batch counters. Similar checks and updates are performed if water batches ($nBatchesW == 0$) are being processed. The algorithm ensures appropriate actions are taken within the limits of batch sizes and steps, resetting and updating various counters and state variables accordingly.

For pulp batches, if no pulp batches are being processed ($nBatchesP == 0$) and the action is not water, the algorithm checks if adding the minimum required pulp batches ($step + Pulp_min$) fits into the remaining available space ($H - 1$). It also checks that the addition won't exceed the maximum limit of pulp batches ($pulp_max$). If these conditions hold, the pumping array is updated

to reflect the maximum allowable pulp batches. Otherwise, if adding the minimum required pulp batches exceeds the array's size ($Pump_size$), the array is updated accordingly while ensuring the pulp batch limit is respected. The state of the pulp and batch counts is then updated.

Similarly, for water batches, if no water batches are being processed ($nBatchesW == 0$) and the action is water, the algorithm checks whether adding the minimum required water batches ($step + water_min$) fits within the array ($Pump_size - 1$). It also ensures that adding water will not exceed the maximum pulp batch count ($pulp_max$). If valid, the pumping array is updated for the water batches. If the addition exceeds the pump size, the array is updated accordingly. The water state and batch counts are updated afterward.

Finally, the algorithm handles the transitions between the materials. If the action is not water and the number of pulp batches is between the minimum and maximum limits ($pulp_min$ and $pulp_max$), it updates the pumping array, water state, and batch counts. Similarly, if the action is water and the number of water batches is within the valid range, it also updates the array, water state, and batch counts.

Algorithm 2 Batch Policy implementation

```

1: if  $nBatchesP == 0$  and  $action \neq water$  then
2:   if  $step + Pulp\_min \leq H - 1$  and  $nBatchesP + pulp\_min \leq pulp\_max$  then
3:     Updates Pumping array to batch limit
4:   else if  $step + pulp\_min > Pump\_size - 1$  and  $nBatchesP + pulp\_min \leq pulp\_max$ 
   then
5:     Updates Pumping array end
6:   end if
7:   Pulp state and batch counts
8: end if
9: if  $nBatchesW = 0$  and  $action == water$  then
10:  if  $step + water\_min \leq Pump\_size - 1$  and  $nBatchesW + water\_min \leq pulp\_max$  then
11:    Updates Pumping array to batch limit
12:  else if  $step + water\_min > Pump\_size - 1$  and  $nBatchesW + water\_min \leq pulp\_max$ 
   then
13:    Updates Pumping array end
14:  end if
15:  Water state and batch counts
16: end if
17: if  $action \neq water$  and  $nBatchesP \geq pulp\_min$  and  $nBatchesP < pulp\_max$  then
18:  Updates Pumping array, water state, and batches counts
19: else if  $action = water$  and  $nBatchesW \geq water\_min$  and  $nBatchesW < water\_min$  then
20:  Updates Pumping array, water state, and batches counts
21: end if=0

```

After the agent makes its decision on what to pump in the current step, the solver is called with the updated pump batches as parameters. The solver uses these parameters to solve the Mixed-Integer Linear Programming (MILP) problem, starting with this feasible solution. The value of the objective function returned by the solver is used to determine the agent's reward, which is based on the improvement in production value from this iteration. This iterative process continues until the specified time horizon is met. The final value of the objective function is stored, and the process repeats until the end of the training steps.

5. Experiments and results

The experiments were conducted on a dedicated machine with the following specifications: a Hexa-core i5-9400F processor, 16GB of RAM, an NVIDIA GTX 1650S graphics card, and Ubuntu 22.04 as the operating system. The algorithm was implemented in Python 3.10, utilizing the PuLP library for the mathematical model. The Gymnasium library¹, alongside the TensorFlow² backend, was used for the implementation and training of PPO environments. The COIN-OR³ and GUROBI⁴ solvers were chosen to compare the results.

The agent was trained over 500,000 episodes, with each episode featuring a randomly selected training instance from a pool of 60 distinct instances. These included both real-world scenarios and randomly generated ones. Solvers were given a time limit of 300 seconds and a target gap of 1%.

5.1. Instances setup

The test instances were designed with the intent to stress the model with extreme scenarios. Ten different instances were created for each of the scenarios, totaling 40 instances.

- **Low storage, high demand (LSHD):** The initial storage values are low (0 or near 0), while the final minimum volume demanded reaches the operational maximum for each product. This scenario aims to stress the concentrator and pelletizing plant production to their maximum capacity and challenge the pipeline with a constant pulp pump.
- **High storage, low demand (HSLD):** The initial storage values are high (maximum or near maximum), and the final minimum volume demanded is low. The goal is to test the storage capacity constraints to ensure that they are respected under conditions of high initial storage and low demand.
- **Varying storage, high demand (VSHD):** The initial storage values vary, with some storage near maximum and others near minimum. The final minimum volume demanded for each product is near the operational maximum. This scenario tests the model's ability to prioritize production and pumping, dealing with uneven initial storage levels while meeting high demand.
- **Average storage, average demand (ASAD):** Both the initial storage values and the final minimum volume demanded are at average levels. This scenario is designed to reflect typical day-to-day operations, ensuring that the model performs efficiently under normal conditions.

Each of these scenarios was created to push the model to its limits and evaluate its performance under a wide range of operational conditions. By covering extremes and average situations, these instances provide a robust testbed for assessing the model's capabilities, reliability, and adaptability.

5.2. Training results

In Figure 1 the reward collected during the agent's training is presented. What stands out is the notable increase in the reward curve during the initial phases, followed by a period of stabilization towards the later stages. This pattern aligns with our expectations, primarily attributable to the validation masks method employed. These masks effectively constrained the agent's actions, ensuring it avoided actions that could lead to infeasibility. Consequently, this precautionary measure

¹<https://gymnasium.farama.org/>

²<https://www.tensorflow.org/>

³<https://www.coin-or.org/>

⁴<https://www.gurobi.com/>

prevented significant decreases in the reward, contributing to the observed stabilization towards the end of the training process.

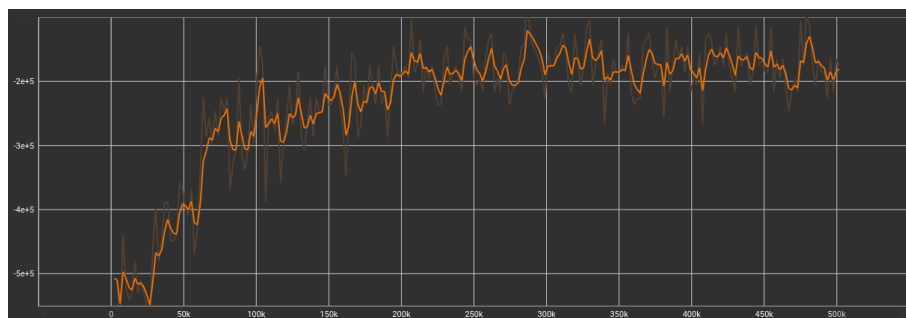


Figure 1: Episodes rewards

5.3. Computational experiments results

In Table 1 we contrast how well the solvers COIN-OR and GUROBI performed on average, on each of the types of instances, both with and without the assistance of the PPO's agent.

Table 1: Time result comparison

instance	COIN-OR		GUROBI	
	time without	with ppo	without ppo	with ppo
LSHD	not solved	89,39	20,38s	1,1s
HSLD	not solved	67,15	4,54s	0,2s
VSHD	not solved	93,55s	21,82s	1,1s
ASAD	not solved	108,41s	31,2s	1,8s

The COIN-OR was unable to solve any of the examples without the assistance of the PPO's agent, on the other hand, GUROBI was able to solve them all. The GUROBI without the ppo timings that have been observed vary from 4.54 seconds for HSLD to 31.2 seconds for ASAD. Which is presumed, by the level of sophistication the GUROBI solver has.

When PPO assistance is applied, there is a significant reduction in the time taken by both solvers, with COIN-OR managing to solve all instances it previously could not. COIN-OR's times range from 67.15 seconds for HSLD to 108.41 seconds for ASAD. GUROBI also shows improved performance, with its solving times reduced to between 0.2 seconds for HSLD and 1.8 seconds for ASAD. This comparison highlights the effectiveness of the ppo technique in enhancing the performance of both solvers, particularly COIN-OR, which was otherwise unable to solve the instances without it.

6. Conclusion

In this paper, we propose a Mixed-Integer Linear Programming (MILP) model for the production planning of a mining operation and a reinforcement learning method to assist solvers in efficiently tackling it. The authors implemented the Proximal Policy Optimization (PPO) algorithm to address particularly challenging components of the model, thereby providing the solver with a feasible initial solution. Although a simplification of the original model had to be made for the purposes of this study, the computational experimental results indicate that even with these simplifications, the free solver struggled to find a solution, having its performance greatly enhanced with the assistance of the PPO algorithm. Conversely, while the commercial solver demonstrated

good results with this version of the problem, it still exhibited performance improvements when augmented by the PPO assistance.

Moreover, the outcomes from this paper reveal a wealth of untapped potential yet to be explored. The results suggest that future research can extend similar methods to larger and more complex instances of mathematical formulations. By scaling up the application of reinforcement learning techniques, researchers can further investigate the potential of artificial intelligence to assist in solving intricate problems, particularly, for cases where even the most sophisticated commercial solvers struggle to find feasible solutions.

References

- Caccetta, L. (2007). *Application of Optimisation Techniques in Open Pit Mining*, p. 547–559.
- Esteso, A., Mula, J., Peidro, D., e Diaz-Madronero, M. (2023). Reinforcement learning applied to production planning and control. *International Journal of Production Research*, 61(16):5772–5789.
- Schulman, J., Wolski, F., Dhariwal, P., Radford, A., e Klimov, O. (2017). Proximal policy optimization algorithms.
- van Hezewijk, L., Dellaert, N., Woensel, T. V., e Gademann, N. (2023). Using the proximal policy optimisation algorithm for solving the stochastic capacitated lot sizing problem. *International Journal of Production Research*, 61(6):1955–1978.
- Wang, L., Hu, X., Wang, Y., Xu, S., Ma, S., Yang, K., Liu, Z., e Wang, W. (2021). Dynamic job-shop scheduling in smart manufacturing using deep reinforcement learning. *Computer Networks*, 190:107969. ISSN 1389-1286.
- Zhang, Y., Haihua, Z., Tang, D., Zhou, T., e Gui, Y. (2022). Dynamic job shop scheduling based on deep reinforcement learning for multi-agent manufacturing systems. *Robotics and Computer-Integrated Manufacturing*, 78:102412.
- Zhu, J., Wang, H., e Zhang, T. (2020). A deep reinforcement learning approach to the flexible flowshop scheduling problem with makespan minimization. In *2020 IEEE 9th Data Driven Control and Learning Systems Conference (DDCLS)*, p. 1220–1225.