Reinforcement Learning for Grinding Circuit Control in Mineral Processing

Mattias Hallén*, Max Åstrand*†, Johannes Sikström‡, and Martin Servin§
*ABB Corporate Research, Västerås, Sweden
Email: mattias.hallen@se.abb.com, max.astrand@se.abb.com
†Department of Automatic Control, School of Electrical Engineering and Computer Science,
KTH Royal Institute of Technology, Stockholm, Sweden
‡Boliden Mines, Boliden, Sweden
Email: johannes.sikstrom@boliden.com
§Department of Physics, UMIT Research Lab, Umeå University, Umeå, Sweden
Email: martin.servin@umu.se

***Abstract*—Grinding, i.e. reducing the particle size of mined ore, is often the bottleneck of the mining concentrating process. Thus, even small improvements may lead to large increases in profit. The goal of the grinding circuit is two-sided; to maximize the throughput of ore, and minimize the resulting particle size of the ground ore within some acceptable range. In this work we study the control of a two-stage grinding circuit using reinforcement learning. To this end, we present a solution for integrating industrial simulation models into the reinforcement learning framework OpenAI Gym. We compare an existing PID controller, based on vast domain knowledge and years of hand-tuning, with a black-box algorithm called Proximal Policy Optimization on a calibrated grinding circuit simulation model. The comparison show that it is possible to control the grinding circuit using reinforcement learning. In addition, contrasting reinforcement learning from the existing PID control, the algorithm is able to maximize an abstract control goal: maximizing profit as defined by a profit function given by our industrial collaborator. In some operating cases the algorithm is able to control the plant more efficiently compared to existing control.**

I. Introduction

A. Problem Description and Motivation

A large part of the costs of running a mineral processing plant can be attributed to the grinding mills. As an example, a breakdown of the costs in a copper concentrator show that 47% of the cost per concentrated copper ton can be accredited to grinding [1]. This cost is mostly due to the energy required to operate the mills. The resulting ground particle size effectively determines the maximum performance of the concentrating process. The particles must be ground to an appropriate size. Too large or too small particles become problematic for the subsequent downstream process. Due to these facts, the grinding process is an interesting process to optimize where small improvements leads to sizeable increases in profitability

This paper focuses on controlling a grinding circuit using Reinforcement Learning (RL), which is a field within artificial intelligence, and in particular within machine learning. Reinforcement learning has received a lot of attention lately by beating the strongest human players in various games. RL is designed to control a system by reinforcing historically beneficial behavior. By designing a suitable reward, the implementer may define *what* the end goal is without having the need to specify *how* to reach that goal. In contrast, in classic control based on PID controllers [2], pursuing a high-level goal requires extensive domain knowledge to characterize the overall behavior of several interconnected PID loops controlling low-level set-points.

The literature on RL is largely focused on applications to various games and toy problems. Early pioneers in this field demonstrated its promise by playing Checkers [3] and Backgammon [4]. More recently, Deep Reinforcement Learning (RL combined with Deep Neural Networks), has rapidly expanded the field to new frontiers. In 2013 an algorithm was capable of reaching human level performance in Atari games by only observing game pixel images [5]. Three years later, in 2016, an algorithm was capable of solving more than 20 simulated physical tasks including cartpole swing-up and legged locomotion [6]. The same year, an algorithm also managed the impressive feat of winning against professional players in the computationally challenging game of Go [7].

Unfortunately, there are only a few published works on industrial process control using RL. Process control brings a different set of challenges compared to studying games; the task is often continuous with no clear definition of a win or a loss, sensors commonly provide continuous signals with noise and uncertainty, and the equipment may have dynamic constraints on the range of allowed actions. The use of RL on industrial process control was however studied by S.P.K. Spielberg et al [8] which demonstrated an RL algorithm capable of controlling simulated linear systems. In contrast, there are more works on RL for robotics. One such notable example is the work by OpenAI [9] in 2018 where a robotic hand was trained in simulation without human demonstrations to manipulate a cube. This simulated experience was then transferred to a physical robot that managed to solve the real-world equivalent task with significant reliability.

In this work, we add to the scarce literature on reinforcement learning for industrial process control. Specifically, we study an ore grinding process within a mineral processing plant. To do this, we use a simulation model of the grinding process that has been calibrated on real operational data. The model is implemented in the popular industrial simulation software Dymola [10] which is based on the Modelica [11] language.

Our first contribution is a solution for integrating Dymola simulation models into OpenAI Gym [12]; a popular framework for RL research. The solution is based on exporting the simulation model as a Functional Mock-up Unit (FMU) adhering to the Functional Mock-up Interface (FMI) standard [13]. Consequently, this makes it possible to apply state-of-the-art RL algorithms found in OpenAI Baselines [14] to any industrial simulation model developed in a software that support the FMI standard.

Our second contribution is a comparison of control strategies for the grinding mills in Aitik, Sweden; a mineral processing plant owned and operated by Boliden. Proximal Policy Optimization, a modern RL method, is compared with a typical PID control strategy. The PID strategy is identical to the strategy used in a real operating plant. It is based on years of experience, vast domain knowledge, and have been hand-tuned for several years. The goal is to investigate if the RL controller can be used to control the grinding circuit in a more effective way compared to the traditional control strategy. Results in this paper confirm that $i)$ it is possible to control the grinding circuit using PPO, $ii)$ the algorithm is capable to maximize an abstract control goal such as maximizing the profitability, and $iii)$ for some operating cases it is able to control the process more efficiently than the PID strategy.

The introduction in Section I covered the main background of this paper, and the following Section II will cover the basic theory underlying the process using grinding mills. An introduction to Reinforcement Learning is presented in Section III, which will cover the general concepts and introduce Proximal Policy Optimization. Then in Section IV the method of adapting a continuous simulation model to a reinforcement learning problem will be described. Results are presented in Section V and finally concluding remarks are discussed in Section VI.

II. Comminution by grinding

In the mineral processing chain, the first step is to separate ore from an ore-body either in an open pit or an underground mine. This ore is crushed to reduce the size of the rock and then transported to a concentrator plant where the ore is ground down to a suitable particle size for the subsequent concentrating steps. After it has been ground, it is treated by various chemical or mechanical operations to separate the valuable minerals from the commercially worthless waste rock. When the ore has been concentrated, it is typically shipped to a smelter for further processing.

Reducing the particle size of rock through grinding is called *comminution*. The main objective for a comminution process is to liberate valuable minerals from the waste rock. Specifically, the goal is to maximize the exposed mineral surface as to efficiently separate the ore from the waste

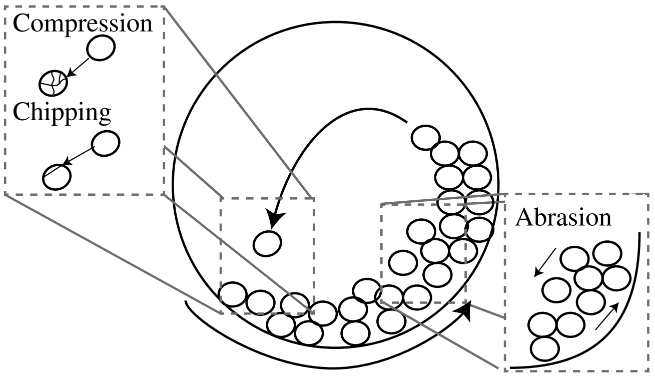


Fig. 1: Overview of the operating principle of a grinding mill in a cross section.

rock in the subsequent process stages focusing on enrichment [1]. When using flotation as a subsequent enrichment stage (i.e. separating mineral from waste rock by using tiny air-bubbles exploiting the hydrophobic and hydrophilic properties of different materials), there is a natural lower limit when it is inefficient to grind further. Grinding the material too fine leads to the small particles sticking to the air bubbles due to their lightness and thus mixing waste rock with the desired minerals which may ruin the grade of the recovered product. In addition, the increase of surface area from splitting the particles decreases as the particle gets smaller, thus it costs more energy to grind the particles finer.

Large grinding mills are used to reduce the size of the ore particles. An *autogeneous* mill reduces the particle size by using the ore itself as grinding media; crushing smaller particles by larger particles. This type of mill is basically a large rotating drum that grinds the ore in a combination of three effects: compression, chipping, and abrasion (see Figure 1). The angular velocity of the grinding mill is a determinant factor in achieving efficient grinding. If the mill is rotating too fast, then the centrifugal force will make the ore particles stick to the inside, resulting in very little grinding. If the speed is too low, the grinding is only due to abrasion, which is not as energy efficient as compression and chipping. A balance in speed is thus sought after, where the particles being heaved hits the particles at the bottom of the drum, maximizing the compression and chipping effects.

A common design of a grinding circuit is a two-staged wet grinding process, in which mixture of ore and water is ground in sequence by two grinding mills. This design consists of a primary autogeneous mill (PM) and secondary *pebble* mill (SM). The pebble mills grinds by adding coarse material (pebbles) from the primary mill selectively. The reasoning is that grinding finer materials with coarser material is more efficient than grinding comparably sized materials. Water is added at the intake of each mill, and the end product is a mixture of water and ground ore that can be pumped to the subsequent concentration steps. However, the ore mixture is not always traversing sequentially through the grinding circuit. Recirculation ensures that only small enough ore particles are passed out of the circuit, while the larger ore particles are sent back to the grinding mills.

III. Proximal Policy Optimization

Reinforcement learning is a subfield of machine learning, focused on learning by interaction. This process is similar to how humans and animals learn. An example of learning by interaction is a child that touches a hot stove. By reaching the hot stove a negative feeling occurs and the child is less inclined to touch the stove again. In other words, the *agent* (child) is in a *state* close to the stove. It takes an *action* in an *environment* by moving its hand, touches the stove, and by touching it the agent transitions into a new state in the environment, and a negative *reward* is perceived by touching the hot surface.

Reinforcement learning problems are typically modelled as Markov Decision Processes (MDPs). MPD is a mathematical model for decision making. Given a state, a limited amount of actions can be taken. With each action follows a probability that the agent ends up in a certain new state, and each transition from one state to another yields the agent a reward. More formally, we have a finite set of states $\mathbb{S}$ and a finite set of actions $\mathbb{A}$ available to be taken in these states. Each state-action pair transition has an associated immediate reward function $\mathbb{R}(s_t, a_t, s_t + 1)$ that follows the transition dynamics $\mathbb{T}(s_{t+1}|s_t, a_t)$ describing what state you end up in after taking an action. A policy $\pi(s)$ determines what action to take given a certain state

$$\pi(s) : s \longrightarrow A. \tag{1}$$

A discount factor γ is introduced to represent the importance of future rewards compared to instantaneous rewards. The *value* of a state is thus defined as the discounted sum of the expected future reward following the current policy

$$V^{\pi}(s) = \mathbb{E}[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s]. \tag{2}$$

Alternatively, *value* can be defined in terms of state-action pairs

$$Q^{\pi}(s, a) = \mathbb{E}[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a]. \tag{3}$$

For process control, the signals encountered are often continuous. One way of representing a continuous control policy is to construct a Gaussian distribution of actions using a neural network. The input of the network is the state. The output of the network is which action to take, represented by a mean and a standard deviation for each possible control action. Here, exploration is done by sampling from a distribution in contrast to only using point estimates. This policy network can be written $\pi_\theta(a|s)$ where θ are the weights and biases of the neural network. As a conceptual illustration, a multilayer perceptron Gaussian probability distribution policy is shown in Figure 2.

Proximal Policy Optimization (PPO) was proposed by John Schulman et al. [15]. The general idea is to do multiple optimization steps over the same trajectory, without destructively large policy updates. Compared to similiar approaches, it strikes a favorable balance between sample complexity, simplicity, and wall-time [15]. We use a clipped surrogate loss function

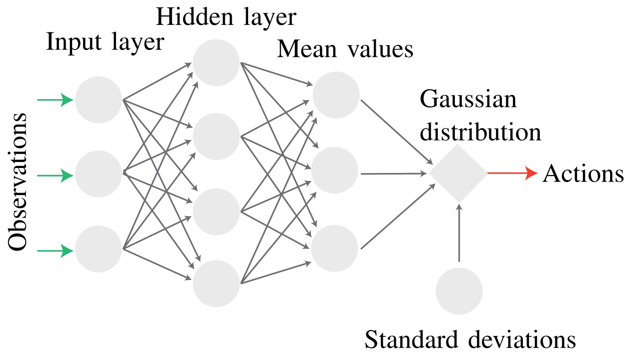


Fig. 2: A multilayer perceptron Gaussian policy. It consists of an input layer for the state observations, one hidden computation layer, and an output layer representing control actions in terms of mean values with corresponding standard deviations.

$$L(\theta) = \hat{\mathbb{E}}[\min(r_t(\theta)\hat{A}_t, \texttt{clip}(r_t(\theta), 1 - \varepsilon, 1 + \varepsilon)\hat{A}_t)] \tag{4}$$

where the ratio

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \tag{5}$$

is the importance sampling between the policy used for gathering experience and the new updated policy. The function `clip` is defined as $\texttt{clip}(x, \alpha, \beta) = \max(\min(x, \beta), \alpha)$. The rationale is that by clipping the ratio $r_t(\theta)$ to be in the interval $[1 - \varepsilon, 1 + \varepsilon]$ one stays within the proximity of the previous policy.

The advantage function $A = Q(s, a) - V(s)$ estimates how much better/worse (advantageous) taking a certain action is compared to the estimated value of the current state. As [15], we use the alternative formulation of Generalized Advantage Estimation (GAE) to estimate the advantage in Equation (4) by

$$\hat{A}_t = \delta + (\gamma\lambda)\delta_{t+1} + \cdots + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1}, \tag{6}$$

$$\text{where} \quad \delta = r_t + \gamma V(s_{t+1}) - V(s_t) \tag{7}$$

and the parameters λ, γ are used to tune the bias/variance trade-off. The state value $V(s_t)$ in (7) is estimated by another neural network $\hat{V}_w(s_t)$ parametrized by w and it is trained from the observed rewards.

IV. Simulation

In this work, we study a grinding circuit in a mineral processing plant. This study is based on a simulation model that has been calibrated and verified against the real plant. In Figure 3 a schematic drawing of the simulated grinding circuit can be seen. Having access to the existing control loops based on PID, this makes it possible to replacing control

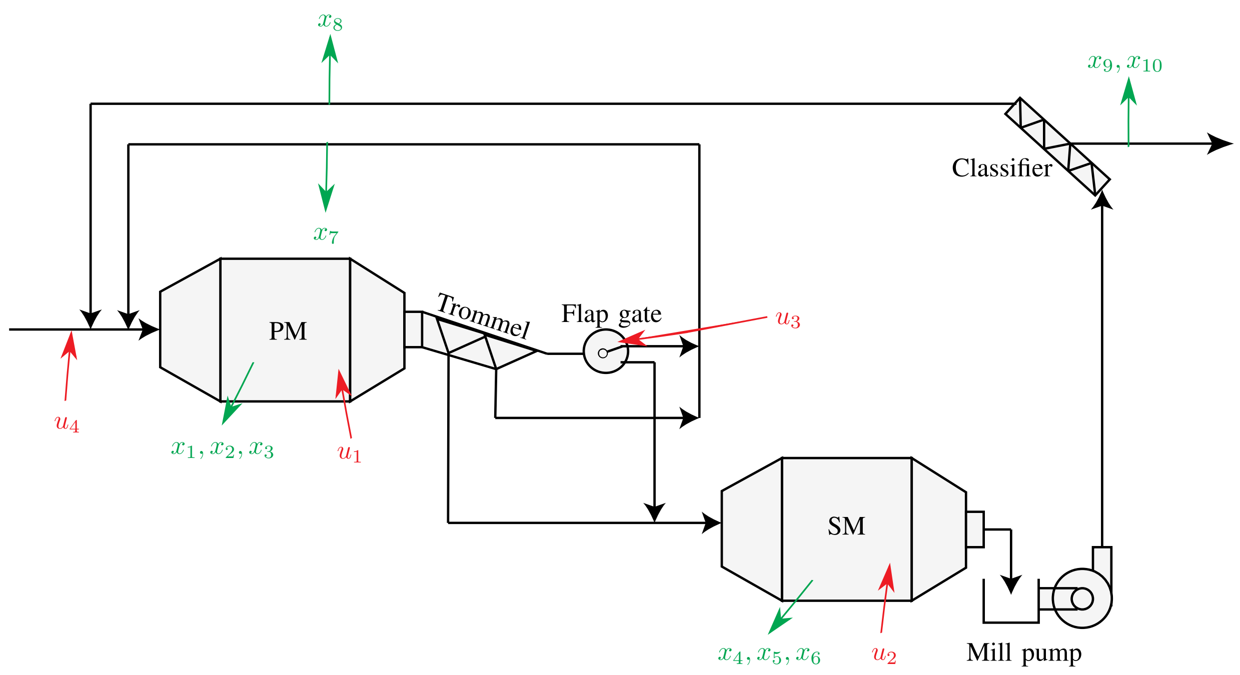


Fig. 3: The simulated grinding circuit with highlighted observations and actions corresponding to Table I.

TABLE I: Implemented observations and actions.

	Observations	**Unit**	**Accepted range**
x_1	Primary mill mass	ton	High & Low limit
x_2	Primary mill torque	%	High limit
x_3	Primary mill power	MW	High limit
x_4	Secondary mill mass	ton	High limit
x_5	Secondary mill torque	%	High limit
x_6	Secondary mill power	MW	No constraint
x_7	Mass flow return	ton/h	No constraint
x_8	Mass flow return classifier	ton/h	No constraint
x_9	Mass flow out	ton/h	No constraint
x_{10}	Particle size K80[1]	μm	No constraint
	Actions		
u_1	Primary mill speed	RPM	Continuous
u_2	Secondary mill speed	RPM	Continuous
u_3	Flap gate	-	Discrete
u_4	Feed flow	ton/h	Continuous

loops with RL controllers while keeping the original control for benchmarking. Table I shows all the possible observations and actions for the RL controller that are varied throughout the later introduced experiments.

The motivation for these observations is three-fold. First, these observations make the performance of the process observable (performance being a function of the mass flow out and the particle size distribution K80 of this mass flow). Second, the main constraining properties of the process are observed to ensure that operation in reality does not damage the equipment. Third, the two return flows are measured to make sure that the state of the back-fed material flow in the circuit is visible to the RL algorithm. The actions in Table I include all major controllable elements in the grinding circuit. The controller can set the speed of the primary and secondary mill, the input feed of ore from the ore storage to the primary mill, and the flap gate that controls when to selectively add pebbles (coarse material) to the secondary mill. The simulation model also includes the possibility of varying the characteristics of the ore being fed into the grinding circuit by varying the hardness and the size distribution of the ore feed.

It may be noted that water is added at the inflow to both mills. This water additive control is kept according to existing PID control as a simplification. In addition, the mill pump after the secondary mill has been removed. This reduces computational expense, and is a fair approximation since at a steady state the flow from the secondary mill is equal to the flow of the mill pump.

A. *Integrating OpenAI with FMU simulation*

To leverage state-of-the-art RL algorithms, the OpenAI Gym [12] is used as a framework. OpenAI Gym is a popular open source framework that has many implemented algorithms for solving Gym *environments*. An environment is the process that the agent interacts with, which it is supposed to learn how to control through its actions. OpenAI Gym thus provides a suitable layer of abstraction, providing a standardized interface

[1]K80 is a particle size distribution measurement indicating that 80% of the particles are smaller than the K80 value.

between the actions/observations (that the RL algorithm need) and how these affect the underlying environment.

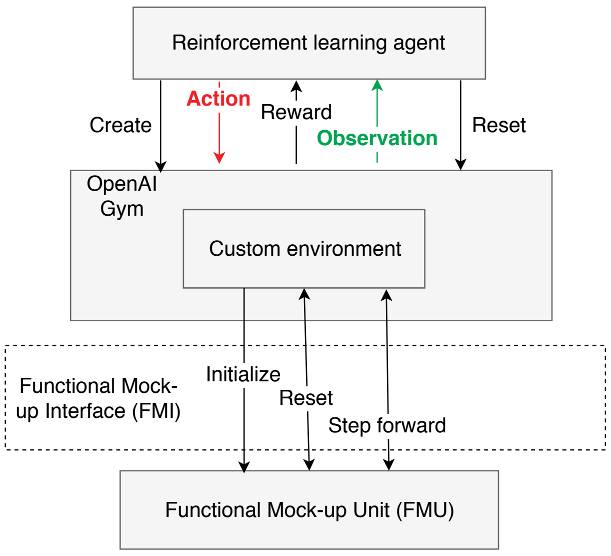
In this work, the environment that we want to control is the calibrated simulation model implemented in the simulation software *Dymola*. To be able to integrate the industrial process model from Dymola into OpenAI Gym, we modified the standard OpenAI framework. Our proposed approach is based on compiling the simulation model into a Functional Mock-up Unit (FMU) communicating by a standard communication protocol. The FMU is thus a standalone executable file that can be integrated into the OpenAI framework to act as the underlying environment. The FMU communicates with the surrounding Python-based framework using the Functional Mock-up Interface provided by the pyFMI library [16]. A schematic view of how these different components interact can be seen in Figure 4. This architecture provides a suitable abstraction layer for RL problems, in which the algorithm is completely separated from the specific process model and the communication via FMI. From the perspective of an RL task, the important part is the ability to take an action and receive a reward coupled with a new observation, the rest of the architecture is dedicated to managing the environment and FMI communication. It is worth to openote that in this work the industrial simulation model was implemented in Dymola. However, the proposed OpenAI integration supports any process model that can be compiled into an FMU adhering to the FMI standard. Another benefit of this integration is that by leveraging internal methods in the pyFMI library, the computational expense is decreased compared to previous approaches to integrating OpenAI and FMU. [2]

In this work, the model's FMU was compiled as a co-simulation model bundled with the Dymola solver Cerk23 (a third order single-step Runge-Kutta integrator). The time discretization, as seen from the RL agent, is 60 seconds meaning that after each action the process is simulated 60 seconds into the future before a reward is calculated. This roughly corresponds to how quickly the simulated grinding circuit model reacts. The binary action corresponding to the flap gate is in the network treated like a continuous signal which at the time of action is compared to a threshold determining whether the flap gate should the activated or not. Lastly, to improve training performance, observations and actions are normalized before being fed to the RL agent.

B. *Proximal Policy Optimization*

Using OpenAI as a framework makes numerous RL algorithms available. In this work, we focus on Proximal Policy Optimization. The implementation of this algorithm is based on the algorithm *PPO2* contained in OpenAI Baselines [14]. It is built around the machine learning framework Tensorflow [17] and has support for parallelization by running parallel environments using Message Passing Interface. All runs are done on a Windows-computer with two Xeon X5650 processors using 23 virtual cores resulting in training and simulation of 23 parallel environments. The loss function Equation (4) is minimized using the Adam implementation in Tensorflow. Lastly, both the value and the policy network are implemented as an MLP Gaussian policy distributions.

[2]The modified framework, and an example using a reversed pendulum, can be found at www.github.com/semahal/FMU-Env



Reinforcement learning agent
Create
Action
Reward
Observation
Reset
OpenAI Gym
Custom environment
Functional Mock-up Interface (FMI)
Initialize
Reset
Step forward
Functional Mock-up Unit (FMU)

Fig. 4: Schematic view of the implemented architecture used for adapting a simulation model (in FMU format) to a reinforcement learning setting.

V. Result

In this section we present numerical results of using Proximal Policy Optimization to control a simulation model of the Boliden Aitik grinding circuit. This is done using the theory and architecture presented earlier. We will study two cases and compare the performance with the existing PID control strategy. The two cases are inspired by different control philosophies. In the first case, we assume that the overall impact of the particle size is negligible as long as the particle size is under a certain threshold. Thus, we aim to maximize ore throughput under a maximum particle size constraint.

The first case represents how the studied grinding circuit and existing control currently operate: the most profitable strategy is almost always to prioritize throughput over particle size. However, in similar grinding circuits exhibiting minor up- or downstream process differences compared to Aitik, the trade-off is not as trivial. Therefore, in the second case, we assume that the performance of the grinding circuit is dependent on both throughput and particle size to an extent where it is non-trivial to trade-off these quantities.

Note that in both cases we assume that there are no upstream or downstream ore flow limitations. The process is thus only limited by the capacity of the grinding circuit itself.

A. *Maximizing throughput*

This experiment is aimed at trying to maximize the throughput of the grinding circuit not explicitly considering the resulting particle size. To accomplish this, the reward function is designed to give +1 for each time step that mass flow out

TABLE II: Hyper parameters for PPO used in the experiments.

Parameter	**Value**
Number of steps per update	115
Minibatches per update	5
Policy entropy coefficient	0
Learning rate	LinearDecay$[1e^{-3}, 3e^{-4}]$
Value function loss coefficient	0.5
Gradient norm clipping coefficient	0.5
Discounting factor γ	0.8
Advantage estimation λ	0.95
n of training epochs per update	4
Clip range ε	0.3
Policy number of hidden layers	2
Policy hidden layer size	32
Value network	copy policy network
Penalty factor	0.2

of the circuit is at its theoretical max. To guide the optimizer, the reward decreases linearly if the mass flow is less than the theoretical max. Lastly, the reward is penalized if actions are selected outside of their bounds due to the unbounded Gaussian policy. This penalty encourages the policy to stay within action bounds and is implemented as a linear function corresponding to how much larger the selected action is compared to its bounds. The impact of this penalization is controlled by a parameter called *Penalty factor*. The algorithm hyperparameters were tuned over several experiments, and Table II depicts the final parameter set.

In Figure 5 the training progress can be seen. The episode reward and thus the throughput for the algorithm continuously climbs as the policy improves. After 7 hours of training it settles. It is interesting to note that even though these algorithms are purely "black-box" (i.e. no domain knowledge is explicitly incorporated in the control strategy), it manages to reach approximately 92 % of the throughput achieved by the PID control which is based on years of tuning and domain-specific reasoning.

PPO manages to reach 92% of the PID performance, but not more. A possible reason for this is that when maximizing throughput, it is vital that some control signals (e.g. the feed signal) is kept close to its maximum bound. Remember that we penalized the PPO reward if it selected actions outside of its bounds, thus the Gaussian exploration will make the developed policy cautious on applying actions near its bounds. Lastly, in this experiment the ore characteristics was static, i.e. the same hardness and size distribution was used throughout all training episodes resulting in no variation between training episodes. The developed policy is thus independent of the specific ore characteristics, yielding constant action values that suitable as a kind of average policy of good actions.

B. Maximizing profit

A grinding circuit has a multifaceted goal boiling down to being as profitable as possible while caring for e.g. wear, tear, and safety. The profitability of a grinding circuit depends on several factors. A simplified profit function was supplied by Boliden that involved the predicted recovery rate from particle

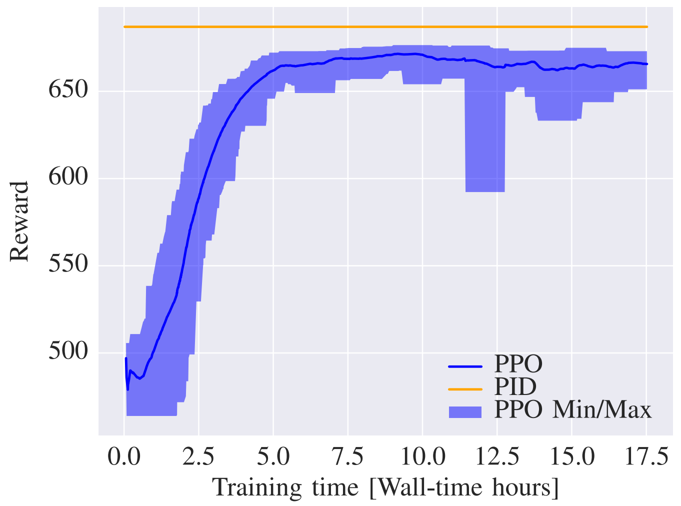


Fig. 5: Training progress for PPO with reward for maximizing throughput. Also shown is the reward achieved for the existing PID control strategy. Note the colored area for PPO which corresponds to the maximum/minimum episode rewards obtained during training.

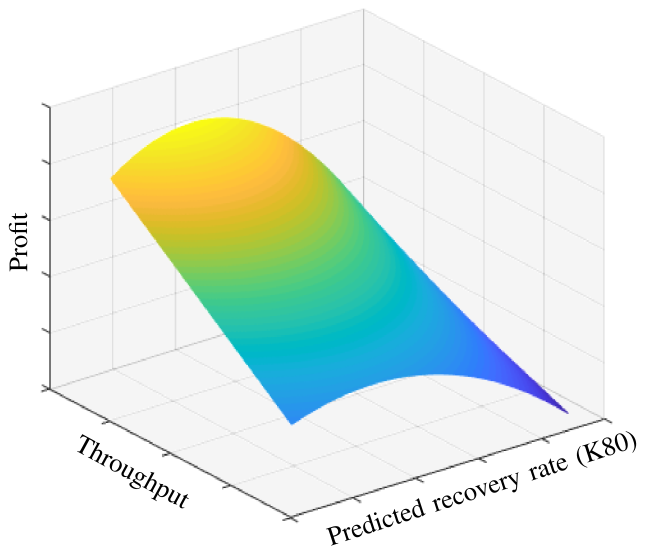


Fig. 6: The profit function implemented as a reward function. Not depicted in this figure is the constraint on achievable predicted recovery rates for a given throughput level, which is determined by the system dynamics.

size of the ground minerals and throughput. For a visual illustration see Figure 6. In this experiment, this function is used as the reward function together with the same action penalty as in the previous experiment. In this experiment, the ore properties are randomized for each episode in order to evaluate the performance of this algorithm when operating conditions change. The parameters of the algorithm are kept the same as for the previous experiment.

In Figure 7 the training progress can be seen. Note that

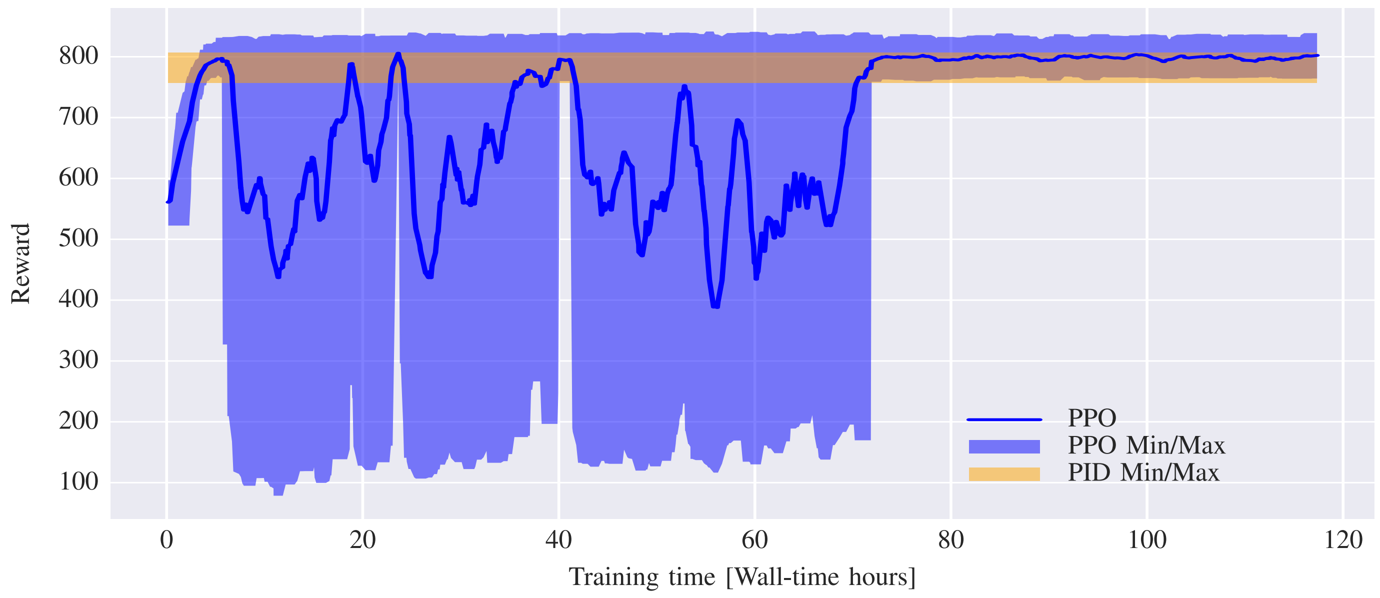


Fig. 7: Training progress for PPO when maximizing profit. The orange interval corresponds to the reward achieved by the existing PID control strategy.

since we are training using varying ore characteristic the PID baseline now corresponds to an interval (in orange) instead of a line as in Figure 5. The lower bound of the interval corresponds to hard ore characteristics while the upper bound corresponds to soft ore. The most interesting thing to note is that around the 80-hour training mark PPO manages to consistently beat the existing control strategy with respect to the given profit function. The training progress to that point is however not strictly increasing. First, we note that the initial values of the policy (which was set at 50% of maximum for all actions) were quite good, however the policy quickly improves even further. After 6h of training, the reward suddenly drops. This is due to one observable being pushed outside its associated operating bounds, and policy is updated to avoid that. In Figure 7, the algorithm slowly gets better until consistently becoming better than the PID control strategy.

The flap gate control signal (u_3) never converged to a stable control strategy. In these experiments we also note that the flap gate was never properly utilized by the emerging policy, while it is crucial for the PID control. This is an interesting observation since the profitability of the two control strategies are comparable. What happens is that the PPO (being very data-intense) is able to find a control strategy that exploits a potential model mismatch between the calibrated model and the reality. On the other hand, the PID control is implemented to mimic reality and thus misses this opportunity. This fact highlights the importance of a (very) accurate simulation model.

In order to assess the difference in the operation of the trained policy and the PID baseline we perform a minor sensitivity analysis of the control strategies, see Figure 8. In this run, the properties of the ore feed is abruptly changed from nominal ore properties to an increased hardness. We can note that overall, PPO consistently sets a lower feed signal and lower mill speeds compared to the PID control. This results in lower throughput, but finer ground material, which in the end returns a higher profit. Another difference is evident when studying the mass of the material in the primary mill. The PID control tries to keep the mass within a given set point, however the PPO algorithm does not try to hold any particular set point w.r.t. mass. It only tries to maximize the reward signal, and hence when the ore properties change the amount of mass of the material in the mill changes as well. The trained policy thus exploits its larger number of degrees of freedom, and is able to more elaborately control the grinding circuit with regard to these varying ore properties.

VI. Conclusion

In this work, we have compared a reinforcement learning controller with the existing PID controller for a grinding circuit on a calibrated simulation model. To do this, we developed a software architecture that integrates FMI compliant simulation models into the reinforcement learning framework OpenAI Gym. This makes industrial process simulations commonly found in e.g. Dymola to be controlled by state-of-the-art RL algorithms. This architecture is used to train a PPO agent controlling a grinding circuit within a mineral processing plant. By training the policy using a calibrated simulation model, the policy is capable of controlling the grinding circuit under the same process constraints as the existing control strategy. In addition, given the profit function supplied to us by our industrial collaborator, PPO is able to control the grinding circuit according to this high-level abstract goal. When comparing the profitability of PPO with the existing PID control strategy, the RL agent is able to achieve a higher profitability under certain operating conditions. This fact is

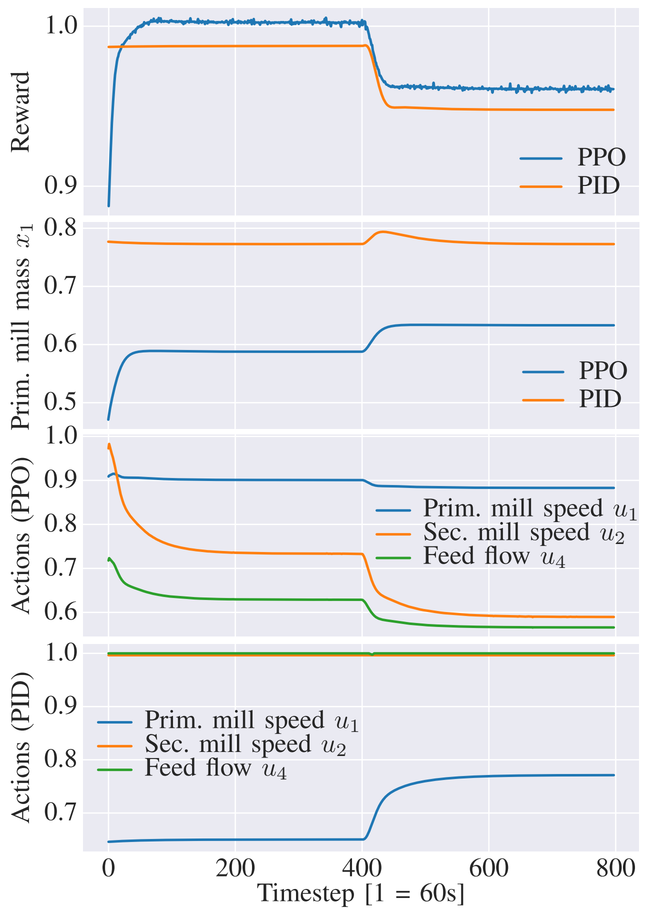


Fig. 8: Sensitivity analysis for PID and PPO. At time step 0 the ore is of nominal properties; at time step 400 ore hardness is abruptly increased.

especially interesting considering the vast amount of domain knowledge and hand-tuning that has gone into developing the PID controller compared to the black-box approach of the PPO agent. The emergent control strategy of the policy is more flexible compared to the PID baseline, and e.g. lowers the feed of ore so that the mills grind more efficiently.

An advantage of using reinforcement learning compared to conventional techniques is the separation of *what* the goal is from *how* to reach it. This makes it possible to leverage these black-box approaches to find non-intuitive control strategies, which also opens up the possibility to gain insights from a trained policy to further develop the on-site process control. However, when it comes to implementation; even if it is shown in simulation that the RL agent is more profitable, these methods still struggles with acceptance when facing a conservative industry. To alleviate this problem, future work includes studying how to combine the PID and the RL agent in an industrial control system to make the control strategy robust and be able to ensure a lowest level of performance.

Conventionally, RL has received most attention from applications in various games and toy problems. However, this work has shown promising results of applying state-of-the-art RL methods for industrial processes as well. Today, the amount of data and simulation models are increasing in many industries. In the future, we believe that RL may be a suitable technology for increased productivity in many industrial processes.

Acknowledgment

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) as funded by the Knut and Alice Wallenberg Fundation. BI Nordic and Optimation has also supplied valuable information in the work leading up to this paper.

References

[1] B. Wills, “Wills’ Mineral Processing Technology: An Introduction to the Practical Aspects of Ore Treatment and Mineral Recovery,” 2011.
[2] K. J. Åström and T. Hägglund, *PID controllers: theory, design, and tuning*. Instrument society of America Research Triangle Park, NC, 1995, vol. 2.
[3] A. L. Samuel, “Some studies in machine learning using the game of checkers,” *IBM J. Res. Dev.*, vol. 3, no. 3, pp. 210–229, Jul. 1959. [Online]. Available: http://dx.doi.org/10.1147/rd.33.0210
[4] G. Tesauro, “Temporal difference learning and td-gammon,” *Commun. ACM*, vol. 38, no. 3, pp. 58–68, Mar. 1995. [Online]. Available: http://doi.acm.org/10.1145/203330.203343
[5] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, “Playing atari with deep reinforcement learning,” *CoRR*, vol. abs/1312.5602, 2013. [Online]. Available: http://arxiv.org/abs/1312.5602
[6] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” *CoRR*, vol. abs/1509.02971, 2015. [Online]. Available: http://arxiv.org/abs/1509.02971
[7] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, “Mastering the game of Go with deep neural networks and tree search,” *Nature*, vol. 529, no. 7587, pp. 484–489, jan 2016.
[8] P. L. S.P.K. Spielberg, R.B Gopaluni, “Deep Reinforcement Learning Approaches for Process Control,” 2017.
[9] OpenAI, “Learning dexterous in-hand manipulation,” *CoRR*, vol. abs/1808.00177, 2018. [Online]. Available: http://arxiv.org/abs/1808.00177
[10] Dassault Systmes, “Dymola.” [Online]. Available: https://www.Dymola.com
[11] Modelica Association, “Modelica.” [Online]. Available: https://www.modelica.org/
[12] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, “OpenAI Gym,” 2016.
[13] Modelica Association, “Functional Mock-up Interface (FMI).” [Online]. Available: https://fmi-standard.org/
[14] P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, and P. Zhokhov, “OpenAI Baselines,” https://github.com/openai/baselines, 2017.
[15] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal Policy Optimization Algorithms,” 2017. [Online]. Available: http://arxiv.org/abs/1707.06347
[16] C. Andersson, J. Akesson, and C. Fuhrer, “PyFMI: A Python Package for Simulation of Coupled Dynamic Models with the Functional Mock-up Interface,” p. 40, 2016. [Online]. Available: https://pypi.org/project/PyFMI/
[17] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems,” 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/