Deep Reinforcement Learning Framework to Optimize Energy Supply for Electric Earthwork Processes

vorgelegt von
M.Sc.
Theodor Svennevig Skaufel

an der Fakultät III - Prozesswissenschaften
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
- Dr.-Ing. -
genehmigte Dissertation

Promotionsausschuss:

Vorsitzender: Prof. Dr. Tom Brown
Gutachterin: Prof. Dr.-Ing. Nicole Riediger
Gutachter: Prof. Dr. Frank Behrendt

Tag der wissenschaftlichen Aussprache: 07. Oktober 2022

Berlin 2023

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Berlin, den November 5, 2022

Abstract

Diesel-powered earthwork processes are responsible for a considerable portion of global greenhouse gas emissions. Electrification of earthmoving machines is generally considered the solution to achieve zero-emission earthwork processes. However, new routines, flexible scheduling, and coordination of smart operations are required to achieve efficient electric earthwork processes [1]. Furthermore, efficient charging of electric earthmoving machinery requires a purpose-built high-power charging infrastructure. The industry regards battery energy storage systems (BESS) as a potential solution to enable efficient high-power charging while avoiding unnecessary upgrades to the distribution network. Therefore, the energy supply has been identified as a significant challenge.

Electrification is a recent development in the field of earthwork and is currently not thoroughly researched. Hence, the factors that affect the energy consumption of electric earthmoving machines are underspecified. Furthermore, previous modeling efforts were identified as case-specific and do not offer generalizability. Therefore, a demand for a generalizable earthwork optimization framework that can handle uncertainties with underspecified influencing factors was identified.

This thesis proposes a novel deep reinforcement learning framework to optimize energy supply for electric earthwork processes by scheduling BESS operations in coordination with the energy consumption and productivity of electric earthmoving machines. The purpose of this framework is to serve as a decision support tool for earthwork contractors to identify new routines that can increase earthwork efficiency and facilitate planning and bidding processes.

The results demonstrate the framework's performance in developing optimized and generalizable decision strategies without relying on external data from existing earthwork projects. Furthermore, the adaptability and scalability of the framework are demonstrated by applying it to various use cases and scenarios with multiple earthmoving machines. The framework is validated by applying it to a case study to compare its performance with data recorded from an actual electric earthwork site.

This thesis builds on existing methods developed to solve problems in the earthwork literature and extends existing knowledge by combining energy optimization methods in related fields to fill the knowledge gap regarding the optimization of electric earthwork processes. Furthermore, this project adds to the field of research aimed at the development of real-world applications of reinforcement learning-based systems by bridging the gap between computer science and energy engineering.

Acknowledgments

I would like to sincerely thank my supervisors Prof. Dr. Frank Behrendt and Prof. Dr.-Ing. Nicole Riediger for granting me the opportunity to pursue this thesis. Thank you for your encouragement, patience, and commitment. I would not have been able to achieve my dream of developing a doctoral thesis without your vast knowledge, guidance, and support.

Thank you for coming together and establishing our research group together with Nico Dabelstein, Ira Lemm, Olga Tcvetkova, and Dr. Zsuzsa Besenyöi. Being part of this research group has significantly enhanced my research experience by allowing me to learn and evolve together with my peers.

I would like to express a special thanks to Nico Dabelstein. You are the reason I got the chance to pursue a doctoral dissertation, and for that I am forever thankful. Thank you for teaching me, motivating me, and hosting me in Berlin. I would also like to thank Dr. Christoph Banhardt for inspiring me to pursue a doctorate. Your positive attitude and endless creativity motivate me to face new challenges with a smile.

In addition, I would like to thank the administrative staff and students at Campus El Gouna for providing me with an inspiring environment where I discovered my passion for research. Furthermore, I would like to thank Aktiv Veidrift AS and Nordic Booster AS for sharing experiences and field data with this project.

Finally, endless thanks to my family and friends for supporting me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Annual emissions of 315 million tons of CO_2 are directly associated with the combustion of fossil fuels on construction sites [2]. The main source of these emissions are diesel-powered earthmoving machines [3]. The earthwork industry is incentivized to reduce emissions in response to volatile fuel prices, carbon pricing, and new legislation that requires earthwork contractors to reduce emissions to win public contracts [4].

Electrification of earthmoving machines is generally considered the solution to achieve zero-emission earthwork sites. However, the management of energy supply has been identified as the main barrier in the transition from diesel-powered to electric earthmoving machines [5]. Electric earthmoving machines have internal battery systems that require significantly more time to recharge compared to refueling a diesel-powered machine. Efficient charging of electric earthmoving machinery can only be achieved with a purpose-built high-power charging infrastructure. The earthwork industry is looking to battery energy storage systems (BESS) to provide efficient high-power charging while avoiding unnecessary upgrades to the distribution network. However, a challenge still remains to optimize the energy supply schedule of electric earthwork processes with a BESS as an interface to avoid scenarios in which earthwork processes are halted due to recharging of the BESS.

The uncertainty surrounding the factors that influence the energy consumption of earthmoving machines is identified as the main challenge in optimizing any earthwork process because earthmoving machines are exposed to varying operating conditions, fleet configurations, and earth materials. Limited attention has been paid to exploring uncertainties in the earthwork literature [6–8]. Therefore, optimization of earthwork processes is an underexplored and underspecified area of research. Furthermore, previous modeling efforts of earthwork processes are based on detailed and specific case study data sets and offer limited or no accuracy when applied elsewhere [6]. This is mainly due to the lack of standardized data collection practices in the earthwork sector. Therefore, a demand for a generalizable earthwork optimization framework that can handle uncertainties when applied to varying earthwork configurations was identified.

This thesis proposes a deep reinforcement learning framework designed to simulate and optimize several earthwork processes, configurations, and scenarios that contain a high level of uncertainty. The purpose of this framework is to assist earthwork contractors in the transition from diesel-powered to electric earthwork processes by identifying routines to improve the efficiency of electric earthwork processes. Furthermore, by simulating and demonstrating how an electric earthwork process will satisfy regulations and meet efficiency targets, earthwork contractors can use this framework in planning and bidding processes, because current practice relies on subjective estimates based on experience [6]. The framework is applied to a case study to validate its performance in a real application and to demonstrate its ability to improve the efficiency of a case study earthwork process.

1.1 Knowledge Gap and Scope

The field of electrified earthwork contains significant knowledge gaps because electrification of earthwork processes is a recent advancement in the industry. The first series of commercially available battery-powered excavators was developed in 2018 [9], the first commercially available earthwork BESS was developed in 2020 [10], and the first zero-emission earthwork projects were commenced in Norway during the second half of 2021 [11]. Therefore, no methods have been developed to optimize the scheduling of BESS operations in coordination with multiple electric earthmoving machines because limited data sets and few related research projects are available.

The scope of this study is designed to partially fill the knowledge gap on optimizing earthwork processes with a BESS as an interface between the grid and electric earthmoving machines. To fill this knowledge gap, literature on diesel-powered earthwork processes and previous modeling efforts are reviewed together with battery optimization techniques to develop an optimization framework.

Simulation models for excavators, loaders, and haul trucks are developed, as they are the most widely represented earthmoving machines in the literature [6]. However, the code for this framework is designed to be modular and to accommodate simulation models for other earthmoving machines and earthwork processes in the future. Other researchers are encouraged to expand on this work because earthwork processes are similar in most infrastructure and mining projects. Furthermore, this work bridges the gap between computer science and energy engineering by demonstrating how deep reinforcement learning can be applied to solve a real-world problem.

1.2 Research Questions and Objectives

The main research question for this project is as follows.

> *How can an earthwork BESS be operated to optimize the energy supply for electric earthwork processes?*

To answer this question, several underlying research questions were identified.

Research Questions

- How can an earthwork BESS make electric earthwork processes more feasible?
- What factors affect the energy consumption of electric earthmoving machines?
- Which mathematical framework is best suited to optimize electric earthwork processes?
- How can electric earthwork processes be optimized to increase efficiency?
- How can electric earthwork processes be optimized to reduce electricity costs?
- What impact does electrification of earthmoving machines have on earthwork processes?

To answer these research questions, the fulfillment of several underlying and ordered objectives is required. The objectives presented below provide direction and summarize the accomplishments the author aims to achieve throughout this thesis.

Objectives

1. Explore the roles and responsibilities of a BESS in the electric earthwork sector.
2. Review existing earthwork literature to explore how factors that influence the fuel consumption of diesel-powered earthmoving machines can impact the energy consumption of electric earthmoving machines.
3. Review previous modeling efforts to establish what mathematical framework is best suited to optimize electric earthwork processes.
4. Review state of the art modeling efforts in related fields to explore what methods and approaches are suitable for this project.
5. Develop a simulation model that represents the dynamic operations of a BESS in coordination with electric earthmoving machines.
6. Develop methods to assess the impact of factors that influence the energy consumption of electric earthmoving machines.
7. Develop an optimization framework that incorporates the simulation model and the methods developed.
8. Make the framework scalable and generalizable to various earthwork scenarios.
9. Explore different use cases for the optimization framework.
10. Apply the optimization framework to a case study to demonstrate its performance in optimizing an electric earthwork process.

1.3 Justification

The sustainable development of cities plays a key role in the transition to fossil-free energy consumption, because cities influence approximately 70% of global emissions [4]. Therefore, the transition from diesel to battery-powered earthmoving machines is an important contribution to reducing emissions, as explained in Chapter 2.

Furthermore, this project is justified by its applicability to several of the Sustainable Development Goals (SDGs) developed by the United Nations (UN) that were adopted by all UN member states in 2015. The SDGs are a collection of 17 interlinked global goals designed to achieve a better and more sustainable future for countries by 2030 [12].

The goals and targets to which this project directly applies are presented in numerical order below.

- Goal 7: Ensure access to affordable, reliable, sustainable and modern energy for all.
  - Target 7.3: By 2030, double the global rate of improvement in energy efficiency.
- Goal 9: Build resilient infrastructure, promote inclusive and sustainable industrialization and foster innovation
  - Target 9.4: By 2030, upgrade infrastructure and retrofit industries to make them sustainable, with increased resource-use efficiency and greater adoption of clean and environmentally sound technologies and industrial processes, with all countries taking action in accordance with their respective capabilities.
- Goal 11: Make cities and human settlements inclusive, safe, resilient, and sustainable.
  - Target 11.3: By 2030, enhance inclusive and sustainable urbanization and capacity for participatory, integrated and sustainable human settlement planning and management in all countries.
  - Target 11.6: By 2030, reduce the adverse per capita environmental impact of cities, including by paying special attention to air quality and municipal and other waste management.
- Goal 12: Ensure sustainable consumption and production patterns.
  - Target 12.2: By 2030, achieve the sustainable management and efficient use of natural resources

1.4 Structure of Thesis

The structure of this thesis is organized into eight additional chapters.

The second chapter provides background information on the ongoing electrification of the earthwork sector and outlines the applicability of BESS technology in the electrified earthwork sector.

The third chapter is a literature review of previous earthwork studies and modeling efforts.

The fourth chapter introduces the chosen mathematical framework and explains its fundamental operations and equations.

The fifth chapter is a review of state-of-the-art applications that utilize the chosen mathematical framework to solve related research problems that this project can benefit from.

The sixth chapter details the proposed methodology that was developed to answer the research questions.

The seventh chapter presents the results of the proposed methodology and its ability to answer the research questions.

The eight chapter is a case study application of the proposed methodology to exhibit its applicability to real-world implementations.

The ninth and final chapter concludes the project by summarizing how the proposed methodology was able to answer the research questions and fulfill the objectives.

Chapter 2

Background

The purpose of this chapter is to provide background information concerning the electrification process and the emergence of BESS technology in the earthwork sector.

2.1 Introduction to Earthwork

Earthwork is a term used to describe the processing of earth materials. Earthwork generally involves the loosening, removing, and handling of earth materials on construction sites. The majority of earthwork processes are performed using heavy duty earthmoving machines, such as excavators, loaders, bulldozers, and dump trucks. Earthwork processes can vary from site to site depending on the fleet configuration and local ground conditions. However, a typical earthwork process consists of an excavator excavating and loading earth material onto a dump truck that transports it to an external dump site before returning to the earthwork site [6]. Furthermore, earthwork is a significant contributor to local and global greenhouse gas emissions because most earthmoving machines are fueled by diesel, which emits approximately 2.7 kilograms (kg) of carbon dioxide equivalents (CO_2e) per liter consumed [13].

2.1.1 Emissions from the earthwork Sector

The construction industry accounts for approximately 23% of global carbon dioxide (CO_2) emissions, where 5.5% of these emissions (315 million tons of CO_2) are directly associated with the combustion of fossil fuels by earthmoving machines and other equipment on construction sites [2]. These numbers are expected to increase due to urbanization and the expansion of cities [4].

Det Norske Veritas (DNV), an international accredited registrar and classification society, estimates that annual emissions from construction sites in C40 cities, which represent almost 10% of the world's population and a quarter of the global economy, are in the range of 120 to 240 megatons of carbon dioxide equivalents ($MtCO_2e$), and between 5% and 10% of the total emissions of individual cities [5]. Furthermore, the total emissions of earthmoving machines in the European Union (EU) was estimated to be $40MtCO_2e$ in 2019, which was higher than the total emissions of eight member countries [4]. Therefore, earthworks processes and earthmoving machines are a significant source of greenhouse gas emissions.

DNV approximates that diesel-powered earthmoving machines have an average carbon footprint of $37kg\ CO_2e$ per square meter of the construction site throughout the construction phase [4]. This average carbon footprint can reach $51kg\ CO_2e$ during the earthwork subphase [14]. However, approximation and reduction of the carbon footprint of earthmoving machines during the earthwork subphase have received limited attention in the literature because earthwork emissions represent a relatively small amount compared to emissions from the operational phase of a vertical construction project [6]. Therefore, a disproportionate amount of attention has been paid to research on reducing emissions from the operational phase of buildings.

However, the earthwork subphase of horizontal infrastructure projects is consistently identified as one of the main sources of emissions [6, 15–17]. In road construction, the earthwork subphase is responsible for between 14-45% of the total emissions, and the lack of adequate methods to estimate earthwork emissions has resulted in estimates that are more than 300% higher than the base case [6]. Therefore, more research should focus on the factors that affect earthwork emissions.

Furthermore, cities with a higher share of construction activity face increased challenges with local area pollutants [5]. These pollutants include carbon monoxide (CO), hydrocarbons (HC), nitrous oxides (NOx), and other particulate matter (PM). Increased concentrations of these pollutants place a burden on public health systems, as they are a known contributor to acute lower respiratory infections, chronic obstructive pulmonary disease, lung cancer, ischemic heart disease, and stroke [15]. Therefore, reducing the combustion of fossil fuels in earthmoving machines can have a significant impact on the health and well-being of people living adjacent to construction sites [6]. In recent years, efforts have been made to reduce emissions from earthmoving machines by imposing emission standards and mandating the use of filters in the EU [4]. However, it is impossible to completely eliminate emissions while using fossil fuels.

2.1.2 Transition to Electric Earthmoving Machines

Electrification of earthmoving machines is generally considered the solution to eliminate local air and noise pollution from construction sites [5]. Earthwork contractors are incentivized to invest in electric earthmoving machines due to increasingly volatile fuel prices, carbon pricing, and new regulations that require earthwork contractors to reduce their emissions to win public contracts [4]. However, there is a low demand for commercially available electric earthmoving machines due to comparably higher capital costs resulting from immature markets and high battery prices [5]. Currently, the capital expense of a new electric earthmoving machine is approximately 20% higher than that of an equivalent diesel-powered earthmoving machine [18]. However, the operating costs of electric machinery are generally lower because electric engines are more energy efficient and have fewer moving parts than internal combustion engines. Furthermore, it is expected that the operating lifetime of electric earthmoving machines will be 50% longer than a diesel-powered equivalent [3].

DNV found that electric excavators are more fuel efficient than diesel-powered equivalents by comparing fuel consumption over an operational lifetime of 15,000 hours. They found that a 25-ton electric excavator can consume up to 2 gigawatt hours (GWh) of energy (284,000 liters of diesel) less than a comparable diesel-powered excavator [3]. Furthermore, DNV estimates that substituting a 25-ton diesel-powered excavator with an electric excavator can reduce fuel costs by between €100,000 and €240,000, depending on the price of diesel and electricity [18]. Thus, electric earthmoving machines are considered a viable business case in countries where the price of diesel is relatively high compared to the price of electricity.

The main challenge with regard to the transition to electric earthmoving machines is the energy supply. Electric earthmoving machines require up to several hours to recharge from a standard grid connection, while diesel-powered earthmoving machines can be refueled in minutes. Currently, the charging times recorded for an electric excavator range from 1 to 10 hours, depending on its battery capacity and its connection to the grid [18]. A purpose-built connection to the grid with a high-power transformer can reduce charging times to the one-hour range for most commercially available electric earthmoving machines [18].

These grid connections are expensive and permanent upgrades to the distribution network that will increase the peak demand from the construction site and often do not serve a purpose when the construction process is complete. Furthermore, it is also not possible to establish high-power grid connections in areas where the capacity of the distribution network is low. Therefore, digging to the nearest transition station may be required. This process is an additional cost and can be very time consuming for the construction process [1]. Therefore, the lack of access to adequate energy supply is the most common objection expressed by earthwork professionals concerning the transition to electric earthmoving machines [4].

The earthwork industry considers BESS as a potential solution to enable efficient charging of electric earthmoving machines without permanently overdimensioning the grid connection and avoiding peak demands [1]. A purpose-built BESS can charge from a standard grid connection and utilize its stored energy capacity to provide high-power charging to electric earthmoving machines. Thereby, reducing the potential increase in peak demand and the cost associated with the earthwork site's grid connection. Furthermore, a BESS can be reused in several earthwork projects and reduce the cost of electricity by performing energy arbitrage.

A review of current earthwork processes is required to facilitate the use of this technology because new routines, flexible scheduling, and coordination of smart operations are necessary to achieve efficient electric earthwork processes [1]. Therefore, there is a demand to develop systems that efficiently meet the energy supply needs of the earthwork process without overinvesting in infrastructure [5]. There are still many unanswered questions concerning the operation, size, and services that an earthwork BESS should provide. Therefore, background information on battery systems is provided in the following sections to inform the reader about the technology.

2.2 Introduction to Battery Cell Technology

A BESS is an energy storage system that uses rechargeable electrochemical battery cells to store and dispatch energy. Battery cells are individual electrochemical units that can store chemical energy and convert it to electrical energy. A battery cell consists of two electrodes, an electrolyte, and a semipermeable barrier separating the electrodes. The electrodes are made from different types of conducting materials, where one of the electrodes must have a stronger standard potential than the other to maintain a reduction-oxidation reaction when the electrodes are connected via an external circuit. The electrode that oxidizes and releases electrons is called the anode, and the electrode that is reduced by attracting electrons is called the cathode.

When a battery cell is discharged, electrons from the anode are attracted to the cathode due to its higher standard potential. This attraction forces the anode to oxidize by releasing its valence electron(s) and ions. Negatively charged electrons flow through an external circuit to reduce the positively charged cathode. The difference between the standard potentials of the electrodes determines the force that moves electrons between the anode and the cathode. This is referred to as the electrical potential or voltage of a battery cell, which is measured in volts. Furthermore, the number of electrons that are passing through a point of the external circuit at any given time is the battery cell's current, which is measured in amperes. The maximum amount of current that can safely flow through the external circuit in one hour is the battery cell's capacity, which is measured in ampere-hours.

To balance the charge and prevent the discharge reaction from ceasing, positively charged ions at the anode are attracted by the increasingly negative charge of the cathode. Ions travel through the electrolyte, which is made of a substance that allows the free movement of ions, and a semi-permeable barrier, which is only permeable for electrolytes and ions, prevents electrons from moving through the electrolyte and short-circuiting the battery cell. The discharge reaction stops when the external circuit is no longer connecting the electrodes or when the anode depletes its supply of electrons and ions. If the electrode materials allow for the reverse flow of electrons and ions, the battery cell can be recharged.

The recharging process occurs when an external power source that produces a voltage higher than that of the battery cell's open-circuit voltage is connected to the external circuit of the electrodes. This process reverses the electrochemical reaction that occurred during discharge because the external energy source applies a higher force to the flow of electrons in the opposite direction, forcing the electrons and ions in the original cathode to return to the original anode. However, this replacement process is not perfect and degradation of the battery cell will occur.

2.2.1 Battery Cell Degradation

The performance of a battery cell is reduced after every discharge and charge cycle because the crystal structure of the electrodes becomes less ordered in the replacement process. The cycle life of a battery cell is an indication of how many charge cycles it can endure without a significant loss of performance. Deeper discharges reduce the cycle life faster because more charging is required. Therefore, a battery cell that discharges 20% of its capacity has a much longer cycle life than a battery that discharges 80% of its capacity [19].

Furthermore, chemical by-products are produced during the reduction-oxidation reaction. These byproducts build up over time within the cell and cause irreversible plating on the negative electrode that increases the internal resistance of the battery cell. Independently of the reduction-oxidation reaction, a rechargeable battery cell will experience self-discharge, which are internal chemical reactions that result in a loss of chemical energy and open-circuit voltage regardless of the electrodes being connected.

The occurrence and significance of the degradation reactions mentioned above is amplified if the battery cell is operated beyond its safe operating limits. If a battery cell is discharged beyond its minimum energy capacity, the voltage of a battery cell will drop below its safe operating voltage. Low voltage will amplify the production of aforementioned by-products, increasing the internal resistance, which can lead to short-circuiting the battery cell. This condition is called overdischarge. Overdischarge also has a significant negative impact on the crystal structure of the electrodes, making it more energy intensive to recharge the battery cell.

When recharging, overcharge can occur if the voltage of the external power source is greater than the safe operating voltage range of the battery cell. Overcharge is a term used to describe when current continues to flow into the battery cell after it has reached its maximum capacity. Excess current causes an increase in the internal temperature, which will degrade the electrolyte by evaporation. If there is effectively no barrier between the positive and negative electrodes due to electrolyte degradation, the chemical reaction will speed up uncontrollably, causing a rapid increase of temperature and an uncontrollable release of energy. This reaction is known as thermal runaway. Furthermore, if the battery cell is operated under conditions where the internal temperature of the battery cell is below a safe operating temperature range, the movement of ions will slow down and lead to irreversible plating on the negative electrode. Therefore, the operating temperature of the battery cell has a significant impact on the cycle life.

The safe operating ranges of a battery cell are defined by the chemistry coupling of the electrodes and electrolyte. Therefore, the electrochemical properties of the materials used in a battery cell affect its performance.

2.2.2 Lithium-ion Battery Cells

Lithium is the metal of preference for battery cell applications because it has the highest electrochemical potential of all metals. This is in part due to the presence of only one valence electron in the outer shell of a lithium atom. Lithium cobalt oxide ($LiCoO_2$), lithium iron phosphate ($LiFePO_4$), and lithium manganese oxide ($LiMn_2O_4$) are the preferred cathode material for most commercially available lithium-ion battery cells [20]. Graphite is the preferred material for the anode, and the electrolyte generally consists of a polymer solution or a lithium salt, like lithium hexafluorophosphate ($LiPF6$).

The emergence of lithium-ion battery cells has made BESS the leading energy storage technology [21]. Commercially available lithium battery cells have a round-trip efficiency of up to 96% [22], which means that up to 96% of the energy charged to the battery cell is available for discharge. A lithium-ion battery cell can retain its charge up to 99% over 24 hours [23] and only self-discharge between 3% and 10% in a month depending on its cycle life [24]. Lithium-ion battery cells have a near-instant response time and ramp rate, which means that they can go from standby to providing power at a desired level at an almost instant rate. Furthermore, lithium-ion battery cells have a higher energy and power density than other commercially available battery cell technologies, ranging from 100 to 265 watt-hours (Wh) per kilogram [24].

2.2.3 Battery Cell Configuration

The configuration of the battery cells determines the voltage and capacity of a BESS. The voltage of a BESS is the total voltage of battery cells that are connected in series because of the additive effect that equals the total force at which electrons move from the anode in the first cell to the cathode in the final cell. The capacity of a BESS is the accumulated capacity of battery cells that are connected in parallel because it increases the possible current, which is the total number of electrons flowing through the battery cells.

Therefore, the configuration of battery cells in a BESS determines the energy and power rating of the BESS. Energy can be defined as the work capacity of the BESS, which is measured in watt-hours (Wh). Power can be defined as the rate at which the work is performed by the BESS, which is measured in Watts (W).

The configuration of the battery cells is modular and provides scalable opportunities for BESS applications [21]. The configuration of battery cells in a BESS is based on the system-specific characteristics of its intended application [25].

2.3 BESS Applications in the Energy System

A BESS is able to provide several services on the supply and demand side of an energy system because it can be considered both a generator and a load. The market applications of a BESS are differentiated as Front-of-the-Meter or Behind-the-Meter.

2.3.1 Front-of-the-Meter Applications

Front-of-the-Meter applications refer to the ancillary services a BESS can provide on the supply side of the energy system. The purpose of these applications is to maintain grid stability and defer system upgrades. In combination with generation assets, a BESS can provide frequency regulation by correcting frequency imbalances in the grid [25]. A BESS can also serve as an operating reserve by providing a continuous power and energy supply if there is a disruption of the generation asset.

A BESS can defer upgrades of the generation capacity by utilizing stored energy for peak power generation. In addition, a BESS can defer upgrades to transmission and distribution networks by providing or storing excess energy in areas that experience congestion [25].

BESS technology plays a key role in the global transition to a sustainable energy system by increasing the flexibility of a grid to accommodate a higher share of fluctuating and non-dispatchable renewable energy generation [26]. A Front-of-the-Meter BESS can regulate frequency and reduce transmission capacity by storing excess renewable energy generated [25].

2.3.2 Behind-the-Meter Applications

Behind-the-Meter applications refer to the various demand response services that a BESS can provide on the demand side of an energy system. The purpose of demand response is to reduce the difference between peak and average load demand levels. Therefore, the main application of a Behind-the-Meter BESS is to serve as a tool for altering customer's energy consumption patterns in response to governmental policies or monetary incentives [27].

A Behind-the-Meter BESS provides customers with the flexibility to shift loads from peak periods to off-peak periods by charging the BESS during periods of low demand and storing this energy for use when demand increases. Tariff rates are often designed to induce this behavior by charging less for electricity during periods of low demand compared to periods of high demand. Therefore, a Behind-the-Meter BESS can be used to perform energy arbitrage by charging and storing electricity when the prices are low for utilization or selling back to the grid when the prices of electricity are comparatively higher.

In combination with a renewable generation asset, a Behind-the-Meter BESS can increase a customer's self-consumption of renewable energy by storing excess generated electricity during periods of renewable generation and using the stored renewable energy when the renewable generation capacity drops. Thus, reducing the customer's climate footprint.

2.3.3 BESS in the Earthwork Sector

In 2021, BESS technology was introduced to the earthwork sector to enable efficient charging of electric earthmoving machines in earthwork projects in Norway. The prototypes developed for the earthwork sector are encased in a standard-sized shipping container for mobility purposes, and the battery system has an integrated multi-interfaced charging system to ensure that the majority of commercially available electric earthmoving machine can charge without providing its own charging solution.

Figure 2.1 shows a 3D-rendering of a purpose-built BESS for the earthwork sector.

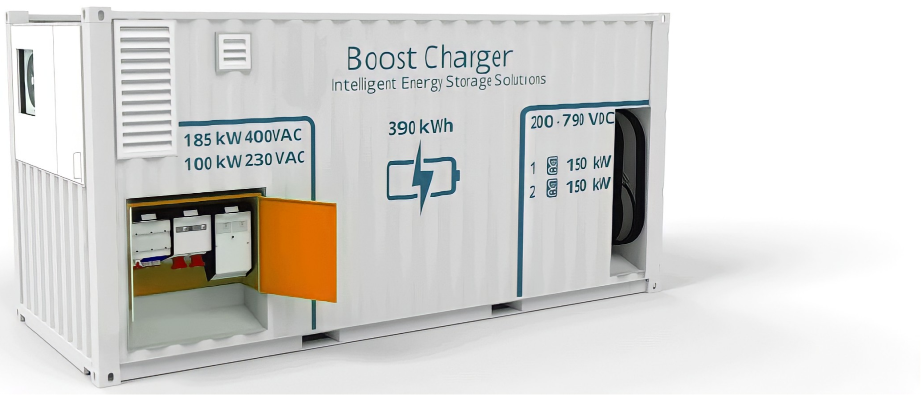


Figure 2.1: *3D rendering of a prototype BESS*

This example is called BoostCharger and it is a prototype developed during 2020 by a Norwegian company called Nordic Booster AS that specializes in earthwork BESS technology. The BoostCharger is designed to meet all electricity demands on a zero-emission earthwork site, making fossil fuel generators redundant. It has a battery energy storage capacity of 390 kilowatt-hours (kWh), several alternating current (AC) connection points, and two direct current (DC) charging cables capable of charging two electric vehicles at a rate of up to $150kW$, simultaneously. How the power electronics are configured in this BESS is beyond the scope of this project. However, the BoostCharger combines galvanic plating with two DC/DC transformers connected in series, where the first transformer raises the voltage to a high and stable level and the second reduces the voltage to the variable voltage of the vehicles during charging.

Figure 2.2 below shows a different example of a BESS prototype called SuperCharge, which is delivered with solar panels to increase the self-consumption of renewable energy generation.

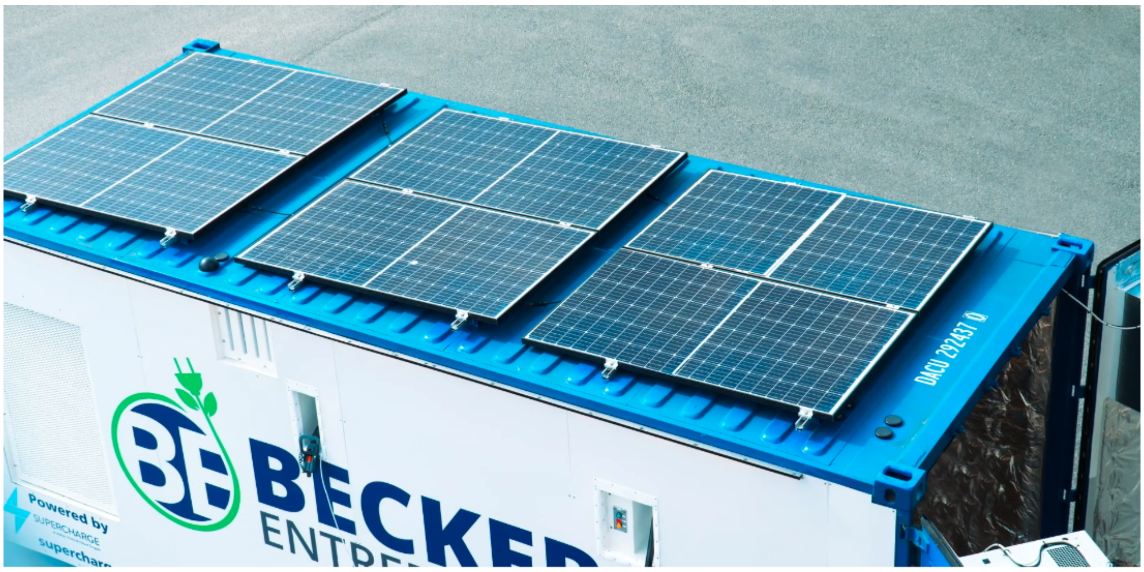


Figure 2.2: *Solar panels on construction site BESS*

When solar panels are used, the carbon footprint of any earthwork project can be reduced. Moreover, by charging the BESS during times of high renewable share in the grid's energy mix, the carbon footprint can be further reduced. Furthermore, SuperCharge is a good example of how the controls of an earthwork BESS are configured. Figure 2.3 below shows an example of the controls to discharge SuperCharge to an electric earthmoving machine.

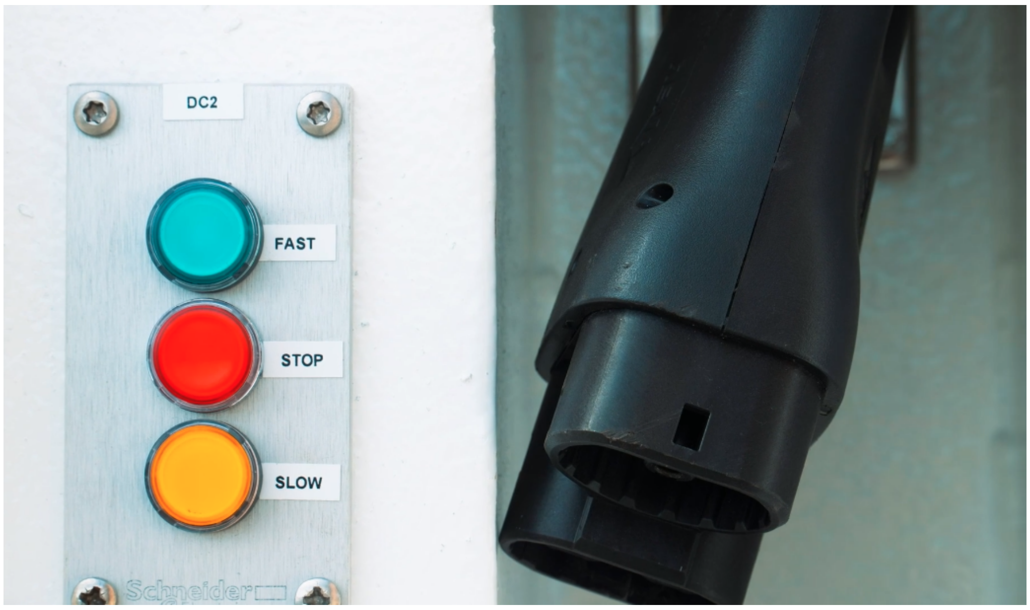


Figure 2.3: *The controls of construction site BESS*

The Supercharge user has the choice between high-power charge (fast), grid-power charge (slow), and stop charging (stop). These controls show how simple it can be to operate the BESS. However, several challenges are identified with respect to the scheduling of operations related to the use of BESS technology on an earthwork site.

2.3.4 Challenges with Earthwork BESS

An earthwork BESS is an intermediate system between the grid and the electric earthmoving machines. It can discharge its stored energy faster than it can recharge itself. Therefore, a challenge arises with scheduling the charging and discharging of the BESS to avoid situations where the earthwork process is halted due to the recharging of the BESS

Optimizing the scheduling for charging and discharging an earthwork BESS has not been possible because methods for estimating future load demands of a electric earthmoving machines have not been developed. This is in part due to the fact that every earthwork project is unique and the energy demand from earthmoving machines is significantly affected by local ground conditions.

The current industry practice, as recorded in the case study in this project presented in Chapter 8, is to constantly charge the BESS when it is not used to charge earthmoving machines. The charging of the earthmoving machines often occurred during lunch breaks. However, lunch breaks were often extended to fully recharge the machines. Moreover, the workday would be extended if additional charging would be required after the lunch break. This was identified as major concern by the case study earthwork contractor.

Furthermore, estimating the optimal energy capacity for a earthwork BESS has also not been possible because it is dependent on the optimal operation schedule. If the capacity is too low, the construction process is slowed down due to situations where the energy demand is greater than the available supply from the BESS. Therefore, the initial commercially available earthwork BESS' have been delivered with a large energy capacity of more than $300kWh$. However, if the capacity is too high, the financial burden of commercially available BESS excludes most earthwork contractors from transitioning to electric construction machines.

DNV has conducted several interviews with earthwork contractors trying to achieve electric earthwork sites, and their conclusion states that optimizing the logistics of earthwork processes is the key to achieving zero-emission earthwork sites [1]. Therefore, the main objective of this project is to develop a methodology to optimize the scheduling of charging and discharging an earthwork BESS to efficiently supply energy to earthmoving machines.

Chapter 3

Literature review

A literature review of academic works was conducted to identify, examine, and expand on previous efforts in the earthwork literature. Electrification is a recent development in the earthwork sector and is currently not comprehensively researched. Methods to optimize energy supply in electric earthwork processes have not yet been developed. However, tracking and modeling fuel consumption during traditional earthwork processes has received increasing attention in the last decade [6]. Therefore, the findings of previous modeling efforts earthwork literature are reviewed together with optimization methodologies in related fields to serve as a basis for the development of a methodology that can fill the knowledge gap on how to optimize the energy supply in electric earthwork processes.

This review consists of two parts. The first part is a review of the factors that affect the fuel consumption and productivity of diesel-powered earthmoving machines with the aim of identifying their application to the modeling of electric earthmoving machines. The second part is a review of previous modeling efforts to justify the selection of reinforcement learning as a mathematical framework for this project.

3.1 Review process

The proposed review process follows a systemic approach with four steps.

The first step is to identify and define keywords that allow the discovery of relevant literature. Primary keywords describe the field of research and were identified as earthwork, construction, earthmoving, excavating, battery, and energy storage. Secondary keywords describe outcomes of interest and were identified as zero-emission, emission, fuel consumption, fuel use, energy consumption, and energy management.

Tertiary keywords describe analysis methods and were identified as optimization, estimation, scheduling, modeling, simulation, minimization, and reinforcement learning.

The second step is to identify the relevant literature based on keyword search results from previous reviews [6] and major academic databases such as ScienceDirect and Google Scholar. Primary keywords are combined with secondary and tertiary keywords to filter the resulting literature. Peer-reviewed journals, conference proceedings, graduate theses, and industry reports were identified as relevant for this project.

The third step is to examine and filter the resulting literature to discover relevant findings and methods that are applicable to this project. In this step, the literature that does not contain relevant findings or methods are discarded from the review process.

The final step is data extraction, where relevant findings and methods are analyzed. Relevant findings are extracted, and the advantages, disadvantages, and practicality of the proposed methods are discussed and evaluated.

3.2 Factors affecting fuel consumption

To accurately estimate and optimize energy consumption in earthwork processes, a detailed understanding of the factors that affect the energy consumption of earthmoving machines is required [28]. In the existing literature, productivity and emission factors have been derived from field studies [29,30], regulatory bodies [31–33], or machine manufacturers' handbooks [34, 35]. However, publicly available data from field studies or regulatory bodies on electric earthwork processes were not discovered, and manufacturers of electric earthmoving machines have not yet developed detailed knowledge on how various factors impact the energy consumption of electric earthmoving machines. Therefore, the identified factors that affect the fuel consumption of diesel-powered earthmoving machines will be investigated to develop an approach to estimate the energy consumption of electric earthmoving machines. These factors are classified as machine factors, operational factors, site factors, or soil factors.

3.2.1 Machine Factors

Machine factors are factors related to earthmoving machines. Engine power is the factor most commonly used to model the fuel consumption of earthmoving machines [6]. It is a measure of the maximum potential output energy of an engine and has the highest correlation with fuel consumption among static engine variables [36, 37]. However, different tasks require different levels of engine power. A load factor defines the amount of engine power that is used during specific tasks, such as digging, loading, or idling. An average load factor is often used to model specific earthmoving tasks because it can be a highly dynamic variable when the earthmoving machine is active [6, 13].

Manufacturers of earthmoving machines typically specify the power and load factors of specific tasks in machine manuals. However, the manufacturers of electric earthmoving machines have not yet recorded enough data to present this information. The traditional approach to identifying the load factor of an earthmoving machine is to measure the absolute pressure of its engine manifold while performing a task [13]. However, electric engines do not have an engine manifold. Therefore, the load factors of electric earthmoving machines must be identified using field analysis or simulation, which has been proven to accurately identify the load factors of diesel-powered earthmoving machines in the past [38].

To estimate the total energy consumption for completing a task, the load factor is multiplied by the amount of time required to complete the task. The duration of a task depends on the number of activity cycles required. Therefore, the duration of a task is broken down into cycle time, which is a measure of the time required to complete a single cycle of a distinctive earthmoving activity [6]. However, the load factor and the cycle time depend on several underlying factors.

Attachment selection is one of the key underlying machine factors, as the selection of attachments has a great impact on the productivity and energy consumption of earthmoving machines. There exist several buckets or blade attachments that affect the effectiveness of performing tasks and the volume of soil that can be moved per cycle. For excavators, larger buckets generally result in reduced fuel consumption even though the cycle time may be longer, as fewer cycles are required to complete a task [6, 34].

Furthermore, maintenance of earthmoving machines can mitigate engine degradation that has a negative impact on fuel efficiency [39]. The most common maintenance approaches in the earthwork domain are run-to-failure maintenance or preventive maintenance [40]. However, electric engines have fewer moving parts and require less maintenance than combustion engines [18]. Therefore, maintenance is not considered in this study.

3.2.2 Operational factors

Operational factors are factors for which machine operators or earthworks contractors have direct decision-making authority. The experience and routines of the machine operators have a significant impact on fuel use. For excavating, the selection of methods such as two-tier excavation over slope-toe excavation, reducing engine speed, and positioning the excavator to reduce the swing angle have been reported to increase fuel efficiency by up to 8%, 14%, and 3% respectively [35]. For loaders, reducing the distance by minimizing the v-angle between the loader's direction and the truck has been reported to increase fuel efficiency by up to 16% [35]. For trucks, coasting or driving at constant speed have been reported to reduce fuel consumption by up to 38% and 16%, respectively [35].

These results apply only to their respective case studies. However, they emphasize that the skill level of machine operators has a significant impact on fuel consumption in relation to productivity. Experienced operators of excavators have been recorded to reduce cycle time and complete tasks twice as quickly as novice operators [41,42]. Moreover, a study on loaders found that experienced operators could be two to three times more fuel efficient than novice operators, while being significantly more productive [43]. Productivity factors that represent excellent, average, and novice operators [1.0, 0.75, 0.6] have been provided in the manufacturer's handbooks to indicate its impact on productivity and subsequently fuel consumption [34]. Therefore, operational factors are implemented in this project to account for earthmoving techniques and operator skill levels.

Furthermore, machine operators and site managers should ensure that trucks are loaded to their maximum capacity, as this can have a large impact on the fuel consumption per m^3 of the soil moved. A study recorded that fuel consumption could increase by up to 32% if trucks were only loaded to 67% of their maximum load capacity [44]. The same study recorded that fuel consumption could increase by up to 120% if truck operators were instructed to leave when another truck arrived. Queuing theory has been applied to estimate the production rate and the queue lengths of earthwork scenarios [6, 45, 46]. These studies found that the fleet configuration was the main factor that affects the queue length, fuel efficiency, and idle time of earthmoving machines

Fleet configuration consists of selecting the number and type of earthmoving machine to be used in an earthwork process. This factor is especially relevant for electric earthwork sites with limited access to charging infrastructure, as queues for charging machines will delay the earthwork process. Furthermore, fleet configuration is the only factor that can produce significant precision between case studies in predicting energy consumption when modeled in isolation [6].

The final operational factor is task scheduling. Only one study was discovered that simulated the impacts on fuel consumption on earthwork task scheduling strategies. The results of this study revealed that the control strategy (where no action is taken when a project is behind schedule) required slightly less fuel, but more time than the catch-up strategy (where more resources are allocated to catch-up if the project had fallen behind schedule) or the crash strategy (where the goal is to stay ahead of schedule at all times) [47].

Task scheduling is especially important for this project because the main objective is to develop a methodology to optimize the scheduling of charging and discharging an earthwork BESS to efficiently supply energy to earthmoving machines. Several studies have revealed a strong correlation between project cost and fuel consumption [45,48–50]. Therefore, scheduling of BESS operations in response to electricity price signals should be investigated to explore its effect on productivity.

3.2.3 Site Factors

Site factors are factors related to the layout and location of the site that machine operators or earthworks contractors do not have direct decision-making authority over. The layout of the site can have a significant impact on fuel consumption and productivity. The average operational efficiency of earthmoving machines ranges between 85% and 65% [51]. However, this can be reduced to 50% in restricted and confined sites due to the lack of optimized movement patterns [52, 53]. These recorded levels of operational efficiency are useful for this project because they are not specific to the type of fuel and are validated by several studies.

The grade of the site has also been found to have an impact on fuel consumption [6]. Steep grades require more energy from trucks and loaders to transport soil to and from the site, and excavators are often required to perform more cut and fill operations to level the site [54]. However, site grade is not considered in this study due to the lack of quantitative data recorded for electric earthmoving machines.

Furthermore, the distance that trucks must travel to and from the site for material transport can be responsible for the majority of fuel consumption in an earthwork project [55]. The only factor an operator can control during transport is speed optimization, as this significantly influences fuel consumption during transport operations, as explained in the previous section [35,56]. Therefore, the hauling distance and the average hauling speed are important factors to include in this project.

3.2.4 Soil Factors

Soil factors refer to the impact of ground conditions on fuel consumption. The impact of soil types on the energy consumption of earthmoving machines has been explored and evaluated in several studies. Direct relationships have been established between soil density and load factor [28,36], emissions [54,57–59], and productivity [58,60,61]. According to these studies, an increase in soil density generally results in a linear increase in fuel consumption [6].

Denser soil types are generally found in deeper sections due to compaction, and several studies that modeled the fuel consumption of excavators found that a greater depth of excavation correlates with an increase in fuel consumption [36,57,59]. Moreover, a greater depth of excavation requires more repositioning and increased movement of the excavator boom to reach deeper sections, also affecting fuel consumption [6].

In addition, soil density can be affected by weather conditions, as rain can increase the density of certain types of soil. Weather conditions are a significant factor that can cause earthwork projects to experience delays and cost overruns [62–65]. However, only one study has been discovered that models the impact of weather on fuel consumption in earthwork processes [6]. Therefore, more research is required to address the impact of weather conditions on earthwork processes [6]. This is especially relevant for electric earthmoving machines because temperature can have a significant effect on battery cells, as explained in the previous chapter.

Furthermore, the grain size of the soil has a significant impact on the fill factor of the machine attachments. Larger grain size can result in lower fill factors due to a higher share of empty space between the particles, whereas soil types with smaller grain sizes can result in fill factors that exceed the rated fill capacity of the machine attachments. Therefore, materials with smaller grain sizes can reduce the number of cycles required to complete an earthwork task, thus reducing energy consumption. However, comparing studies that have modeled the impact of soil types on fuel consumption is challenging because qualitative descriptions of soil types are rarely associated with quantitative properties [6]. Therefore, discrepancies can be encountered when comparing models with identical input variables.

3.3 Quantifying the Impact on Fuel Consumption

Every construction site is unique and can vary significantly from site to site due to the range of influencing factors. Therefore, modeling energy consumption and supply using a single variable is not recommended. Previous modeling efforts have revealed several ways to estimate fuel consumption and productivity in earthwork processes. However, comparing the methodologies revealed that they would produce different results when identical input values were used [6]. Therefore, a review of selected methodologies is conducted to explore how previous studies have considered the impact of factors on the fuel consumption of earth moving machines.

According to Adrian Roy's review of factors affecting earthworks greenhouse gas emissions and fuel use [6], the most impactful factors are the engine power and the type of soil. However, their impacts are subject to great variation between various studies, highlighting that the impacts of other factors are still relevant and should be considered [6]. This section will review approaches related to excavators and dump trucks, as these are generally the most used earthwork equipment.

The volume-based equations developed by Hajji [59] are reviewed because the p-values, which express the statistical relationship between the predicted and actual results, were less than 0.0001. Indicating that the developed approach has a significant correlation with the actual fuel consumption of the investigated earthmoving machines.

Hajji divided the amount of soil moved, dumped, or excavated by the sum of several site factors and an intercept to define a load factor before multiplying this load factor by engine power, engine efficiency, and an emission factor. The most important aspects of Hajji's equations are extracted and expressed as follows.

$$Excavator\ Fuel\ Use = \frac{Q}{(Yi + fs - d + B + ft)} \cdot E_p \cdot BSFC \cdot TAF \tag{3.1}$$

$$Dump\ Truck\ Fuel\ Use = \frac{Q}{(Yi + C + S - D - t)} \cdot E_p \cdot BSFC \cdot TAF \tag{3.2}$$

Where Q is the volume of soil moved, dumped, or excavated in m^3, Yi is the resulting intercept of a case-specific regression analysis, fs is a soil factor, d is an excavating depth factor, B is the bucket capacity, ft is a case-specific excavator type factor, E_p is engine power, $BSFC$ is the brake specific fuel consumption of the engine type (fuel efficiency), TAF is a transient adjustment factor (used for modeling emissions), C is the maximum load capacity of the truck, S is hauling speed, D is the hauling distance, and t is the cycle time of the truck.

However, the accuracy of this approach is completely dependent on parameter results from case-specific multi-linear regression analysis, e.g. the soil factor had different quantitative values depending on the type of equipment used. Therefore, its usefulness to this project is limited to its use of different factors to define machine-specific load factors.

Jassim, Lu, and Olofsson, developed a different approach to estimate fuel consumption of earthmoving machines. Their approach also depends on results from a multi-linear regression analysis. However, their equations are more general because they used an artificial neural network to calculate the results. Their estimation equations are expressed as follows.

$$Excavator\ Fuel\ Use = \frac{S_{fc} \cdot E_p \cdot L_f \cdot E_{cf}}{P_{fuel}} \tag{3.3}$$

$$Dump\ Truck\ Fuel\ Use = \frac{S_{fc} \cdot H_{pt} \cdot E_{cf}}{P_{fuel} \cdot T_{pr}} \tag{3.4}$$

Where S_{fc} is a specific fuel consumption factor, L_f is a specific load factor, E_{cf} is a fuel-to-energy conversion factor (efficiency), P_{fuel} is the relative density of diesel fuel, T_{pr} is a productivity rate factor associated with the time required to move the soil per hour, and H_{pt} is a truck-specific load factor that incorporates engine power, hauling speed, site grade, rolling resistance, and truck weight.

This approach is also volume-based and dependent on specific parameter estimations resulting from a multi-linear regression model. However, it emphasizes that the accuracy can be derived from the product of the machine's engine power and load factor, given that the load factor is sufficiently specific to the task.

According to the most comprehensive data collection effort currently in the field of earthwork emission modeling, time-based models have been shown to be more accurate than volume-based models [37]. Therefore, the time-based estimation approach developed by Roy [6] is also reviewed. This approach is very specific to static values that were underspecified. Therefore, the following equations are simplified from its original state to improve the readability.

$$Excavator\ Fuel\ Use = C_t \cdot (E_p \cdot L_f \cdot fs) \cdot \eta \tag{3.5}$$

$$Dump\ Truck\ Fuel\ Use = Haul_t \cdot (G + fs) \cdot R_r \cdot (E_p \cdot L_f) \cdot \frac{\eta}{O_s} \tag{3.6}$$

Where C_t is the cycle time in minutes, η is an engine efficiency factor, $Haul_t$ is the duration of transport, G is a grade factor, R_r is a rolling resistance factor, and O_s is the operator's skill factor. Although this representation is heavily modified from its original state, it represents a time-based approach that is more useful to this project, as time-based scheduling of energy consumption and supply is required for the optimization process.

This time-based approach lacks quantification of equipment productivity. However, a productivity equation [35] can be used to complement the time-based fuel consumption equations. A productivity equation can be expressed as follows.

$$Productivity = q \cdot (60/C_t) \cdot OEE \tag{3.7}$$

Where q is the production of soil (m^3) per cycle, 60 represents the number of minutes in an hour, and OEE is the operating efficiency of the machine, which is a measure of the time spent active in relation to idling.

The approaches reviewed to estimate fuel consumption in earthwork processes reveal that specific values representing various factors generated from regression analysis of large and detailed data sets are required for accurate results. However, detailed and publicly available data sets for electric earthwork processes do not exist in the current state of the electric earthwork literature. Therefore, this study will focus on developing an optimization framework that can incorporate specific values that represent the various factors as they become available in the future.

3.3.1 Challenges with Data Collection

Previous literature on earthwork reveals a lack of quantity, quality, and uniformity of data due to the inherent challenges associated with data collection in the earthwork domain [6]. There are fewer opportunities to perform detailed measurements, as the duration of the earthwork subphase is generally much shorter than the operational phase of a construction project. Additionally, coordination and mobilization of measurement infrastructure and personnel in earthwork processes is challenging because the earthwork subphase is often initiated within a short time period after a contractor wins a bid. The data collection process is further complicated by variances in site conditions, fleet configurations, and subcontractors, which can have their own fuel sources and privacy concerns. Therefore, data dissemination is limited by decentralized data collection and privacy concerns [66, 67]

Various data collection methodologies have been developed [68–74]. However, these approaches reveal a lack of standardization for comprehensive data collection. Furthermore, additional costs are associated with the required measurement infrastructure and tracking personnel, and collecting a sufficient amount of data can lead to modifications of current practices [6].

Roy's recommendations for standardized data collection state that data should be collected directly from equipment reports and interviews with site supervisors regarding the type and capacity of machine attachments, the duration of duty cycles, the volume of soil moved in each duty cycle, engine power (total and during cycles), fuel consumption, active and idle time of equipment, soil density, soil fill factor, soil particle size, and weather data [6]. These recommendations are considered for the case study of this project.

3.4 Previous modeling efforts

3.4.1 Multi-linear Regression

Multi-linear regression (MLR) techniques have been shown to be useful in modeling the effect of various influencing factors on the fuel consumption of earthwork processes [13, 36, 59, 75]. However, these models are specific to their intended case study and do not offer generalized precision when applied to other case studies [76]. Furthermore, comparing the result of several MLR models [36, 59, 75] using identical input values results in inconsistent fuel consumption estimates because the models do not consider the same number of factors and the qualitative descriptions of the soil types in one model do not correspond to the properties of other models that state the same soil type [6].

According to Adrien Roy's review of factors affecting earthwork emissions and fuel use [6], there are no known attempts in which MLR has been used to model earthwork processes where several earthmoving machines interact. Therefore, MLR should not be applied to this project because interactions between electric earthmoving machines are required to optimize the electric earthwork process.

3.4.2 Supervised Machine Learning

The lack of standardization and consistency in data collection has delayed the introduction of data-intensive machine learning methods that have led to significant optimization advances in other sectors such as medicine and manufacturing [6]. However, methods have been developed that use artificial neural networks to estimate the fuel consumption and emissions of earthmoving machines [36, 77].

These supervised machine learning modeling efforts have only been shown to demonstrate precision in the scenarios for which they were designed [6]. This is due to the lack of available data from earthwork processes, as the precision of these methods depends on the accessibility of vast amounts of data detailing the correct behavior in various earthwork processes to provide generalization. Supervised machine learning would not be beneficial for this project, as recorded data from electric earthwork processes are currently scarce and lack the level of detail required for this optimization approach.

3.4.3 Discrete Event Simulation

Discrete event simulation (DES) is the most common method for modeling fuel consumption and emissions in earthwork processes with several interacting machines. This method is used in the field of earthwork because it can solve logically complex problems that involve a certain level of uncertainties and interdependent components [78]. It is also capable of generating multiple output signals, such as schedules, equipment utilization rate, critical path identification, and insights into how components and the system as a whole are affected by changes in input variables [38]. Therefore, different scenarios can be easily tested [79].

Several models have been developed to track and estimate emissions using discrete event simulation [38, 45, 48, 49, 80–85]. However, they are often case study specific. Other approaches have focused on modeling variations in load-haul configurations, where one or more loaders interact cyclically with several trucks that transport soil to a dump site [38, 48, 77, 78, 81, 82, 86]. However, these approaches are generally applied to a single case study in which different conditions are simulated.

This review has discovered only one generalizable discrete event simulation framework that has been developed for the earthwork process [6]. This approach is designed to estimate the emissions and fuel consumption of diesel-powered earthmoving machines. However, it does not offer an automatic optimization algorithm. Other studies have used the DES framework to generate various operating scenarios to manually discover the scenarios that produce the least emissions [77, 80, 87].

The greatest challenge with the DES approach is its inherent inability to estimate optimal solutions because it lacks an interface between the DES and an optimization algorithm that can perform actions as a result of the optimization process [88]. Therefore, this project cannot rely on DES, as the purpose is to develop optimal schedules.

3.4.4 Reinforcement learning

Reinforcement learning has been shown to be applicable to modeling earthwork processes [89–91]. The reinforcement learning approach is used to consider uncertainty when optimizing earthwork processes. Reinforcement learning does not rely on existing data sets of correct behavior, as it learns from its own experience in the simulation process. Recent advances in the field of reinforcement learning have led to the development of methods that provide a generalized response through the integration of artificial neural networks.

The reinforcement learning approach is suitable for this project because it can generalize and optimize electric earthwork processes while considering the uncertainty surrounding the influencing factors without relying on detailed data sets from electric earthwork processes. This project will utilize the temporal difference (TD) learning approach. TD learning is a combination of Monte Carlo simulation and dynamic programming. Similarly to Monte Carlo simulation, TD learning learns from experiences without requiring an accurate model of the system's dynamics or detailed data sets. However, TD learning uses a dynamic programming feature called bootstrapping or online learning in which it can update multiple decision variables with respect to the optimization goal after each simulation time step rather than waiting for the simulation to terminate [92].

The review did not discover any literature on how TD learning can be applied to optimize energy supply scheduling for earthwork processes. Therefore, the mathematical framework of reinforcement learning is explained in detail in Chapter 4 to inform the reader. Furthermore, a review of state of the art literature that has used reinforcement learning to solve similar problems is presented in Chapter 5 to highlight methodologies that are relevant to this project.

Chapter 4

Mathematical Framework

This chapter serves as a general introduction to the methods that make reinforcement learning suitable for this project. Reinforcement learning is selected as the mathematical framework for this project because of its ability to optimize dynamic problems with a high level of uncertainty and generalize its response to unseen scenarios. These features are required when developing a framework that can optimize the energy supply for electric earthwork processes, as the impact of influencing factors is underspecified, and earthwork processes can vary substantially from one earthwork site to another. Moreover, the lack of standardization in electric earthmoving machines and BESS technology further emphasize the need for a framework that handles uncertainties well.

4.1 Introduction to Reinforcement Learning

Reinforcement learning is considered the third paradigm of machine learning, together with supervised and unsupervised machine learning. Reinforcement learning is an iterative machine learning method to learn desired behaviors in sequential decision making. It is a computational approach to learn which actions yield the highest return when interacting with a dynamic system over time. The purpose of reinforcement learning is to develop optimal strategies to solve stochastic control tasks by trial and error. A reinforcement learning framework consists of an agent software that interacts with an environment software. The agent is a decision maker that develops optimal strategies to solve stochastic control tasks, whereas the environment is a demonstration of a specific stochastic control task. The agent interacts with the environment in discrete time steps by performing actions. Figure 4.1 shows the iterative process of reinforcement learning.

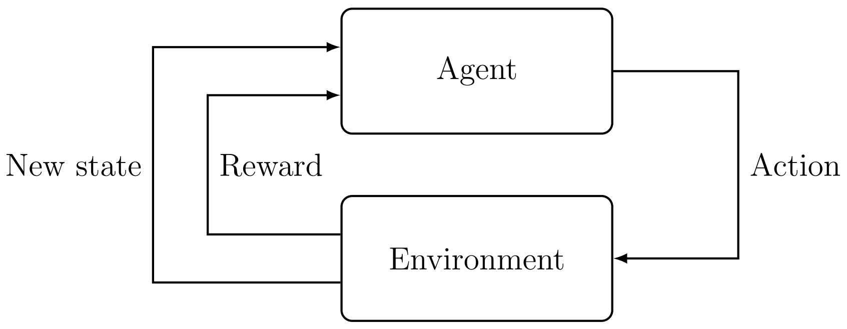


Figure 4.1: *The interaction process of an agent and an environment*

Following each action, the agent receives information on how the action affected the state of the environment in the form of a new state. The agent also receives a reward from the environment that reflects the quality of performing the specific action with respect to completing the task. The agent interprets the feedback from the new state and the reward to decide which actions to perform in future interactions to yield the highest sum of rewards.

Although agents can be applied to many environments, environments must be uniquely developed to represent a specific task. An environment is a dynamic system that provides an agent with a set of actions, states, and rewards at each time step of the simulation. The actions can represent low-level controls, such as how much voltage should be applied to a motor, or high-level decisions, such as what day to run a machine. The states can represent low-level sensations, such as direct sensor readings, or high-level information, such as the time and date of operation. The reward signals are designed to guide the agent's selection of actions by rewarding favorable state-action combinations or penalizing unfavorable ones in terms of the tasks to be solved.

However, the environment does not require system identification or an exact mathematical representation of the physical system with which the agent is intended to interact. Reinforcement learning is considered model-free because an agent can discover and identify relevant system parameters by associating different state and action combinations with its corresponding rewards. Therefore, an agent that has trained in a theoretical environment can adapt the learned parameters when applied to a real-world system. Furthermore, agents are designed to learn an optimal control strategy exclusively from the data generated by their own interactions with the environment. Therefore, no external data sets are necessary. These features make reinforcement learning techniques very versatile and suitable for implementation in stochastic scenarios where there is limited external data available.

An agent learns through a process referred to as training, where the agent tests different actions at each state and progressively favors state-action combinations that return higher rewards over time. Depending on an agent's approach, each action is tried several times in the same state to gain a reliable estimate of its expected reward. Therefore, the agent should only be rewarded for achieving the main goal of the task. If sub-goals are rewarded, the agent's performance in finding the most efficient way to achieve its main goal may be reduced [92].

The agent maps all state and action combinations it encounters in a probability matrix known as the policy. The purpose of the policy is to store the probability of which action to take in any perceived state of the environment. An agent develops several policies during training. However, the agent's purpose is to develop an optimal policy that maximizes the expected cumulative rewards over time. Therefore, the optimal policy is the optimal strategy to solve the task.

To develop an optimal policy, an agent has to balance the trade-off between exploration and exploitation during training. Exploration occurs when the agent chooses actions stochastically to discover the rewards of new state-action combinations in an attempt to improve its policy. Exploration is necessary for the agent to learn how the environment works and discover relevant system parameters. Without exploration, the agent would be stuck exploiting immediate rewards. When an agent exploits, it uses a policy it already knows to make greedy decisions. An agent has to balance when to exploit and when to explore to reduce the chance of failing a task. There are several different ways to define how an agent explores. A common method for balancing the trade-off between exploration and exploitation is called Epsilon-greedy, where epsilon refers to the probability of choosing to explore at every time step. A decay is often applied to the epsilon to reduce the amount of exploration as training progresses to solidify a policy and to avoid distorting discovered sequences of favorable state-action combinations.

The trade-off between exploration and exploitation sets reinforcement learning apart from the other machine learning paradigms, as this challenge does not arise in supervised or unsupervised learning. The objective of supervised learning is to correctly label unlabeled data. A supervised model learns by identifying patterns and correlations in prelabeled and external data sets. Therefore, supervised learning is not suitable for solving stochastic control tasks, as prelabeled data representations of all possible state-action combinations of a dynamic system rarely exist. Reinforcement learning overcomes this challenge by generating its own data rather than relying on external data.

Furthermore, the objective of unsupervised learning is to group or cluster data points based on patterns, hidden structures, and correlations discovered from unlabeled external data sets. Both unsupervised learning and reinforcement learning do not rely on examples of correct behavior. However, reinforcement learning's aim of maximizing the expected cumulative rewards sets it apart from unsupervised learning. The formalization of a goal using the reward signal is one of the most distinctive features of reinforcement learning [92].

4.2 Markov Decision Process

Reinforcement learning is built on a mathematical framework called Markov Decision Process (MDP) to define interactions between an agent and an environment. MDP is an extension of a Markov Process, which is a stochastic model that describes a sequence of possible events where the probability of each event depends on the information attained in the previous event. Furthermore, MDP is an abstraction of goal-oriented learning through interaction that reduces any problem of learning goal-oriented behavior to three signals between an agent and its environment: the decisions made by the agent (actions), the basis on which the decisions are made (states), and a signal to define the agent's goal (rewards). This framework is widely useful and applicable to represent most decision-making problems.

In a discrete MDP, the number of elements that represent the states, actions, and rewards of an environment is known. The grouping of state signals is referred to as the environment's state space (S), while the grouping of actions is referred to as the environment's action space (A). Representations of states and actions vary greatly from one environment to another. How they are represented has a large effect on the performance of the model, and these representational choices are currently more art than science [92].

4.2.1 Dynamics of a MDP

An agent interacts with an environment in a sequence of time steps, $t = 0, 1, 2, 3, \ldots$. The time steps can represent real-time or successive stages of decision making. An environment with a finite number of time steps is referred to as an episodic environment, and a sequence of discrete time steps is referred to as an episode. At the final time step, the agent reaches a terminal state and the episodic environment resets to a standard starting state or to a sample from a standard distribution of starting states to make every episode independent. When an environment does not have a final time step, it is referred to as a continuous environment, and the sequence of time steps is continuous after the initial state.

At each time step t, the agent is presented with the state of the environment $s_t \in S$ and selects an action available for that state $a_t \in A(s)$. As a consequence of the selected action, the agent receives a reward $r_{t+1} \in R$ and a new state s_{t+1}. The next state s_{t+1} and the future reward r_{t+1} have a discrete probability distribution that depends only on the preceding state and the action. The dynamics of a discrete MDP is expressed as follows.

$$p(s', r \mid s, a) = Pr\{s_{t+1} = s', r_{t+1} = r \mid s_t = s, a_t = a\}, \tag{4.1}$$
$$\forall\ s', s \in S,\ r \in R,\ and\ a \in A(s)$$

Where Pr is the probability of transitioning into a new state s' and receiving reward r, given the current state s, and action a. The '|' is the notation of conditional probability and is a reminder that p specifies a probability distribution for every choice of s and a. The probabilities returned by p represent the full dynamics of the environment because the probabilities for s_{t+1} and r_{t+1} depend exclusively on the current state s_t and the choice of action a_t. This is possible when the current state s_t contains all the information of past interactions between the agent and the environment. This is commonly referred to as the Markov Property, which is the fundamental expression of transitions between states in any Markov Process. The Markov Property is expressed as follows.

$$P[s_{t+1} \mid s_t] = P[s_{t+1} \mid s_1, s_2 ..., s_t] \tag{4.2}$$

Where the transition from the current state s_t to the next state s_{t+1} is entirely independent of the past since s_t captures the information of the past states.

4.2.2 Rewards

The probability of transitioning to a new state depends on the reward received by the agent. The goal of any reinforcement learning agent is to maximize the sum of rewards received. The cumulative sum of rewards is referred to as the return G_t, and for episodic environments it is expressed as follows.

$$G_t = r_t + r_{t+1} + ... + r_T \tag{4.3}$$

Where the reward received in the initial time step is r_t, the reward received in the second time step is r_{t+1}, and r_T is the reward received in the final time step.

For continuous environments where $T = \infty$, a discount rate γ is introduced to allow exploration and ensure that the sum of rewards does not diverge into ∞. The discount rate quantifies how much value should be considered for future rewards compared to immediate rewards. This enables the agent to select actions that will maximize the sum of discounted rewards it receives in the future based on the present value of future rewards. The discount rate is implemented in the returns function as follows.

$$G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ... = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \tag{4.4}$$

Where k is the number of time steps the agent is looking into the future. A reward received at k steps in the future is worth only γ^{k-1} times what it would be worth if it were received immediately.

4.2.3 Policy

Every agent follows a policy when it selects an action in a state. A policy defines the probability distribution for all possible action choices in every state. The policy function is expressed as follows.

$$\pi(a|s) = P[a_t = a \mid s_t = s] \tag{4.5}$$

Where π is the policy and $\pi(a|s)$ is the probability of choosing action a in state s. When combining a deterministic MDP with a trained policy, the MDP behaves similarly to a Markov Process because all decisions regarding state-action combinations have been made.

The goal of an MDP agent is to develop an optimal policy that will yield the highest return from the environment. To achieve this, the agent must use a set of value functions to estimate the expected return from being in a state and performing every action in every consecutive state. There are several approaches to value functions, and an agent is defined by its approach to value function. Therefore, there are several different ways for an agent to develop an optimal policy. However, most value functions are either a variation of the state-value function or the action-value function.

4.2.4 Value Functions

The state-value function determines the value of being in a particular state following a policy. Beginning in state s, it calculates the expected return by following policy π for the following states until it reaches a terminal state. It is expressed as follows.

$$\upsilon_\pi(s) = \underset{\pi}{\mathbb{E}}[G_t \mid s_t = s] = \underset{\pi}{\mathbb{E}}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1,} \mid s_t = s\right], \forall\, s \in S \tag{4.6}$$

Where $\upsilon_\pi(s)$ is the value of state s following the policy π, and $\mathbb{E}_\pi$ denotes the expected return following policy π.

The action-value function calculates the value of performing a specific action in a given state. This function is commonly referred to as the Quality function or the Q value function, and returns the value of performing action a in state s following the policy π. The action-value function can be defined as follows.

$$q_\pi(s,a) = \underset{\pi}{\mathbb{E}}[G_t \mid s_t = s, a_t = a] = \underset{\pi}{\mathbb{E}}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1,} \mid s_t = s, a_t = a\right] \tag{4.7}$$

Where $q_\pi(s,a)$ is the value of performing action a in state s.

A value function is calculated for every state that the agent encounters during training. When an agent follows policy π, the average returns of all the states following state s is that specific state's $\upsilon_\pi(s)$, and the separate averages for each action taken in every state will converge to $q_\pi(s,a)$.

Furthermore, value functions define a partial order of policies where policy π is better than policy π' if the value returned by policy π is greater than the value returned by policy π' [92]. However, the optimal policy is not found before the optimal return from the value functions are calculated.

The optimal return function, known as the Bellman Equation, is used to find the optimal return from the value functions, which in turn will reveal the optimal policy [92]. The Bellman Equation finds the optimal value of a state following a policy by calculating the value of a state based on the value of all its possible successor states following the policy π. The Bellman Equation for a state can be expressed as follows.

$$v_\pi(s) = \sum_a \pi(a \mid s) \sum_{s'} \sum_r p(s', r \mid s, a)[r + \gamma v_\pi(s')], \tag{4.8}$$
$$\forall\ a \in A,\ \ s \in S,\ \ and\ r \in R$$

The Bellman Equation is different from the state-value Equation 4.6 because it calculates the return from all possible future states-action combinations, weighting each by its probability of occurring. It returns the expected value of a state by computing the probability p for each triplet of s', a, and r and weighting the quantities in the brackets by its probability.

Therefore, the optimal state-value function $v_*(s)$ is found using the Bellman equation without reference to any specific policy [92]. This function is referred to as the Bellman Optimality equation and is expressed as follows.

$$v_*(s) = \max_a \sum_{s'} \sum_r p(s', r \mid s, a)[r + \gamma v_*(s')] \tag{4.9}$$

Where the Bellman Optimality equation is written in a form without reference to any specific policy because it expresses the fact that the value of a state under an optimal policy must equal the expected reward for the best action from that state($\max_a$). v_* is calculated for every state and outputs the maximum return received from the environment.

The Bellman Optimality equation is also used to find the optimal action-value function q_*, which is expressed as follows.

$$q_*(s, a) = \sum_{s'} \sum_r p(s', r \mid s, a)[r + \gamma \max_{a'} q_*(s', a')] \tag{4.10}$$

The Bellman Optimality equation for action-value outputs the expected return from taking the best action a in state s and following an optimal policy thereafter. It is the maximum action-value function over all policies.

When the optimal value functions are calculated for all combinations of states and actions, the optimal policy π_* can be found by maximizing over q_* in terms of v_*, as follows:

$$\pi_*(a \mid s) = argmax\, \mathbb{E}[r_{t+1} + \gamma v_*(s_{t+1}) \mid s_t = a, a_t = a],\ \forall\ s \in S \tag{4.11}$$

Where *argmax* is an operation that finds the argument that gives the maximum value of the target function.

4.2.5 Solving an MDP

A Markov Decision Process is considered solved when at least one optimal policy is found. There can be several optimal policies as long as they have the same return and share the same optimal state-value function v_* and optimal action-value function q_*. However, actual optimal policies are rare in practice because they can only be found when the environment satisfies the Markov Property (Equation 4.2) and there is enough enough computational resources to calculate the Bellman Optimality equation for every state and action combination [92]. The computational burden of reinforcement learning is referred to as the curse of dimensionality, because the computational requirements grow exponentially with the number of state variables [92]. For the game of Backgammon, the environment can be accurately modeled with the Markov Property, but solving the Bellman Optimality equation for v_* could take hundreds of years with today's computers since there are 10^{20} states [92].

Therefore, settling for approximate value functions to solve the task is common in reinforcement learning. By approximation optimal value functions, the agent is able to put more effort into learning to make good decisions for more frequently encountered states at the expense of less effort spent on infrequently encountered states. There may be several states with such low probability of occurring that selecting sub-optimal actions for them has little impact on the return received by the agent. Function approximation also makes it possible for the agent to generalize its response to states it has not encountered during training. Generalization is one of the key properties that distinguishes reinforcement learning from other approaches to approximately solve Markov Decision Processes [92].

Function approximation generalize the value-estimation when it is computational impossible for an agent to memorize the value of every possible state and state-action combination. This is often true for real-life environments where the number of state and action combinations can be very high or continuous. Function approximation identifies and utilizes features of states to generalize the value-estimation of states with similar features. This approach will never find the true value of a state nor the true optimal policy, but it is able to estimate a near-optimal policy that will reach similar results to the true optimal policy with much less computational effort.

There are many methods for computing function approximators. However, artificial neural networks have become widely accepted as the best approach due to their non-linear function approximation possibilities that does not require hand-picked features or domain knowledge like most linear approaches would have.

4.3 Deep Reinforcement Learning

Deep reinforcement learning is when the policy is developed by an agent utilizing a parameterized artificial neural network with one or more hidden layers as a function approximator. The purpose of the artificial neural network is to function approximate state and action-value functions and to provide generalization for responses to unseen states. The artificial neural network functions as a policy by finding patterns in links between the state signals, hyper-parameters of the state signals, and actions that will generate the highest rewards. Figure 4.2 shows how a deep reinforcement learning agent interacts with an environment using a artificial neural network as its function approximator.

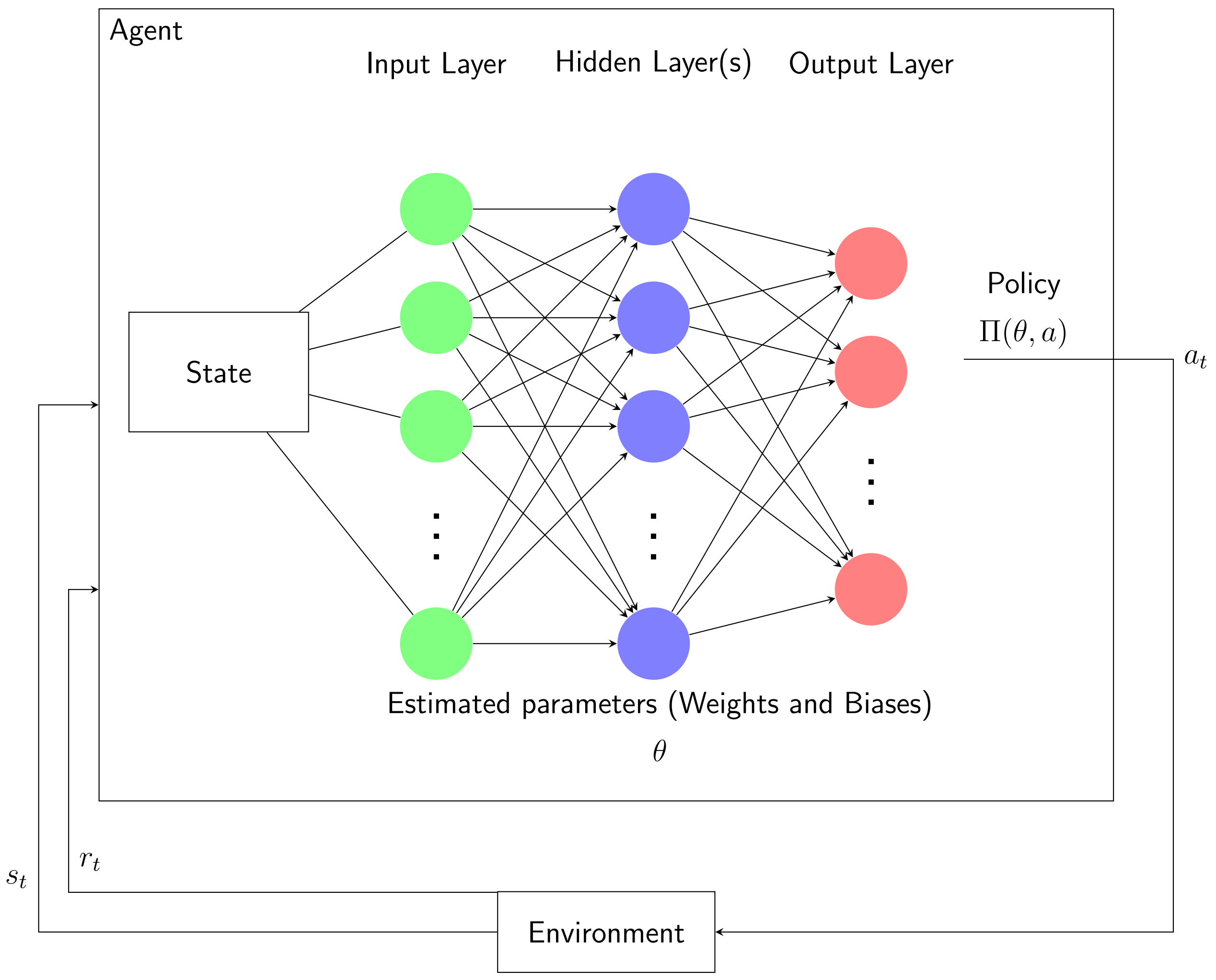


Figure 4.2: *Deep Reinforcement Learning Agent*

Every deep reinforcement learning agent uses at least one artificial neural network for developing its policy, where the network is either categorized as a critic or an actor.

A critic returns the expected value of the return for a given state and action combination. An agent that use a critic approximator relies on an indirect policy representation, and are considered value-based because they use the approximator to represent the state-value and/or the action-value. In general, critic agents function well with environments that have discrete action spaces. However, they can become computationally expensive when the action space is continuous.

An actor is better with a continuous action space, because it only returns the action that will maximize the return in a given state. Actors are considered policy-based because they rely on a direct policy representation that's either deterministic or stochastic. In general, actor agents are simpler and their training algorithms can be sensitive to noisy measurements and may converge on a local minimum.

Furthermore, agents that are able to use both actor and critic approximators have been developed. These actor-critic agents function with the actor learning the best action to select based on the feedback from the critic. Simultaneously, the critic learns the value function from the rewards so that it can criticize the choices of the actor. In general, actor-critic agents can handle both discrete and continuous action spaces. However, actor-critic agent is not explained further because the use of multiple artificial neural networks as function approximators is not required for this project.

4.3.1 Artificial Neural Networks

An artificial neural networks consist of a network of layers with interconnected operations referred to as neurons. The network must have one input layer, one output layer, and one or more layers in between that are commonly referred to as hidden layers. The amount of hidden layers depends on the tasks that will be solved. It is generally considered that one hidden layer offers a near universal approximation property for reinforcement learning tasks. However, experience show that multiple hidden layers can make this universal approximation property more efficient [92].

Jeff Heaton's textbook "Introduction to neural networks with Java" [93] is often referred to when answering questions regarding the architecture of artificial neural networks. It states that if there are no hidden layers, the network is only capable of representing separable linear functions and decisions. However, a network with one hidden layer can approximate any function that contains a continuous mapping from one finite space to another. Moreover, a network with two hidden layers can represent an arbitrary decision boundary to an arbitrary accuracy with rational activation functions and can approximate any smooth mapping to any accuracy. Therefore, there is no theoretical reason to use networks with more than two hidden layers for reinforcement learning tasks.

In deep reinforcement learning, the input for the neurons in the input layer represents the state signals from the environment, while the neurons of output layer represents the actions of the environment. The neurons in the hidden layers represents hyper-parameters or further abstract representations of features from the neurons in the input layer, where each successive hidden layer compute increasingly abstract representations of these features. Each neuron in the hidden layers defines one specific feature that contributes to a hierarchical representation of the overall input-output functions of the network [92].

Therefore, the hidden layers may have more or less neurons than the input layer depending on the complexity of the state signals and the level of abstraction the user wants to achieve. The amount of neurons in each hidden layer is subject to trial and error and can vary greatly from tasks to tasks. It is generally considered that the number of neurons should be kept as low as possible. However, too few neurons can result in under-fitting where the model is not able to learn enough hyper-parameters to generalize its response accurately. Furthermore, too many neurons is computational expensive and can result in over-fitting where the network is able to memorize the training data to perfection and will have a reduced accuracy when encountering unseen states.
Figure 4.3 below shows an example of a neural net with two hidden layers.

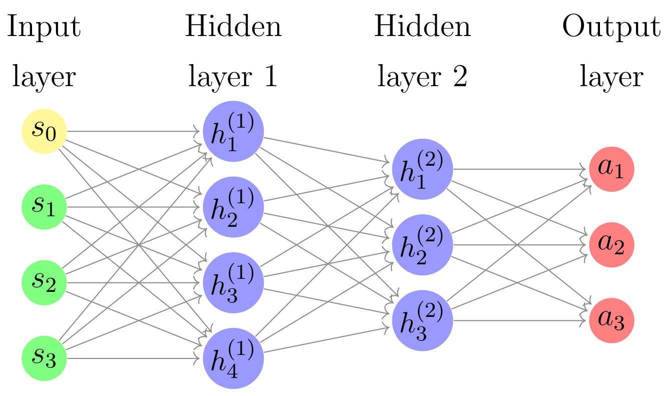


Figure 4.3: *An example of a artificial neural network with two hidden layers*

The connections between neurons in the network have a weight associated with it. Depending on the network's objective, the output value of a neuron is either amplified, suppressed, or inverted by this weight before serving as the input for the succeeding neuron. The succeeding neuron has connections with all the neurons in the previous layer, and must compute a weighted sum of all its input values. Furthermore, neurons apply a nonlinear function that converts the real value of the weighted sum into a number in a range between zero and one to produce the neuron's output signal. This nonlinear function is commonly referred to as an activation function, and the Sigmoid function or the rectified linear activation function are examples of activation functions. Calculating a neuron's activation can be expressed as follows.

$$h_1^{(1)} = ReLU((w_1 * s_0) + (w_2 * s_1)...(w_n * s_n)) \tag{4.12}$$

Where $ReLU$ is a rectified linear activation function, $h_1^{(1)}$ refers to the activation value of the initial hidden neuron in the first hidden layer, and w_1 is the weight of the connection applied to the activation value of the first neuron in the previous layer s_0.

The output of a neuron's activation function is the activation value of that neuron, which is the output signal sent to the succeeding connected neurons. The activation value determines how much a neuron is considered in the process. If the activation value is closer to one, the neuron is considered activated because a higher value is likely to activate subsequent connected neurons. If the activation value is closer to 0, the neuron is considered deactivated because its output signal is not likely to activate subsequent neurons. The activation value of neurons in the output layer represents the likelihood that the agent will choose a specific action. However, the activation function in the output layer may differ from that of the hidden layers. In regression problems, the activation function of an output layer is linear because the values are unbounded, whereas in classification problems a softmax or sigmoid function is sufficient.

In the hidden layers, a bias value is added to the weighted sum of every neuron to adjust the value in a favored direction. The purpose of the bias is to ensure that neurons in the hidden layer only activate when its weighted sum is within a given range. This feature allows neurons to activate if a specific parameter is in the correct range or to ignore it if it is not in the correct range.

Mathematically, all activation values are organized as a column of a vector ϕ, and all weights and biases are organized in a matrix θ. Therefore, the state-value function is the product of ϕ with respect to θ, and can be expressed as follows.

$$v_t(s) = v(\phi(s) \mid \theta_t) \tag{4.13}$$

Where $\phi(s)$ is a vector of the activation values of state s and θ_t is a transpose matrix of all the weights applied to these features. Furthermore, the action-value function can be expressed as follows.

$$q(s_t, a) = q(\phi(s_t, a) \mid \theta) \tag{4.14}$$

Where all activation values ϕ and weights θ related to selecting an action in a state are combined.

During training, the weights and biases are updated after each iteration through the network to strengthen the connections between neurons that generate a good policy. Weights and biases are essentially updated to produce activation values that result in actions that yield high rewards in every given state. Backward propagation of errors, commonly referred to as backpropagation, is the process of updating these weights and biases by calculating the gradient of a loss function with respect to all the weights or biases in the network. Proper tuning of the weights and biases reduces the error rates and makes the model reliable by increasing its generalization. The weights and biases θ are updated by minimizing the difference between the previously received reward and the new reward received by performing the action a in state s. This loss function is known as the temporal difference error δ, and is expressed as follows.

$$\delta \leftarrow r_t + \gamma \max_{a \in A} q(\phi(s_{t+1}, a) \mid \theta_i^-) - q(\phi(s_t, a) \mid \theta_i) \tag{4.15}$$

Where previously known values of θ are referred to as θ^-, and i in θ_i refers to a specific position in the transpose matrix of weights. Following the calculation of δ, θ can be updated as a gradient descent as follows:

$$\theta_{t+1} = \theta_t + \alpha(\delta_t - q(\phi(s, a) \mid \theta_t))\Delta_{\theta_t} q(\phi(s, a) \mid \theta_t) \tag{4.16}$$

Where θ_{t+1} are the updated weights, δ_t is the temporal difference error of the current time step, and α is a constant step-size parameter for the gradient descent commonly referred to as the learning rate. The learning rate controls how much to change the model in response to the estimated error every time the model weights are updated.

4.4 Challenges with Reinforcement Learning

The adoption of reinforcement learning for real-world applications is slow. Current methods have proven their abilities in finding high performing policies. However, utilizing them for real-world applications is generally been viewed as difficult or impractical.

Dulac-Arnold et al. [94] paper "Challenges of real-world reinforcement learning" highlights several challenges of real-world application of reinforcement learning systems. The challenges are explained below.

- Reinforcement agents should not be trained directly on a real system because exploratory actions may damage the real system or violate safety constraints. Therefore, a theoretical training environment is necessary.
- The transfer of an agent trained on a theoretical training environment to a real system environment often comes with a loss of performance due to unknown delays or unexpected faults of the real system.
- A significant loss of performance is expected when the real system is partially observable, meaning that a comprehensive state space is not available. For example, on a physical system where humans are interacting, there likely no sensors that can accurately signal wear and tear on motors or the mental state of the users.
- Furthermore, the high dimensionality of a real system's state and action space makes the formulation of an adequate training environment challenging.
- In systems where inference must be achieved in real-time, the control frequency of the system dictates time available for analyzing the state signals and selecting the best action by the agent.
- The formulation of the reward function is a challenge because it is the only motivation that guides the behavior of an agent to solve the task. Therefore, the efficiency of solving the task with reinforcement learning relies on the formulation of the reward function. Furthermore, the reward must also guide the behavior without violating any constraints expected from the real system in the model-free training environment. Moreover, If the tasks has multiple objectives the formulation of the reward function becomes extra challenging because the agent must be motivated to achieve all objectives simultaneously at every time step.
- It is very challenging to explain the true motivation behind the agent's policies on a human-level to system operators who have the responsibility of the real system. This is especially true for deep reinforcement learning policies because it is an on-going field of research regarding how to identify the actual hyper-parameters of the hidden-layers in artificial neural networks.

Chapter 5

State of the Art

This chapter focuses on recent literature in the field of applied reinforcement learning. The literature reviewed in this Chapter has been discovered using the review process explained in Chapter 3. However, the literature has been further surveyed and analyzed to discuss theories, hypotheses, methods, and results that are applicable to this project.

The main focus of this process has been on the applicability of relevant methods, as no literature was found that would satisfy the research questions. Therefore, the advantages and drawbacks of specific methods from the relevant literature are highlighted together with the justification for selecting the reinforcement learning framework.

5.1 Reinforcement Learning and Construction problems

The review process discovered three articles in the field of construction that were identified as relevant to this project, as they contained methods from which this project could benefit. These articles are reviewed in the following subsections.

5.1.1 Optimizing Scheduling of Construction Tasks

Soman and Molina-Solana [95] paper ”Automating look-ahead schedule generation for construction using linked-data based constraint checking and reinforcement learning” presents a reinforcement learning framework for generating conflict-free look-ahead schedules for construction tasks. The purpose of their framework is to act as a decision support tool during look-ahead planning meetings and to assist construction professionals to efficiently plan construction tasks faster than conventional methods. Their article was published in Automation in Construction Volume 132, 2022.

Reinforcement learning was selected due to its applicability in solving highly constrained optimization problems. Their reinforcement learning environment is episodic, and each episode is initialized by presenting the agent with a list of construction tasks, available resources, and task constraints. The goal of the agent is to schedule all the tasks without violating the constraints while minimizing costs.

Tasks represent operations that must be performed sequentially to complete construction processes. Each task requires time and resources to complete. The state space of the environment represents tasks that are completed, in progress, or not yet scheduled, and the resources required to complete the individual tasks. The action space represents the scheduling of a set of possible tasks for the following state.

The agent is rewarded based on three different metrics: if the schedule satisfies the tasks' constraints, the completion of tasks, and the total cost of resources.
The reward for satisfying the constraints of a task is zero. However, if a constraint is violated, the agent receives a negative number greater than any achievable positive episodic return. This enables the agent to quickly learn to avoid making schedules where constraints are violated.
The reward for the completion of tasks is a positive number associated with the completion or progress of the task. This reward motivates the agent to schedule tasks so that they are completed with the greatest efficiency.
The total cost reward represents the total cost of resources associated with a task. This includes the costs of mobilizing or demobilizing a task. Therefore, the agent is penalized for rescheduling tasks.

The paper uses a Q-learning agent to schedule the tasks. This type of agent uses a variation of the action-value equation 4.7 to calculate the value of performing an action in a given state. This value is stored in a matrix that contains the value of all state-action combinations encountered. During training, the agent updates the values for state-action combinations that it encounters on the basis of the state-action combinations' previous values. Following training, the agent will always select the state-action combination with the highest value when encountering states on which it has been trained. However, this agent does not provide generalization for unseen states and can therefore become unreliable in states that it has not encountered several times during training.

The results of this article found that the methodology could produce meaningful schedules in seconds, which proved its usefulness in situations where manual methods are too time consuming. Moreover, the framework was proven to be scalable by being able to produce results of similar quality when the number of tasks was increased. However, the results showed that the output of the approach was not as accurate as that of the manual critical path method. This is probably due to the choice of agent. A value-based deep reinforcement learning agent would likely produce more efficient results due to its function approximating capabilities with weights and biases that are able to examine more features of every state.

The main method extracted from this article is the reward method because the rewards motivate the agent to schedule tasks in an order that does not violate constraints, minimize costs, and maximize completion rates. This is a great example of how multiple rewards can be used to achieve multiple objectives.

5.1.2 Optimizing fulfillment of Earthmoving Tasks

Shitole et al. [90] article ”Optimizing earth moving operations via reinforcement learning” presents a reinforcement learning framework to optimize the loading, transport, and unloading of earth materials with earthmoving machines. Their approach combines discrete event simulation (DES) of the earthwork process with reinforcement learning to enable optimization of the DES. This article was published in proceedings of Inform’s 2019 Winter Simulation Conference.

The authors justify the use of reinforcement learning by defining the earthwork process as a collection of recurring activities carried out toward an end goal. Therefore, a complete overview of all construction activities at a given time defines a Markov state, while the progress deduced from the construction activities defines a return to be optimized. Furthermore, the control variables used to control construction activities define action variables, and the different settings of these control variables correspond to different policies that govern the efficiency of the operation.

The earthwork process in this framework represents the operations required to load and transport material from a loading area to a dump area by a fleet of trucks. They applied this framework to a case study example of two excavators and five dump trucks, where the excavators are operating at two different load sites within the same general loading area and where the dump site is bound by an upper limit on the number of trucks that can unload simultaneously. The unloaded trucks return to the loading area via a common return route.

The purpose of the agent is to learn the most efficient routing strategy as its policy. However, the action space consists of only two actions, which are route truck to excavator A or route truck to excavator B. This limits the agent's impact on solving the task and does not showcase the full potential of the reinforcement learning approach. Therefore, the discrete event simulation is likely more rule-based than dynamic.

The environment is episodic and terminates when a specified amount of material is delivered to the dump site. They decided that 200 episodes were the maximum number of iterations for training the model. However, it is not clear why this number was selected as it could be a limiting factor in the outcome of the policy. Furthermore, each episode began with the same initial state in which the amount of material delivered to the dumpsite was set equal to zero.

They trained and evaluated the framework in slightly different versions of the environment to compare the performance of the policies learned from different scenarios. However, this step could be avoided by making the environment more dynamic by substituting static values with semi-stochastic values, especially the initial state of the dumpsite and the number of machines in the scenario.

The main method extracted from this article is the interaction between the DES and the reinforcement learning framework. This approach enables the reinforcement learning environment to simulate recurring activities associated with different construction activities without involving the action space, which reduces the computational burden of the model and ensures the consistency of dependent construction activities.

5.1.3 Energy Management of Construction Machines

Zhang et al. [91] article "Reinforcement learning-based intelligent energy management architecture for hybrid construction machinery" presents a reinforcement learning-based methodology to improve the energy management of operating hybrid construction machines. The purpose of this energy management model is to minimize fuel consumption and emissions by allocating energy to auxiliary power sources that drive the walking system and actuators most efficiently. The article was published in Applied Energy volume 275 on the first of October in 2020.

The reinforcement learning approach was selected due to its dynamic self-learning and adaptive abilities in unknown environments to assess complex and variable working environments of construction machines. The authors agree with the findings of

Wang et al. [96] that generic energy management systems cannot be implemented in hybrid construction machines due to the significant differences in the working environment and working conditions.

Their reinforcement learning environment represents the operating characteristics of a hybrid construction machine and its working conditions. Relevant operating characteristics are how the remaining energy capacity of the hybrid construction machine's battery is limited by a maximum and minimum value to avoid overcharge and overdischarge. However, the model does not consider the cycle life of the battery by limiting the frequency of charging and discharging behavior.

Moreover, a virtual-world model designed to approximate a real-world environment by subdividing and sampling different cyclical working conditions for the machine to operate in, is introduced. However, the working conditions in this virtual-world model are cyclical and follow a pattern specific to a case study. To increase the versatility and adaptability of the model, a stochastic approach should be implemented to select the next working conditions to further assess the complex and variable working environments of construction machines.

The reinforcement learning agent used in this framework is a combination of DYNA-learning, explained in Robert Sutton's book "Reinforcement learning: An Introduction" [92], and Q-learning where a variation of equation 4.7 is used to update the probability of moving into the next state. The results indicate that their DYNA-Q agent outperforms standard Q-learning and rule-based generic energy management systems in terms of adaptability, real-time performance, and optimality.

They verify their reinforcement model by applying it to a real-world example where 25% of the real-world data samples are used to define the virtual-world model and the remaining 75% are used for performance verification. They applied their trained agent to control a simulation model of a hybrid construction machine in Matlab's Simulink where the initial state of charge for the hybrid construction machine's battery storage was set to different values to explore its adaptability. The outcome of this verification was that their DYNA-Q approach was closer to a global optimum than rule-based strategies. This verification process on a simulation model in Matlab's Simulink is relevant for this project, as this approach is able to demonstrate how a reinforcement learning agent can be adapted to interact with real-world systems.

The main method extracted from this paper is the virtual-world model that generates construction scenarios for the hybrid construction machine. This approach is relevant for this project because it generates unique training episodes that allow the reinforcement learning agent to learn and adapt its response to various scenarios.

5.2 Reinforcement learning and Battery problems

There were several articles related to reinforcement learning and battery problems. However, many of these articles had a similar approach or did not offer any relevant methods for this project. Therefore, the five articles reviewed below were identified as the most relevant in the review process due to their unique and applicable methods.

5.2.1 Battery Charging Costs

Chang et al. [97] article "Control of battery charging based on reinforcement learning and long short-term memory networks" presents a novel reinforcement learning methodology to solve the task of minimizing charging costs for a BESS in an electricity market with time-varying electricity prices. The article was published in Computers and Electrical Engineering volume 85, 2020.

The reinforcement learning approach was selected because of the model-free applicability that does not limit the method to any specific energy storage system and the adaptive capability in updating the decision-making scheme as new data are presented to it. Their model combines a reinforcement learning framework with a data-driven forecasting technique based on a long short-term memory network to predict electricity prices.

The action space consists of charging actions that specify how much a battery should be charged at each time step. The reward function represents the cost associated with charging and the increase in the state of charge in relation to its maximum state of charge. Therefore, discharging is not possible.

They trained their framework on baseline data estimated from historical energy prices rather than using direct historical electricity prices to improve the efficiency of the model. However, this limits the framework's ability to make good decisions when encountering an electricity price on which it has not been trained. Furthermore, they used a Q-learning agent that does not offer any generalization for the response to unseen electricity prices. Therefore, direct historical prices should have been used in combination with a value-based deep reinforcement learning agent to improve the response and efficiency of the framework.

Their approach was validated by applying the trained agent in an environment that was adapted to control a simulation model of a battery in Matlab's Simulink. The results showed that their framework performed better than two other model-free controller techniques, which were fuzzy control and constant charging. However, these other controllers do not seem to be comparable because their results do not show any response to varying battery prices.

The main method extracted from this article is the process of validating a trained agent on an adapted environment that controls a battery simulation model in Matlab's Simulink, as this demonstrates the applicability of the reinforcement learning framework in interacting with real-world systems.

5.2.2 Deep Reinforcement Learning for Optimal Battery Arbitrage

Kwon and Zhu [98] article titled "Reinforcement Learning Based Optimal Battery Control Under Cycle-based Degradation Cost" presents a reinforcement learning framework that finds the optimal arbitrage control of battery with time-varying electricity prices while reducing battery degradation. The article is currently being revised for IEEE's Transactions on Smart Grid, but was uploaded to ArXiv on February 17th, 2022.

Deep reinforcement learning was selected because of its ability to adapt to uncertainty in future information and its model-free features that do not require exact system dynamics and can perform well with uncertain statistical information.

Their state space includes the state of charge, the price of electricity, and several state-of-charge switching points to keep track of the battery's depth of discharge. These switching point state signals are used in a degradation cost calculation that affects the rewards, in an attempt to optimize the cycle life of the battery model. However, these switching points can be integrated as rule-based decisions to ensure that they are not violated unless certain conditions are met. This would increase the performance of the model due to fewer state signals.

The action space consists of multiple discrete actions that represent the intensity of charging or discharging. However, the authors state that the framework also works with a continuous version of this action space. The discrete actions are normalized to a range between -1 and 1 where actions that are greater than zero represent charging and actions that are lesser than zero represent discharging. This approach limits the applicability of the model to real-world applications because actions represent different intensities rather than actual control of a battery.

The framework includes a Deep Q-Network agent that utilizes a neural network with two hidden layers to function approximate the optimal action-value function equation 4.10. The ReLU-function is used as the activation function, which is explained in section 4.3.1. To train the network, a gradient-based optimization called Adam is used, which is explained in section 6.7. They also used the Epsilon-greedy method for exploration as explained in section 4.1.

The results conclude that the framework is more effective than existing linearized approximation approaches with respect to cycle-based degradation. However, comparing the performance of this framework to a different non-linear programming effort would be more applicable.

The main method extracted from this paper is the configuration of the Deep Q-Network agent because this article demonstrates how the agent can handle multiple state signals, a continuous action space, and the formation of subgoals.

5.2.3 Deep Reinforcement Learning for Optimal Control of lithium-ion based Batteries

Sui and Song's [99] article "A Multi-Agent Reinforcement Learning Framework for Lithium-ion Battery Scheduling Problems" presents a deep reinforcement framework for solving battery scheduling problems that can be adapted to various battery models and applications. The article was published in Energies volume 83, 2020.

The deep reinforcement learning approach was selected because of reinforcement learning's model-free ability to identify relevant system parameters without predefining them, the artificial neural network's ability to adapt to new cases, and its ability for real-world deployment with complicated use cases.

The action space of the framework consists of three actions: charge, discharge, and wait. These actions are conditional and are only available for the agent to perform in situations that can produce beneficial outcomes for the agent.

Their environment theoretically calculates the impact of actions on a lithium-ion battery system, and battery-specific parameters and dynamics are identified and isolated by the weights and biases of the artificial neural network of a value-based deep reinforcement learning agent. Therefore, the battery-specific parameters are represented as the hyper-parameters in the hidden layers based on a state of charge state signal and a battery cell temperature state signal.

The results demonstrated that this framework was able to approximate optimized results for battery system without requiring information on the battery's electrical model or its exact charge/discharge current. Therefore, it was able to solve complex tasks without solving complicated analytical equations. However, only three applications are shown in the paper. Therefore, specific applications like arbitrage that require substantial changes to the reward function and state space of the environment are not proven for the model.

The main method extracted from this article is the action space because it can be applied to any battery system by relying on the artificial neural network to discover system-specific parameters.

5.2.4 Deep Reinforcement Learning for Optimal Sizing of Battery

Li et al. [100] article "Battery Optimal Sizing under a Synergistic Framework with DQN Based Power Managements for the Fuel Cell Hybrid Powertrain" presents a synergistic framework that utilizes reinforcement learning to estimate the optimal capacity for a battery system in hybrid fuel cell vehicles. This article was published in IEEE Transactions on Transportation Electrification in 2021.

The reinforcement learning approach was selected because of its ability to quickly produce near-optimal performance compared to global optimization algorithms and its model-free applicability for systems where global information is not available.

Their synergistic framework combines a sizing optimization space with a deep reinforcement learning model to solve the sizing task, where the sizing optimization space produces battery candidates that the reinforcement learning model tests on the power management of a simulated hybrid vehicle. The outcome of the model forms the basis for the sizing optimization space to increase or decrease the capacity of the next battery candidate.

The optimal capacity is found by a comprehensive analysis of the equivalent hydrogen cost, battery operational cost, and fuel cell operational cost of each candidate. Therefore, finding the optimal capacity also finds the optimal power management for the hybrid vehicle.

This approach trains a Deep Q-Network agent on each battery candidate. However, if the agent is able to produce a robust policy on an initial training, the consecutive training can be avoided by testing the different candidates on a pretrained model and comparing the results to the initial candidate the agent trained on.

The main method extracted from this article is the integration of a sizing optimization space because this method expands the possible real-world applications for a reinforcement learning framework by using it as a supporting application within a larger framework.

5.2.5 Deep Reinforcement Learning for fleet management of multiple batteries

Wang et al. [101] article ”Reinforcement Learning for Real-time Pricing and Scheduling Control in EV Charging Stations” presents a reinforcement learning framework for an electric vehicle charging station, where the goal of the agent is to maximize profits by minimizing the cost of electricity. The article was published in IEEE Transactions on Industrial Informatics Volume 17, 2019.

The reinforcement learning approach was selected because of its model-free applicability, where decisions do not rely on any assumed stochastic model of uncertain future events, and where decisions can be made exclusively based on past events.

The framework simulates the operation of an electric vehicle charging station where vehicles arrive and depart at random times. Therefore, the action space and the state space are continuous to adapt to vehicles at the charging station at any given time step. The state space consists of the residual charging demand and the parking time of the vehicles. The action space consists of the charging price and the charging rate of every vehicle.

However, they found that it was sufficient to consider the total charging rates of all vehicles at the charging station at a given time rather than the individual vehicle’s charging rate. This dramatically reduces the action spaces without compromising the results. To reduce the state space, they could have done the same to the residual charging demand for every vehicle at the charging station at a given time.

The main task of the framework is to fulfill the charging demand for each vehicle before it departs. This task is achieved conditionally by allocating the available capacity of the charging station proportionally to the load demand from the vehicles as they arrive. Furthermore, the reward is designed to have the agent maximize the profit of the charging station, which is based on the hourly price of electricity and revenues received from the individual charging vehicles. Therefore, the framework is able to achieve two goals.

They developed their own agent based on the SARSA-algorithm. With this agent, they stated that they achieved between 20.2% to 138.5% higher charging-station profits than representative benchmark algorithms. However, these profits margins rely heavily on how the revenue from charging vehicles is calculated.

The main methods extracted from this paper are the optimization of the total charging rates for all vehicles instead of the individual vehicle's charging rate at a given time step because this can greatly increase the performance of a reinforcement learning model by reducing the action and state spaces.

Chapter 6

Methodology

This chapter outlines the structure and methods of a deep reinforcement learning framework. The purpose of this framework is to generate optimized schedules for charging and discharging a BESS to efficiently supply energy to electric earthmoving machines. The framework is designed to optimize earthwork processes with three different optimization goals: maximizing the workloads of electric earthmoving machines, minimizing the time required to complete a productivity goal, and minimizing the costs of electricity while completing a productivity goal.

The fundamental component of this framework is a reinforcement learning environment that simulates the stochastic nature of unique earthwork scenarios with various electric earthmoving machines without relying on external data sets. The environment contains several variables that earthwork professionals can specify to simulate user-specific scenarios and increase the level of detail. The assumptions, limitations, and rationale are highlighted as they become relevant.

Furthermore, the framework is developed using the object-oriented programming language Python because of its flexibility, interoperability, and access to relevant programming libraries and examples. Moreover, developing this project in Python makes it more accessible to other data scientists, as a worldwide survey of 19,717 data professionals found that 93% of data scientists used Python [102]. The full code for this project can be found in the Appendix.

6.1 Framework Introduction

This framework allows for the optimization of electric earthwork scenarios that involve excavation, filling, loading, and hauling because these are the most common earthwork processes found in the literature [6]. The framework is intended to assist earthwork professionals in decision-making processes regarding fleet configuration and scheduling the charging of electric earthmoving machines to increase work efficiency. Furthermore, the framework enables the user to investigate how different earthwork scenarios affect the expected productivity.

The framework simulates the interactions between a BESS and a number of electric excavators, loaders, and trucks, where the BESS is an intermediate energy supply source connected to the grid with a static power supply. A visualization of an optimizable earthwork scenario is provided in Figure 6.1.

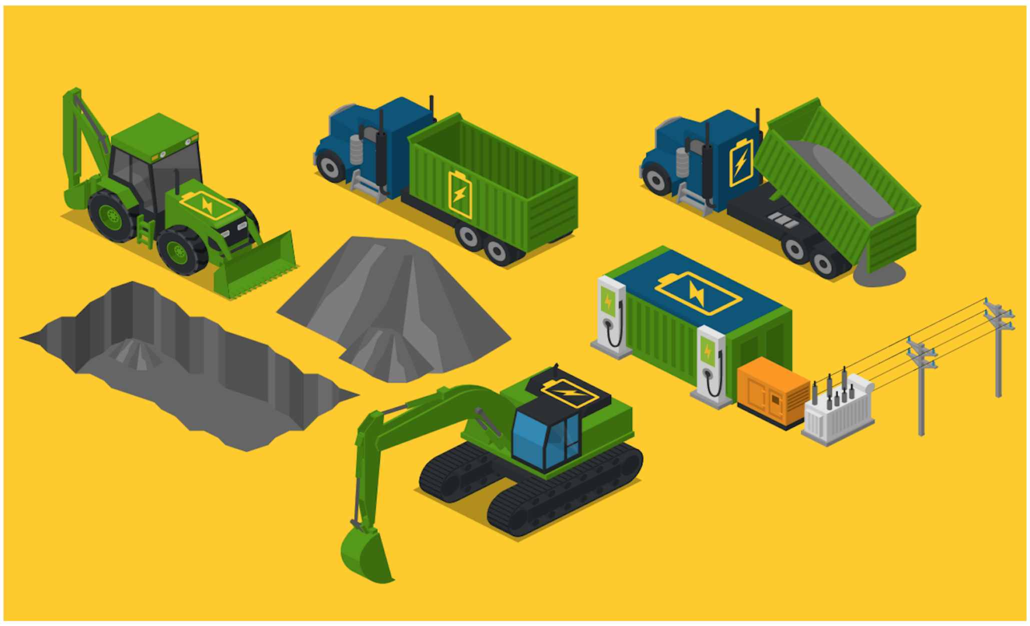


Figure 6.1: *Visualization of Earthwork Scenario*

The scenario in Figure 6.1 shows two trucks, one excavator, one loader, and a BESS. However, the user can specify the fleet configuration by setting the number and type of earthwork machine to be simulated.

To develop a unique earthwork scenario, the user must provide a set of input values to account for factors that affect the energy supply and energy consumption of earthmoving machines, which was discussed in Chapter 3. The input values are presented below.

BESS Factors

- State of charge at the beginning of the earthwork process ($B_{t_0}^{RC}$)
- Maximum energy capacity in kWh (B^{MaxCap})

- Minimum energy capacity in kWh (B^{MinCap})
- Maximum charging power in kW (B^{CP})
- Maximum discharging power in kW (per cable) (B^{DP})
- Number of charging cables available for simultaneous charging.
- Charging efficiency in percentage (B^{C}_{η})
- Discharging efficiency in percentage (B^{D}_{η})

Machine Factors

- State of charge at the beginning of the earthwork process ($M(x)^{RC}_{t_0}$)
- Maximum energy capacity in kWh ($M(x)^{MaxCap}$)
- Minimum energy capacity in kWh ($M(x)^{MinCap}$)
- Maximum Charging Power in kW ($M(x)^{CP}$)
- Charging efficiency in percentage ($M(x)^{C}_{\eta}$)
- Discharging efficiency in percentage ($M(x)^{D}_{\eta}$)
- Maximum output power in kW ($M(x)^{Power}$)
- Idling power in kW ($M(x)^{idle}$)
- Attachment specification (e.g. bucket size or loading capacity) in m^3
- Cycle time (e.g. digging, excavating, loading, hauling, etc.) in percentage of hour.
- Average hauling speed (for trucks) in km/t

Where x in $M(x)$ refers to a specific machine.

Operational Factors

- Operator skill level in percentage
- Operational efficiency (productivity level to account for site layout)
- Fleet Configuration (number of machine types)

- Work dependencies (optional)

Soil Factors

- Soil types
- Volume of soil to be excavated in m^3
- Layer location of soil types
- Soil power factors for machines
- Soil fill factor
- Maximum volume of soil that can stockpiled on site in m^3
- Distance to dump site in km

These input values can be collected from interviews with a site manager, machine reports, or machine handbooks. However, if certain values are unknown or underspecified, the user can set a range determined by the literature or experience-based estimates because the framework can handle uncertainties and generalize its response. The user can input these values into a specified section of the Python code.

The output values generated by the framework are curated to provide earthwork professionals with enough information to make high-quality decisions to optimize future earthwork scenarios. The output information for optimized scenarios is provided to the user as a CSV file and graphs that represent a timeline of optimized decisions. The output contains the following information for each run of the simulation.

Output Values

- Energy consumption and charge received by individual machines
- Charging and discharging of the BESS
- Number of cycles performed by individual machine
- Amount of soil moved to dump site
- Total amount of soil excavated
- Total amount of soil filled
- Detailed and total cost of electricity

- Comprehensive time schedule of activities
- The total duration of the earthwork activity

The quality of the output information is limited by the quality of the input values. Since there is a low quantity and quality detailing the relationship between electric earthmoving machines and energy consumption, ranges of stochastic values are used to demonstrate the impact of certain factors on the energy consumption of machine. These values can be substituted for more accurate values as they become available in the future.

6.1.1 Environment Type and Time Steps

The framework environment consists of a state space, an action space, and a rewarding system that are configured to enable current and future value-based deep reinforcement learning agents to interact with it.

The environment is episodic, where each episode represents a working day or a user-specified earthwork period. The episodic approach is chosen because every earthwork process has a beginning and an end that can be represented with an initial and final time step. A continuous approach would not be suitable because an earthwork process does not continue indefinitely after its initial time step.

The simulation time steps are dynamic and represent the successful completion of an action. However, the time steps are limited to one simulated hour. Therefore, any action will be terminated if the simulation reaches the next simulated hour while an action is active. Furthermore, the simulated hourly time steps are fractionated by a user-specified numerator to increase the level of detail for intrahourly energy transfer calculations.

This time-based approach allows the agent to perform new actions within the simulated hour if the previous action was completed before the full hour is simulated. In addition, this approach allows the agent to make new decisions as the price of electricity is updated, since most dynamic tariff systems are generally updated hourly [103].

6.1.2 Energy Transfer Calculations

This framework simulates electrical energy transfers between battery systems as the product of a power rating (kW), an intrahourly timestep, and a round-trip efficiency. This approach may not reflect the detailed dynamics of charging and discharging a specific type of BESS or electric earthmoving machine, but it does enable the agent to make high-quality decisions and generalize its response to all types of battery systems. Furthermore, it allows earthwork professionals to use the framework with the technical information available from manufacturers of electric earthmoving machines and BESS.

This approach was selected based on a study by Rosewater et al. for Sandia National Laboratories [104] that compares multiple representations of battery models in linear and non-linear programming. By gaining access and comparing the results of their models, it was revealed that the linear energy reservoir model, where the battery capacity was measured in kWh, deviated by less than 5% from a non-linear concentration-based model, where the battery capacity was measured in units of the concentration (mol/L) of the active materials of the electrodes. Furthermore, the conclusion of this study states that the linear energy reservoir is more appropriate for optimizing systems where multiple battery systems interact, which is the case for this project. Therefore, the energy transfer in this framework will be represented in kilowatt hours (kWh).

Battery degradation and self-discharge are not considered because the purpose of the framework is to optimize daily schedules and limited time periods that would not have a significant negative impact on battery cells in isolation. However, the depth of discharge can be limited by the user-specified minimum capacity variables for machines $M(x)^{MinCap}$ and BESS B^{MinCap}.

6.2 State Space

The state space consists of several state signals that the agent observes at every time step during an episode. These state signals reflect the state of the earthwork process and the state of charge of the battery systems. The purpose of these signals is to provide the agent with enough information to make high-quality decisions regarding the selection of actions in following time steps.

6.2.1 Remaining Capacity Signals

The remaining capacity signals $M(x)_t^{RC}$ reflect the individual state of charge of the simulated earthmoving machine $M(x)$ in kWh. The purpose of providing the agent with these signals is to enable the agent to monitor the state of charge of individual machines to make decisions regarding when specific machine should be charged to solve the tasks most efficiently. Therefore, the number of simulated earthmoving machines is equal to the number of remaining capacity state signals. These state signals are automatically generated by the code after the user inputs the fleet configuration.

Furthermore, a remaining capacity signal is also provided to reflect the state of charge of the BESS B_t^{RC}. This BESS signal enables the agent to monitor how much of the remaining capacity is available to perform actions with respect to its maximum or minimum energy capacity. During training, the agent uses this signal to learn when the BESS is approaching its minimum or maximum capacity because these limits have a significant impact on how the agent can navigate the action space. When this signal indicates that the BESS is approaching its maximum capacity, the agent is penalized for performing the charge action to avoid overcharge. Conversely, when the signal indicates that the BESS is approaching its minimum capacity, the agent is penalized for performing a discharge action to avoid overdischarge. The impact of this state signal on the action space is explained in more detail in Section 6.3.

The remaining capacity state signals can be normalized to limit its numerical range and reduce the computational burden of the agent by dividing the dynamic values representing the remaining capacities by the user-specified input values representing the maximum capacities of the earthwork machine and BESS.

6.2.2 Workload State Signals

The workload state signals $M(x)_t^{WL}$ are signals that reflect the individual amount of energy consumed by the simulated earthmoving machine in kWh in a time step. These signals allow the agent to monitor the amount of energy consumed by the individual earthmoving machine while being active or idle during each time step.

Furthermore, they allow the agent to identify which machine(s) can perform the most work and prioritize charging this machine before its minimum capacity limits the amount of work that can be completed in future time steps. These signals are especially important when optimizing the schedule to maximize workloads.

Similarly to the remaining capacity state signals, the number of simulated earthmoving machines equals the number of workload state signals. These state signals are automatically generated by the code after the user inputs the fleet configuration.

6.2.3 Productivity State Signals

Productivity state signals reflect the amount of soil excavated, filled, stockpiled, and delivered to a dump site, in m^3. These signals allow the agent to monitor the productivity of the earthwork process and tell the agent when a productivity goal has been achieved.

The productivity state signals can affect the agent's action decisions process by signaling when the agent should avoid charging specific machines that do not contribute to the productivity goal or prioritize charging machines that are required to avoid exceeding the stockpile limit on the site.

Furthermore, these signals can be normalized to reduce the computational burden of the agent if the corresponding maximum values are specified by the user.

6.2.4 Electricity Cost State Signals

An electricity price state signal and multiple time and date state signals are introduced to enable the agent to make decisions regarding when to select charge and discharge actions. However, these signals are only relevant when the agent is configured to minimize the cost of electricity during an episode.

The time and date signals are made up of four separate state signals that include the hour of the day, the day of the week, the week of the year, and the month of the year. The purpose of these signals is to enable the agent to associate electricity prices with specific hours of the year and to discover patterns in the interdaily, daily, weekly, and seasonal variations of electricity prices. Furthermore, these signals give context to the time steps of the episode, allowing the user to train the agent on specified hours and dates. Time and date signals are normalized to reduce the computational burden by dividing hours by 24, weekdays by 7, weeks by 52, and months by 12.

The electricity price signal reflects the price of electricity in hours represented by the hourly and intrahourly time steps. For training purposes, data sets of historical price data or forecasted estimates of a target location are used to adapt the agent's training to the currency and price range. This is the only training data that the framework should not generate by itself due to its regional and political dependencies. The price signal is not normalized because it is impossible to accurately predict what the highest electricity price will be in a real-world application. However, unprocessed price signals are relatively computationally light because prices per kWh are usually numerically low and in a limited range.

The performance of the framework was significantly reduced when the price signal was normalized. Specifically, the agent's ability to identify which hours had the lowest or highest price during an episode took significantly longer. When unprocessed price data were used, it allowed the agent to quickly identify high and low prices during an episode and subsequently act on small differences in periods with low price variations. Therefore, this signal can be derived directly from a utility provider in a real-world application.

6.2.5 Challenges with state signals

Grouped state signals were initially tested for the remaining capacity and workload signals of the machines, inspired by the paper by Wang et al. [101]. However, the performance of the model was reduced because the agent was unable to correctly identify which machine had the highest priority to receive charge, leading to issues in scaling the framework with multiple machines. The purpose of the grouped signals was to reduce the computational burden of the framework because there were fewer state signals to consider for the agent at each time step. However, increasing performance and scalability was deemed necessary at the expense of the computational burden.

A challenge did arise in collecting the machines' individual state signals for the agent when scaling the framework with multiple machines. Therefore, a method was developed to automatically collect the state signals of individual machines at each time step to avoid manually altering the framework every time a different fleet configuration was simulated. This method is a function that iterates over the machines selected by the user and stores relevant state signals in an ordered array where the individual signals are in an order corresponding to the individual machines. Figure 6.2 shows an example of the flow of operations for this method.

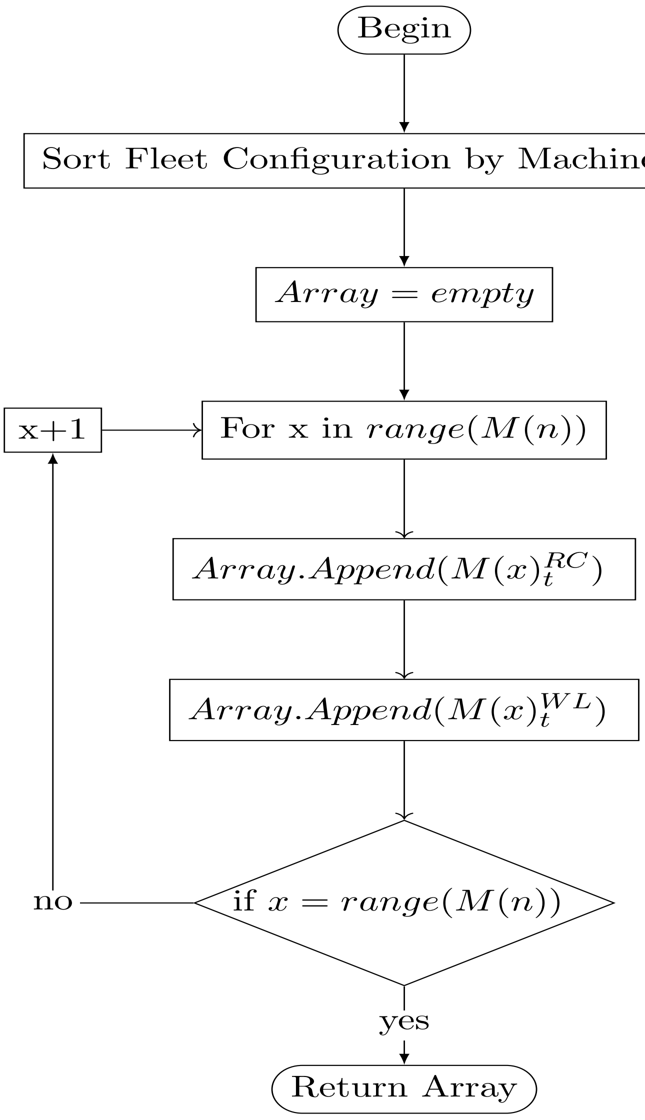

Begin
Sort Fleet Configuration by Machine Name
$Array = empty$
x+1
For x in $range(M(n))$
$Array.Append(M(x)_t^{RC})$
$Array.Append(M(x)_t^{WL})$
no
if $x = range(M(n))$
yes
Return Array

Figure 6.2: *Return Machine States Function*

Where a table containing the fleet configuration is alphabetically sorted by the user-specified names of the individual machines to iterate and store the relevant state signals for each machine in the same order for each execution of the function. The code for this function is available in the Appendix A.8.

6.3 Action Space

The action space of the environment consists of all available actions that the agent can choose from in each state and at each time step. The action space is discrete and consists of one charge action to charge the BESS, one waiting action to wait until the machines have depleted their remaining capacities, and multiple actions that discharge the BESS to charge individual machines or groups of machines simultaneously. Each action is represented by a number for the agent. Therefore, the number of actions depends on the number of machines selected by the user to simulate. These actions are developed to represent the real choices that exist on a earthwork site and to ensure that the BESS cannot be charged while discharging. This approach was inspired by the experience of the case study detailed in Chapter 8 and the paper by Sui and Song [99].

6.3.1 Charge Action

The charge action represents the charging process of the BESS from a standard grid connection. This action enables the agent to increase the remaining capacity of the BESS during a time step. This action is conditional and will only be allowed by the environment if specific conditions are met.

When the agent selects the charge action at a time step, the environment will check if the remaining capacity of the BESS is less than its maximum capacity to avoid overcharge. If the remaining capacity of the BESS is equal to its maximum capacity, the agent is forced to perform the waiting action as a penalty for selecting the charge action. The purpose of this penalty is to force the agent to learn to avoid selecting this action at a time step when it should have selected a different action.

If the remaining capacity of the BESS is not equal to its maximum capacity, a charging function begins. This function is a loop of operations that simulates the charging of the BESS from the grid. The loop adds the product of the time fraction, the charging efficiency of the BESS, and the user-specified power available from the grid limited by the charging power of the BESS in relation to its remaining capacity. This operation is expressed as follows:

$$B^{RC}_{t+\Delta t} = B^{RC}_{t} + min(GP, B^{CP}) \cdot \Delta t \cdot B^{C}_{\eta} \tag{6.1}$$

Where $B^{RC}_{t+\Delta t}$ is the remaining capacity of the BESS after charging for the period of the time fraction Δt, min() is a minimization function that returns the lowest value of two variables, GP is the power available from the connection to the grid, B^{CP} is the charging power of the BESS, and B^{C}_{η} is the charging efficiency of the BESS.

This loop continues until the remaining capacity of the BESS equals its maximum capacity or until the time spent charging exceeds the hourly time step. The flow of operations for the charge action is presented in Figure 6.3 below.

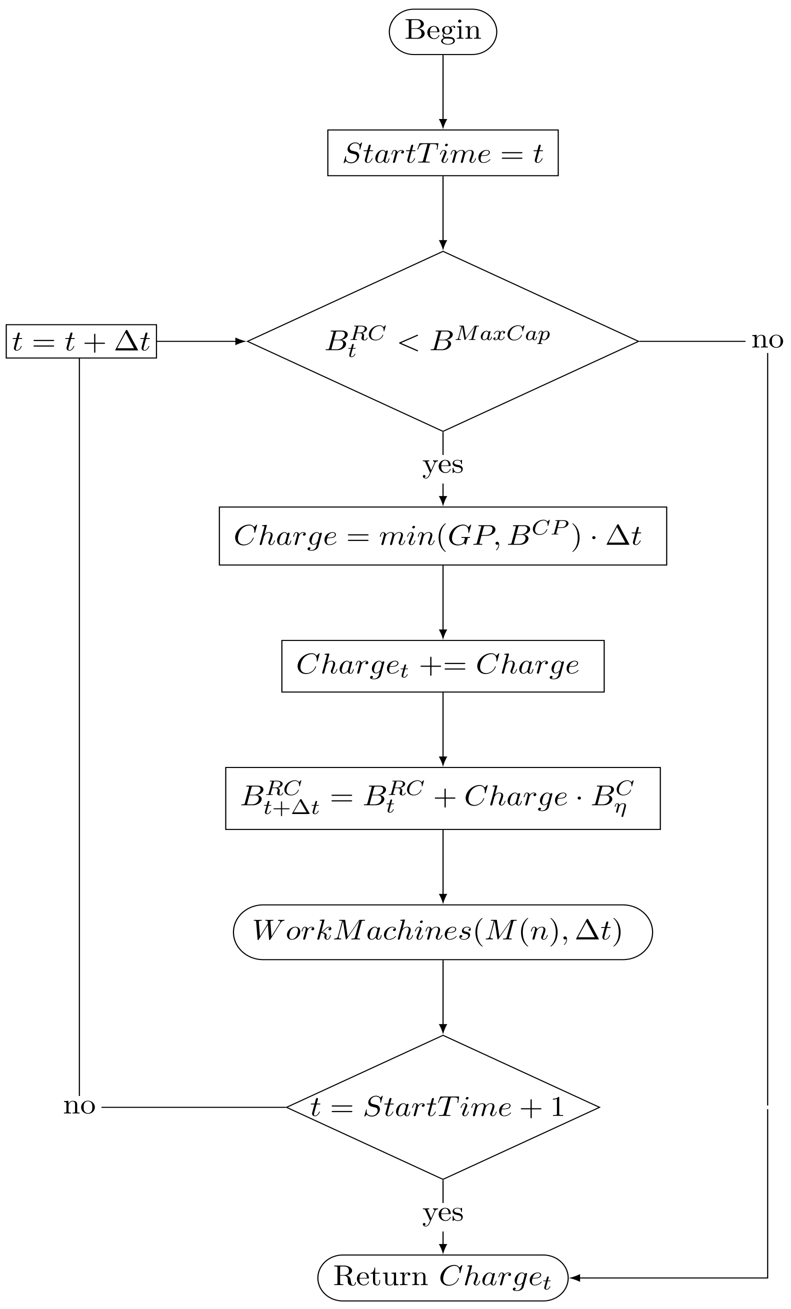

Begin
$StartTime = t$
$t = t + \Delta t$
$B_t^{RC} < B^{MaxCap}$
no
yes
$Charge = min(GP, B^{CP}) \cdot \Delta t$
$Charge_t += Charge$
$B_{t+\Delta t}^{RC} = B_t^{RC} + Charge \cdot B_\eta^C$
$WorkMachines(M(n), \Delta t)$
no
$t = StartTime + 1$
yes
Return $Charge_t$

Figure 6.3: *Charge-action dynamics*

Where $StartTime$ is set equal to the time step t at the beginning of the action, B^{MaxCap} is the maximum energy capacity of the BESS, $Charge_t$ is the accumulated charge from the grid during the action, and $WorkMachines$ is a function that simulates the completion of workloads by the earthmoving machines, which is explained in Section 6.5.2.

This action seeks to maximize the amount that will be charged to the BESS, and its output can represent all numbers between 0 and the BESS' maximum charging capacity over one hour, limited by the remaining capacity of BESS at the time step.

This action also incorporates the working of the machines to ensure that the same time spent charging the BESS is available for the machines to work. The code for this action is available in the appendix A.7.

In a previous version of this framework, the charge action was calculated as follows:

$$B_{t+1}^{RC} = B_t^{RC} + \min((B^{MaxCap} - B_t^{RC}), B^{MaxCharge}) \tag{6.2}$$

Where $B^{MaxCharge}$ is the maximum energy in kWh that could be charged to the BESS over one hour. This approach was replaced because it did not account for the non-linear characteristics of charging the BESS and it would spend the entire hour of the time step in situations where the charging process would not require a full hour.

6.3.2 Discharge Actions

The discharge actions simulate the discharging process of the BESS and the charging process of electric earthmoving machines. These actions decrease the remaining capacity of the BESS while increasing the remaining capacity of the electric earthmoving machine(s). The discharge actions aim to maximize the amount of kWh discharged from the BESS limited by the depth of discharge of the BESS and the depth of discharge of the earthmoving machine(s) selected for charging.

There are two discharge actions for each earthmoving machine simulated, where one discharges the BESS to charge a specific machine exclusively, and the other discharges the BESS to charge a specific machine together with a number of additional machines, simultaneously. The number of machines that the BESS can charge simultaneously is limited by the user-specified number of charging cables available from the BESS. However, the flow of operations that constitute the discharge actions is the same for charging one or more machines.

Any discharge action begins with checking if the remaining capacity of the BESS is greater than its user-specified minimum energy capacity B^{MinCap}. This minimum capacity can be set to limit the depth of discharge or avoid overdischarge. If the remaining capacity of the BESS is equal to its minimum capacity, the agent is penalized by having to perform the waiting action to avoid overdischarge. The purpose of this penalty is to force the agent to learn to avoid selecting this action at a time step when it should have selected a different action.

If the remaining capacity of the BESS is greater than its minimum capacity, the discharge action continues. The next operation separates the machines into two lists; one list for the machine(s) that will be charged ($M(n)^{Charging}$) and one for the machine(s) that will not receive charge ($M(n)^{Working}$). The action continues by checking the remaining capacities of the machine(s) selected for charging. If the remaining capacity of a machine selected for charging is greater than 98% of its maximum capacity, the machine is moved from the list of machines available for charging to the list of working machines. However, this will only occur in the early stages of training until the agent learns to avoid selecting machines that are fully charged for charging.

For the remaining machine(s) selected for charging, a function will begin to simulate the discharging of the BESS and the charging of the machine(s). This function is a loop of operations that iterates for each time fraction and calculates the discharging of the BESS and the charging of the machine(s) individually. The first operation in this loop subtracts the product of the user-specified time fraction and the machine's maximum charging power, limited by the BESS discharge power, from the BESS. This operation is expressed as follows.

$$B^{RC}_{t+\Delta t} = B^{RC}_{t} - min(M(x)^{CP}, B^{DP}) \cdot \Delta t \cdot B^{D}_{\eta} \tag{6.3}$$

Where $M(x)^{CP}$ is the charging power of a specific machine, B^{DP} is the discharging power of the BESS, and B^{D}_{η} is the discharge efficiency of the BESS. Following the subtraction of energy from the BESS, the charge to the machine is calculated as follows.

$$M(x)^{RC}_{t+\Delta t} = M(x)^{RC}_{t} + min(M(x)^{CP}, B^{DP}) \cdot \Delta t \cdot M(x)^{C}_{\eta} \tag{6.4}$$

Where $M(x)^{RC}_{t+\Delta t}$ is the remaining capacity of the machine after receiving charge and $M(x)^{C}_{\eta}$ is the charging efficiency of the machine receiving charge.

This operation loops until the remaining capacity of BESS is equal to its minimum capacity, the next hour is reached, or if all the machine(s) selected for charging are fully charged.

If multiple machines are being charged simultaneously, the action will check if any of the machines have reached 98% of its maximum capacity during the process. Machines that satisfy this condition are moved from the list of machines selected for charging to the list of machines available to work until there are no machines available for charging, and the action terminates. Therefore, the machine(s) that are not being charged are working for the same amount of time.

In an attempt to maximize future workloads by reducing charging time, an additional condition can be activated to terminate the action for machines that reach a state of charge that the framework expects to be sufficient to maximize its workloads in the remaining time steps of the episode. This condition, referred to as the charge-limit function, checks if the remaining capacity of the machine, minus its minimum capacity, has become greater than the sum of expected future workloads. If this condition becomes true, the machine is moved to the list of machines not being charged to avoid allocating more charge to this machine. The sum of future expected workloads for the remaining time steps is calculated as the product of the average workload completed by the machine and the remaining time steps of the episode. It is expressed as follows.

$$M(x)_t^{WLDemand} = \frac{\sum \left(M(x)_{t_0}^{WL} + M(x)_{t_1}^{WL} ... + M(x)_{t-1}^{WL} \right)}{t-1} \cdot (T-t) \tag{6.5}$$

Where $M(x)_t^{WLDemand}$ is the expected demand for future workloads in a time step, $M(x)_{t_0}^{WL}$ is the workload or energy consumed by the machine in the initial time step, $M(x)_{t_1}^{WL}$ is the workload or energy consumed by the machine in the second time step, and $M(x)_{t_{t-1}}^{WL}$ is the workload or energy consumed by the machine in the previous time step.

The general flow of operations for discharge actions is presented in Figure 6.4 below:

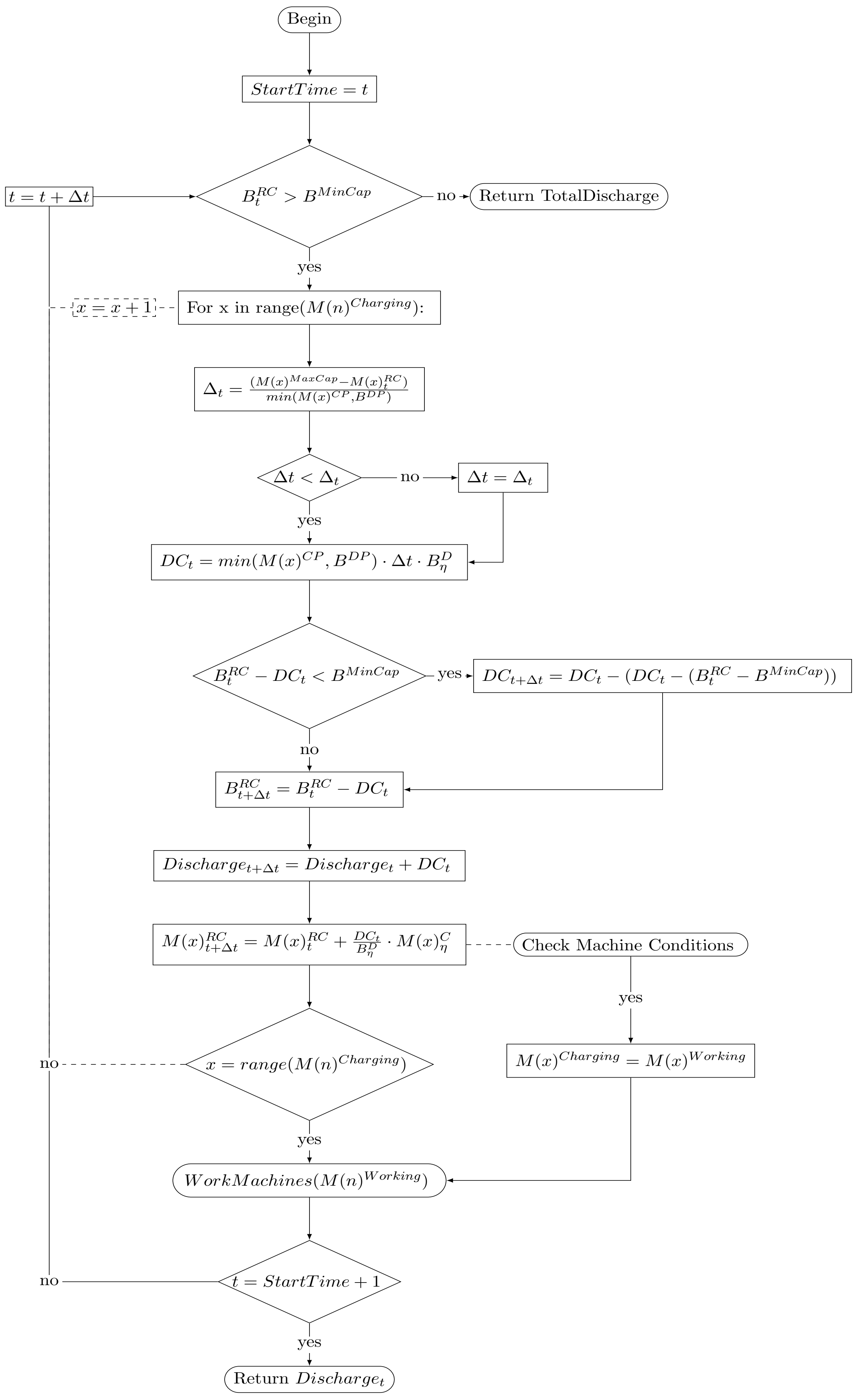

Begin
$StartTime = t$
$t = t + \Delta t$
$B_t^{RC} > B^{MinCap}$
no
Return TotalDischarge
yes
$x = x + 1$
For x in range($M(n)^{Charging}$):
$\Delta_t = \frac{(M(x)^{MaxCap} - M(x)_t^{RC})}{min(M(x)^{CP}, B^{DP})}$
$\Delta t < \Delta_t$
no
$\Delta t = \Delta_t$
yes
$DC_t = min(M(x)^{CP}, B^{DP}) \cdot \Delta t \cdot B_\eta^D$
$B_t^{RC} - DC_t < B^{MinCap}$
yes
$DC_{t+\Delta t} = DC_t - (DC_t - (B_t^{RC} - B^{MinCap}))$
no
$B_{t+\Delta t}^{RC} = B_t^{RC} - DC_t$
$Discharge_{t+\Delta t} = Discharge_t + DC_t$
$M(x)_{t+\Delta t}^{RC} = M(x)_t^{RC} + \frac{DC_t}{B_\eta^D} \cdot M(x)_\eta^C$
Check Machine Conditions
yes
$M(x)^{Charging} = M(x)^{Working}$
no
$x = range(M(n)^{Charging})$
yes
$WorkMachines(M(n)^{Working})$
no
$t = StartTime + 1$
yes
Return $Discharge_t$

Figure 6.4: *Discharge-actions dynamics*

Where DC_t is the amount of discharged energy from the BESS in the time fraction Δt, $Discharge_t$ is the accumulated amount of discharged energy from the BESS during the action, and Δ_t is the time required to fully charge a machine. Δ_t is calculated using an equation that outputs the remaining time required to charge a machine $M(x)$ to its maximum capacity $M(x)^{MaxCap}$. Δ_t is used to adjust the user-specified time fraction Δt when the remaining time to charge a machine to its maximum capacity is less than the user-specified time fraction, to maximize the efficiency of the intrahourly time steps. The code for this action is available in the Appendix A.4 and A.6.

6.3.2.1 Challenges with the Discharge-action

A challenge did arise when developing multiple machine-specific discharge actions, as it would require a lot of manual work to scale the framework and accommodate more machines. Therefore, a new method was developed to automatically generate multiple discharge actions and minimize the manual work required by the user when simulating different fleet configurations.

This method assigns a range of ordered numbers to identify individual machines that correspond to the numbers that the agent associates with selecting a discharge action for a specific machine. If the user has specified that the BESS can charge more than one machine simultaneously, a second discharge action is generated for each machine with an assigned number that is twice as great as the number representing charging the machine by itself. However, to avoid generating an exponential number of discharge actions for simultaneous charging, the other machines that are being charged simultaneously are selected by a rule-based prioritized ranking function.

The priority ranking function iterates over all simulated machines and updates the individual machine's priority based on its depth of discharge and its charging power in relation to the difference between its current remaining capacity and its maximum capacity. This function balances how a greater depth of discharge results in higher priority, while a greater charging power results in a lower priority. This is estimated using Equation 6.6.

$$M(x)_t^{Priority} = 1 - \frac{M(x)_t^{RC}}{M(x)^{MaxCap}} \cdot \frac{M(x)^{MaxCap} - M(x)_t^{RC}}{M(x)^{CP}} \tag{6.6}$$

Where $M(x)_t^{Priority}$ is a machine's priority for receiving charge in a time step. The code for the priority function is available in appendix A.10.

This discharge action method ensures that no discharge action occurs when the agent selects the charge or waiting action as long as the assigned numbers for the other actions are greater than twice the number of machines. An example of the flow of operations for this action assigning method with two charging cables is shown in Figure 6.5 below.

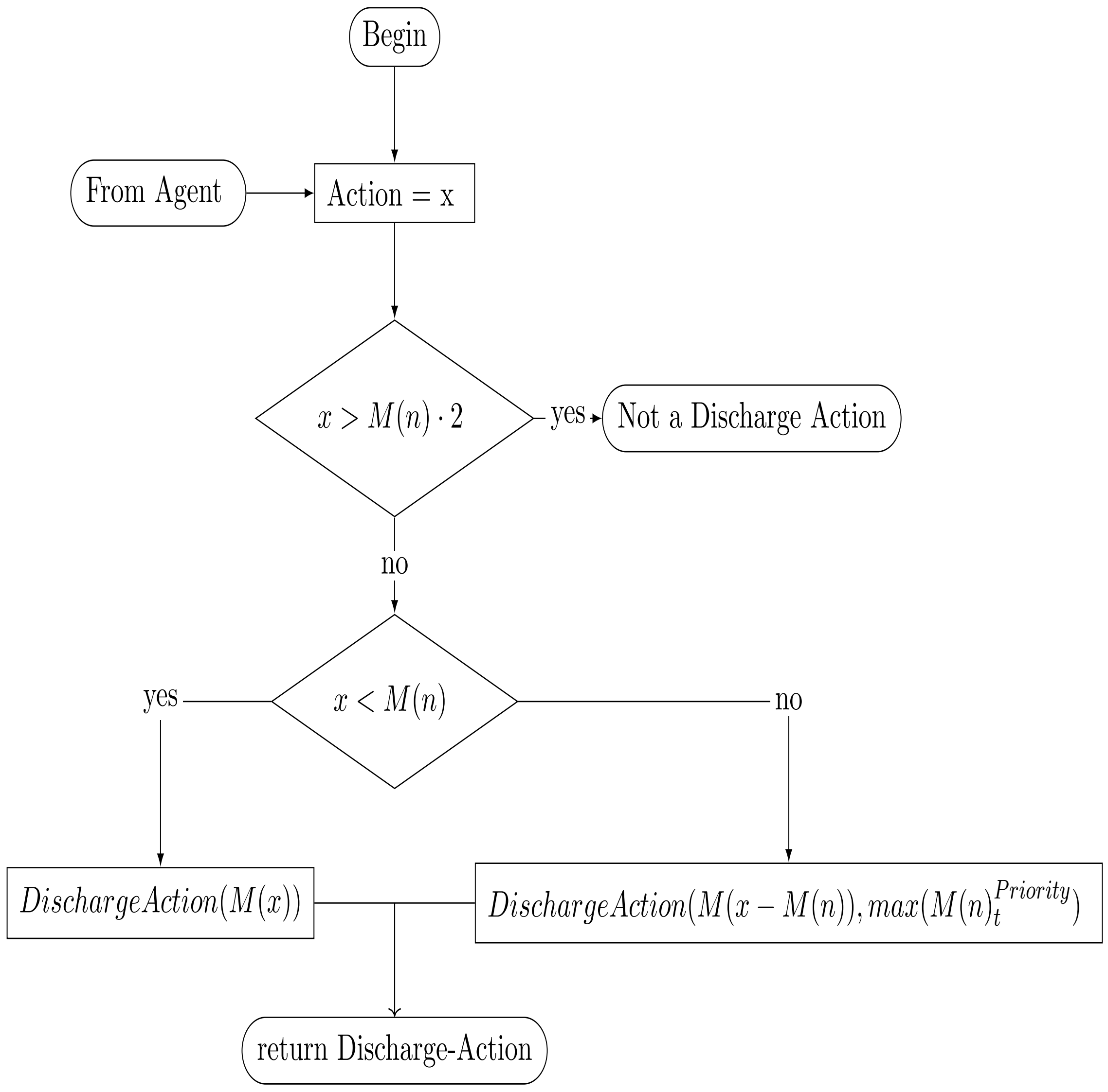


Figure 6.5: *Example of Action Generating Function*

Where x is the action number received from the agent, $M(n)$ is the total number of machines selected by the user, $M(x)$ is a specific machine associated with the action number, $max()$ is a function that returns the highest value of an array of numbers, and $M(n)_t^{Priority}$ is an array of the machine's priority ranking. The code for this function is available in the appendix A.7.

6.3.3 Waiting action

The waiting action represents that the BESS is not charged or discharged during a time step. When the agent selects this action, the agent cannot discharge or charge the BESS during that time step. The purpose of this action is to help the agent learn when to strategically wait for the time steps where the charge or discharge actions are more favorable. Therefore, the agent can learn to select this action when it estimates that performing a charge or discharge action in subsequent time steps may yield a higher episodic return.

The waiting action also enables the agent to wait for the earthmoving machines to complete workloads and deplete their remaining capacities, thus increasing the transfer capacity of the BESS' discharge amount. The waiting action ends when the time step has reached the following hour or when all machines have exhausted their remaining capacity. The flow of operations for this action is presented in Figure 6.6 below.

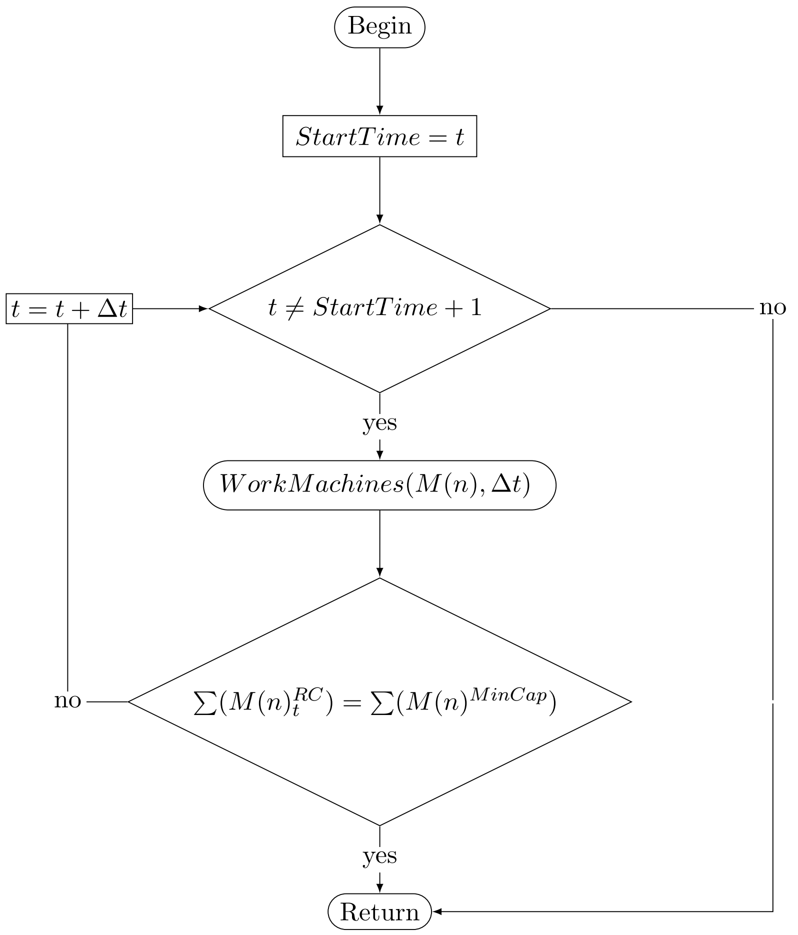


Figure 6.6: *Wait-action dynamics*

Where $\sum(M(n)_t^{RC})$ is the sum of the remaining capacities of all machines in the current time step and $\sum(M(n)^{MinCap})$ is the sum of the minimum capacities of all machines. The code for this action is available in the Appendix A.7.

6.4 Reward Signals

The purpose of the following reward signals is to guide the behavior of any value-based agent in solving the tasks of maximizing the machines' workloads, minimizing the time required to complete a productivity goal, and minimizing the costs of electricity while completing a productivity goal.

6.4.1 Reward Method for Maximizing Workload

The reward method to maximize workloads is designed to motivate the agent to learn the most optimal schedule to charge and discharge the BESS to maximize the amount of work completed by the machines during an episode of finite time steps. The purpose of this method is to allow the user to estimate how much work can be completed in a limited time frame. Furthermore, this reward mechanism must ensure that the agent learns to maximize the total amount of workload accumulated during the episode, rather than the workloads in individual time steps. Therefore, only one reward is presented to the agent at the final time step, which reflects the total amount of work completed during the episode. This reward signal is expressed as follows.

$$r_T(Maximize\ Workloads) = \sum \left(M(n)_{t_0}^{WL} + M(n)_{t_1}^{WL} ... + M(n)_T^{WL} \right) \tag{6.7}$$

Where r_T is a reward received in the final time step T, $M(n)_{t_0}^{WL}$ is the workload completed by all machines in the initial time step of an episode, $M(n)_{t+1}^{WL}$ is the workload completed by all machines in the second time step, and $M(n)_T^{WL}$ is the workload completed by all machines in the final time step of the episode.

This reward signal emphasizes that the purpose of the agent is to maximize the total workload for all machines, as the reward increases when the agent is able to have the machines complete more work during an episode. Furthermore, there are no other rewards to avoid the formation of subgoals, which could lead to a reduction in performance [92]. Therefore, the agent must learn by trial and error to select actions in an order that allows machines to maximize their workloads.

The drawback of this reward method is that the agent will favor the most powerful machines that consume more energy during individual time steps. Therefore, a less powerful machine may be neglected from receiving charge in favor of more powerful ones in scenarios with multiple machines. However, the presence of a stockpile volume limit or other productivity-related limits will force the agent to learn how to maximize the workloads of the machine(s) that can be the most productive in a scenario.

6.4.2 Minimize the time required to complete a productivity goal

The reward method to minimize the time required to achieve a productivity goal is designed to motivate the agent to learn the most optimal schedule to charge and discharge the BESS to the machines to efficiently achieve a productivity goal during an episode. The user can customize this productivity goal to focus on the amount of soil excavated, filled, or dumped at a dump site during the episode.

This reward method is inspired by Soman and Molina-Solana [95], where the completion of tasks is rewarded based on the resources consumed. In this case, the resource is time.

This reward method provides the agent with a static negative reward for each time step required to achieve the productivity goal. The episode terminates when the productivity goal is reached to limit the negative return. Therefore, the agent's goal is to minimize the negative return by reducing the number of time steps required to achieve the productivity goal. This reward method is expressed as follows.

$$r_t(Minimize\ Completion\ Time) = \begin{cases} -100 & \text{if } Dumped\ Soil < 200m^3 \\ 0, & \text{otherwise} \end{cases} \tag{6.8}$$

Where -100 is an example of a static negative reward received by the agent at each time step where the amount of dumped soil is less than the productivity goal, represented by $200m^3$ in this example.

6.4.3 Reward Method for Minimizing Electricity Cost

The reward method to minimize electricity costs is designed to guide a value-based agent in selecting the charge action when electricity prices are relatively low compared to when selecting the discharge actions. This is achieved by calculating the reward conditionally on the action.

6.4.3.1 Charge Reward

The reward for performing the charge action is formulated to incentivize the agent to perform the charge action in time steps when it considers the price of electricity to be low during an episode. When the charge action is selected, the reward for the time step is the sum of kWh charged to the BESS ($Charge_t$) converted to a negative number multiplied by the price of electricity of the hour represented by the time step. This reward equation is expressed as follows.

$$r_t(Charge - action) = -Charge_t * price_t \tag{6.9}$$

Where $price_t$ is the price of electricity at the hour of the year represented by the time step. This reward equation will always produce a negative value, which will motivate the agent to reduce this reward as much as possible. The agent can minimize this reward by charging during time steps when the price of electricity is lower during an episode. Hence, motivating the agent to predict the price of electricity according to its training data. However, the agent would learn to avoid selecting the charge action due to its negative reward if the discharge action and its corresponding reward were not dependent on it.

6.4.3.2 Discharge Reward

The discharge reward is formulated to incentivize the agent to select any of the discharge actions when electricity prices are comparatively higher during an episode. When the discharge action is selected, the reward is the positive sum of kWh discharged from the BESS ($Discharge_t$) at the time step multiplied by the price of electricity for the hour represented by the time step. This reward equation is expressed as follows.

$$r_t(Discharge - action) = Discharge_t * price_t \tag{6.10}$$

This reward equation motivates the agent to maximize the amount of kWh discharged during the time steps of an episode when electricity prices are relatively higher. Hence, motivating the agent to learn to predict when the price of electricity peaks according to its training data.

The agent is able to maximize this reward by learning to perform the charge action in previous time steps to increase the BESS' remaining capacity and wait for the electric earthwork machines to deplete their remaining capacities. Therefore, this reward indirectly motivates the agent to select the charge action in previous time steps, especially when electricity prices are lower.

6.4.3.3 Waiting Reward

The reward for performing the waiting action is zero. When different reward values for this action were tested, it was found that a negative value will reduce the occurrence of the agent strategically waiting or rush the agent's decision to select either the charge or discharge actions. Furthermore, if the reward for selecting the waiting action presented a positive value, a greedy agent would learn to do nothing during the entire episode.

6.4.3.4 Completion of a Productivity Goal Reward

The completion of a productivity goal reward is a reward signal similar to Equation 6.8. However, this reward is directly dependent on the actions performed in the previous time steps and is only available to the agent if the productivity goal has been reached.

This approach multiplies the sum of previously received rewards by a user-defined factor to amplify the importance of selecting the charge action during time steps when the prices of electricity are lower and selecting a discharge action when prices are higher. It is expressed as follows.

$$r_T(Completing\ Productivity\ Goal) = \begin{cases} \sum(r_1, r_2 ... r_{T-1}) * 10 & \text{if } Excavated\ Soil >= 450m^3 \\ 0, & \text{otherwise} \end{cases} \tag{6.11}$$

Where $r_T(Completing\ Productivity\ Goal)$ is a reward available at the final time step r_T and 10 is a factor defined by the user.

The main challenge with this reward is that it becomes the main goal of the agent and reduces the importance of action-dependent rewards to subgoals during training.

However, it is the only guarantee that the agent will learn to complete a productivity goal during an episode. This performance sacrifice is considered necessary, as the primary focus should be on achieving a productivity goal while the secondary focus should be on reducing electricity costs. Moreover, if the agent is only rewarded for arbitrage, it will not consider the earthwork process.

A secondary challenge with this reward signal arises when the agent struggles to achieve the productivity goal during training. If the agent cannot generate a positive return from action-dependent rewards during an episode, $r_T(Minimize\ Completion\ Time)$ will be negative and penalize the agent for achieving the productivity goal. Therefore, the performance of the framework can become unpredictable when the productivity goal is too great and when there is a limited number of time steps available.

6.5 Earthwork Dynamics

The environment includes several novel methods and functions developed to simulate the earthwork process. These methods include a scenario generator that generates semi-stochastic earthwork scenarios for every training episode, workload functions that simulate the earthwork process with the machines, and a work dependency method that enables work dependencies and specific tasks to be simulated.

6.5.1 Earthwork Scenario Generator

The Earthwork Scenario Generator is a method to generate earthwork scenarios that the agent must learn to navigate while optimizing the schedule. During training, these scenarios can include semi-stochastic values to make every episode unique and increase the agent's ability to generalize when solving unseen scenarios. The generator contains a data bank where the user can store the input values that represent the influencing factors.

For training purposes, the generator assigns a semi-stochastic numerical value to represent the remaining capacity of each machine specified in the fleet configuration, limited by the machines' maximum and minimum capacities. The purpose of this process is to force the agent to explore different scenarios for every training episode and, ultimately, to develop a robust policy to generalize its response to unseen scenarios. This approach can also be applied to the formation of a productivity goal, bounded by an upper and lower limit provided by the user to not exceed what is possible during the time frame of an episode. Furthermore, the initial charge priority is set by the priority function explained in Section 6.3.2.

The generator stores the generated scenario in several tables that represent the state of the earthwork scenario at the initial time step of an episode. The dynamic variables in these tables are modified by actions and other methods during each time step to reflect the progress of the earthwork process.

Table 6.1 is an example of a table of machine factors generated by the earthwork scenario generator.

Parameter	**Excavator**	**Loader**	**Truck**
Maximum Capacity	300kWh	210kWh	80kWh
Minimum Capacity	30kWh	21kWh	8kWh
Remaining Capacity	(30,300)kWh	(21,210)kWh	(8,80)kWh
Maximum Charge Power	150kW	100kW	40kW
Maximum Output Power	50kW	30kW	20kW
Idle Power	10kW	5kW	3kW
Round-trip Efficiency	0.95	0.95	0.95
Cycle Time	0.01	0.03	0.08
Attachment Capacity	$1.5m^3$	$3m^3$	$23m^3$
Operator Skill Level	0.9	0.8	0.7
Operational Efficiency	(0.65, 0.85)	(0.65, 0.85)	(0.65, 0.85)
Priority	1	2	3
Average Hauling Speed	-	-	80km/t

Table 6.1: *An example of a table of machine factors generated by the earthwork scenario generator.*

The generator also generates a table of soil factors that affect the estimates of energy consumption and productivity. Table 6.2 is an example of a table of soil factors generated.

Parameter	Rock	Clay	Sand
Soil to be excavated	100 m^3	150 m^3	400 m^3
Location of Soil	lower	middle	upper
Soil Power Factor	(0.7,1)	(0.5, 0.7)	(0.1, 0.5)
Fill Factor	0.7	0.8	0.9

Table 6.2: *An example of a table of soil factors generated by the construction scenario generator program.*

The soil power factor represents the impact of the type of soil on the load factor of specific machine types. Since quantitative studies have not yet studied the impact of soil types on the energy consumption of electric earthwork machines, a range is set to generate stochastic values for every cycle to expose the agent to a varying load factor in an attempt to increase its ability generalize its response.

Additionally, the distance to the dump site and the number of expected work dependencies are established at this stage. The code for the earthwork scenario generator is available in the Appendix A.16.

6.5.2 Workload Functions

The workload functions are a set of functions designed to simulate earthwork processes and the energy consumption of machines. These functions calculate machine-specific workloads and productivity estimates based on user-inputted machine, operational, and soil factors. It also records the amount of workload and productivity achieved by individual machines during the time step for the workload and productivity state signals, as explained in Section 6.2.

The flow of operations that make up these functions is executed for the same number of time fractions as the actions performed by the agent, as explained in Section 6.3. Furthermore, these functions impact machines that are not being charged in the current time step because a machine(s) that is being charged cannot perform workloads simultaneously.

The functions are machine-specific, meaning that there are separate functions for different types of earthmoving machines because the machines serve different roles and responsibilities in earthwork processes. This project has developed functions for excavators, loaders, and trucks.

For machines that are active on-site (e.g. excavators and loaders), the energy consumed and productivity rates are estimated with the same approach due to the lack of detailed quantitative data on of electric earthwork machines energy consumption and productivity. The energy consumed by an on-site machine is referred to as the machine's workload (WL), and it is estimated as follows.

$$WL = M(x)^{WL}_{t+\Delta t} = M(x)^{Power} \cdot Spf \cdot M(x)^{OE} \cdot \Delta t \cdot M(x)^{D}_{\eta} \tag{6.12}$$

Where $M(x)^{WL}_{t+\Delta t}$ is the estimated energy consumption of a machine during a time fraction Δt, and $M(x)^{Power}$ is the maximum output power of the machine, Spf is a soil power factor, $M(x)^{OE}$ is the operational efficiency of the machine, and $M(x)^{D}_{\eta}$ is the machine's discharge efficiency.

However, Equation 6.12 does not consider that overdischarge may occur. To avoid overdischarge, a supporting function is developed to limit the workload of any type of earthwork machine. This function is presented in Figure 6.7 below.

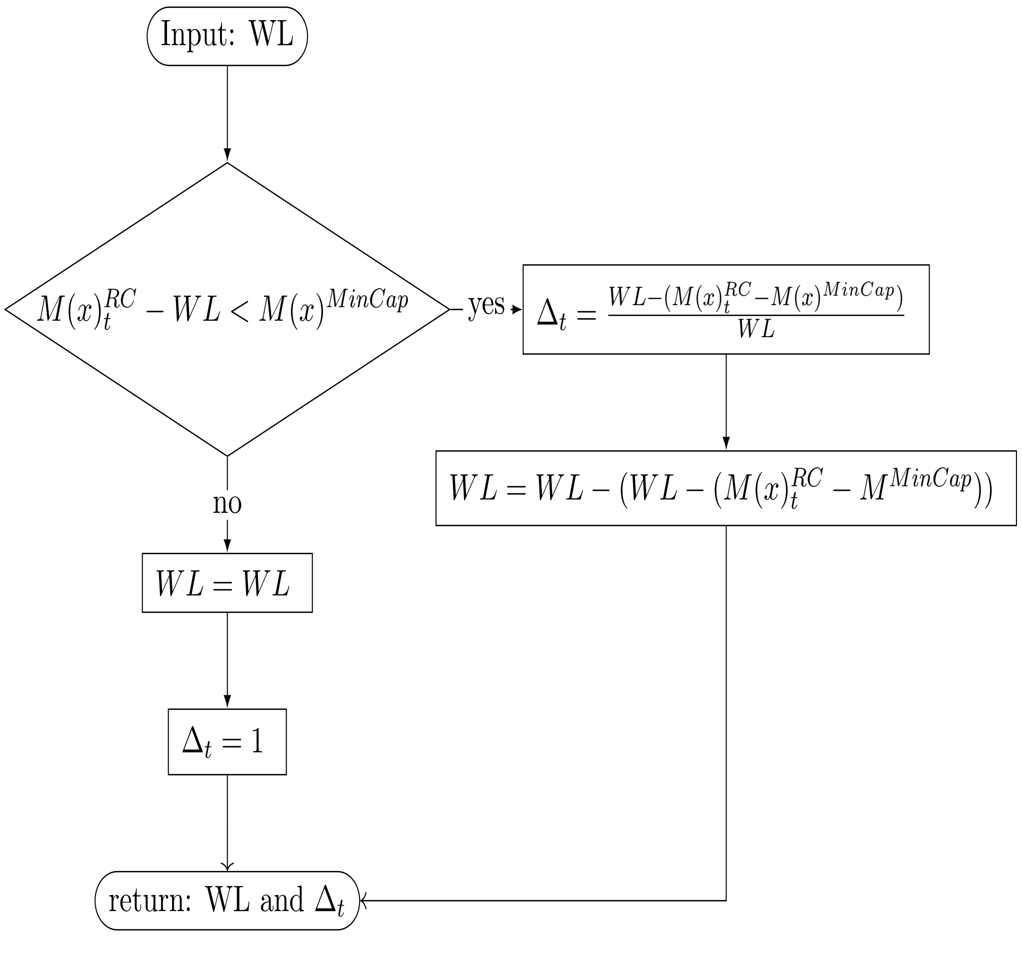

Input: WL
$M(x)^{RC}_{t} - WL < M(x)^{MinCap}$
yes
$\Delta_t = \frac{WL-(M(x)^{RC}_{t}-M(x)^{MinCap})}{WL}$
$WL = WL - (WL - (M(x)^{RC}_{t} - M^{MinCap}))$
no
$WL = WL$
$\Delta_t = 1$
return: WL and Δ_t

Figure 6.7: *Function to avoid overdischarge of electric earthwork machines*

Where $M(x)_t^{RC}$ is the remaining capacity of the machine in the current time step, $M(x)^{MinCap}$ is the minimum energy capacity of the machine, and Δ_t is the limited amount of time spent active.

Following the energy consumed and the time spent active, the productivity of the on-site machines can be estimated as follows.

$$M(x)_t^{Prod} = (M(x)^{BS} \cdot Sff) \cdot \frac{\Delta t \cdot \Delta_t}{M(x)^{CT}} \cdot M(x)^{OE} \cdot M(x)^{OSL} \tag{6.13}$$

Where $M(x)_t^{Prod}$ is the volume of soil handled in m^3 by the machine during the time fraction, $M(x)^{BS}$ is the attachment capacity of the machine, Sff is the fill factor of the type of soil handled, $M(x)^{CT}$ is the cycle time of the machine operation, and $M(x)^{OSL}$ is the skill factor of the machine operator.

6.5.2.1 Excavator Workload Function

The excavator function simulates the excavation of the user-specified volume of soil to be excavated to the on-site stockpile. However, if the volume of the on-site stockpile exceeds the maximum amount specified by the user, the excavator will load the excavated soil directly into an available truck. On the rare occasion that the volume of the on-site stockpile exceeds its maximum volume and there are no trucks available to load, the excavators are forced to idle. If an excavator is forced to idle, no productivity will be recorded, and equation 6.12 will be replaced by the following equation.

$$WL = M(x)^{idle} \cdot \Delta t \cdot M(x)_\eta^D \tag{6.14}$$

Where $M(x)^{idle}$ is the idling power of the machine. However, if the excavators are not forced to idle, the excavator function is activated. This function iterates over all excavators specified in the fleet configuration. For each iteration, the energy consumed and productivity are estimated. Furthermore, impacts on the volume of soil to be excavated and the volume of soil stockpiled are simulated. Figure 6.8 shows the flow of operations for the excavator function.

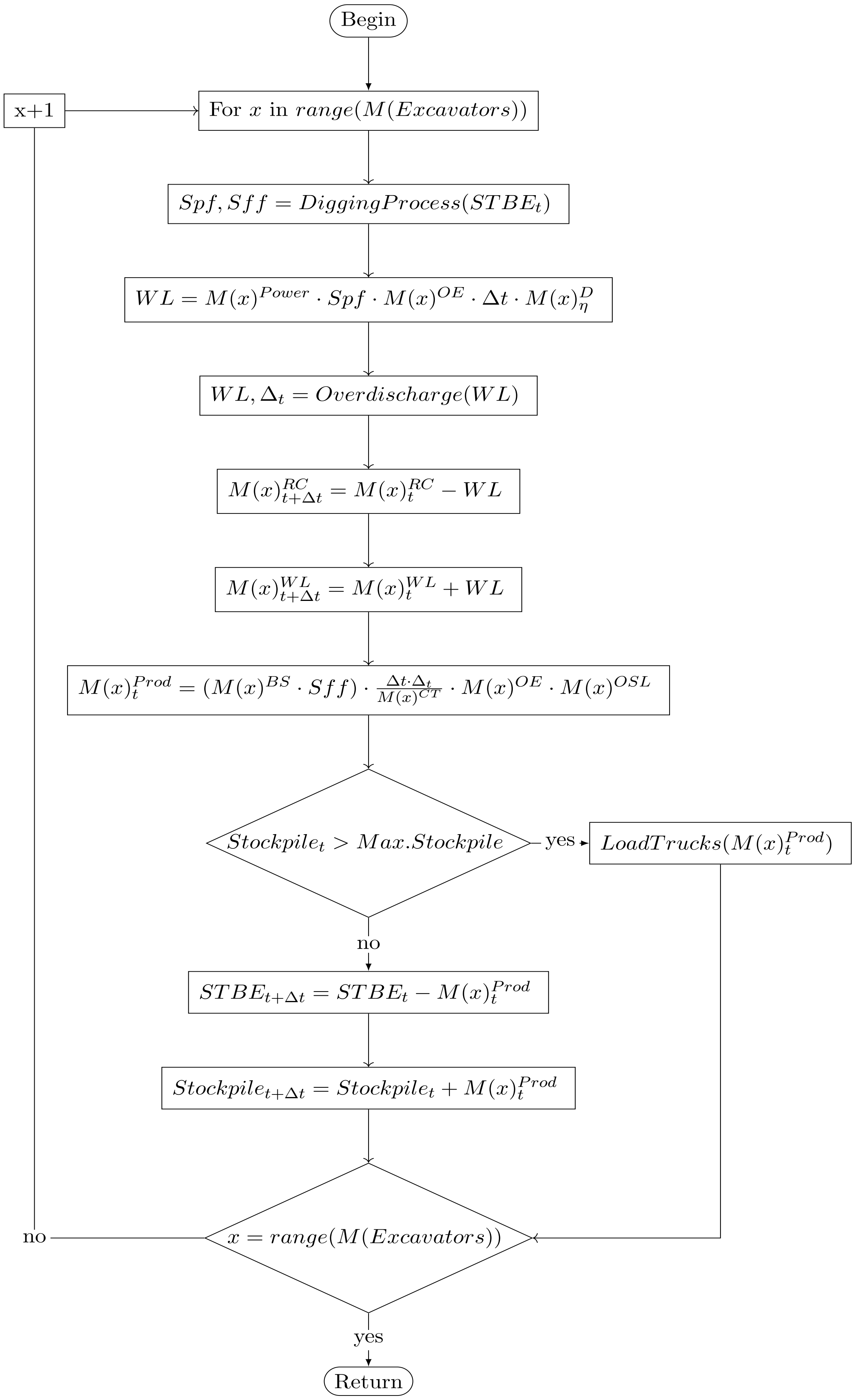

Begin
For x in $range(M(Excavators))$
$Spf, Sff = DiggingProcess(STBE_t)$
$WL = M(x)^{Power} \cdot Spf \cdot M(x)^{OE} \cdot \Delta t \cdot M(x)^{D}_{\eta}$
$WL, \Delta_t = Overdischarge(WL)$
$M(x)^{RC}_{t+\Delta t} = M(x)^{RC}_{t} - WL$
$M(x)^{WL}_{t+\Delta t} = M(x)^{WL}_{t} + WL$
$M(x)^{Prod}_{t} = (M(x)^{BS} \cdot Sff) \cdot \frac{\Delta t \cdot \Delta_t}{M(x)^{CT}} \cdot M(x)^{OE} \cdot M(x)^{OSL}$
$Stockpile_t > Max.Stockpile$
yes
$LoadTrucks(M(x)^{Prod}_{t})$
no
$STBE_{t+\Delta t} = STBE_t - M(x)^{Prod}_{t}$
$Stockpile_{t+\Delta t} = Stockpile_t + M(x)^{Prod}_{t}$
$x = range(M(Excavators))$
no
x+1
yes
Return

Figure 6.8: *Simulated working of excavators when not idle*

Where $STBE_t$ is the remaining volume of soil to be excavated in m^3. $DiggingProcess$ is a separate function that checks the volume of soil that has already been excavated during the episode and returns the current soil power factor Spf and the soil fill factor Sff. The $DiggingProcess$ is shown in figure 6.9 below.

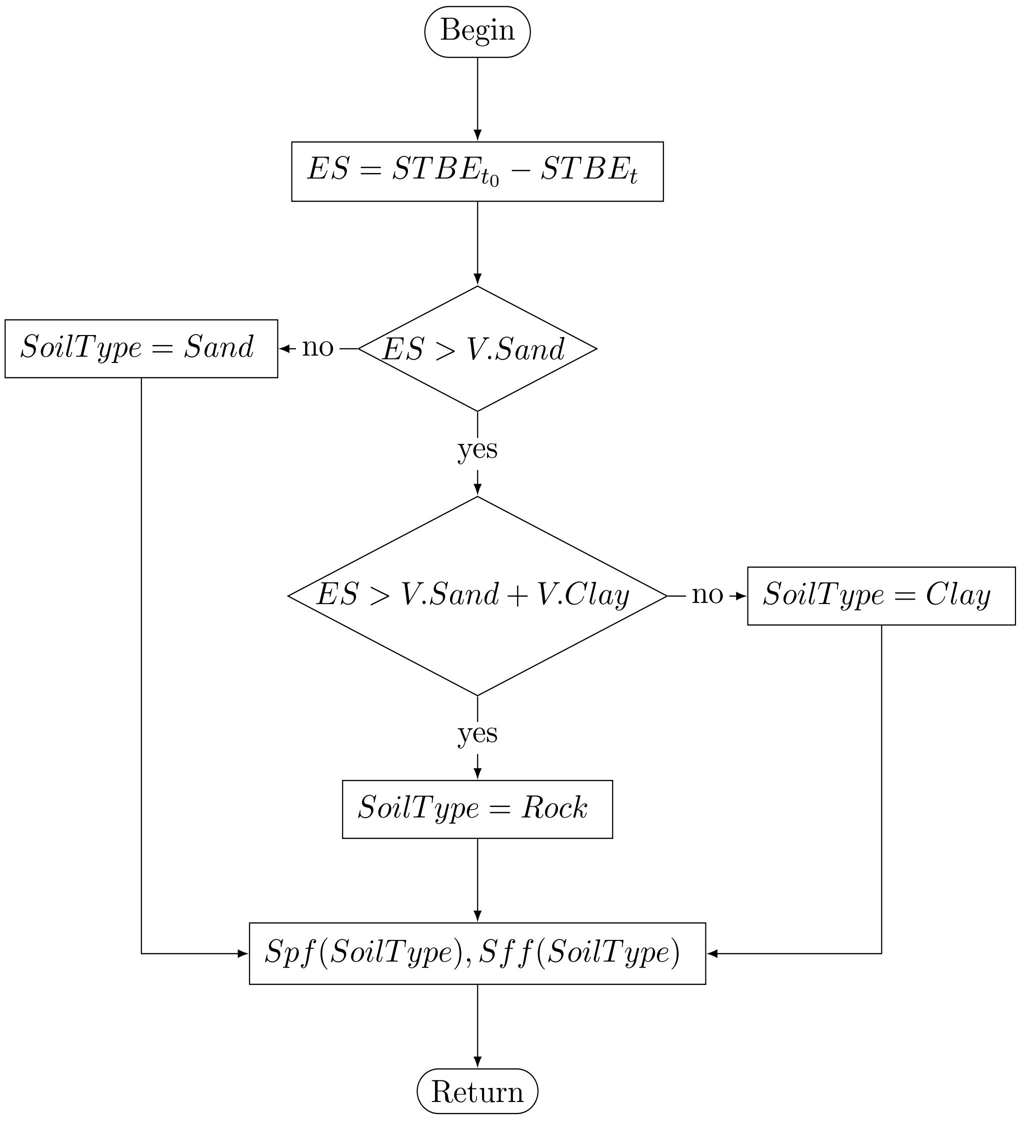


Figure 6.9: *Digging Process Function*

Where ES is the current volume of excavated soil in m^3, $V.Sand$ is the volume of sand in m^3 to be excavated, and $V.Clay$ is the volume of clay in m^3 to be excavated.

Furthermore, $LoadTrucks()$ is another separate function that is activated if the volume of the stockpiled soil exceeds the maximum allowed volume. This function simulates the loading of soil into trucks. The function is based on the first-in first-out (FIFO) queuing principle, where the first available truck is selected for loading. This flow of operations for this function is presented in Figure 6.10 below.

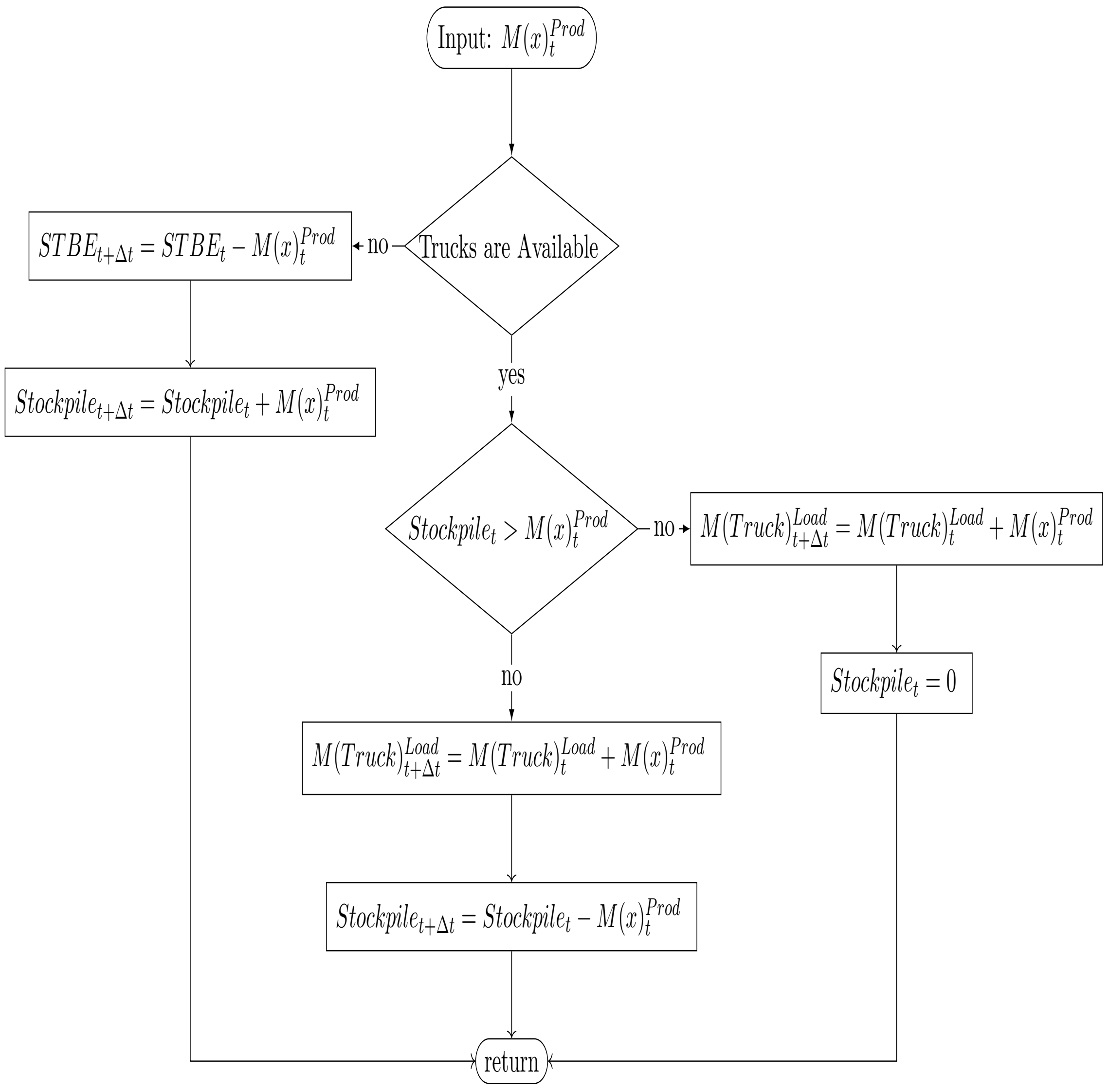

Input: $M(x)_t^{Prod}$
Trucks are Available
no
yes
$STBE_{t+\Delta t} = STBE_t - M(x)_t^{Prod}$
$Stockpile_{t+\Delta t} = Stockpile_t + M(x)_t^{Prod}$
$Stockpile_t > M(x)_t^{Prod}$
no
$M(Truck)_{t+\Delta t}^{Load} = M(Truck)_t^{Load} + M(x)_t^{Prod}$
$Stockpile_t = 0$
no
$M(Truck)_{t+\Delta t}^{Load} = M(Truck)_t^{Load} + M(x)_t^{Prod}$
$Stockpile_{t+\Delta t} = Stockpile_t - M(x)_t^{Prod}$
return

Figure 6.10: *Function of the loading process*

Where $M(Truck)_t^{load}$ is the current loaded capacity of the first available truck. If no trucks are available at the current time fraction, the loader or excavator is allowed to stockpile the excavated soil.

6.5.2.2 Loader Workload Function

The loader function is quite similar to the excavator function. However, the main goal of the loader is to carry out the loading process 6.10. The flow of operations of the loader function is presented in Figure 6.11 below.

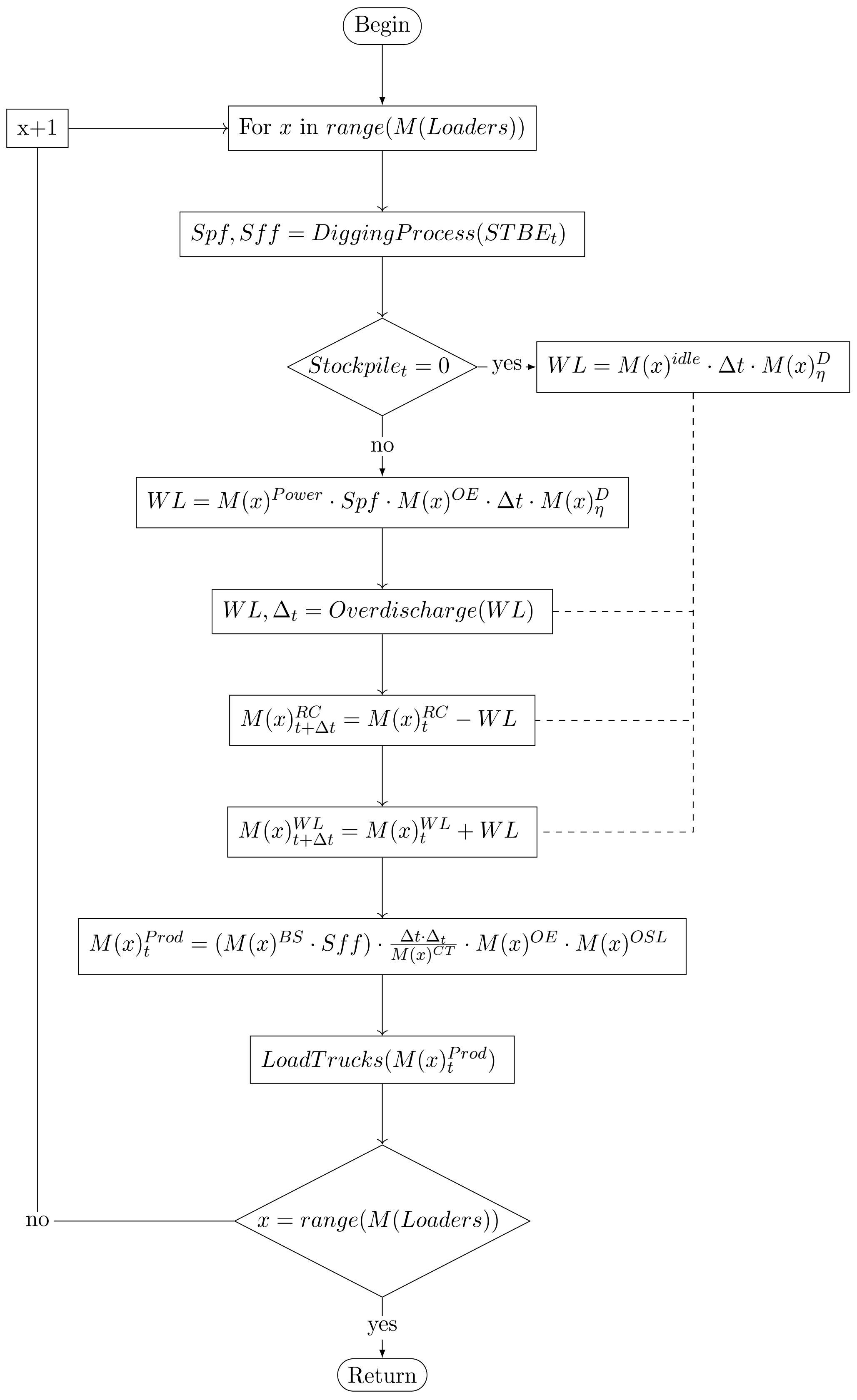

Begin
x+1
For x in $range(M(Loaders))$
$Spf, Sff = DiggingProcess(STBE_t)$
$Stockpile_t = 0$
yes
$WL = M(x)^{idle} \cdot \Delta t \cdot M(x)^D_\eta$
no
$WL = M(x)^{Power} \cdot Spf \cdot M(x)^{OE} \cdot \Delta t \cdot M(x)^D_\eta$
$WL, \Delta_t = Overdischarge(WL)$
$M(x)^{RC}_{t+\Delta t} = M(x)^{RC}_t - WL$
$M(x)^{WL}_{t+\Delta t} = M(x)^{WL}_t + WL$
$M(x)^{Prod}_t = (M(x)^{BS} \cdot Sff) \cdot \frac{\Delta t \cdot \Delta_t}{M(x)^{CT}} \cdot M(x)^{OE} \cdot M(x)^{OSL}$
$LoadTrucks(M(x)^{Prod}_t)$
no
$x = range(M(Loaders))$
yes
Return

Figure 6.11: *Simulated working of loaders*

This function forces the loaders to idle if the volume of stockpiled soil becomes 0 during the simulation.

6.5.2.3 Truck Workload Function

The final machine-specific workload function developed for this project is the truck function. This function simulates that the trucks are idle while being loaded. However, when a truck reaches its maximum loading capacity, the truck begins its journey to the dump site. The time required to reach the dump site and return to the site is the quotient of the round-trip distance to the dump site and the average hauling speed of the trucks.

For each iteration, the simulated time fractions are deduced from the time required for the round-trip. Therefore, trucks that have started the round-trip trip will not be available in the loading process 6.10. The truck function is presented in Figure 6.12.

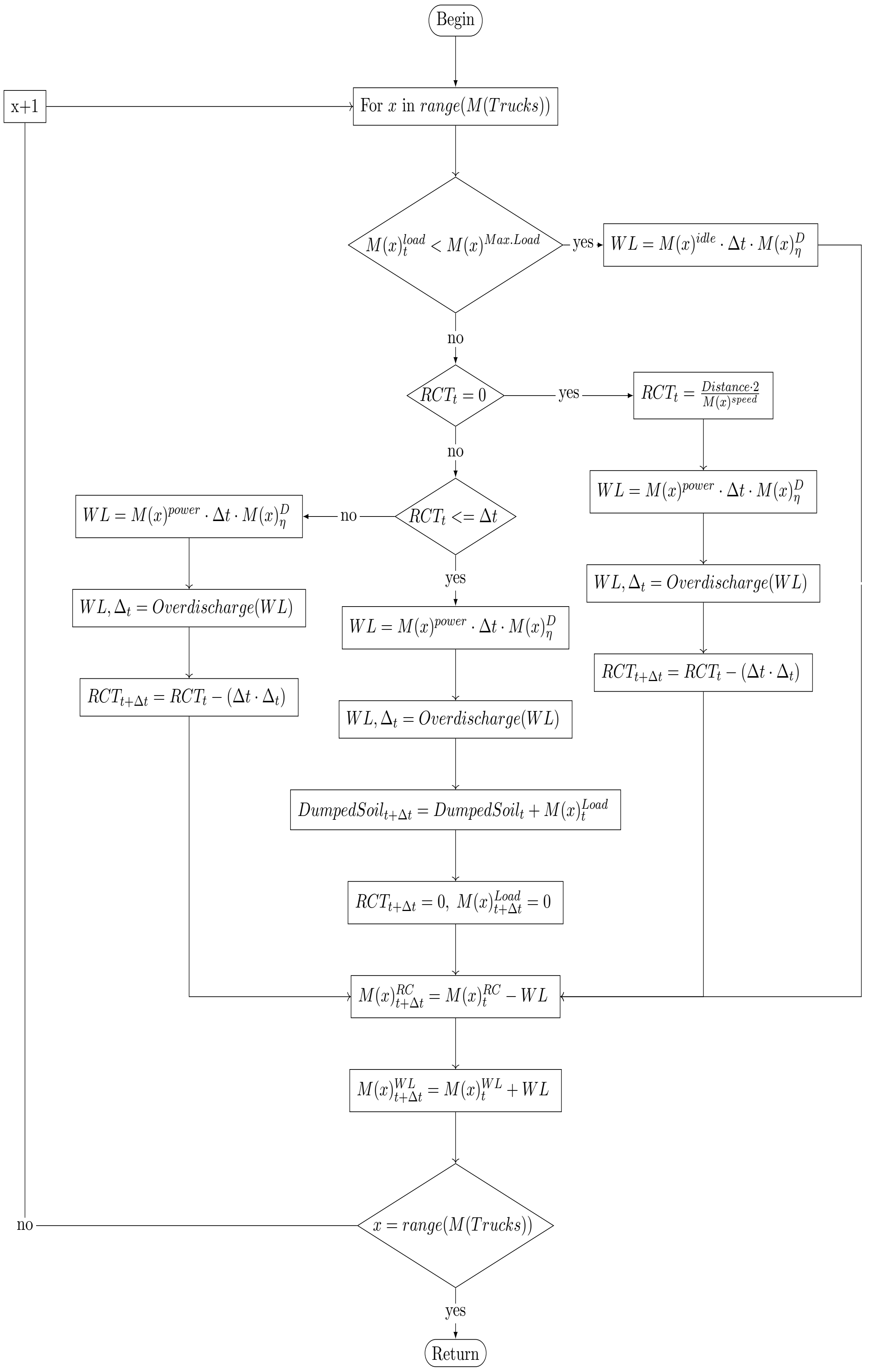

Begin
x+1
For x in $range(M(Trucks))$
$M(x)_t^{load} < M(x)^{Max.Load}$
yes
$WL = M(x)^{idle} \cdot \Delta t \cdot M(x)_\eta^D$
no
$RCT_t = 0$
yes
$RCT_t = \frac{Distance \cdot 2}{M(x)^{speed}}$
no
$WL = M(x)^{power} \cdot \Delta t \cdot M(x)_\eta^D$
$RCT_t <= \Delta t$
no
$WL = M(x)^{power} \cdot \Delta t \cdot M(x)_\eta^D$
$WL, \Delta_t = Overdischarge(WL)$
yes
$WL, \Delta_t = Overdischarge(WL)$
$WL = M(x)^{power} \cdot \Delta t \cdot M(x)_\eta^D$
$RCT_{t+\Delta t} = RCT_t - (\Delta t \cdot \Delta_t)$
$RCT_{t+\Delta t} = RCT_t - (\Delta t \cdot \Delta_t)$
$WL, \Delta_t = Overdischarge(WL)$
$DumpedSoil_{t+\Delta t} = DumpedSoil_t + M(x)_t^{Load}$
$RCT_{t+\Delta t} = 0,\ M(x)_{t+\Delta t}^{Load} = 0$
$M(x)_{t+\Delta t}^{RC} = M(x)_t^{RC} - WL$
$M(x)_{t+\Delta t}^{WL} = M(x)_t^{WL} + WL$
no
$x = range(M(Trucks))$
yes
Return

Figure 6.12: *Flow of operations for the truck function*

Where $M(x)^{MaxLoad}$ is the maximum loading capacity of a truck and RCT_t is remaining time required for the round-trip journey to the dump site. This function assures that when RCT_t is reduced to 0, the loaded soil is dumped at the dump site and the loaded capacity of the truck $M(x)_t^{Load}$ is set to 0.

These function terminates the action ends and all the machines available to work have been iterated over. The code for the workload functions are available in the Appendix A.5.

6.5.3 Work Dependency Method

The work dependency method allows the user to add and customize work dependencies to simulate unique events during an episode. The structure of the work dependency is subject to the user. However, an example is provided to explain its functionality.

When the capacity of the main dump site is full, the trucks have to be redirected to another dump site that is further away.

In this example, the work dependency can be a conditional statement that changes the static value that represents the distance to the main dump site to a new static value that represents the distance to another dump site.

Furthermore, the method ensures that each work dependency occurs only one time during an episode by checking a variable that is initially set to ”not complete” for each work dependency. Following the activation of any work dependency, the ”not complete” variable is updated to ”complete” to ensure that it is only activated ones during the episode. The dynamics of this method is visualized by using the work dependency example in Figure 6.13 below.

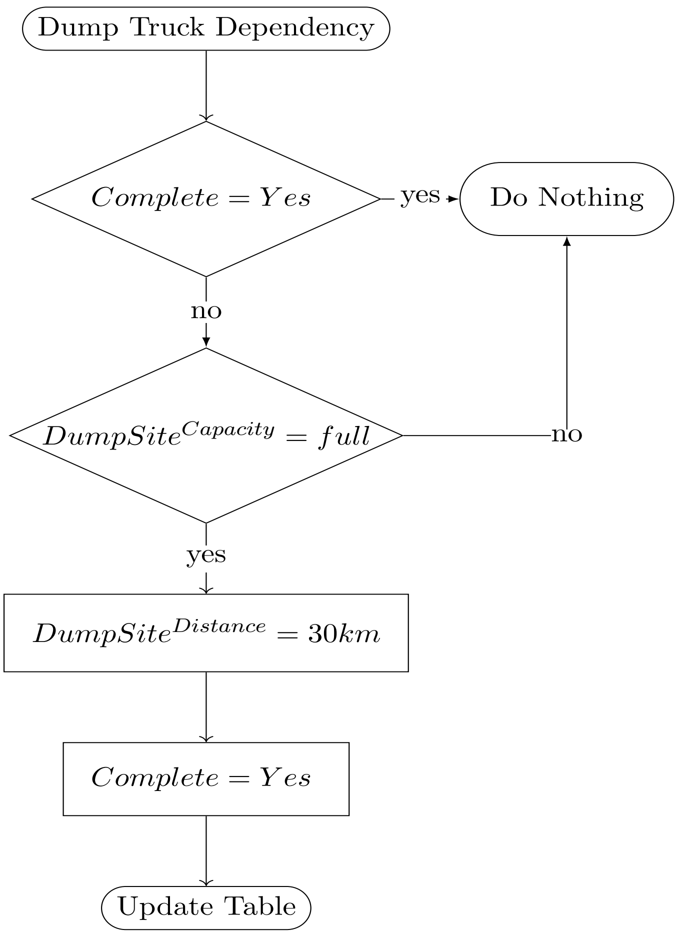


Figure 6.13: *Example of a work dependency*

The code for the work dependency example is available in the Appendix A.9.

6.6 Environment Structure

The environment compiles all previously explained methods and functions into a structure that value-based reinforcement learning agents can interact with. The structure of this environment is formulated to fulfill the requirements of OpenAI's Gym, which ensures that the environment remains comprehensible for multiple existing and future reinforcement learning agents [105]. The environment structure begins with an initialization process, followed by a step process, and resets with a reset process.

6.6.1 Initialization process

The main purpose of the initialization process is to initialize the default values that must be declared before any episode can begin. These values include the influencing factors presented in Section 6.1. These values are set by the user and represent the dynamic variables and static values that the agent must comply with when interacting with the step process.

6.6.1.1 Action and State Space

In this process, the dimensions of the action and state spaces are established to let the agent know the number of actions it can choose from and the number of state signals it is required to observe in every time step.

The action space is the sum of the available actions. This number is equal to twice the number of machines selected by the user since there are two discharge actions per machine, plus 2 to represent the charge action and the waiting action of the BESS. The size of the action space is automatically detected when the user specifies the fleet configuration by equation 6.15 below.

$$ActionSpace = (M(n) \cdot 2) + 2 \tag{6.15}$$

Furthermore, the state space is the sum of all observable state signals. This number also depends on the number of machines selected by the user, as explained in Section 6.2. The state space is set equal to twice the number of machines in the fleet configuration to represent the remaining capacity and workload signals, plus one that represents the remaining capacity of the BESS.

Four productivity state signals are added if the user wants to minimize the time required to achieve a productivity goal. If the user wants to minimize the electricity cost, five additional signals are added to the state space representing the hour of the day, day of the month, week of the year, month of the year, and the electricity price of the hour. The size of the state space is automatically detected by equation 6.16 below.

$$StateSpace = \begin{cases} (M(n) \cdot 2) + 2 & \text{Maximize Workload} \\ (M(n) \cdot 2) + 2 + 4 & \text{Minimize Time Required} \\ (M(n) \cdot 2) + 2 + 4 + 5 & \text{Minimize Electricity Cost} \end{cases} \tag{6.16}$$

6.6.1.2 Dynamic Values

For training purposes, the initialization process can set the state space signals to semi-stochastic values within its corresponding limits for the initial time step in the step process. The purpose of using semi-stochastic state signals at the beginning of an episode is to make every episode unique. This forces the agent to explore more state-action combinations to increase its ability to provide high-quality generalization for unseen states.

The initial remaining capacity of the BESS can be set to a semi-stochastic value, bounded by its maximum and minimum capacities. This is done to resemble real-world scenarios in which the remaining charge of the BESS is defined by previous work, transportation, or overnight charging. This operation is expressed as follows.

$$B_{t_0}^{RC} = Random(B^{MinCap}, B^{MaxCap}) \tag{6.17}$$

Where $B_{t_0}^{RC}$ is the remaining capacity of the BESS in the initial time step of the step process.

The remaining capacity state signals of the earthwork machines are set by the earthwork scenario generator in this process to produce a unique earthwork scenario for every episode. A semi-stochastic remaining charge can be set for every machine, bound by its maximum and minimum capacities, similarly to Equation 6.17.

Furthermore, the productivity goal and various influencing factors can be set to stochastic values if the user does not have access to values that accurately represent the impact of certain factors on the earthwork process. However, all of these values can be set to a static value if the user wants to train the model with a specific starting point.

6.6.1.3 Time Steps

The user defines the number of time steps of every episode by setting a time step limit, which will terminate the episode when reached. To minimize the time required to achieve a productivity goal, a maximum number of time steps is not required, as the episode will terminate when the productivity goal is reached. However, the user must provide a time step limit when maximizing workloads because the limited time frame will motivate the agent to explore more efficiently.

Furthermore, to minimize the electricity cost, the user must define a time and date range on which the framework should train. If the user wants to train the framework on data for a full year, the time and date range must include all 8760 hours of a normal year. However, training can be limited to certain periods within a year by setting this range between the hours of specific dates.

The user can also specify the hour of the day at which every episode will begin. Additionally, the framework stores the user-provided hourly electricity price data in this process. It is important to note that the user must provide hourly price data that correspond to the selected time and date range.

For training purposes, the state signals for the time and date range are set by choosing a random day of the year within the given time and date range. This operation forces the agent to explore different time and date ranges with various electricity prices for each episode. If the time and date range is a full year and every episode should begin at the 9_{th} hour of a day. The initial time step is expressed as follows.

$$t_0 = Random(9, 8736, 24) \tag{6.18}$$

Where t_0 is the initial time step of the step process, *Random* is an operation that chooses a number stochastically in the given range of 9 to 8736 by increments of 24, where 9 is the eighth hour of the year and 8736 is the first hour of the last day of a normal year. t_0 modifies the values of the state signals: hour of year, day of year, week of year, month of year, and current electricity price.

The full dynamics of the initialization process is shown in Figure 6.14.

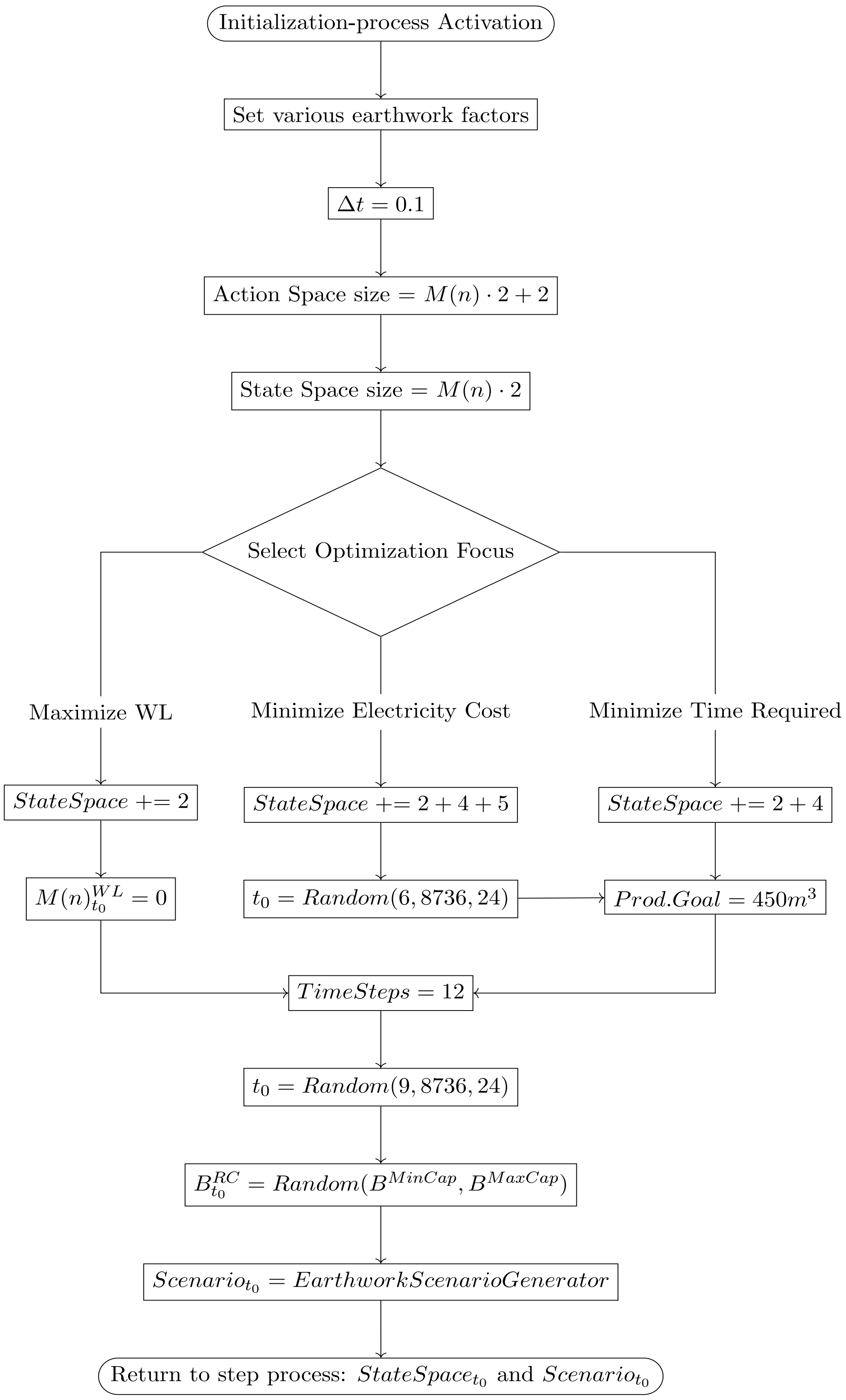

Initialization-process Activation
Set various earthwork factors
$\Delta t = 0.1$
Action Space size = $M(n) \cdot 2 + 2$
State Space size = $M(n) \cdot 2$
Select Optimization Focus
Maximize WL
Minimize Electricity Cost
Minimize Time Required
$StateSpace \mathrel{+}= 2$
$StateSpace \mathrel{+}= 2 + 4 + 5$
$StateSpace \mathrel{+}= 2 + 4$
$M(n)^{WL}_{t_0} = 0$
$t_0 = Random(6, 8736, 24)$
$Prod.Goal = 450m^3$
$TimeSteps = 12$
$t_0 = Random(9, 8736, 24)$
$B^{RC}_{t_0} = Random(B^{MinCap}, B^{MaxCap})$
$Scenario_{t_0} = EarthworkScenarioGenerator$
Return to step process: $StateSpace_{t_0}$ and $Scenario_{t_0}$

Figure 6.14: *Initialization process for the proposed framework*

The code for this process is available in the Appendix A.12.

6.6.2 Step process

The step process defines the agent's interactions with the environment for each time step of every episode. The step process begins with receiving a number that represents the action selected by the agent for the initial time step.

If the action-number is equal to twice the number of selected machines plus one, the agent has selected the waiting action, and the waiting process is simulated as explained in Section 6.3.3. If the action number is greater than twice the number of selected machines plus one, the agent has selected the charge action, and the charging process is simulated as explained in Section6.3.1. If the action-number is less than or equal to twice the number of selected machines, the agent has selected a discharge action, and the discharge process is simulated for the specific machine(s) selected as explained in Section 6.3.2.

If the environment is configured to solve the tasks of minimizing electricity cost, a action-dependent reward is calculated based on the result of the action performed in the time step, as explained in 6.4.3. If the user has implemented any work dependencies, the work dependencies are checked to see if any should be fulfilled at the current time step, as explained in Section 6.5.3.

Following this process, all updated states are recorded. This logged information is used to generate schedules for the user to interpret and visualize how the agent was able to optimize the task.

The final process is to check if the current time step is equal to the final time step of the episode. If it is not the final time step, the step process returns the reward and updated state space to the agent, and the step process repeats itself for the following time step. If it is the final time step, the step process terminates and a final reward is calculated and returned to the agent together with the final state space.

The dynamics of the step process configured to minimize electricity cost is shown in figure 6.15 below.

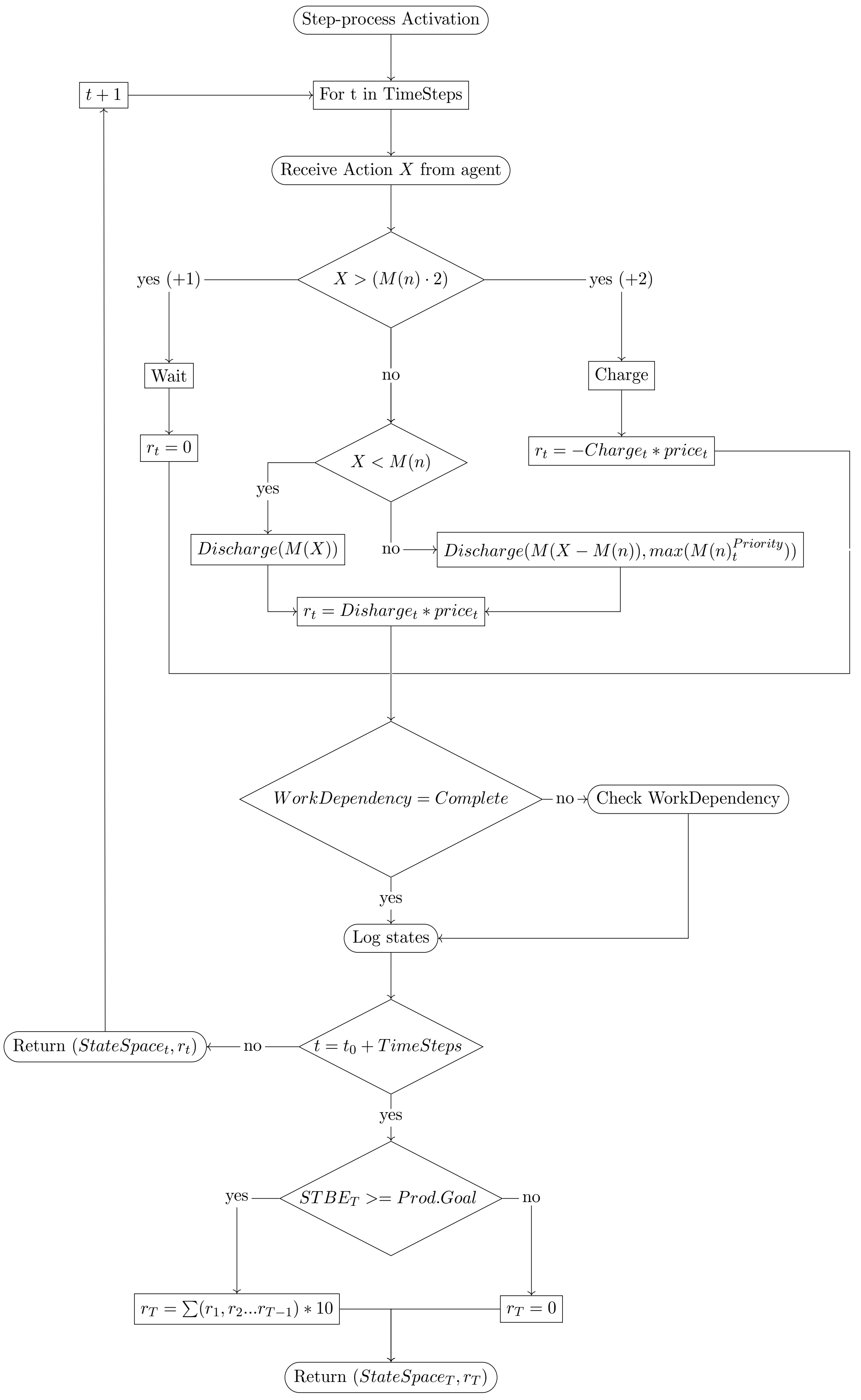

Step-process Activation
For t in TimeSteps
$t+1$
Receive Action X from agent
$X > (M(n) \cdot 2)$
yes (+1)
yes (+2)
Wait
no
Charge
$r_t = 0$
$X < M(n)$
$r_t = -Charge_t * price_t$
yes
$Discharge(M(X))$
no
$Discharge(M(X - M(n)), max(M(n)_t^{Priority}))$
$r_t = Disharge_t * price_t$
$WorkDependency = Complete$
no
Check WorkDependency
yes
Log states
Return $(StateSpace_t, r_t)$
no
$t = t_0 + TimeSteps$
yes
yes
$STBE_T >= Prod.Goal$
no
$r_T = \sum(r_1, r_2 ... r_{T-1}) * 10$
$r_T = 0$
Return $(StateSpace_T, r_T)$

Figure 6.15: *Step process for solving the task of minimizing electricity cost*

The code for this process is available in the Appendix A.12.

6.6.3 Reset process

The reset process is activated after the completion of the step process to reset the environment and provide the agent with a new independent episode. This process continues until a user-specified number of training episodes is completed.

Similarly to the initialization process, the reset process can set the state signals to semi-stochastic values for the initial time step of the step process. Therefore, the reset process can provide the agent with new earthwork scenarios generated by the earthwork scenario generator, as explained in Section 6.5.1. It can also set the remaining capacity of the BESS to a stochastic value with Equation 6.17. However, all static factors defined in the initialization process remain unchanged.

To solve the tasks of minimizing the cost of electricity, a new day is selected within the given time and date range with Equation 6.18 to expose the agent to new electricity price patterns. To solve the task of maximizing workloads, the completed workloads for the initial time step of the new episode are set to zero.

The purpose of resetting the semi-stochastic values for each episode is to force the agent to explore new action-state combinations during training and expose the agent to new earthwork scenarios. This allows the agent to learn a more robust policy that can be used to generalize its response to unseen scenarios. If the initial state signals remain the same for every episode during training, the agent's policy may become overfitted, and the trained agent will fail at maximizing its return when applied to unseen scenarios.

The dynamics of the reset process is shown in Figure 6.16 below.

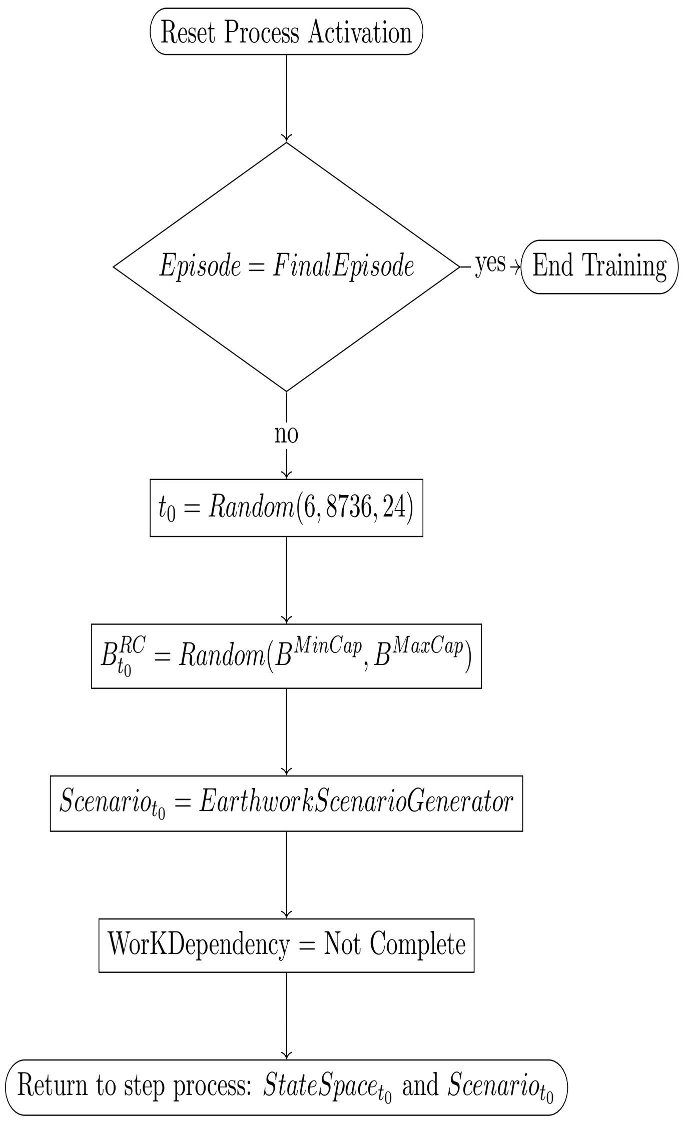

Reset Process Activation
$Episode = FinalEpisode$
yes
End Training
no
$t_0 = Random(6, 8736, 24)$
$B^{RC}_{t_0} = Random(B^{MinCap}, B^{MaxCap})$
$Scenario_{t_0} = EarthworkScenarioGenerator$
WorKDependency = Not Complete
Return to step process: $StateSpace_{t_0}$ and $Scenario_{t_0}$

Figure 6.16: *Reset process for solving the task of minimizing electricity cost*

The code for this process is available in the Appendix A.12.

6.7 Agent

The type of agent selected for this framework is Deep Mind's Deep Q-Network Agent (DQN) [106]. This agent was chosen based on the experience detailed in Kwon and Zhu's paper [98], and because it is a well-proven value-based agent that utilizes a fully connected artificial neural network as a critic to approximate an optimal action-value function, which is expressed as follows.

$$q^*(s, a) = \max_{\pi} \mathbb{E}[r_1 + \gamma r_{t+1} + \gamma^2 r_{t+2} + ...|s_t = s, a_t = a, \pi] \qquad (6.19)$$

Where $q^*(s, a)$ maximizes the expected sum of rewards $\mathbb{E}$ discounted by γ for each time step.

Furthermore, using a DQN agent can significantly reduce the time it takes an agent to learn a high-quality policy because it periodically updates its policy with random samples from the agent's experience with the environment, which reduces training instabilities caused by constant small updates of the policy and the computational burden of assessing all discovered states, actions, and rewards when updating the policy.

This process is achieved by storing the agent's discovered states, actions, and rewards in a data set D, dividing the data set into samples U, and updating the weights and biases of the artificial neural network θ by applying a loss function on samples chosen uniformly at random. The loss function for the DQN agent is expressed as follows.

$$L_i(\theta_i) = \underset{(s,a,r,s')\sim U(D)}{\mathbb{E}} \left[\left(r + \gamma \max_{a'} q(s', a'|\theta_i^-) - q(s, a|\theta_i) \right)^2 \right] \tag{6.20}$$

Where $L_i(\theta_i)$ is the loss function (δ_t) for the parameters θ in iteration i, $\mathbb{E}_{(s,a,r,s')\sim U(D)}$ is the expected value of a sample in the data set $U \in D$ in terms of the difference between a target reward $r + \gamma \max_{a'} q(s', a'|\theta_i^-)$ and the actual reward $q(s, a|\theta_i)$

The occurrence of updating the weights by the samples is calculated using Diederik P. Kingma and Jimmy Lei Ba's Adam Optimizer [107]. Adam Optimizer was chosen because it is widely adopted for deep learning applications and has become a standard for stochastic gradient-based optimization for deep reinforcement learning agents. The benefit of using Adam over a standard stochastic gradient descent method is that it provides a learning rate for each of the network's weights and biases and adapts it separately as learning unfolds. However, Adam requires an initial learning rate to base its adaptations from. For this project, the initial learning rate was set at 0.001 because higher learning rates were found to make the framework unstable and slow, while lower learning rates did not allow the agent to adapt to minor changes in state signals.

6.7.1 Network Architecture

The architecture of the artificial neural network made available for the DQN agent in this framework consists of an input layer with the number of neurons representing the state signals of the environments configuration, two hidden layers with 24 neurons in each layer, and an output layer with a number of neurons dependent on the number of machines simulated representing the actions. The number of neurons in the hidden layers was subject to trial and error.

The number of adjustable parameters is bound by the number of neurons in each connected layer. The number of adjustable weights in the first hidden layer is the product of neurons in the input layer and neurons in the first hidden layer. The number of adjustable biases is equal to the number of neurons in the first hidden layer. However, the number of neurons in the input layer must be equal to the size of the state space, which depends on the number of simulated machines. Therefore, the number of adjustable parameters in the first hidden is calculated as follows.

$$Adjustable\ Parameters\ 1_{st}\ hidden\ layer = (M(n) \cdot 2 + X) \cdot 24 + 24 \tag{6.21}$$

Where X is the number of state signals for the configuration of the environment.

The number of adjustable parameters in the second hidden layer is the product of the number of neurons in the first hidden layer and the number of neurons in the second hidden layer, in addition to 24 adjustable biases. Therefore, the number of adjustable parameters in the second hidden layer is calculated as follows.

$$Adjustable\ Parameters\ 2_{nd}\ hidden\ layer = 24 \cdot 24 + 24 \tag{6.22}$$

The number of adjustable parameters in the output layer is the product of the number of neurons in the second hidden layer and the neurons in the output layer, plus the number of adjustable biases, which is equal to the number of actions. Therefore, the adjustable parameters for the output layer are calculated as follows.

$$Adjustable\ Parameters\ output\ layer = 24 \cdot (M(n) \cdot y + 2) + (M(n) \cdot y + 2) \tag{6.23}$$

Where y is the number of charging cables available in the configuration of the environment.

The ReLU function was chosen as the activation function for both hidden layers because it is generally more efficient than the Sigmoid function. The activation function for the output layer is a linear regression function because it predicts actions based on unbounded quantities. The code for the training process is available in the Appendix A.14.

6.7.2 Training the agent

The number of episodes in which the agent is required to train in a configuration of the environment to learn a high-quality policy is estimated by training the agent for a substantial number of episodes and plotting the returns of each episode. An optimized policy has been found if the returns from the episodes have reached a steady rate. Therefore, the number of episodes can be reduced to the number required to reach this steady rate of return. However, the number of training episodes should be increased if returns are still approaching a steady rate, as this indicates that the policy can be further improved by training.

Figure 6.17 below shows an example of training a DQN agent for 3000 episodes in an environment where a reward of approximately 0 indicates that the task is solved.

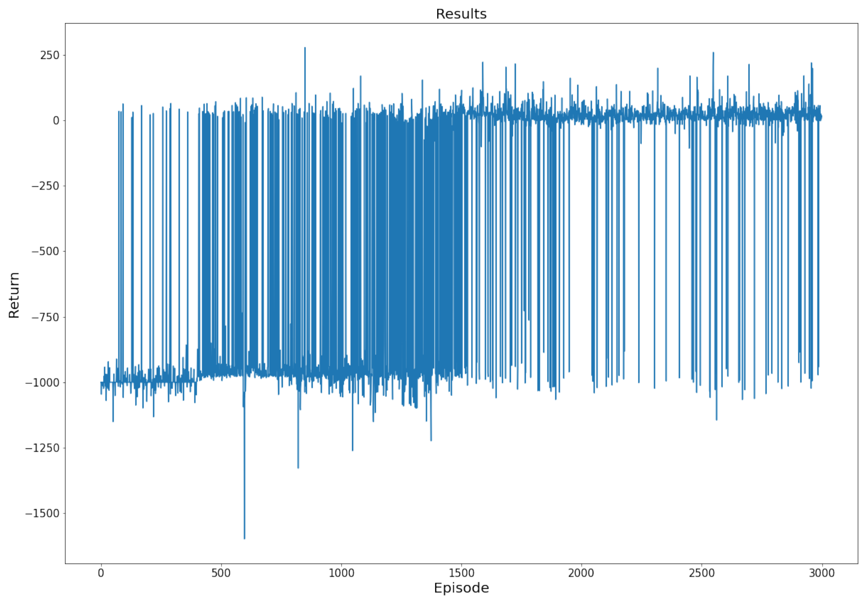


Figure 6.17: *Returns from training an agent*

Figure 6.17 shows that the agent was exploring different policies to familiarize itself with the environment in the first 500 episodes. In the following 1000 episodes, the agent was able to develop certain policies that were able to solve the task in certain scenarios. However, it still explored different variations of this policy to improve the consistency of the returns. Between episodes 1500 and 3000, the agent developed an optimized policy that was able to reach a steady rate of return. However, the performance is somewhat reduced between episodes 2500 and 3000. Therefore, this agent should train in this environment for approximately 2500 episodes. The code for this visualization process is available in the Appendix A.15.

6.8 Methodology for using the trained framework

There are several real-world applications for this deep reinforcement learning framework. The main application is to provide earthwork professionals with a decision support tool to plan efficient electric earthwork processes by generating optimized energy supply schedules for electric earthwork processes.

The process of generating an optimized schedule is achieved in 7 steps.

1. Select the optimization goal and configure the environment.
2. Select the type and number of machines to be simulated.
3. Set the user-specific static and semi-stochastic variables that represent influencing factors of real-world earthwork scenarios.
4. Train the agent in the configured environment.
5. Substitute the semi-stochastic values in the environment with static values that represent a specific scenario to be optimized.
6. Log the dynamics of the BESS and earthmoving machines for each time step when applying the trained agent in the static environment for one episode.
7. Plot the data logged from the previous step and review.

For different variations of the schedule, steps 5, 6, and 7 can be repeated. However, the user must repeat all steps if the user wants to optimize a different goal or simulate other machines. The codes for logging and visualizing the generated schedules are available in the Appendix A.17 and A.11.

Furthermore, this framework can be used to generate optimized schedules during the earthwork process. The purpose of this application is to assist earthwork professionals to decide which machine(s) has the highest priority to charge during the work day.

This is achieved by applying a trained agent's policy on a version of the environment that is modified to represent the actual earthwork scenario at the current hour.

However, the trained agent must have been trained in a version of the training environment that represents the semi-stochastic nature of the intended earthwork site.

To modify the environment to represent the current earthwork scenario, the semi-stochastic variables in the environment's initialization process 6.6.1 and reset process 6.6.3 must be replaced with actual values or sensor readings from the earthmoving machines and BESS. The time and date range must be set to the actual date and time of the working day, and the remaining time steps must be defined to represent the remaining hours of the specific working day. Additionally, expected electricity prices can be collected directly from a utility company's intraday prognosis.

This process can be automated by developing a version of the environment that is able to collect sensor readings from machines and BESS through a data acquisition and supervisory control system. However, no such system has been developed yet for electric earthwork processes.

6.9 Other Use cases

The framework can be applied to solve related optimization problems in the electric earthwork field. This section presents three novel applications that validate the generalizable performance and real-world applicability of the framework.

6.9.1 Optimized Overnight Charging of BESS and Machines

The environment can be configured to optimize the overnight charging of BESS and machines. The purpose of this application is to optimize the schedule to charge the BESS and the machines to their maximum capacity during longer periods when there are no earthwork activities. This configuration is set to minimize the electricity cost by automatically charging the BESS during hours when the price of electricity is comparatively lower than the hours when the agent is discharging it to charge the machines.

This is achieved by training the agent in a modified version of the environment configured to solve the task of minimizing the cost of electricity, where the workload functions are deactivated. Furthermore, the remaining capacities of the BESS and machines are set to low values for the initial time step of each episode to force the agent to select the charge action multiple times.

The action-dependent reward signals are the same as described in Section 6.4.3. However, the completion of a productivity goal reward signal 6.11 is substituted for a reward signal that rewards the agent if the machines and BESS are fully charged in the final time step. This modified reward signal is expressed as follows:

$$r_T(Fully\ Charged\ Reward) = \sum(M(n)_T^{RC}) + B_T^{RC} + \sum(r_1, r_2...r_{T-1}) \tag{6.24}$$

Where $\sum(M(n)_T^{RC})$ is the sum of the remaining capacity of all machines in the final time step, B_T^{RC} is the remaining capacity of the BESS in the final time step, and $\sum(r_1, r_2...r_{T-1})$ is the sum of all rewards previously received before the final time step.

This reward signal motivates the agent to maximize the remaining capacities of the machines and BESS by the final time step, and it considers the action-dependent rewards received in previous time steps in an effort to motivate the agent to minimize electricity cost.

The challenge with this reward signal is that the importance of the action-dependent rewards are reduced to a sub-goals. Therefore, performance is unreliable in scenarios where the agent struggles to completely recharge the BESS and machines.

After training the agent in the modified training environment, the trained agent can generate schedules that optimize overnight charging process. This application can be automated using a rule-based method that executes the schedule generated by the framework. However, this only works if the electricity prices for the charging period are known. If the electricity prices in the period are unknown, the agent must have direct control of the BESS to apply its trained generalization in real time to adapt to unseen electricity price patterns as they are updated.

6.9.2 Estimating the minimum required capacity of a BESS

This framework can be used to estimate the minimum required capacity of a earthwork BESS. The purpose of this application is to help earthwork professionals estimate the minimum required energy capacity of an earthwork BESS. This is an important use case because earthwork professionals can use it to avoid overinvesting in BESS technology and subsequently increase the economic feasibility of an electrified earthwork site.

Finding the minimum required capacity of a BESS can be achieved manually following the steps to generate an optimized schedule as outlined in Section 6.8. When the agent has developed a robust policy, the user can substitute the value representing the maximum capacity of BESS B^{MaxCap} in the training environment initialization process and repeat steps 6 and 7. The user can then learn if the updated B^{MaxCap} is capable of generating returns similar to the original B^{MaxCap} used during training. However, this approach is subject to manual trial and error.

Therefore, an estimation method has been developed to automate the task of finding the minimum required capacity of an earthwork BESS, inspired by the method presented in the paper ”Battery Optimal Sizing under a Synergistic Framework with DQN Based Power Managements for the Fuel Cell Hybrid Powertrain” by Li et al. [100].

This method consists of a function that automates the task of finding the minimum required capacity by sequentially testing a trained agent in its training environment over a user-specified of episodes. Where B^{MaxCap} is initially set to zero and is increased by a user-specified value for each consecutive sequence of episodes. The function terminates when the average return of a sequence indicates that the updated value of B^{MaxCap} consistently generates similar optimization results as the original B^{MaxCap} used during training.

Figure 6.18 shows the dynamics of the estimation function to find the minimum required BESS capacity.

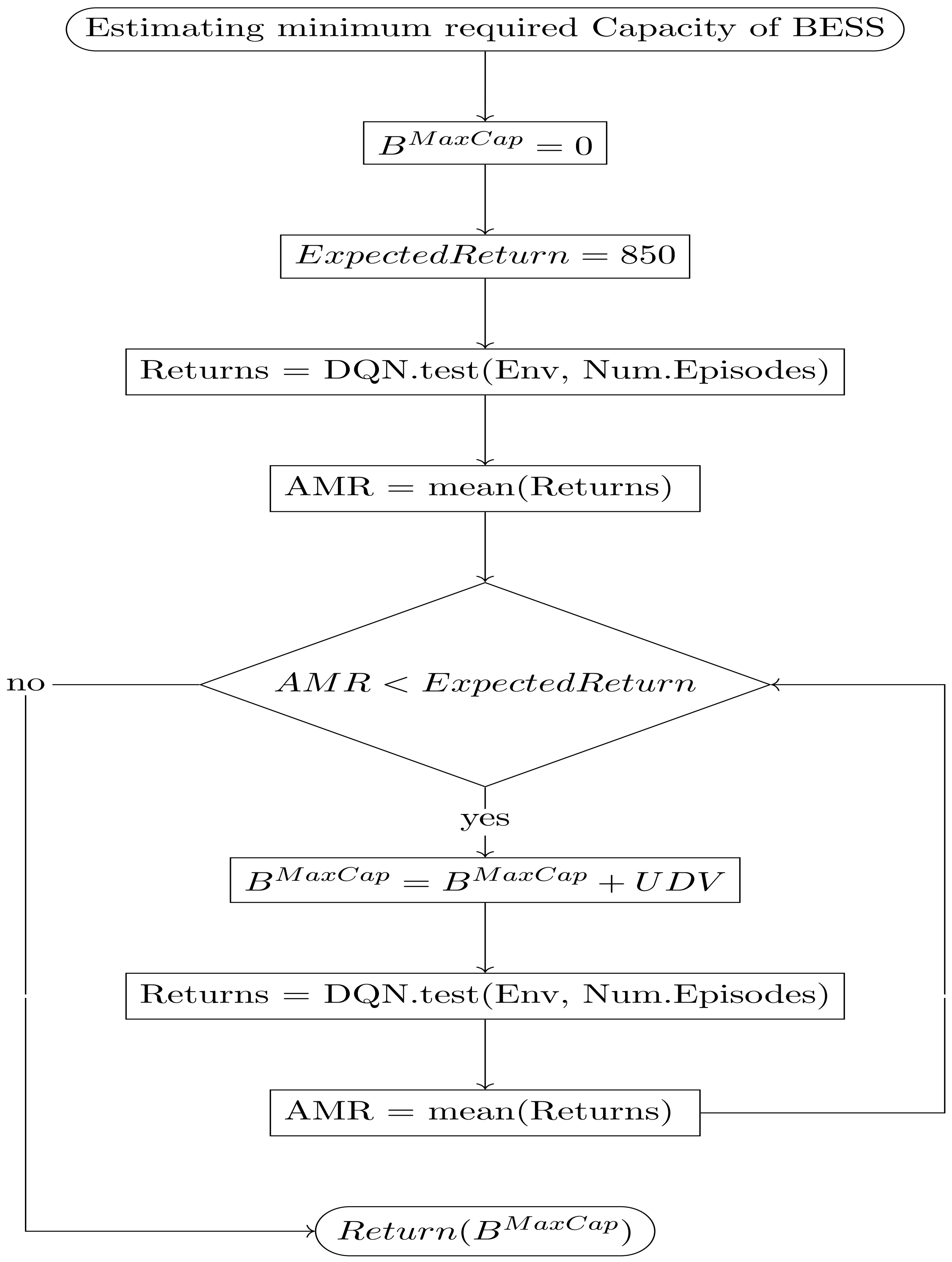

Estimating minimum required Capacity of BESS
$B^{MaxCap} = 0$
$ExpectedReturn = 850$
Returns = DQN.test(Env, Num.Episodes)
AMR = mean(Returns)
$AMR < ExpectedReturn$
no
yes
$B^{MaxCap} = B^{MaxCap} + UDV$
Returns = DQN.test(Env, Num.Episodes)
AMR = mean(Returns)
$Return(B^{MaxCap})$

Figure 6.18: *Estimating minimum required BESS*

Where *ExpectedReturn* is the optimized return from training the agent, *DQN.test* represents the process of testing the trained agent in the environment *Env*, *NumEpisodes* is a user-specified number of episodes in a sequence, AMR is a function that finds the average return of the sequence, and UDV is the user-specified value to increase B^{MaxCap}. The code for this estimation method is available in the Appendix A.18.

This application shows that the framework is able to generate policies that are robust enough to handle changes of the static values that serve as limits for most of the framework's equations during training because the maximum discharge power and the minimum capacity of the simulated BESS are dependent on the maximum capacity of the BESS. Therefore, the same rate of change applied to the maximum capacity is applied to the maximum discharge power and the minimum capacity.

6.9.3 Direct Control Environment of a BESS

This deep reinforcement learning framework is designed with the real-time control of a BESS in mind, inspired by methods presented in the paper ”Control of battery charging based on reinforcement learning and long short-term memory networks” by Chang et al. [97]. A new training environment and a direct control environment are developed to demonstrate the performance of this framework with real-time control. The main application for this use case to perform automatic energy arbitrage if the connection to the grid allows the BESS to be reimbursed for selling electricity back to the grid.

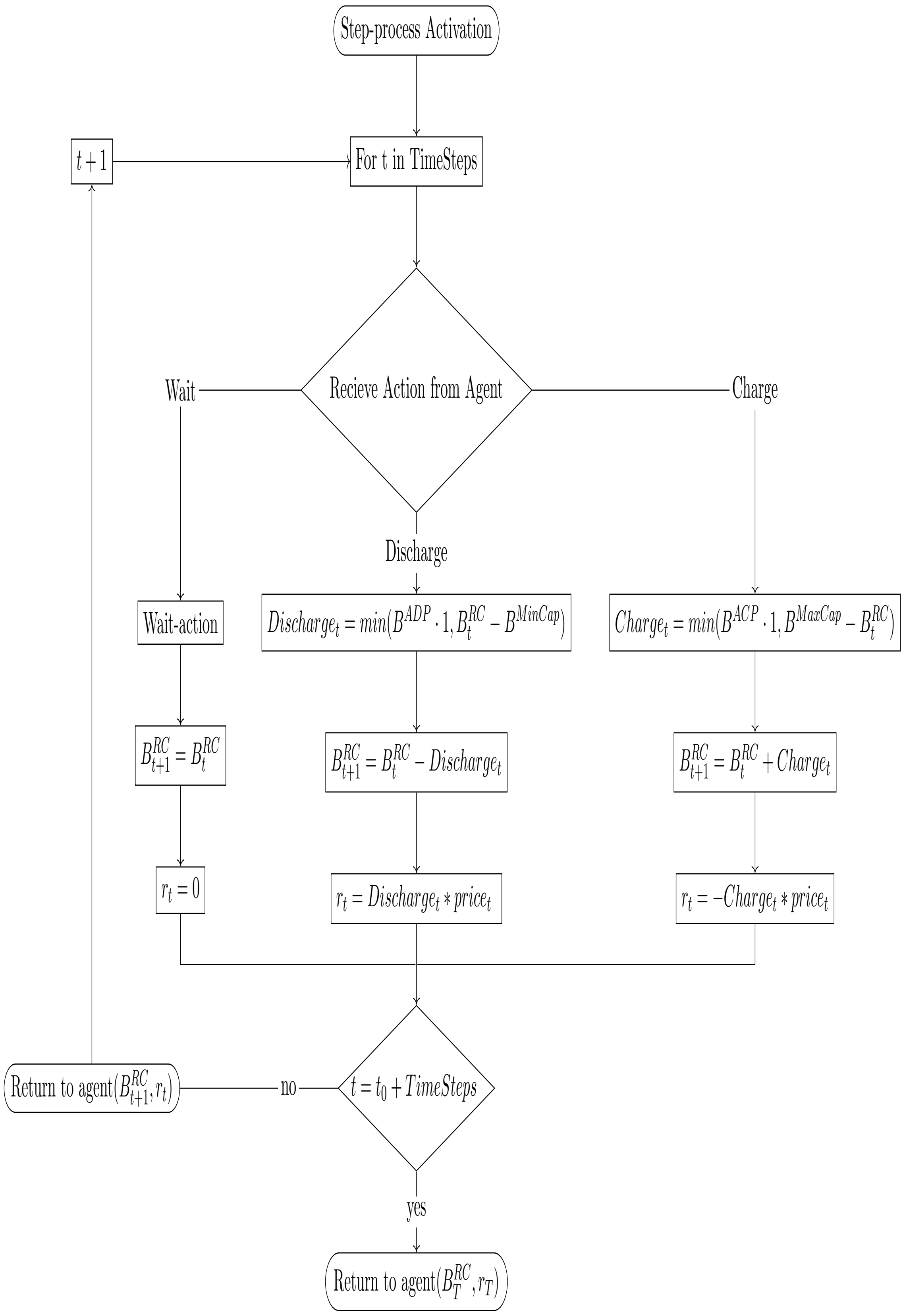
The first step in using this application is to modify the training environment configured to solve the tasks of minimizing the cost of electricity by removing all functions, state signals, and actions related to the earthwork machines. The modified training environment has the same structure and rewards as the original training environment. However, the state space is limited to only monitor the remaining capacity of the BESS, electricity prices, and the time and date range. Furthermore, the action space has only one discharge action in addition to the charge and waiting actions. The new mathematical operation for the charging action is expressed as follows.

$$Charge_t = \min(B^{ACP} \cdot 1, B^{MaxCap} - B_t^{RC}) \tag{6.25}$$

Where $Charge_t$ is the maximum charge receivable by the BESS for one hour, limited by its current depth of discharge, and B^{ACP} represents a static hourly amount of charged energy to the battery. Furthermore, the new mathematical operation for the discharging action is expressed as follows.

$$Discharge_t = \min(B^{ADP} \cdot 1, B_t^{RC} - B^{MinCap}) \tag{6.26}$$

Where $Discharge_t$ is the maximum discharge achievable by the BESS for one hour, limited by its current depth of discharge, and B^{ADP} represents a static hourly amount of discharged energy from the battery. The full dynamics of the step process of the modified training environment is presented in Figure 6.19 below.



Step-process Activation
$t+1$
For t in TimeSteps
Recieve Action from Agent
Wait
Charge
Discharge
Wait-action
$Discharge_t = min(B^{ADP} \cdot 1, B_t^{RC} - B^{MinCap})$
$Charge_t = min(B^{ACP} \cdot 1, B^{MaxCap} - B_t^{RC})$
$B_{t+1}^{RC} = B_t^{RC}$
$B_{t+1}^{RC} = B_t^{RC} - Discharge_t$
$B_{t+1}^{RC} = B_t^{RC} + Charge_t$
$r_t = 0$
$r_t = Discharge_t * price_t$
$r_t = -Charge_t * price_t$
Return to agent(B_{t+1}^{RC}, r_t)
no
$t = t_0 + TimeSteps$
yes
Return to agent(B_T^{RC}, r_T)

Figure 6.19: *Modified Step Process of training environment for direct control of Battery*

The second step is to set up a programmable controller for the intended BESS. As proof of concept, a battery simulation model is used. The chosen battery simulation model was developed by Olivier Tremblay and Louis-A. Dessaint [108], and features an automatic charge and discharge process of a battery connected to a constant load. This model is developed in Matlab Simulink and is featured in Matlab's documentation as a verified example of a battery simulation model.

This simulation model was chosen because it can accurately represent the behavior of various types of batteries, provided the battery parameters are well defined [108]. The parameters of the battery simulation model are deduced from the discharge characteristics of the real types of batteries and are assumed to be the same for charging. However, the internal resistance is constant during the charge and discharge cycles and does not vary with the amplitude of the current. Similarly to the representation of a battery in the previous training environments, the battery capacity does not change with the amplitude of the current, the temperature does not affect the model's behavior, self-discharge is not represented, and the battery has no memory effect.

The original configuration of the simulation model would automatically charge the battery if it went below a 40% state of charge and discharge the battery if the state of charge exceeded 80%. However, the simulation model was adapted for this application by replacing the automatic switching between charge and discharge with a manual switch that is controlled by the direct control environment. Figure 6.20 below shows the adapted Matlab Simulink model.

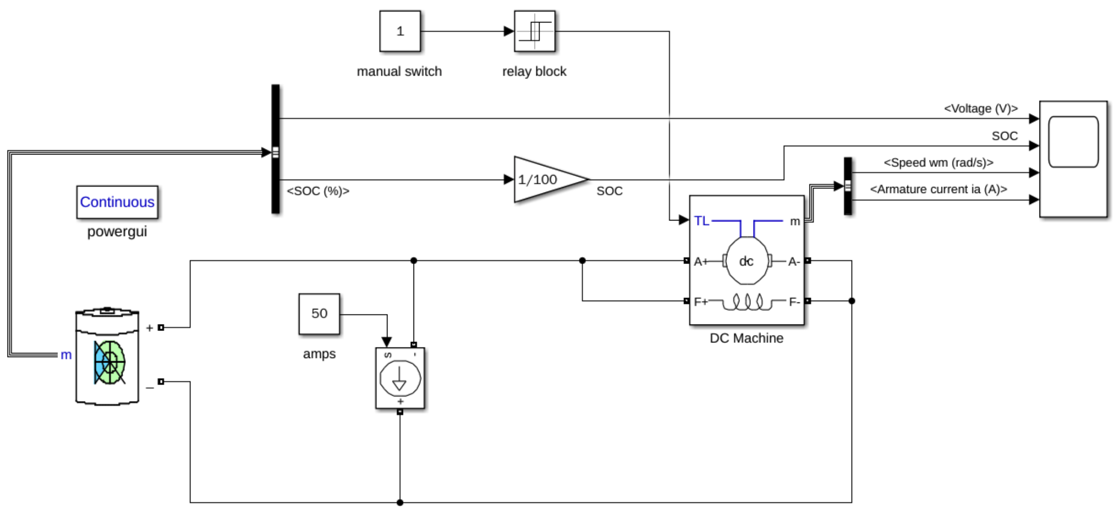


Figure 6.20: *Simulink Battery model*

Where a constant block controls the manual switching between the charge and discharge actions.

To control the battery simulation model, a direct control environment with a programming-bridge between Python and Matlab Simulink is developed. The mathematical operations of the actions in this direct control environment are replaced by direct sensor readings and controls of the simulation model. Figure 6.21 below shows the step process of the modified control environment for direct control of the battery simulation model.

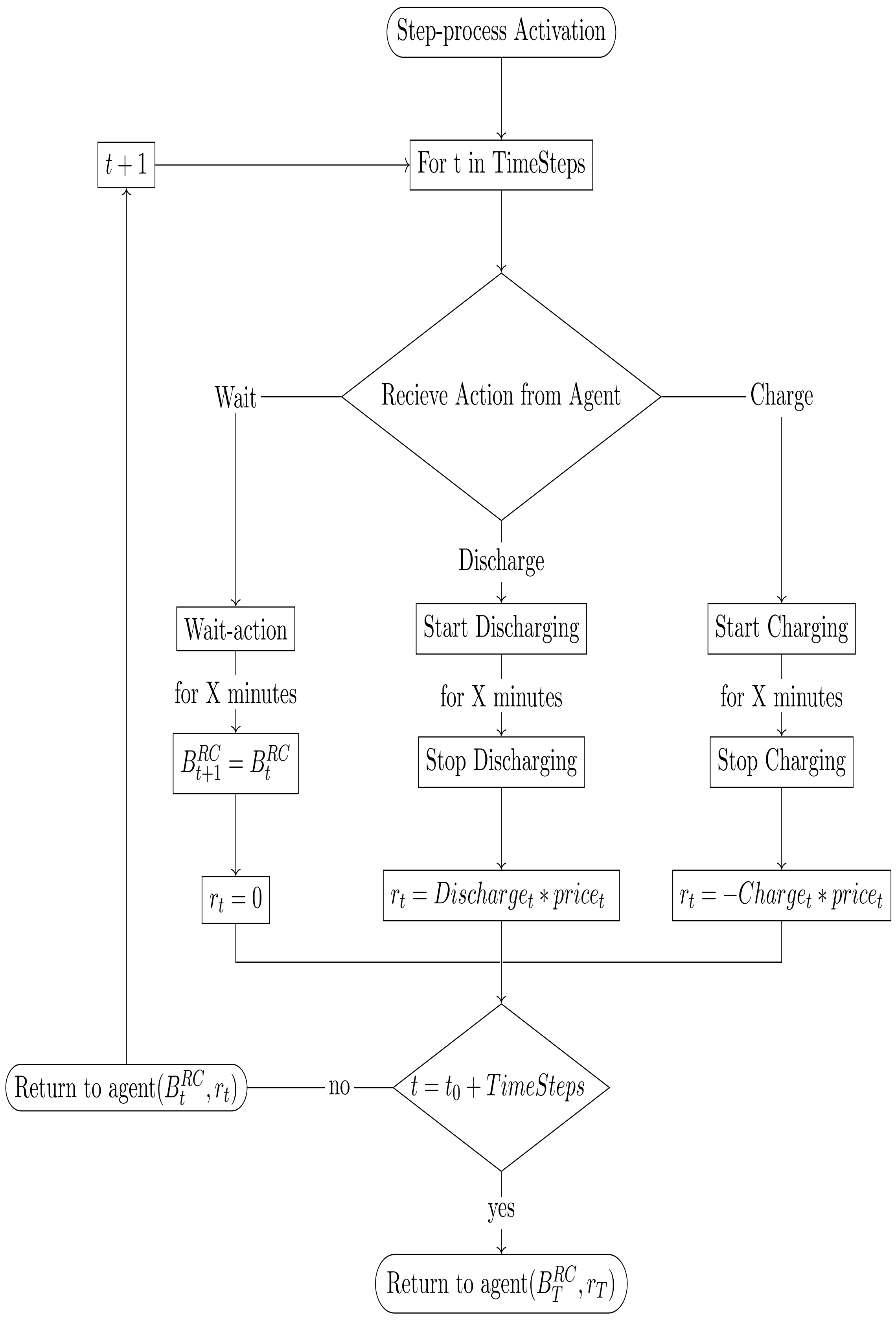


Figure 6.21: *Modified Step Process for direct control of Battery*

When the agent selects the charge action, the control environment will first record the current remaining capacity B_t^{RC} from the simulation model. If B_t^{RC} is less than the maximum capacity of the simulated battery model B^{MaxCap}, it will set the manual switch block to 0, which represents the charging process. Subsequently, it will send a command to the simulation model to start a simulation period that represents one time step. If the battery reaches B^{MaxCap} before the simulation period terminates, the simulation model will not overcharge the battery, because this is limited by the simulation model itself. After the simulation finishes, the direct control environment will record the new remaining capacity of the simulated battery as B_{t+1}^{RC} and subtract B_{t+1}^{RC} from B_t^{RC} to calculate $Charge_t$, which is used to calculate the reward in Equation 6.9.

When the agent selects the discharge action, the control environment will first record the current remaining capacity B_t^{RC} from the simulation model. If B_t^{RC} is above the batteries depth of discharge threshold B^{MinCap}, it will set the manual switch block to 1 representing the discharging process. Subsequently, it will send a command to start a simulation period for one time step. If the battery reaches its B^{MinCap} before the simulation period terminates, the simulation model will not overdischarge the battery because this is limited by the simulation model itself. After the simulation ends, the modified environment will record the new remaining capacity of the simulated battery as B_{t+1}^{RC} and subtract B_{t+1}^{RC} from B_t^{RC} to calculate $Discharge_t$, which is used to calculate the reward in Equation 6.10.

When the agent selects the waiting action, the modified environment will record the current remaining capacity B_t^{RC} from the simulation model and set B_{t+1}^{RC} equal to B_t^{RC}. The reward received by the agent is 0 as explained in Section 6.4.3.3. The code for the direct control environment is available in the Appendix A.19.

Furthermore, by training the agent in this direct control environment, the agent can develop a policy that satisfies the detailed dynamics of a real battery system. This may further increase the performance of BESS-specific applications. However, training would require significantly more time as a result of the latency caused by simulating every time step in Matlab Simulink.

Chapter 7

Results

This chapter details how the electrified earthwork process can be optimized using the deep reinforcement learning framework presented in the previous chapter. The purpose of this chapter is to demonstrate the performance and generalizability of the framework in optimizing electrical earthwork processes. Specifically, the framework's training environment will be configured to maximize the machines' workloads within a specified time frame, minimize the time required to complete a productivity goal, and minimize the cost of electricity while completing a productivity goal.

The optimization results serve as a sensitivity analysis by demonstrating how different influencing factors, fleet configurations, and optimization goals affect the performance of the framework. Furthermore, the framework is applied to optimize the other use cases presented in Section 6.9 as validation of the generalizability of the framework.

The training of framework's agent was conducted on a personal computer with a 2.2 GHz Intel Core i7 processor and 16 GB 1600 MHz DDR3 RAM. The training time ranged from 5 minutes (<2000 episodes) to 20 minutes (6000 episodes), depending on the framework configurations. Therefore, training time was not identified as a significant obstacle to this framework.

7.1 Influencing Factors

The training environment is configured to represent different scenarios on the same construction site. Therefore, several influencing factors are static for all scenarios. However, arbitrary and estimated ranges are provided for influencing factors that were not discovered or were underspecified in the literature to account for the lack of specification.

The uncertainty surrounding these ranges selected for underspecified factors will increase the number of unique scenarios, as the agent will be exposed to more stochastic values during training. Hence, the generalizability of the framework will increase.

7.1.1 BESS Factors

The same BESS factors were used for all configurations. The BESS is modeled after the BoostCharger prototype, introduced in Section 2.3.3.

BESS Factors

- Maximum energy capacity of 390kWh
- Minimum energy capacity of 39kWh
- Maximum charging power (30, 37)kW
- Maximum discharging power of 150kW (per cable)
- 2 charging cables available for simultaneous charging.
- Charging efficiency of 95%
- Discharging efficiency of 105%

The maximum charging power range was set to account for variations in the grid's power supply, which was recorded for the case study in the following chapter.

7.1.2 Machine Factors

The type of machines simulated by this framework are excavators, loaders, and dump trucks. These types of machines were selected on the basis of their prominent representation in the literature. The machine factors are collected directly from the manufacturer's brochures and site visits. However, certain performance factors are not yet known to manufacturers and are not represented in the literature. These unknown factors are represented as ranges. The machine factors are presented in Table 7.1 below.

Parameter	Excavator	Loader	Truck
Maximum Capacity	300kWh	200kWh	180kWh
Minimum Capacity	30kWh	15kWh	18kWh
Maximum Charging Power	150kW	75kW	40kW
Maximum Output Power	50kW	30kW	20kW
Round-Trip Efficiency	0.95	0.95	0.95
Idling Power	(0, 5)kW	(0, 3)kW	(0, 1)kW
Attachment	Bucket	Bucket	Loading Bed
Attachment Capacity	1.5m^3	3m^3	23m^3
Cycle Time	36sec	106sec	18min

Table 7.1: *Machine Factors*

The cycle time set for excavators represents the excavation and stockpiling of the soil, while the cycle time set for loaders represents moving the stockpiled soil to load a dump truck. These cycle times are based on the duration of specific tasks presented in [6] and estimates derived from observations at a real construction site. The cycle time set for trucks is the time required to travel from the construction site to an adjacent dump site and back. This value is deduced from the division of an average travel speed of 80 km/t [53] and an arbitrary round-trip distance of 14.5km.

7.1.3 Operational and Site Factors

The operational and Site factors included in this framework are presented in Table 7.2 below.

Parameter	Excavator	Loader	Truck
Number of machines	(1,2)	(0,2)	(0,1)
Operator Efficiency	95%	95%	95%
Operational Efficiency	(65, 83)%	(65, 83)%	100%

Table 7.2: *Operational and Site Factors*

Here, the number of machines represent the fleet configurations for the various configurations. Furthermore, the operator efficiency represents the impact of the operators skill level on productivity. The operational efficiency impacts the energy consumption and productivity of machines to account for the impact of the site layout. The operational efficiency ranges are established on the basis of recommendations from the literature [51].

7.1.4 Soil Factors

The soil factors are presented in Table 7.3 below.

Soil Type	**Sand**	**Clay**	**Rock**
Min. volume to be excavated	200 m^3	200 m^3	200 m^3
Location of soil	upper	middle	lower
First Soil Power Factor	(0.4, 0.6)	(0.6, 0.8)	(0.8, 1)
Second Soil Power Factor	(0.2, 0.4)	(0.2, 0.5)	(0.5, 0.8)
Soil Fill Factor	1	0.8	0.6

Table 7.3: *Soil Factors*

Here, the volume to be excavated represents the amount of soil, clay, or rock located in the upper, middle, or lower layer of the excavation site. The soil power factors represent the impact of soil density on the energy required to move them. These are set to arbitrary ranges because the literature on electric earthmoving machines has yet to define them. However, the selected ranges correlate with soil factors for diesel-powered earthmoving machines, where denser soil types have a greater impact on the energy required to move them [59].

7.1.5 Other Factors

Other factors that are specific to the configuration are specified in the following sections. However, The time fraction Δt for all configurations is set to 0.1 to simulate the productivity, energy consumption, and charging of equipment in intervals of 6 minutes.

7.2 Maximize Workloads of Machines

The optimization framework's training environment is configured to maximize the workloads of the machines by including workload state signals and setting the reward signal to maximize workloads, as described in Sections 6.2.2 and 6.4.1, respectively.

Furthermore, the time step range is set to 8 time steps, which provides the agent with 8 simulated hours to maximize the workloads.

The construction scenario generator is configured to generate scenarios with two excavators and two loaders for each training episode. The excavators are using the first soil power factor, while the loaders are using the second soil power factor. The remaining capacities of these machines at the initial time step of every episode are set stochastically within a range of 50% and 90% of the machine's maximum energy capacity. Furthermore, The maximum volume of stockpiled soil is set to $300m^3$, and the loaders are set to fill the excavated soil on site because there are no trucks to haul the soil to a dump site. Figure 7.1 shows a visualization of the earthwork scenario for this configuration.

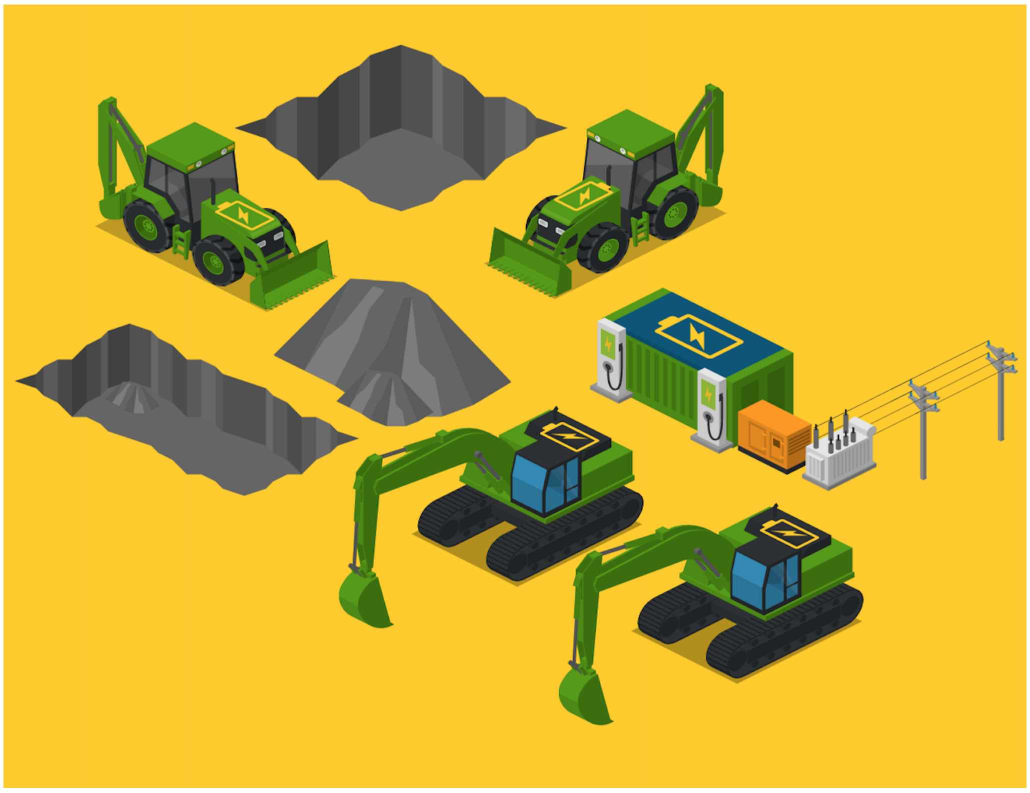


Figure 7.1: *Visualization of the maximization scenario*

For every episode, the remaining capacity of the BESS is set to its maximum capacity in t_0 by setting $B^{RC}_{t_0}$ equal to B^{MaxCap} in the initialization and reset process. This approach is selected to maximize the available energy capacity to charge machines in order for them to efficiently complete workloads, which in turn will allow the agent to maximize its return. The rewards are calculated using Equation 6.7.

Furthermore, the number of training episodes for this version is set to 2000. This number was identified using the method explained in Section 6.7.2.

7.2.1 Results from Maximizing Workloads

The returns of training the framework to solve the task of maximizing workloads are shown in the graph 7.2 below.

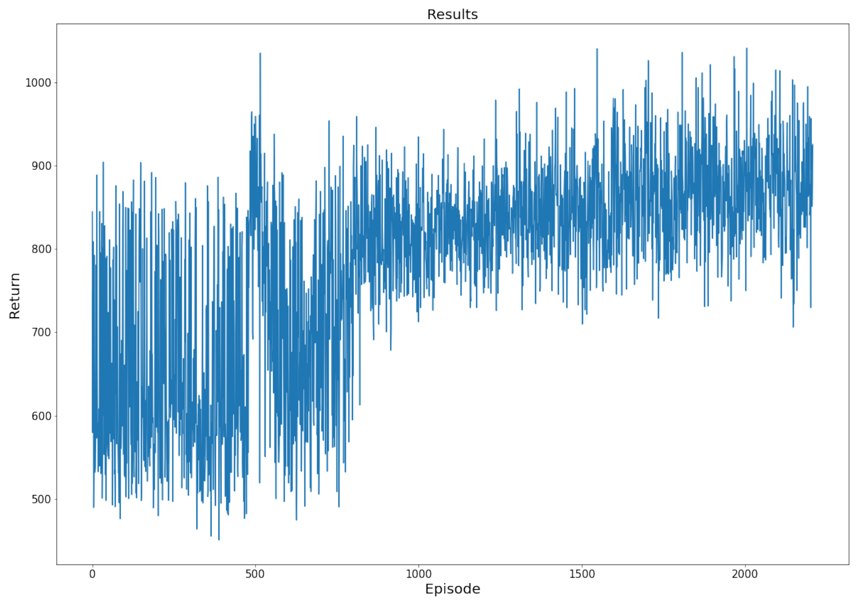
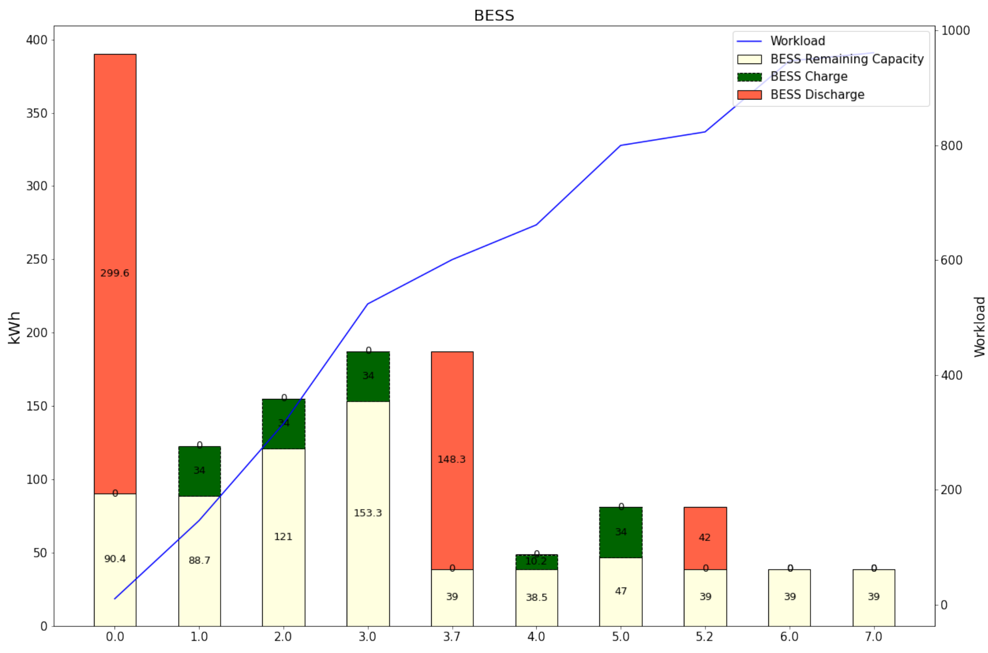


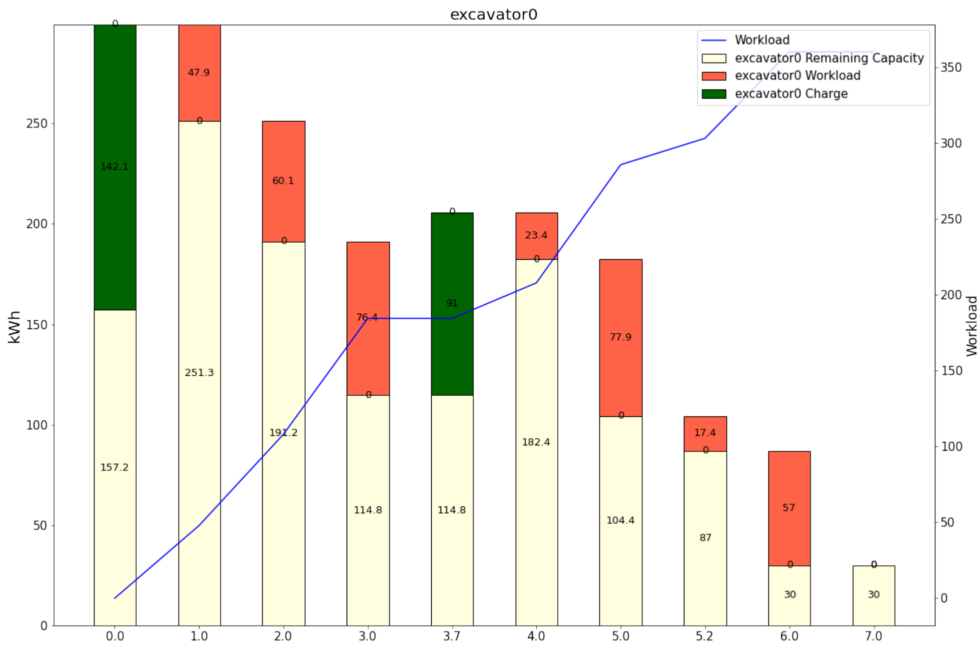
Figure 7.2: *Returns from training to solve the task of maximizing workloads*

The graph shows that the agent was testing different policies to for the first 400 episodes. Between episodes 400 and 500, the agent develops a policy that was able to generate a return of up to 1000 from some scenarios. However, this policy was not stable, as the returns quickly drops in the following 100 episodes. Therefore, the agent chose to explore different policies in the following 200 episodes. This exploration leads to the development of a more stable policy in which the performance does not drop and more returns of 1000 are achieved. In the remaining 1000 episodes, the agent improves this policy with minor adjustments, and a slight increase in maximum return is noticeable.

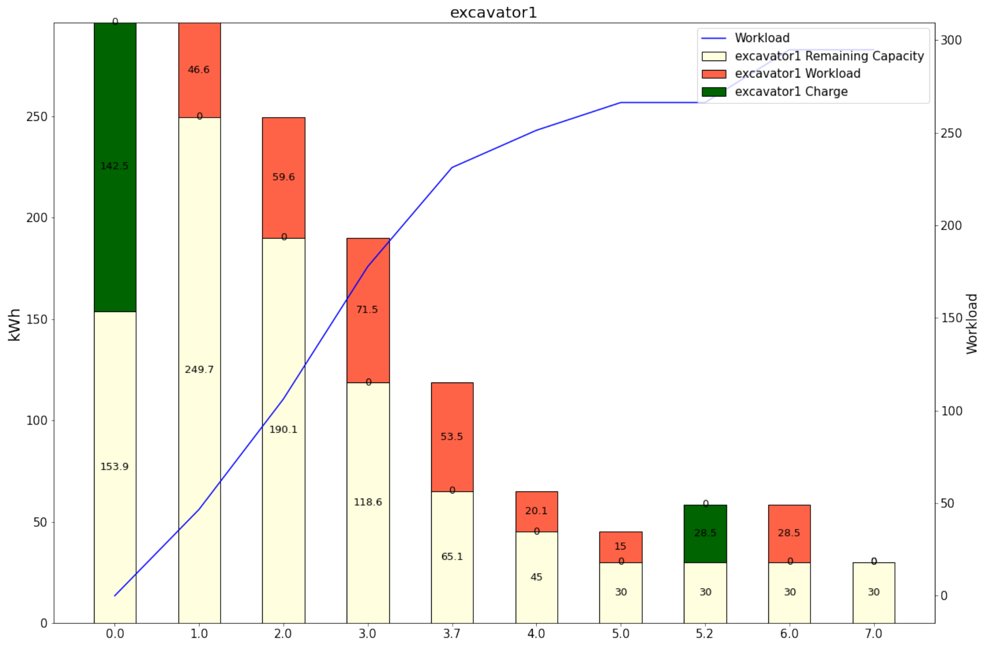
An example of how the agent maximizes the workloads was generated using this policy and is presented in Figure 7.3 below.



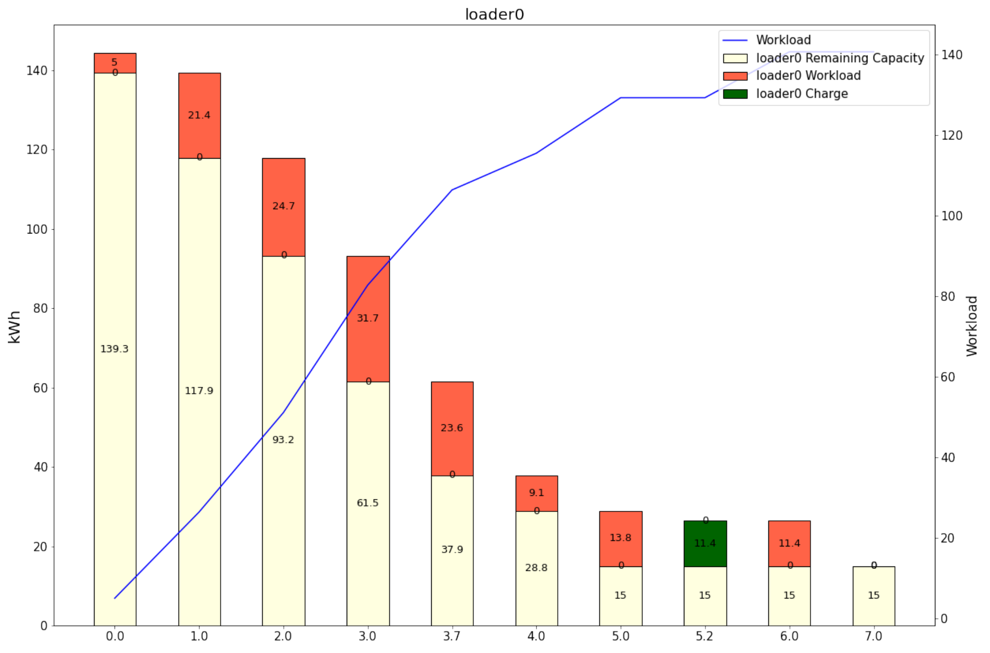
(a) BESS Energy Flow



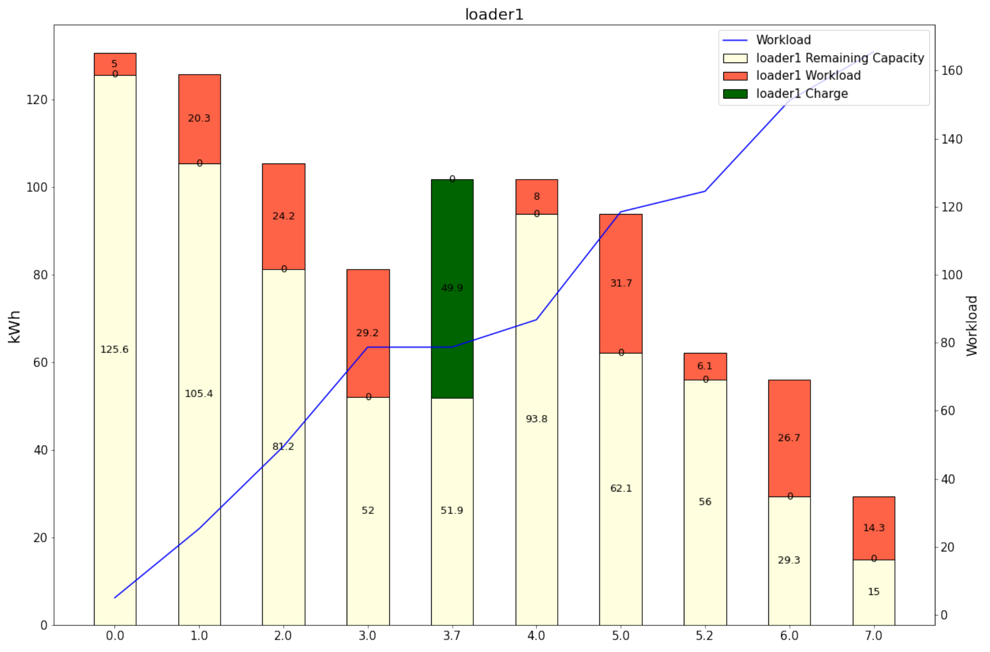
(b) First Excavator Energy flow



(c) Second Excavator Energy Flow



(d) First Loader Energy Flow



(e) Second Loader Energy Flow

Figure 7.3: *Optimized Schedule for Maximized Workloads*

Graph (a) shows that the agent performed a discharge action using both available charging cables in the initial time step. The agent decided to charge the first excavator (Graph (b)) and the second excavator (Graph(c)) for a period equivalent to 1 hour. In the next three time steps, the agent is recharging the BESS because it is expecting to discharge the BESS again. It is noticeable that the excavators were reaching deeper layers of soils as the workloads increase between time steps 1.0 and 3.0.

At time step 3.7, the agent discharges the BESS to the first excavator and the second loader (Graph (e)). The action is terminated before the end of the simulated hour because the BESS reaches its minimum energy capacity. Therefore, time step 4.0 represents only 18 minutes, and allows the agent to make other decisions for the remaining time of the simulated hour.

Furthermore, the agent decides to charge the BESS in time steps 4.0 and 5.0 in order to discharge it in time step 5.2. The discharged amount was allocated to charge the second excavator and the first loader (Graph (d)), which enabled these machines to perform a final workload in the next time step. The episode ends when the 8 hours are simulated (in time step 7.0). This example proves that agent was able to maximize the workloads of the machines during the episode because the remaining capacities of all machines are equal to their minimum energy capacities at the final time step.

The productivity result of this example is shown in Figure 7.4 below.

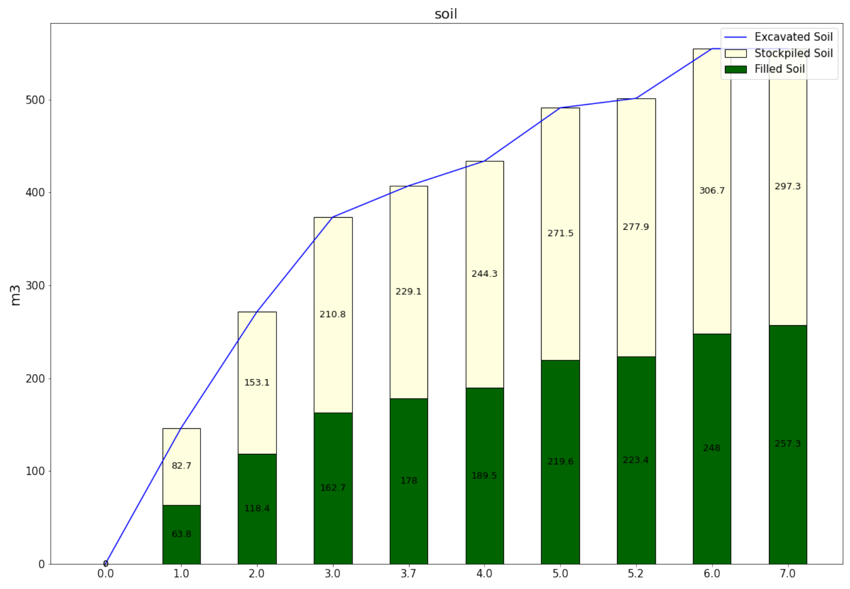


Figure 7.4: *Amount of Excavated and filled Soil in m^3*

This graph shows that the machines were able to excavate more than $500m^3$ of soil consisting of sand, clay, and rock during the 8 hours. Additionally, the agent did not breach the maximum stockpile volume of $300m^3$, and the loaders were able to fill $257m^3$ of the excavated soil.

7.3 Minimize the Time Required to Complete a Productivity Goal

The optimization framework's training environment is configured to minimize the time required to complete a productivity goal by including the productivity state signals and setting the reward signal to minimize the number of time steps required to complete a productivity goal, as described in Sections 6.2.3 and 6.4.2, respectively. Furthermore, there is no maximum number of time steps, as the purpose of this configuration is to minimize the number of hours required to complete the productivity goal.

The construction scenario generator is configured to generate scenarios with one excavator, one loaders, and one truck for each training episode. Both the excavators and loaders are using the second soil power factor. Similarly to the previous configuration, the remaining capacities of these machines at the initial time step of every episode are set stochastically within a range of 50% and 90% of the machine's maximum energy capacity. Figure 7.5 shows a visualization of the earthwork scenario for this configuration.

Figure 7.5: *Visualization of the productivity goal scenario*

For every episode, the remaining capacity of the BESS is set to a stochastic value calculated using Equation 6.17. This is done to force the agent to develop a robust policy that is capable of minimizing the time required to complete a productivity goal in several different scenarios.

Furthermore, the productivity goal is set to $200m^3$ of dumped soil. Therefore, an episode will not terminate until the agent has developed a strategy in which $200m^3$ of soil is excavate and stockpiled by the excavator, loaded onto the truck by the loader, and dumped at the dump site by the truck. This productivity goal is formulated for the agent using Equation 6.8.

Furthermore, the maximum allowed volume of soil stockpiled is set at $200m^3$. This makes the task more complicated for the agent, as it must develop a strategy in which the volume of soil stockpiled does not exceed this amount at any time step during the episode.

The number of training episodes for this version is set to 1300. This number was identified using the method explained in Section 6.7.2.

7.3.1 Results from Completing a Productivity Goal

The returns of training the framework to minimize the time required to complete a productivity goal are shown in the graph 7.6 below:

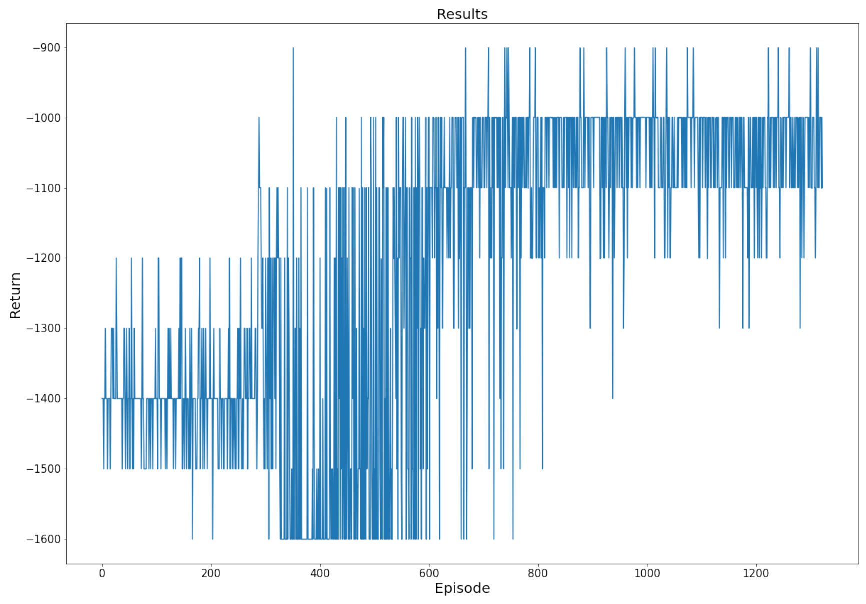
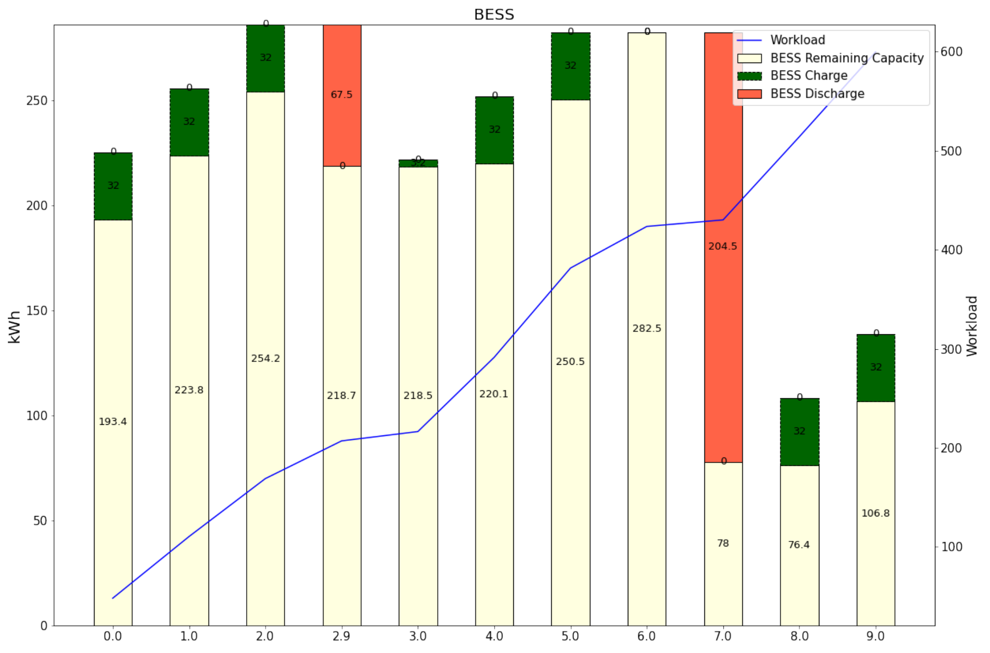


Figure 7.6: *Returns from training to solve the task of minimizing completion time of productivity goal*

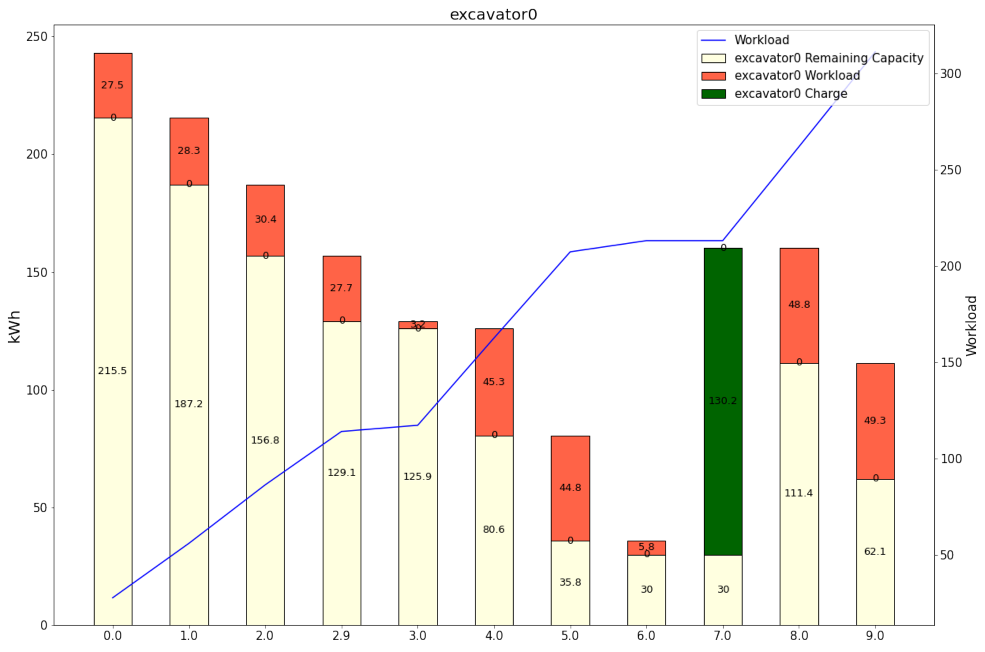
This graph shows that the agent was developing different policies that were struggling to complete the productivity goal in less than 14 time steps in the first 300 episodes. In the following 200 episodes, the agent explored several new policies. The exploration clearly shows a performance drop where more time steps are required to complete the productivity goal. However, the agent was able to use this exploration to develop a more robust policy that is evident from Episode 600 and onward. This robust policy is able to consecutively complete the productivity goal in 10 simulated hours for various scenarios because a return of -1000 indicates that the agent received the time step reward of -100 10 times in a row.

Following Episode 900, the agent was unable to reduce the average completion time further, except for a few scenarios where it was completed in 9 hours. This is likely because several stochastic values were favorable in those scenarios. Therefore, 10 hours is considered the optimized time required to reach this productivity goal.

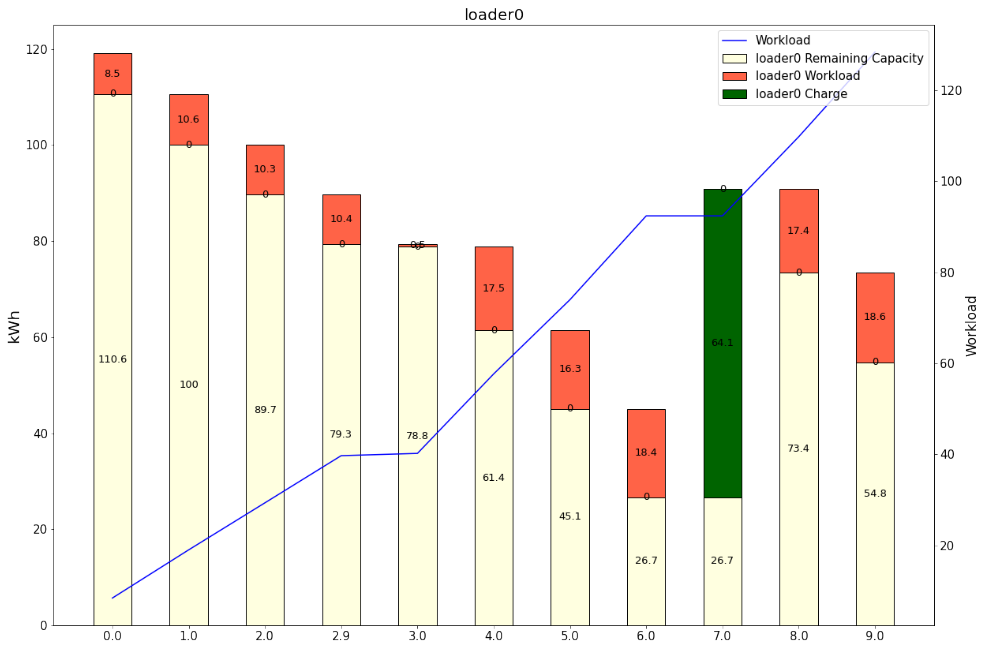
An example of how the agent uses this policy to complete the productivity goal is presented in Figure 7.7 below.



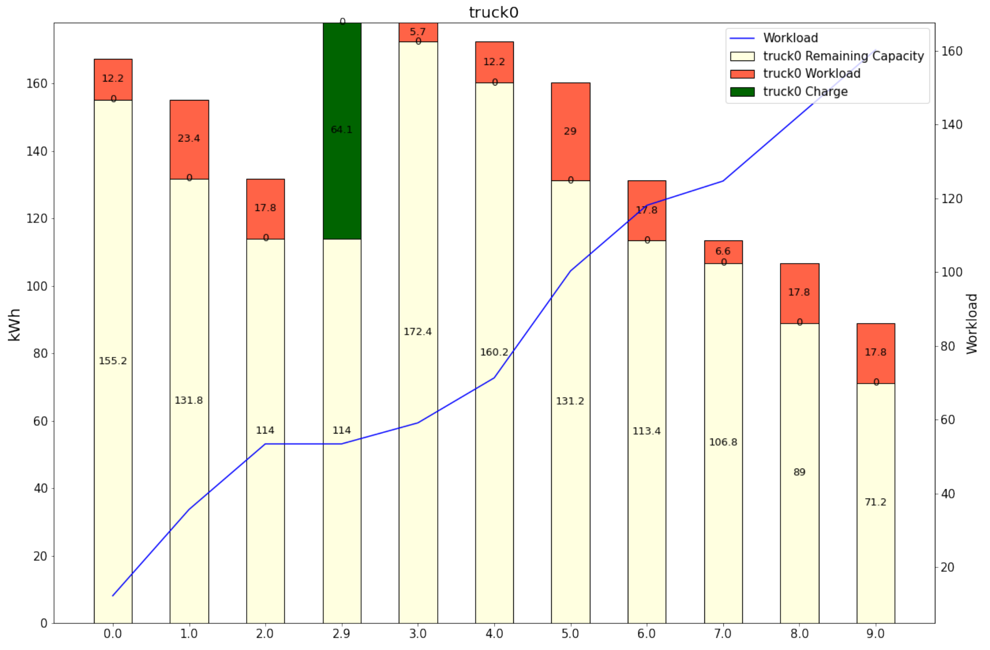
***(a)** BESS Energy Flow*



(b) Excavator Energy Flow



(c) Loader Energy Flow



(d) Truck Energy Flow

Figure 7.7: *Optimized Schedule to Complete a Productivity Goal*

Graph (a) shows that the agent charges the BESS for the first three time steps. During time step 2.9, the agent discharges the BESS to charge the truck (Graph (d)). This is the only charge that the truck receives because the agent estimates that its remaining capacity is more than enough to complete its expected workloads in the remaining time steps. The time step terminates at 2.9 and not 3.0 because the truck reached its maximum energy capacity before the full hour was complete. Therefore, time step 3.0 represents only 6 minutes and is available to the agent to perform other actions to maximize the utilization of the simulated hour.

In the following two time steps, the agent charges the BESS again. During these time steps, it is noticeable that the excavator (Graph (b)) has reached a new layer of a denser soil type, as the workloads increase for both the excavator and the loader (Graph (c)). In time step 6.0, the agent performs the waiting action. This is probably due to the truck being loaded by the loader. However, it can be discussed whether charging the excavator would be a better decision since it reaches its minimum energy capacity in this time step.

In time step 7.0, the agent discharges the BESS to recharge the excavator and the loader to avoid reaching the loaders minimum energy capacity. If the agent had not recharged the excavator and loader in this time step, the earthwork process would have halted and more waiting would be required to complete the productivity goal.

The productivity goal is completed in time step 9.0, which is shown in Figure 7.8 below.

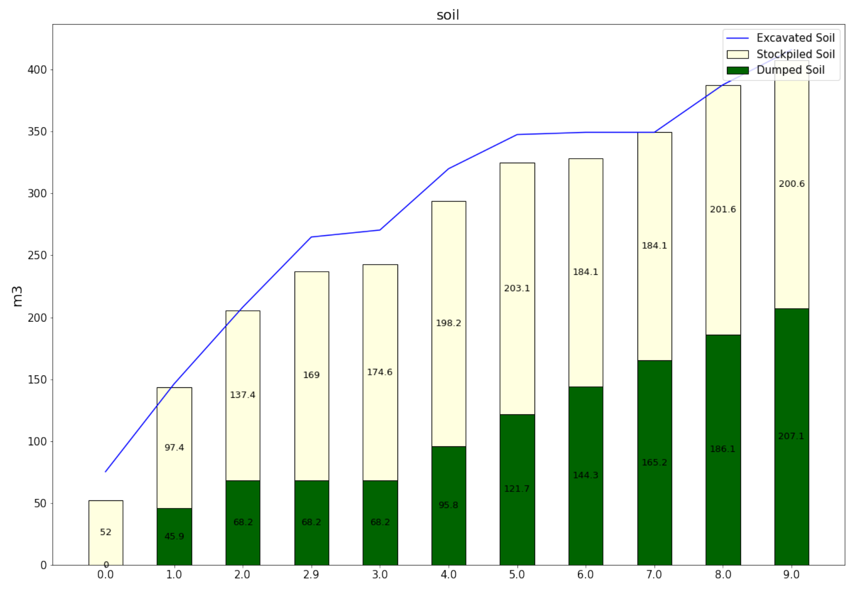


Figure 7.8: *Amount of Soil Excavated or Dumped in* m^3

This graph shows that the agent was able to complete the productivity goal of dumping $200m^3$ soil within 10 simulated hours. Furthermore, the agent did not exceed the maximum allowed volume of stockpiled soil by more than 1.6 m^3, which is considered acceptable.

Figure 7.8 also shows that the earthwork process was not halted, as there is a steady increase in the volume of soil excavated and dumped. Therefore, the agent did not cause delays by not charging the excavator in time step 6.0, as the volume of stockpiled soil was already sufficient to complete the productivity goal. This serves as a reminder of how impactful the formulation of the reward signal is in reinforcement learning.

7.4 Minimize Electricity Cost While Completing a Productivity Goal Configuration

The optimization framework's training environment is configured to minimize electricity cost while completing a productivity goal by including the electricity cost state signals and setting the reward signal to minimize electricity cost, as described in Sections 6.2.4, and 6.4.3, respectively.

The time step range is set to maximum 10 time steps. However, the agent can reduce the number of time steps by completing the productivity goal on an earlier time step. The purpose of the time step range is to limit the agent's attempts to minimize the electricity cost within a certain number of hours.

The construction scenario generator is configured to generate scenarios with two excavator for each training episode. The excavators are using the second soil power factor. Both excavators begin every episode with a full charge, while the initial remaining capacity of the BESS is set to a stochastic value calculated using Equation 6.17. This is done to force the agent to charge the BESS and experience the negative rewards associated with the charging action. Figure 7.9 shows a visualization of the earthwork scenario for this configuration.

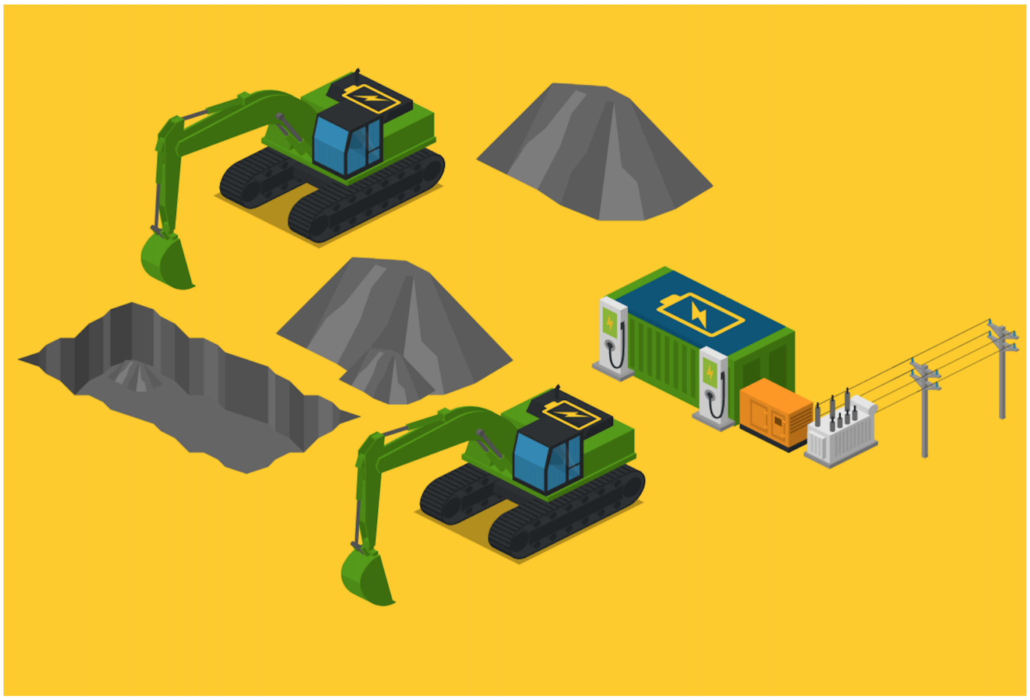


Figure 7.9: *Visualization of the Minimize Electricity Cost Scenario*

The time and date range are set to a full year and every episode begins at the 9_{th} hour of the day. This is achieved by applying Equation 6.18. The electricity prices for an entire year are collected from NordPool [109] and represent the electricity prices for Oslo, Norway, in 2021.

Furthermore, there are multiple reward signals for this configuration, as explained in Section 6.4.3. The agent is rewarded based on the actions performed and for completing the productivity goal. When the agent selects the charge action, the agent's reward signal is calculated using Equation 6.9. When the agent selects any of the discharge actions, the agent's reward signal is calculated with Equation 6.10. If the agent is able to complete the productivity goal during the episode, the agent receives a reward signal calculated with Equation 6.11. The productivity goal of this configuration is to excavate $800m^3$ of soil because there are no trucks or loaders in this configuration.

The number of training episodes for this version is set to 6000. This number was identified using the method explained in 6.7.2.

7.4.1 Results from Minimizing Electricity Cost

The returns of training the framework to minimize electricity cost while completing a productivity goal are shown in the graph 7.10 below.

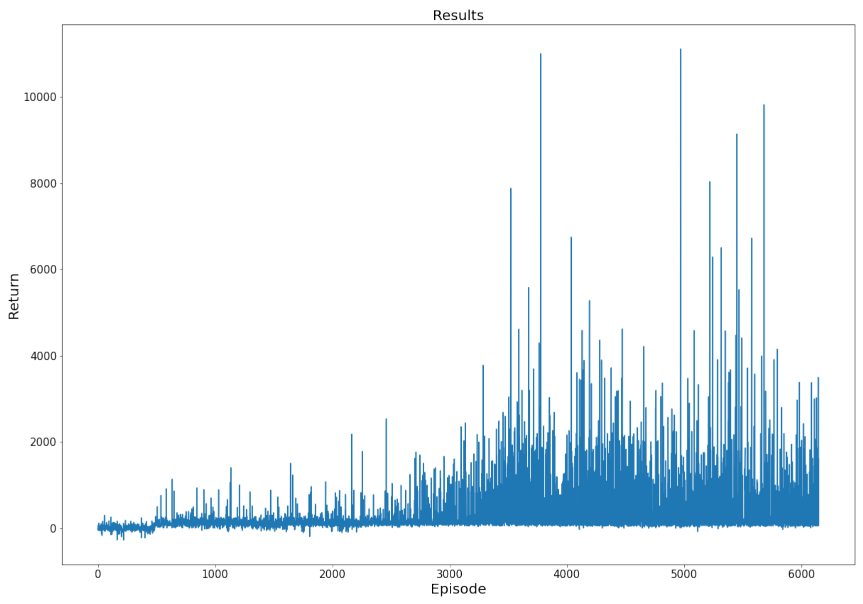
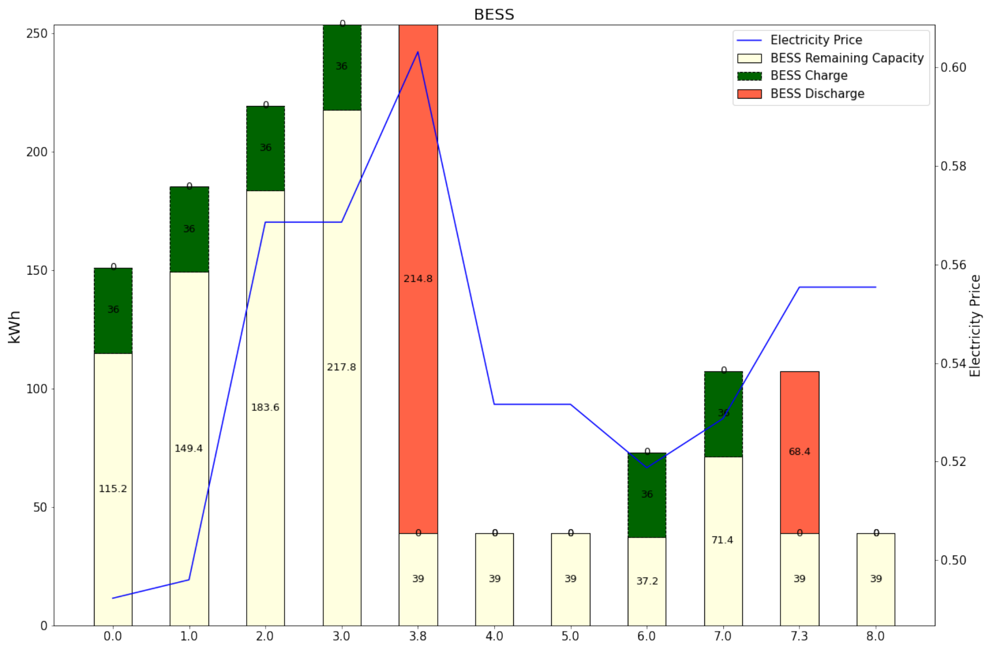


Figure 7.10: *Returns from training to solve the task of minimizing electricity cost*

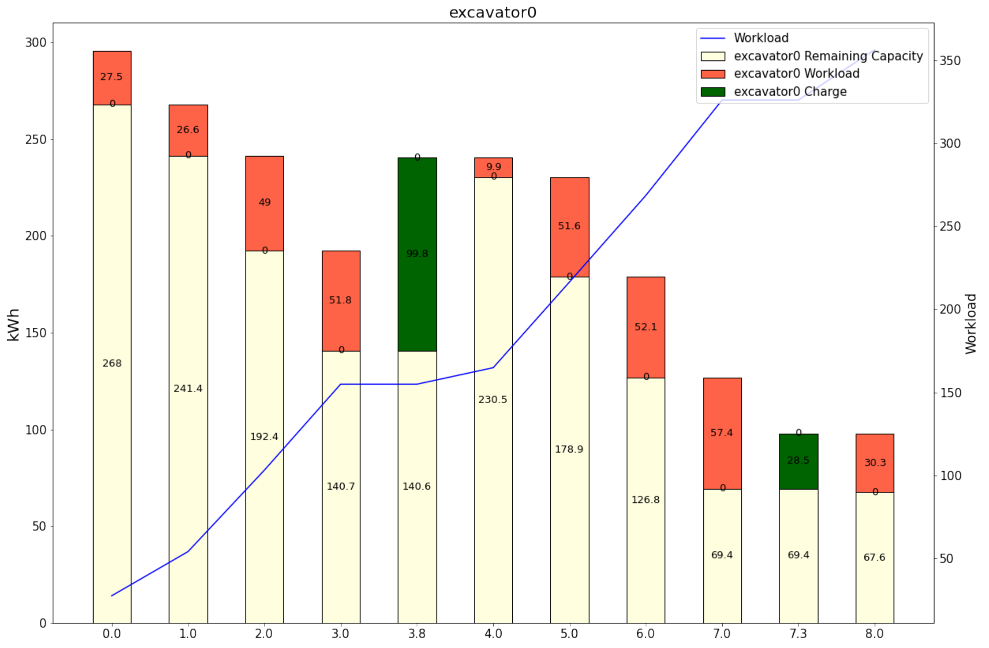
The results show that the agent was struggling to complete the productivity goal while reducing electricity costs for the initial 3000 episodes. In the following episodes, the agent developed a policy that was able to complete the productivity goal and produce a positive return from the sum of action-dependent rewards. For certain days, this resulted in returns that were significantly higher than others. This is due to the high variety of electricity prices on which the agent was trained.

For most episodes, the agent was presented with unseen price pattern for electricity. This forces the agent to recognize the features of the electricity pricing structure that is applicable to most days. Therefore, the generalized response to unseen scenarios for this policy is deemed strong enough after 6000 episodes.

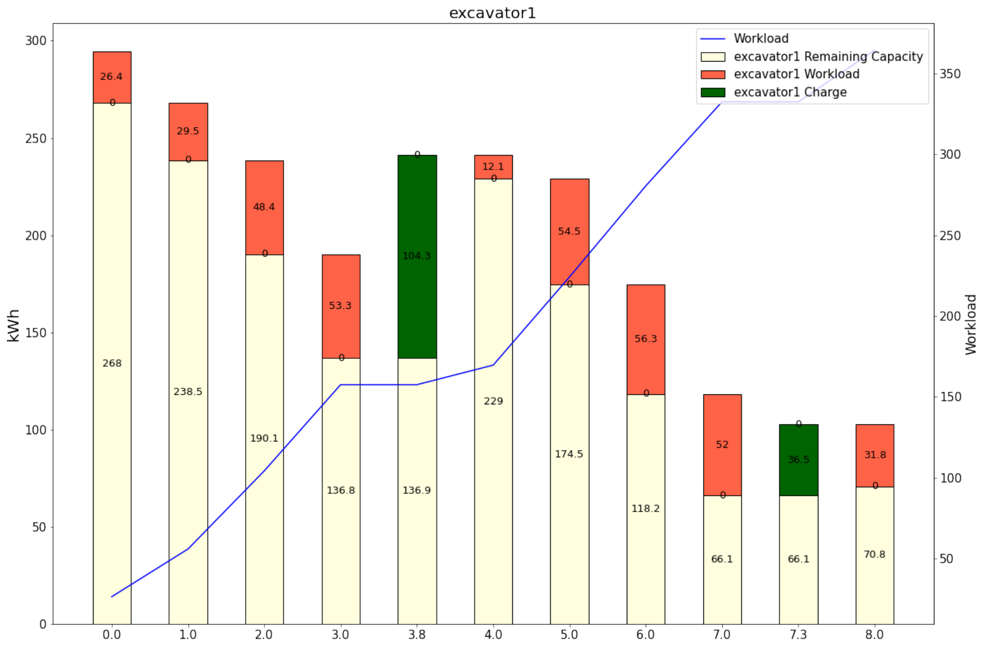
An example of how the agent solves the task for an unseen scenario was generated using the trained policy and is presented in Figure 7.11 below.



(a) BESS Energy Flow



(b) First Excavator energy flow



(c) Second Excavator Energy Flow

Figure 7.11: *Example of an optimized schedule to minimize electricity cost*

Graph (a) shows that the agent charged the BESS for the first four time steps while the price of electricity increased. The agent correctly identified that the price of electricity would peak in time step 3.8 and selected a discharge-action to charge both excavators. The agent was able to foresee this peak because of pattern recognition derived from its training and its generalized response. The agent was able to maximize its discharge reward because the excavators had time to deplete their remaining capacities in the previous time steps.

In the following two time steps the agent waits as the price of electricity drops. However, the BESS is at its minimum energy capacity and the agent must perform a charge action to be able to discharge at a later time step. Therefore, the agent selects the charge-action for time steps 6.0 and 7.0. In time step 7.3, the agent discharges the previously charged amount of energy to both excavators to capitalize on the second peak electricity price peak.

This is a very good example on how the agent is able to minimize the electricity cost while completing a productivity goal. However, most of the agent's attempts are not this great due to high variation of electricity price patterns, as seen in Figure 7.10. Therefore, a limited time and date range should be used to train the agent on more specific periods rather than a full year.

The productivity result of this example is shown in Figure 7.12 below.

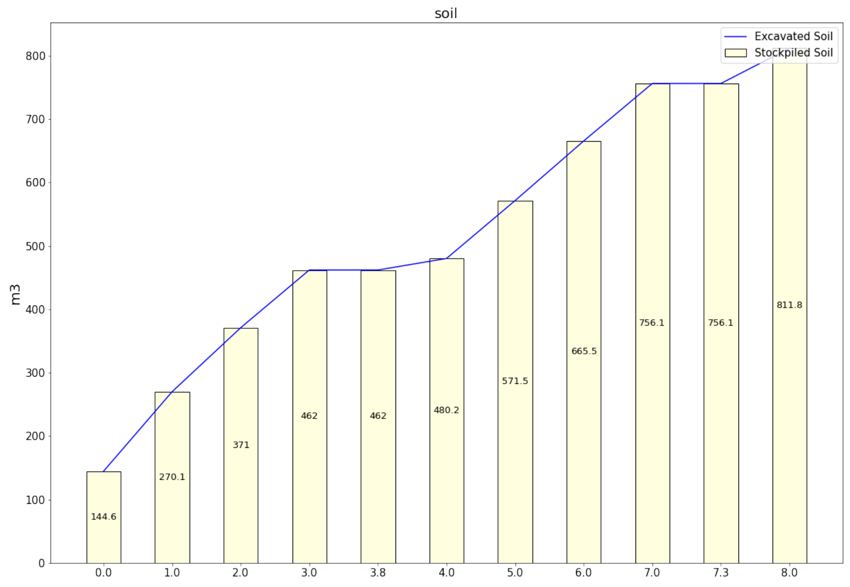


Figure 7.12: *Amount of Excavated Soil in* m^3

This graph shows that the agent was able to complete the productivity goal of excavating $800m^3$ of soil within 9 simulated hours. Therefore, the agent did not require the 10 simulated hours that was available to it.

7.5 Results from other use cases

The framework was configured to optimize the other use cases that were explained in Section 6.9. The purpose of applying this framework to other use cases is to demonstrate the framework's applicability to optimize other use cases in the electric earthwork domain. Therefore, this section serves as further validation of the generalizability and applicability of the framework.

7.6 Minimize Electricity Cost for Overnight Charging Configuration

The optimization framework's training environment can be configured to minimize the electricity cost for overnight charging by including the electricity cost state signals and setting the reward signal to minimize electricity cost, as described in Sections6.2.4, and 6.4.3.

However, the reward for completing a productivity goal is replaced by a reward that motivates the agent to reach the maximum energy capacity of the machines and BESS at the end of the episode, as explained in Section 6.9.1. Furthermore, the workload functions 6.5.2 are deactivated to ensure that the workloads are not simulated for the machines.

The generated earthwork scenarios consist of one excavator and one loader. Each training episode begins with a low remaining capacity for the BESS and the machines. This is done to train the agent to charge the BESS multiple times in order to provide enough charge to the machines. Therefore, the initial remaining capacity of the BESS is calculated using Equation 6.17 with a limit of maximum $150kWh$.

Furthermore, the time and date range is set to the month of October, where each episode begins at the seventeenth hour of the day, and the time step range is set to 18 hours. The same electricity prices are used as in the previous configuration. However, only the prices for October are available to the agent to increase the training results.

There are multiple action-dependent reward signals for this configuration, as explained in Section 6.4.3. When the agent selects the charge action, the agent's reward signal is calculated using Equation 6.9. When the agent selects any of the discharge actions, the agent's reward signal is calculated with Equation 6.10. However, Equation 6.24 is used as the reward signal for the final time step.

The number of training episodes for this version is set to 1400. This number was identified using the method explained in 6.7.2.

The returns of training the framework to solve the task of minimizing the electricity cost for overnight charging are shown in the graph 7.13 below.

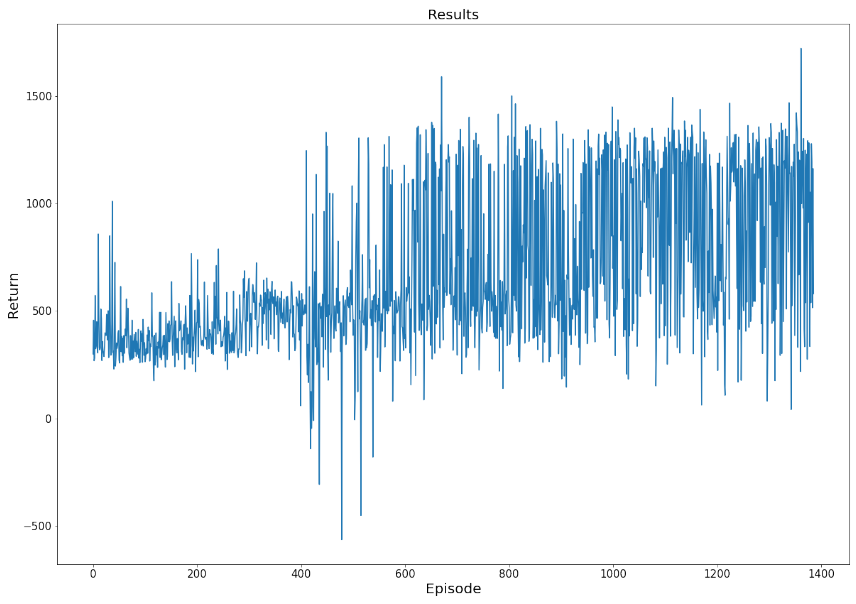
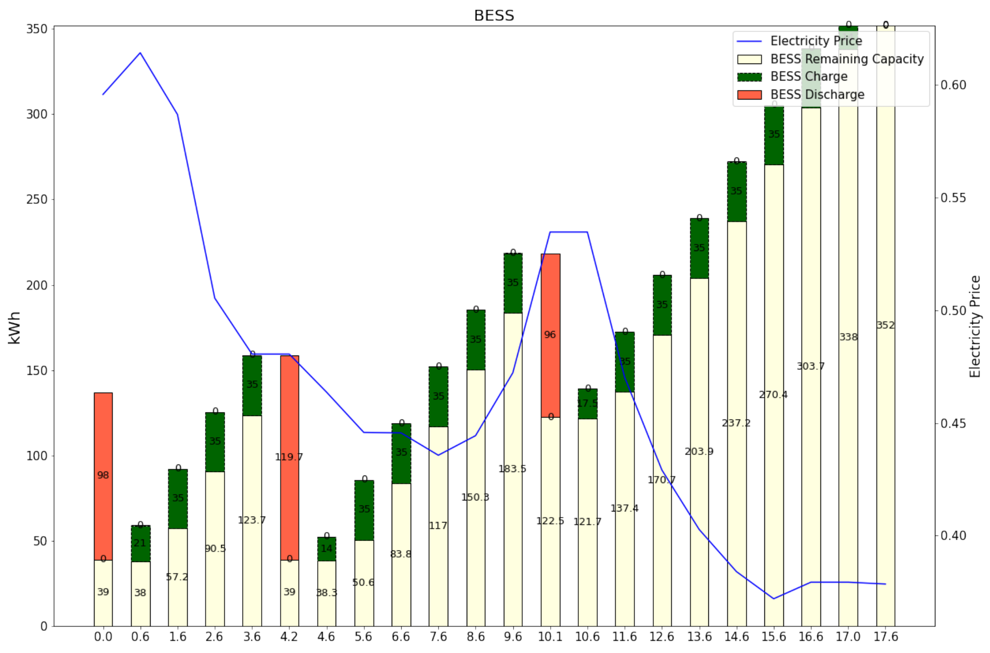


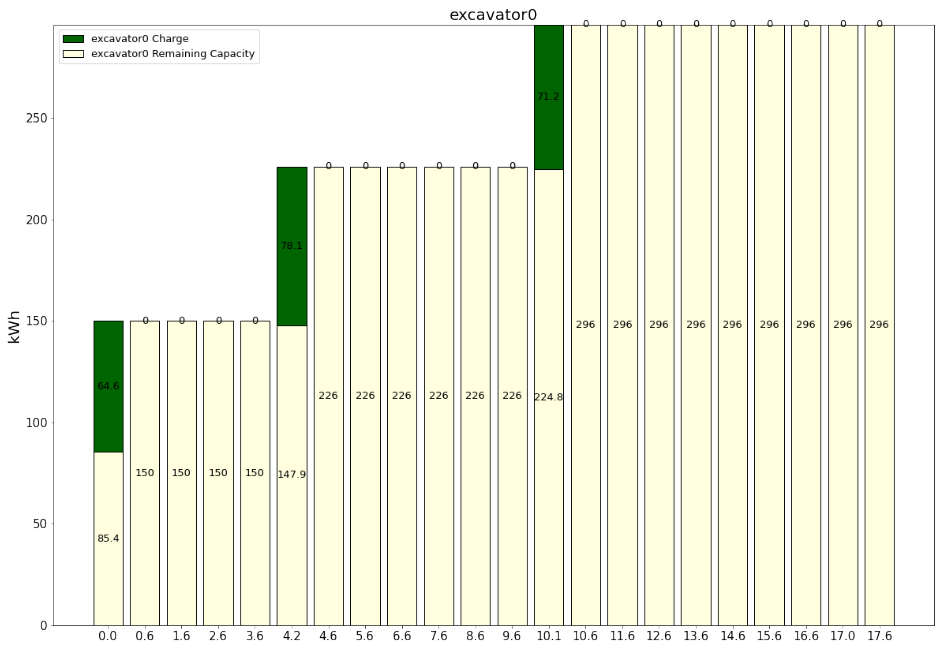
Figure 7.13: *Returns from training to solve the task of overnight charging*

The results show that the agent struggled to increase the return from 500 in the first 400 episodes. Between episode 400 and 600, the agent explored different policies and attempted to exploit the varying electricity price patterns. From Episode 600, the agent's maximum return is identified as approximately 1500. However, the noise in the return for the remaining episodes is due to the varying electricity price patterns. Therefore, the agent can only maximize its return on certain days in October. However, these returns are more consistent than from the previous configuration.

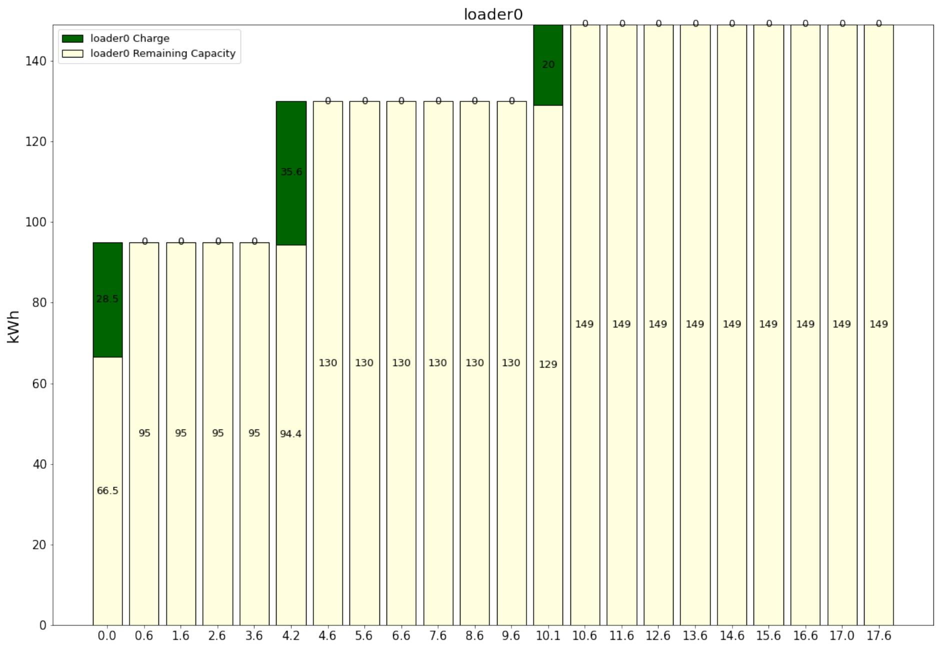
An example of how the agent solves the task for an unseen scenario was generated using the trained policy and is presented in Figure 7.14 below:



(a) BESS Energy Flow



(b) Excavator energy flow



(c) Loader Energy Flow

Figure 7.14: *Example of an Optimized Scheduling for Overnight Charging*

Graph (a) shows that the agent immediately discharges the BESS to capitalize on the high electricity price. In the following time steps, the BESS is charged as the price of electricity drops. The agent discharges the BESS again as the electricity price stabilizes in the fourth time step. The agent continues to charge the BESS until a final peak in the electricity price is expected. The agent correctly identified the peak and was able to charge the remaining amounts to the machines shown in graphs (b) and (c). This allowed the agent to fully recharge the BESS in the remaining time steps while the electricity price was at its lowest during the episode.

7.6.1 Estimating the Minimum Required Capacity of a BESS

The method to automatically estimate the minimum required capacity of a BESS was applied to the framework that was trained to maximize workloads, as explained in Section 6.9.2.

The results from this use case serves as a further sensitivity analysis of the input parameters because the maximum discharge power and the minimum capacity of the simulated BESS are dependent on the maximum capacity of the BESS. Therefore, the same rate of change applied to the maximum capacity is applied to the maximum discharge power and the minimum capacity.

For visualization purposes, an average reward of 10000 indicates that the updated maximum capacity of the BESS was able to generate a consistent return within the range of 800 to 900, as identified in figure 7.2 depicting the training results of maximizing the workloads. This was done to improve the visualization of the results of the estimation function and did not affect the policy, as the agent was already trained with reward equation 6.7.

The user-defined value to increase the maximum capacity of BESS was set at $10kWh$, and the number of episodes in each sequence was set to 10. Graph 7.15 shows the results of the estimation process.

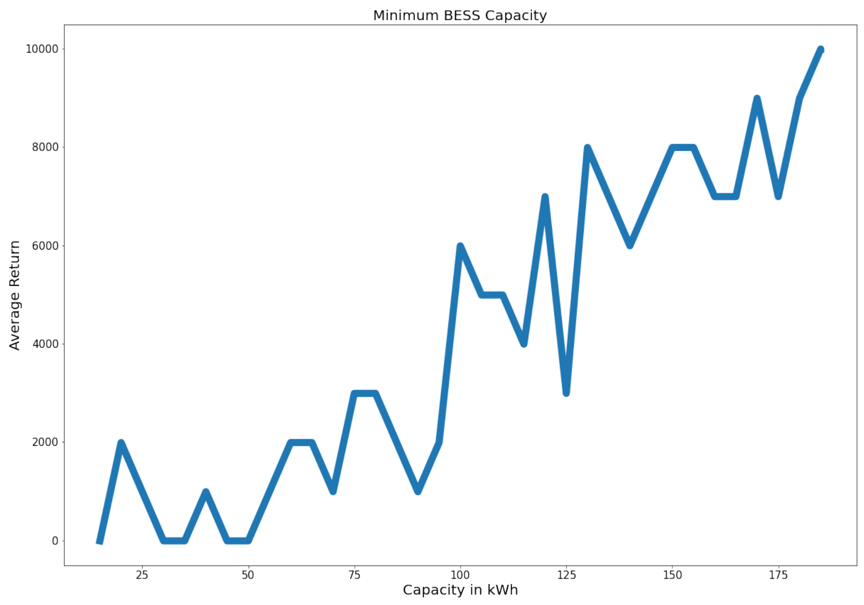


Figure 7.15: *Estimating minimum required Capacity for solving the task of minimizing completion time of assigned workloads*

For this configuration of the framework, the minimum required capacity of the BESS was identified as $190kWh$. The graph shows a clear trend towards improved completion rates as the capacity of the BESS increases.

7.6.2 Results from Direct control of a BESS

This use case demonstrates the frameworks ability to control a system in real time after training in a theoretical environment.

The methods explained in Section 6.9.3 were applied to the agent to allow it to perform direct control of a battery simulation model in real time. The agent was trained in the modified training environment shown in Figure 6.19. As explained in Section 6.9.3, this environment is a simplified version of the environment configured to solve the task of minimizing the electricity cost. This modified environment does not include any earthwork dynamics methods, and there is only one discharge action. Therefore, its identified use case is battery arbitrage, as explained in 6.9.3.

After training the agent in the simplified environment, the trained agent is applied to the direct control environment presented in figure 6.21. This environment uses the programming bridge to control actions and read state signals from the Matlab Simulink model 6.20.

An example episode from the direct control version of the framework is presented in graph 7.16 below.

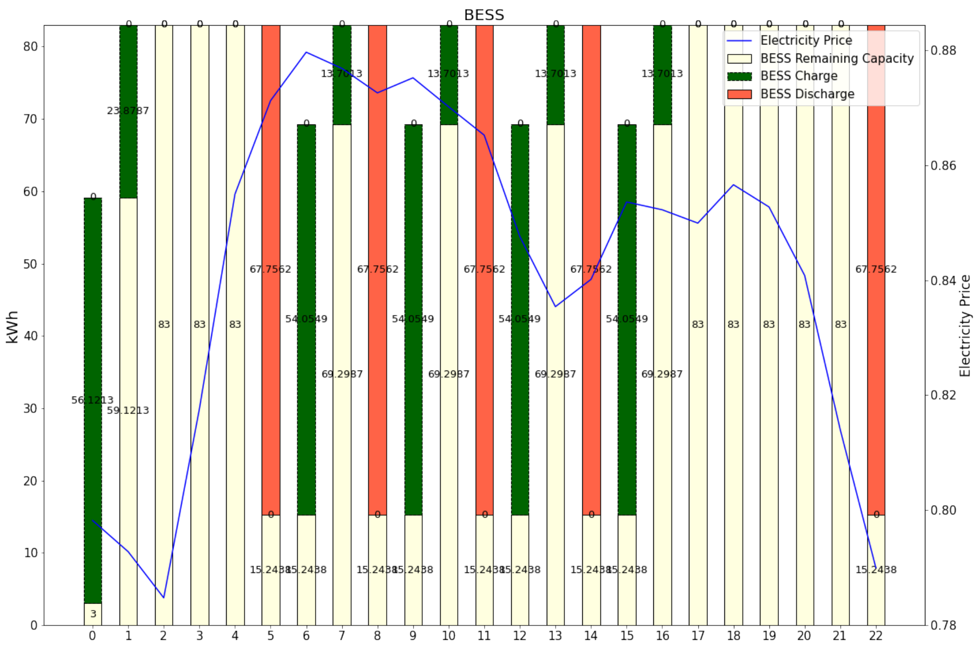


Figure 7.16: *Example episode of the direct control version*

In the first two time steps of the example 7.16, the agent charged the battery simulation model from nearly empty to its maximum capacity while the prices were at its lowest. In the next three time steps, the agent performs the wait action while the price of electricity is increasing. In the fifth time step, the framework predicts that the price of electricity has reached its first peak of the episode and performs the discharge action. The framework was almost correct as the peak occurred in the following time step. From time steps 6 to 11, the framework performs two charge and discharge cycles in which the agent fails to charge when the prices are lower than when it discharges. However, it is again successful from time steps 12 to 14.

Following time step 14, the agent predicts that there will be a second peak in the price of electricity in the remaining time steps, but the prediction is wrong. Therefore, the agent performs the discharge action in the final time step to secure a final discharge reward. The performance of the framework in this example is not perfect. However, it shows that the agent can perform well enough to break even, since the total cost of electricity for this example was approximately €2.6.

Chapter 8

Case Study

A case study is introduced to compare the performance of the deep reinforcement learning framework with a real world scenario. A zero-emission earthwork site in the Greater Oslo region of Norway was selected because it tested the use of a purpose-built BESS as a single source of energy supply for electric earthmoving machines.

8.1 Case Study Background

In Norway, 340,000 tons of CO_2e are associated with construction sites, where 211,000 tons of CO_2e are directly related to earthmoving machines [110]. Oslo, the capital of Norway, contributes the most to these emissions due to the rapid expansion of the city. In 2019, it was estimated that 7% of Oslo's total CO_2 emissions were related to construction site activity and that earthwork machinery accounts for up to 30% of all traffic emissions in the city [18].

Furthermore, the Norwegian Environmental Agency (Miljødirektoratet) [111], has collected data on direct emissions from earthwork machines in Oslo. Figure 8.1 shows these data from 2009 to 2020.

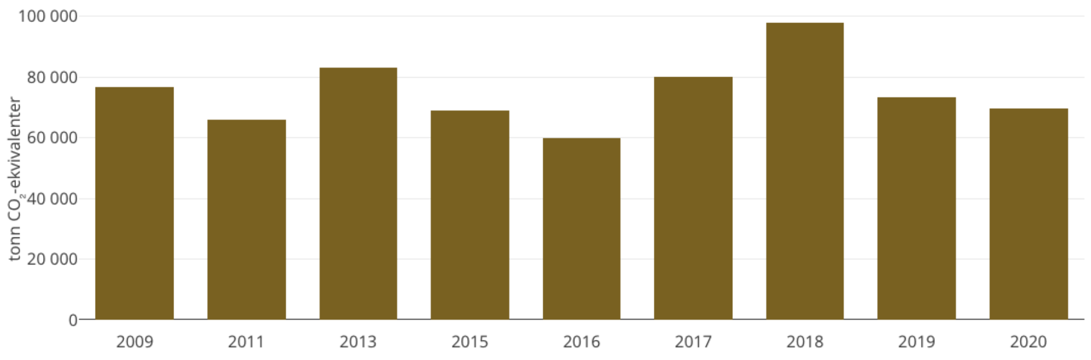


Figure 8.1: *Emissions in CO_e^2 from construction machines use in Oslo*

The data presented in Figure 8.1 show that the annual emissions of the earthwork machines in Oslo range from 59,000 tons of CO_2e to 97,000 tons of CO_2e. The average annual emissions were approximately 74,000 tons of CO_2e. However, there is no indication that annual emissions are being reduced, as the varying annual emissions may be the result of normal variation.

Oslo Municipality estimates that they can reduce future annual emissions by up to 100,000 tons of CO_2 by replacing diesel-powered machinery with electric earthmoving machines [18]. Moreover, Oslo's municipality plans to reduce construction site emissions by 95% by 2030, requiring all public construction sites to operate electric earthmoving machinery by 2025. Therefore, earthwork contractors are required to use electric earthwork machines to win public contracts.

Furthermore, the time required to increase the capacity of a local distribution network in Norway is a minimum of two months and can take up to several years depending on the level of power demanded [1]. Therefore, earthwork BESS have been introduced to enable the transition from using diesel-powered earthmoving machines to electric earthmoving machines.

8.2 Earthwork Scenario

The zero-emission construction site of the case study was established in September 2021 with the purpose of developing a new suburban road. The earthwork machines included a 25-ton electric hydraulic excavator with a battery capacity of $300kWh$ and a 12-ton electric hydraulic excavator with an original battery capacity of $150kWh$. However, the battery capacity of the 12-ton excavator was reduced to

$130kWh$, due to unknown circumstances. On working days, the excavators would work simultaneously from 07:00 in the morning to 17:00 in the evening.

The BESS used in the case study was the BoostCharger developed by Nordic Booster AS, as explained in Section 2.3.3. The maximum capacity of the BESS is $390kWh$, and it has two charging cables that are capable of charging two machines simultaneously at a rate of up to $150kW$ each. The BESS was connected directly to the local grid with a standard connection to the grid that could provide power up to $36kW$.

The BESS was leased to the construction company from an energy supply company. The construction company paid a flat rate per week to rent the BESS, which included the price of the electricity charged from the grid. Therefore, the hourly price of electricity did not affect the earthwork contractor. However, it did affect the energy supply company, which had no control over when the BESS was charged or discharged.

8.3 Quantitative Data Collection and Analysis

Quantitative data on energy consumption were recorded during the month of November 2021. The excavator data was recorded by the earthwork contractor that was responsible for the earthwork processes, while the BESS data was recorded by the company that developed the BESS. Access to these data sets was provided by the site manager and the BESS developers.

8.3.1 Excavator Data

The excavators' state of charge was recorded by machine operators for each working day. The machine operators were instructed to record the state of charge at the beginning of the work day (07:00), at 09:00, before lunch (11:00), after lunch (12:00), at 14:00, and at the end of the work day (17:00). For every recording, the workers would also specify the type of workload achieved by the machine as light, moderate, or heavy. Figure 8.2 shows a summary of the data recorded for the 12-ton excavator.

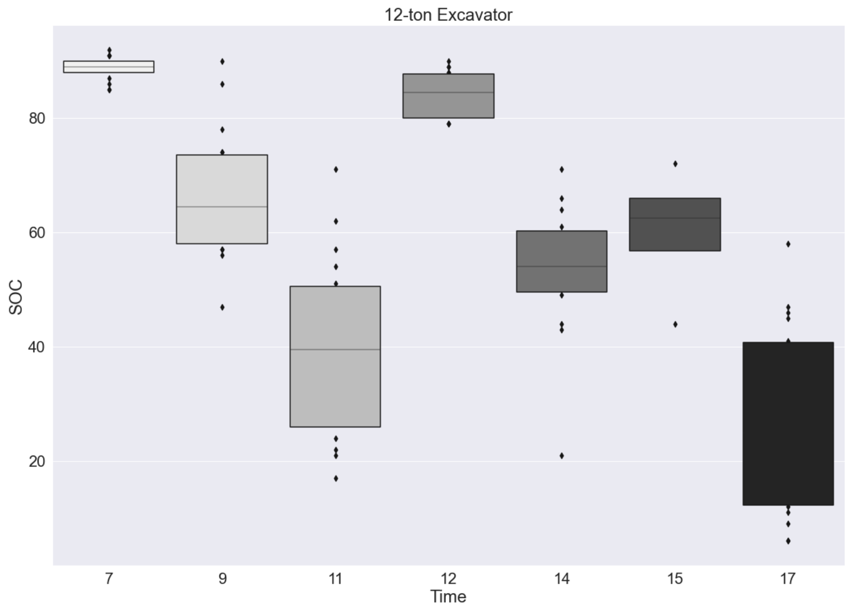


Figure 8.2: Data recorded from the 12-ton excavator

The box plot graph shows that the 12-ton excavator began each working day with a state of charge of between 80% and 90%. This was its full charge as a result of the reduction in its battery capacity. At 09:00, the state of charge would be reduced to approximately 70%. At 11:00, the state of charge could be as low as 20%. However, the average state of charge at this time was around 40%.

From 11:00 to 12:00, the 12-ton excavator was recharged from the BESS while the operators were on lunch break. However, it is noticeable that the excavator was not charged to its maximum capacity, since the average state of charge at 12:00 is slightly lower than at the beginning of the workday. This is also noticeable at 14:00, where the state of charge is slightly lower than at 09:00.

Furthermore, the state of charge was recorded at 15:00 on certain days when the operator had to charge the 12-ton excavator a second time to continue working for the remaining two hours. At 17:00, the state of charge was recorded as low as 6%. However, the average state of charge was approximately 25%.

Figure 8.3 shows a summary of the data recorded for the 25-ton excavator.

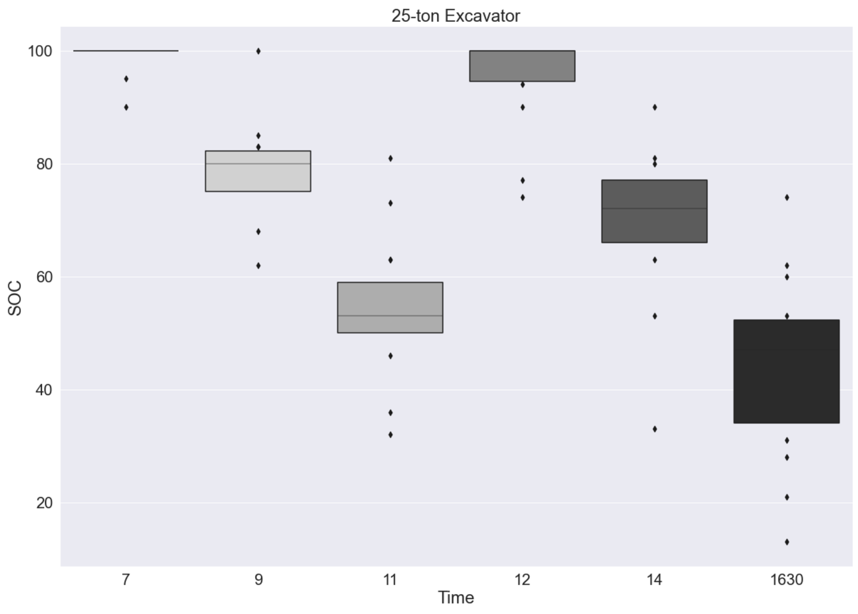


Figure 8.3: *Data recorded from the 25-ton excavator*

The box plot graph shows that the 25-ton excavator began each working day with a state of charge of 100%. This shows that there was no damage to its battery packs. At 09:00, the state of charge would be reduced to approximately 80%. Before lunch break, the average state of charge was approximately 55%. However, on some days it could be as low as 35%.

From 11:00 to 12:00, the 25-ton excavator was also recharged from the BESS while the operators were on lunch break. The average state of charge at 12:00 is also slightly lower than at the beginning of the workday. This is also noticeable at 14:00, where the average state of charge is slightly lower than at 09:00.

This excavator did not require a second charge on any day as its remaining capacity at 14:00 was sufficient to continue working for the remaining time. The state of charge was recorded at 16:30, rather than at 17:00. This indicates that the 25-ton excavator tasks were completed before the 12-ton excavator tasks. However, it is unknown if this is the case. The average state of charge at 16:30 was approximately 45%.

The box plot graphs reveal a linear trend in the average values for the state of charge. However, the outliers depict that there are many influencing factors that can have a non-linear and unpredictable impact on the excavators' energy consumption.

The type of workload was the only recorded influencing factor for this case study. Figure 8.4 shows when the different types of workloads were recorded for both excavators.

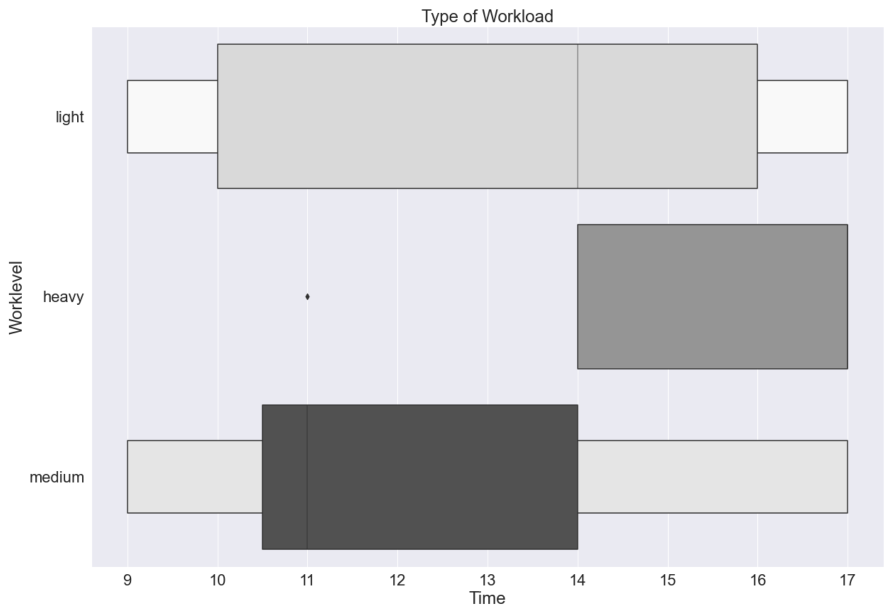


Figure 8.4: *Type of workloads recorded*

In Figure 8.4, a trend can be observed where medium workloads are recorded more frequently between 11:00 and 14:00, while heavier workloads are recorded after 14:00. This is correlated with the deeper depth of discharge recorded by the excavators at the end of the working day. The reason for this occurrence has not been specified. However, it is feasible to assume that the excavation process reaches deeper layers with denser soil types that demand more energy to be excavated as the excavation process progresses and that similar excavation processes are repeated daily as the development of the road unfolds.

In 2022, the company began to record detailed energy consumption data directly from the excavators. The quality of the data is poor, as the data points are scattered and several timestamps are missing. However, all recorded data points are collected for the month of July in 2022 in an attempt to identify the energy consumption trends of the two excavators. The data recorded from the 12-ton excavator were the most incomplete. Figure 8.5 shows the energy consumption of the 12-ton excavator.

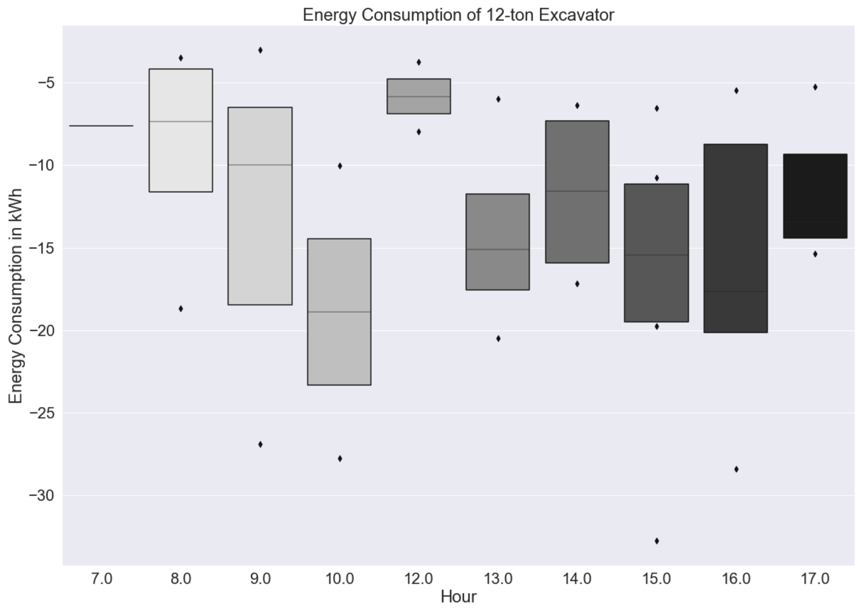


Figure 8.5: *Energy Consumption of 12-ton Excavator*

This graph gives an indication that the 12-ton excavator consumes between $10kWh$ and $20kWh$ an hour, and that it can consume up to $35kWh$ in rare occasions. However, some data are missing (e.g., for timestep 07:00 and 11:00). Therefore, the average values cannot be trusted.

Figure 8.6 shows the energy consumption of the 25-ton excavator.

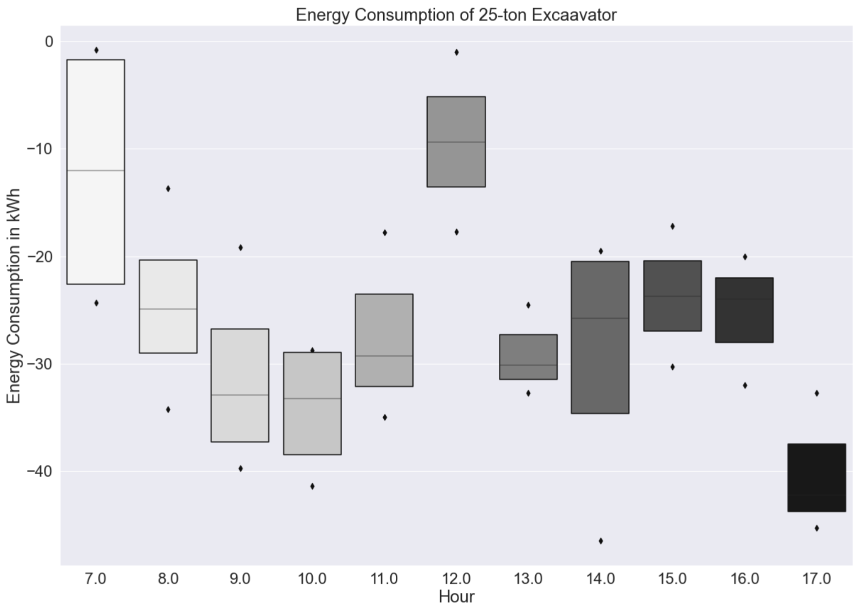


Figure 8.6: *Energy Consumption of 25-ton Excavator*

This graph shows that there were more data points recorded for the 25-ton excavator than for the 12-ton. The graph indicates that the 25-ton excavator consumes between $20kWh$ and $40kWh$ an hour.

8.3.2 BESS Data

The energy consumption data from the BESS used in the case study were collected directly from a human-machine interface that recorded data points every five minutes. During November, the BESS consumed a total of $9,612kWh$ where $5,809kWh$ was supplied as AC to the excavators for overnight charging and $3,803kWh$ as DC to fast charge the excavators during working hours.

A maximum discharge power of $252kW$ was recorded when both excavators were charged simultaneously, and the average duration of these DC fast charge processes was 57 minutes. Furthermore, the charging power from the grid to the BESS ranged from $31kW$ to $36kW$, and the average daily consumption from the grid was $480kWh$ with several days reaching up to $600kWh$.

Figure 8.7 shows a summary of the remaining capacity of the BESS during the working days in the month of November.

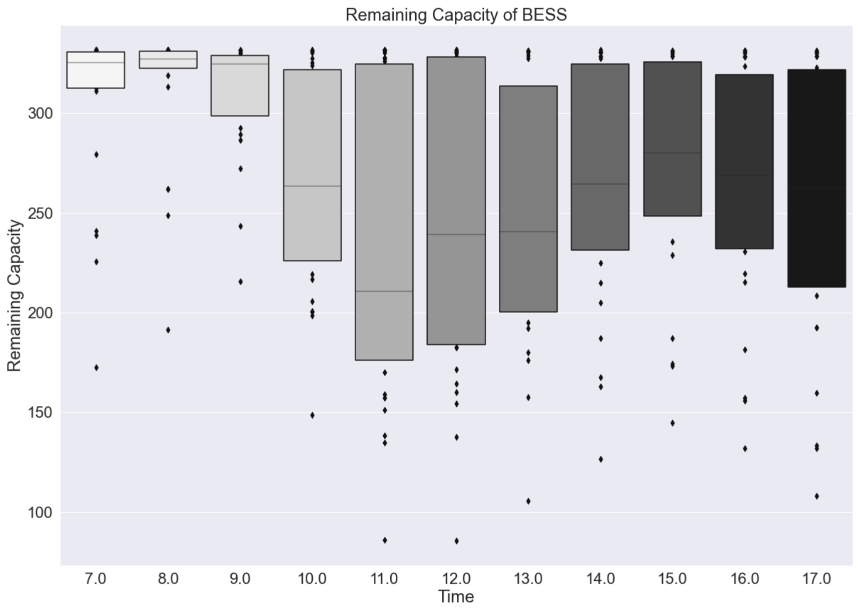


Figure 8.7: *Data recorded from the BESS*

The box plot graph shows that the BESS began each working day with a capacity of around $330kWh$, and that the excavators were charged between 10:00 and 12:00. However, the deepest discharges are recorded between 11:00 and 12:00, which is consistent with the data collected from the excavators. Furthermore, the graph reveals that the BESS began recharging immediately after the excavators were charged. However, the graph shows that the BESS was sometimes discharged for the second time between 15:00 and 17:00 with lesser intensity. This is in correlation with the 12-ton excavator that demanded an additional charge on certain days. However, this could also be the result of other load demands that were not recorded (e.g., electric cars or trucks visited the site). The lowest state of charge of the BESS was recorded at 11% while the highest was 95%.

8.4 Qualitative Data Collection and Analysis

Separate interviews were conducted with the site manager of the case study earthwork site and the company that developed the purpose-built BESS for the earthwork process. The purpose of the interviews was to explore experiences related to electrification of earthwork processes.

8.4.1 Summary from the Interview with the Construction Site Manager

The site manager owned an earthwork company that specialized in road construction. The company had acquired two electric excavators in 2021. However, this was the company's first project in which a BESS was used as an interface between the grid and the excavators.

He explained that the electrified earthwork processes were kept as similar as possible to the diesel-powered earthwork processes by purposefully charging the excavators during lunch breaks to reduce the impact of the transition from diesel to electric-powered machines. Therefore, the construction process was not adapted to accommodate the use of electric earthmoving machines.

To enable efficient charging at lunch time, the excavators had to start the working day fully charged. This was achieved by overnight AC charge from the BESS to the excavators. This process did not account for fluctuating electricity prices, as the electricity price did not directly impact the construction company.

The construction manager's main concern regarding the transition to electrified earthmoving machines was to identify processes that would make the work more efficient. This could potentially save costs by reducing the time workers are paid to wait while machines are charging because lunch breaks were often extended from 45 minutes to an hour to allow excavators to charge to their respective maximum capacities. Therefore, his goal was to reduce the charging time per excavator to a maximum of 45 minutes per day.

Furthermore, daily productivity goals were deduced from three-week plans. These productivity goals represented excavation from one point to another, and a cross section of the path would indicate how intense the work would be based on the layers of soil types. However, no other influencing factors were used when planning the earthwork process.

8.4.2 Summary of the Interview with the BESS developer

The BESS developer explained that earthwork companies and other stakeholders are slowly adapting to new roles and responsibilities associated with zero-emission construction sites. However, the developer's service technicians were often called to assist earthwork professionals with the new BESS technology on site.

The main challenge emphasized by the developer was estimating the appropriate maximum energy capacity of the BESS. A BESS with a large energy capacity faced challenges with internal temperature, as battery cells would generate a lot of heat, which proved to be difficult to manage with the available air conditioning systems. Therefore, the earthwork process could be halted when the BESS was not available due to temperature-mitigating measures.

This problem could be avoided by minimizing the number of battery cells required to operate efficiently. However, neither developers nor earthwork professionals have access to a method to estimate the minimum energy capacity required for an earthwork BESS.

Furthermore, battery cell technology was identified as the main cost per unit sold. Therefore, the estimation of optimal capacities for the intended use of their products was identified as a key performance indicator to reduce costs and increase sales.

The method used by the BESS developer to decide on the maximum energy capacity of the BESS prototype was to ensure that it could fully recharge two 12-ton excavators over a period of one hour. However, testing the product on the market resulted in a demand for a BESS with lower energy capacity.

8.5 Framework Application

The performance of the framework was compared with the actual results of the case study to evaluate how the electric earthwork process can be improved. The framework was used to generate a schedule that would represent the case study approach using the quantitative data collected. The earthwork scenario generator was configured to generate scenarios with one 25-ton excavator and one 12-ton excavators. Figure 8.8 is a visualization of the case study scenarios.

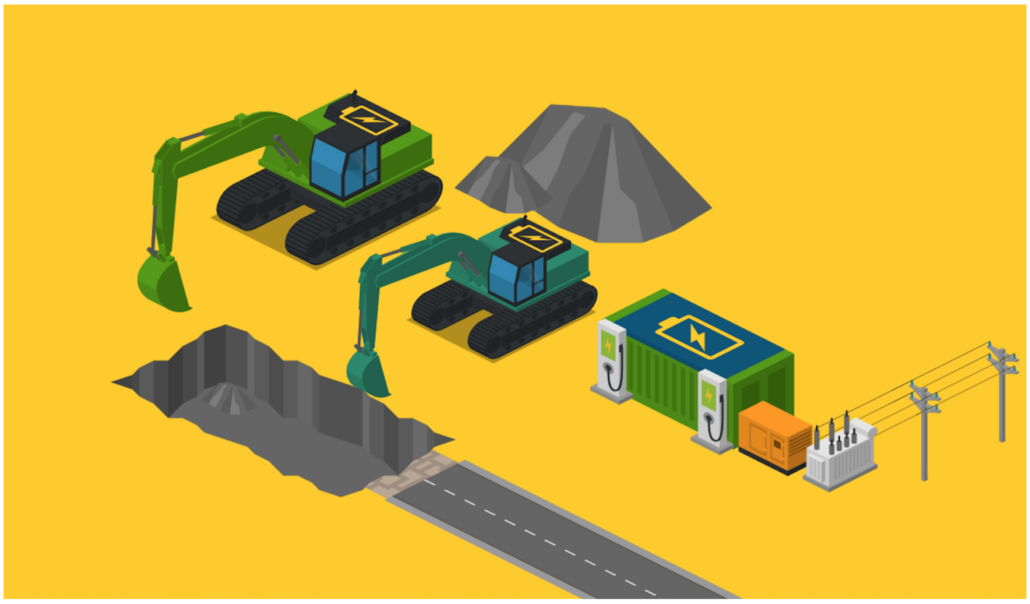


Figure 8.8: *Visualization of the case study scenario*

.

In addition, the input values of the framework's environment were configured to represent these specific machines, as shown in Table 8.1.

Parameter	**25-ton Excavator**	**12-ton Excavator**
Remaining Capacity at t_0	300kWh	130kWh
Maximum Capacity	300kWh	130kWh
Minimum Capacity	30kWh	13kWh
Maximum Charge Power	150kW	70kW
Maximum Output Power	120kW	50kW
Round-Trip Efficiency	0.95	0.95
Idling Power	5kW	3kW
Bucket Capacity	$1.5m^3$	$1m^3$
Cycle Time	36sec	36sec
Operator Efficiency	95%	95%
Operational Efficiency	(65, 83)%	(65, 83)%

Table 8.1: *Case Study Machine Factors*

Operators were assumed to be experienced and the operational efficiency is represented by the recommended range in the literature [51].

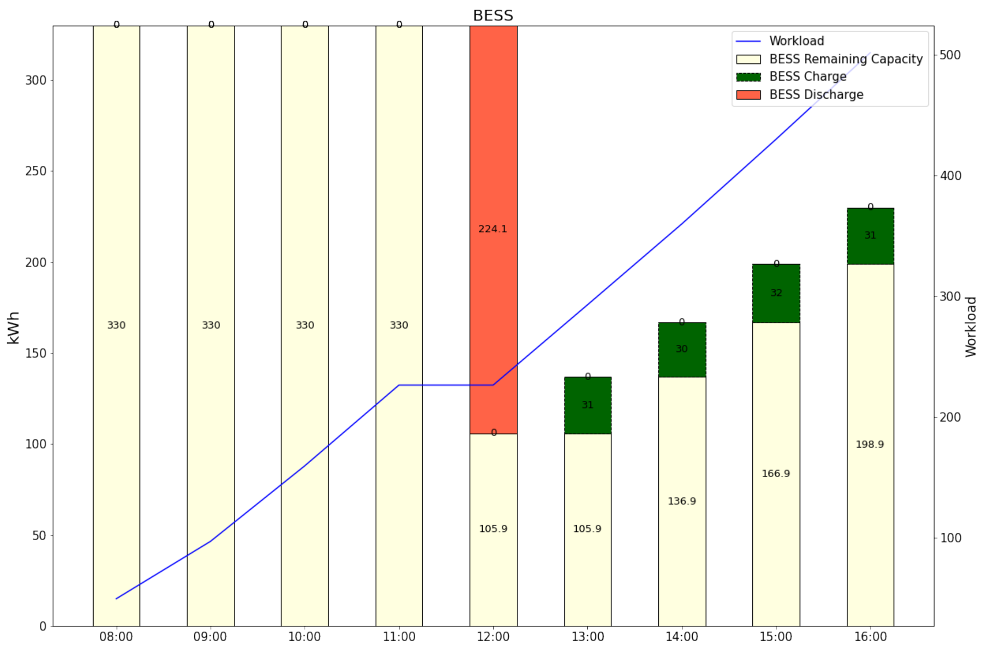
Furthermore, the types of workloads recorded by machine operators were represented by adapting the soil power factors of the framework to generate excavator-specific workloads that are within the ranges provided in Figure 8.6 for the 25-ton excavator and Figure 8.5 for the 12-ton excavator. Table 8.2 shows how these workloads are represented.

Workload Type	**Light**	**Medium**	**Heavy**
Volume of Soil Assoc. with Workload	350 m^3	400 m^3	400 m^3
Location of soil	upper	middle	lower
12-ton Soil Power Factor	(0.3, 0.45)	(0.55, 0.6)	(0.6, 0.8)
25-ton Soil Power Factor	(0.3, 0.35)	(0.35, 0.45)	(0.5, 0.55)

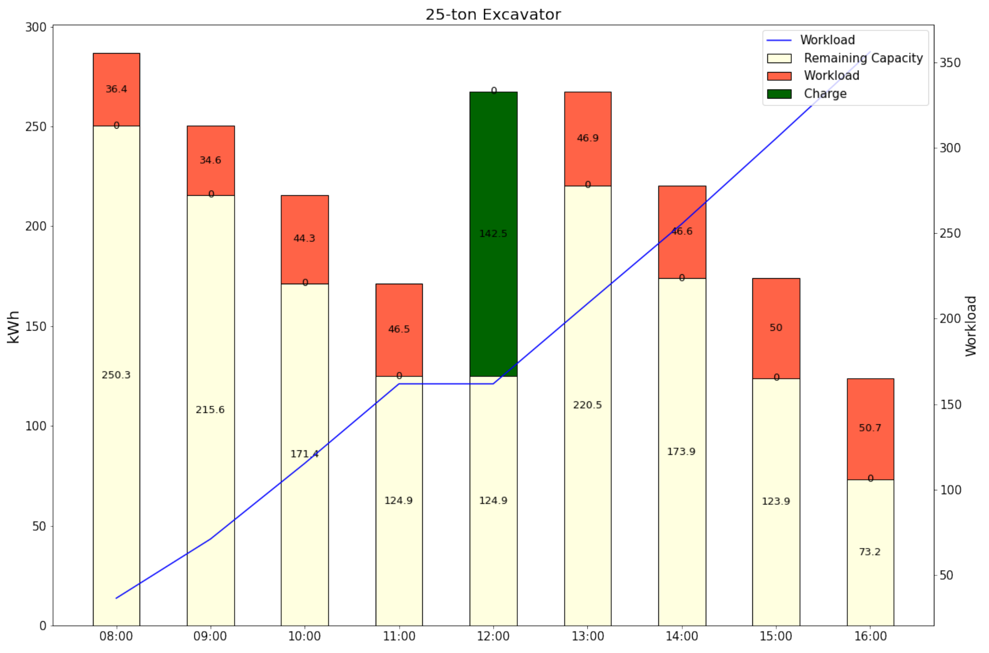
Table 8.2: *Workload Factors*

Here, the type of workload changes when excavators have excavated the volume of soil associated with the type of workload to generate similar energy consumption patterns as recorded in Figure 8.3 for the 25-ton excavator and Figure 8.2 for the 12-ton excavator.

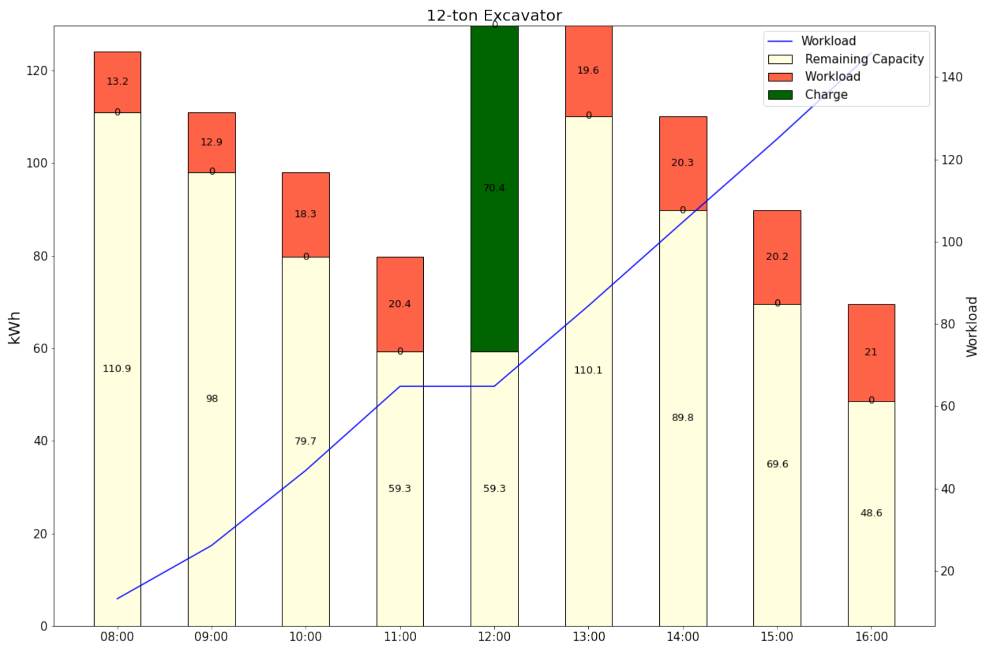
The case study configured framework was used to generate a schedule to represent the case study approach. This was done by manually selecting when actions were to be performed, without training any agent. Figure 8.9 is a generated schedule in which the case study approach is used.



(a) BESS Energy Flow



(b) 25-ton Excavator Energy flow



(c) 12-ton Excavator Energy Flow

Figure 8.9: *Case Study Schedule*

Graph (a) shows that the BESS began the working day with full charge and that it was discharged to charge both excavators between 11:00 and 12:00. Following discharge, the BESS was recharged with the same intensity as recorded in 8.7. Graphs (b) and (c) show the energy consumption patterns of the excavators. This schedule represents the case study approach because the hourly energy consumption values are within the ranges presented in Figures 8.6 and 8.5, and the remaining capacities of the excavators are within the state of charge ranges presented in Figures 8.3 and 8.2. The accumulated workloads simulated for the excavators in this scenario amounted to $493kWh$.

8.5.1 Optimization Results

The optimization process was applied to the environment representing the case study approach by training the agent to maximize the workloads of the two excavators for 4000 episodes. The training results are shown in figure 8.10.

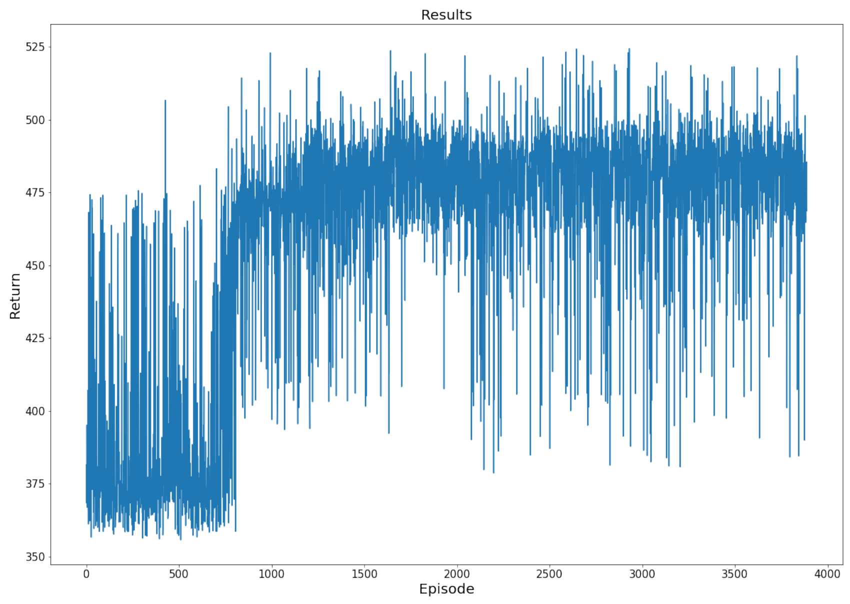
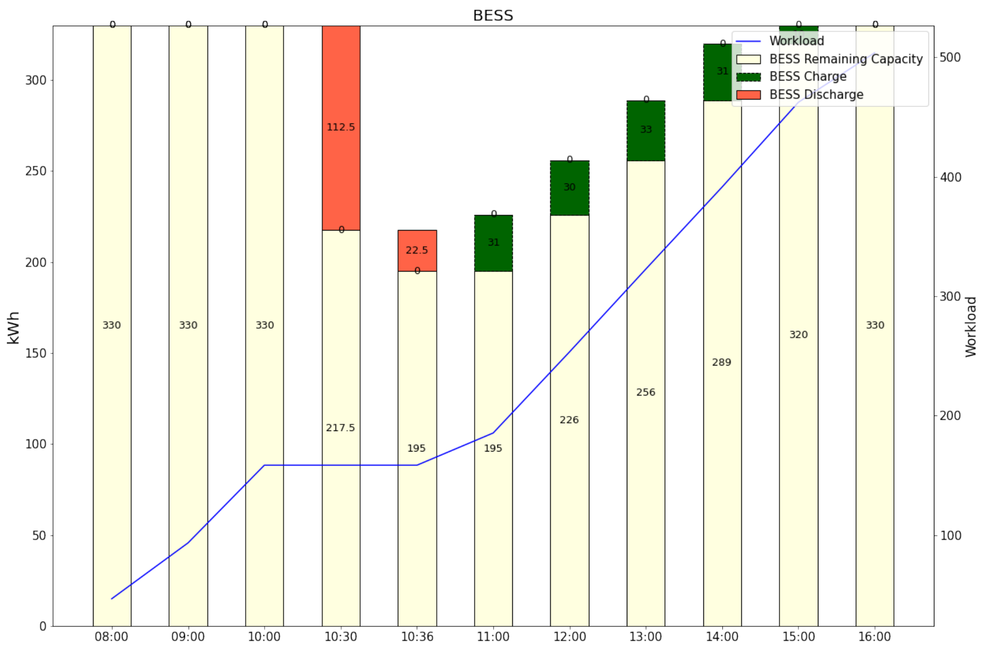


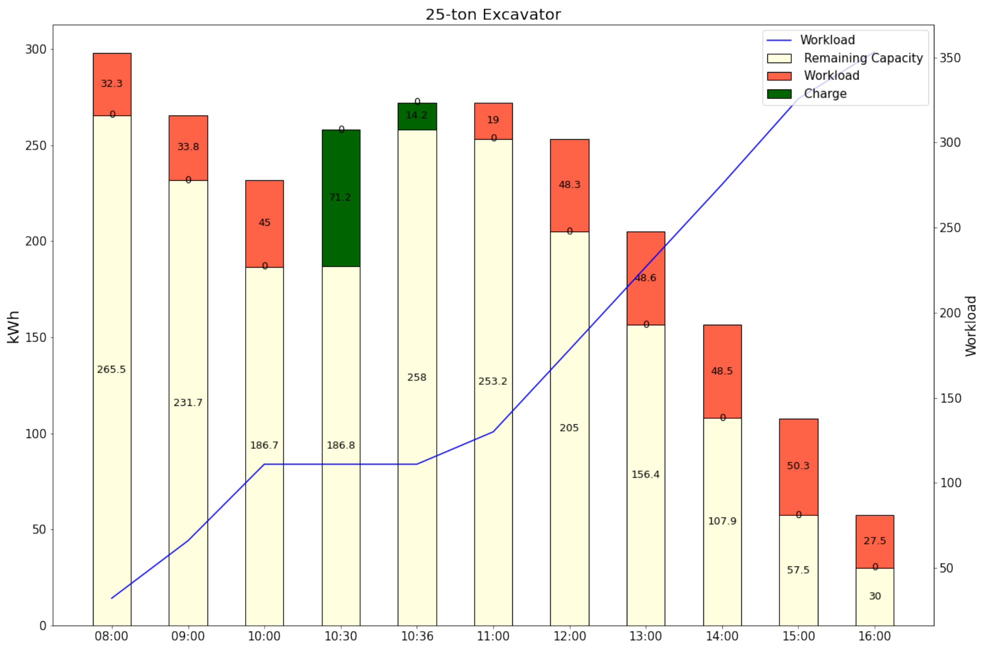
Figure 8.10: *Returns from training the agent to optimize the case study schedule*

The results show that the agent explored different policies for 800 episodes. Between episodes 800 and 1000, the agent developed a policy that was able to generate returns of approximately 500. In the remaining time steps, the agent explored different ways to improve the developed policy. However, it was not successful in significantly improving the developed policy. Therefore, a return of approximately 500 is considered optimal.

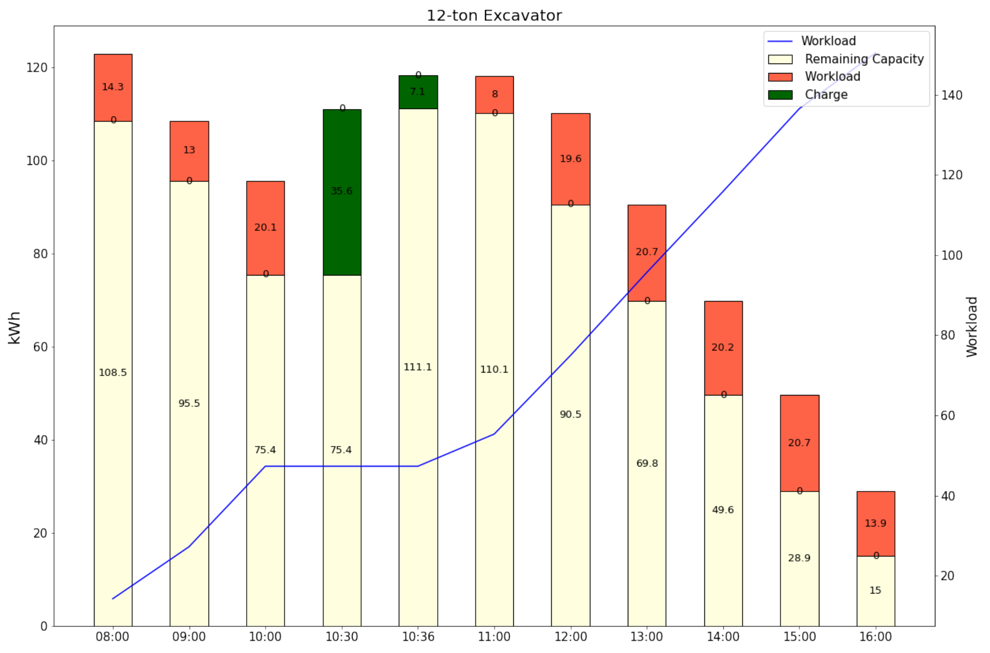
Two optimized schedules are presented to show the performance of the framework in optimizing the case study schedule. Figure 8.11 shows the first optimized schedule.



(a) BESS Energy Flow



(b) 25-ton Excavator Energy flow



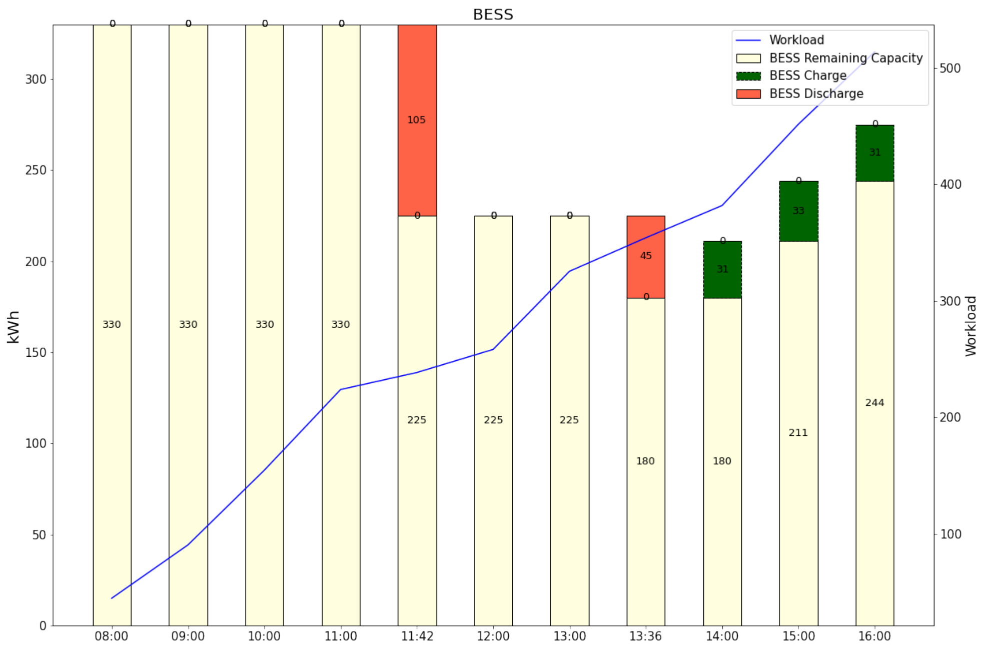
(c) 12-ton Excavator Energy Flow

Figure 8.11: *First Optimized Schedule for Case Study*

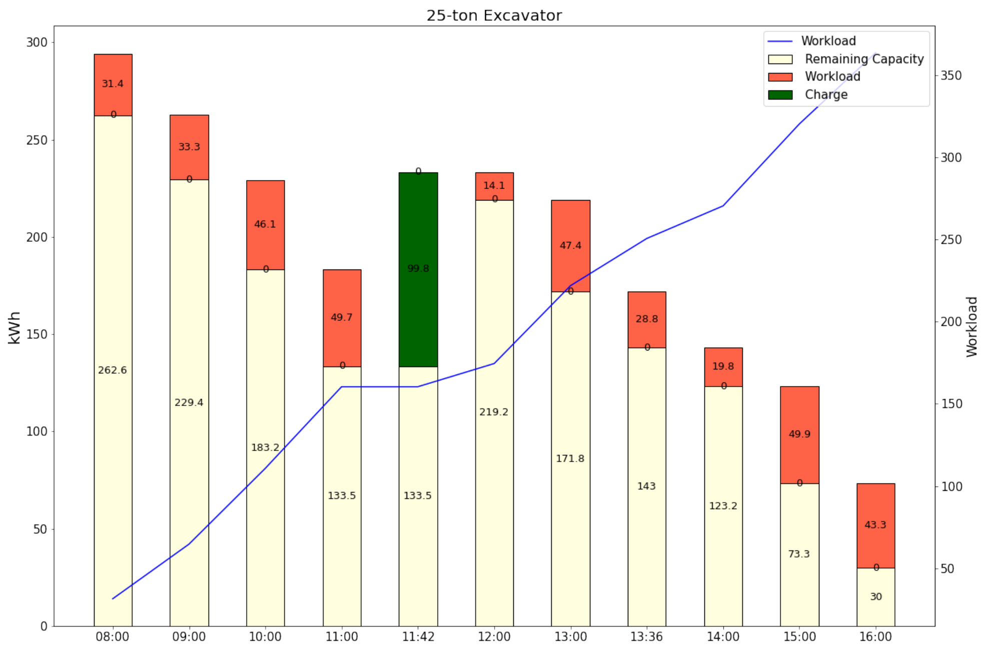
The first optimized scheduled generated a total workload of 508. The agent was able to increase the total workload by spending less time charging the excavators. The excavators were charged between 10:00 and 10:36, and the agent did not use the full hour to charge the excavators because the charge-limit function determined that the excavators had enough individual charge to maximize workloads for the remaining time steps.

However, the agent was unable to maximize the excavators' workload in the final time step, as the excavators reached their respective minimum energy capacities. This revealed an interesting feature of the framework, in which the agent has learned that using the rule-based charge-limit function early in the simulation process may lead to reduced performance. Therefore, the agent decided to continue charging the excavators after the initial charging process was terminated at 10:30 in an attempt to further increase the remaining capacities of the excavators to maximize the workloads in the final time step. However, this charging process was also terminated by the charge-limit function at 10:36. This is a great example of how the agent can develop ways to optimize the schedule when a rule-based function limits its behavior.

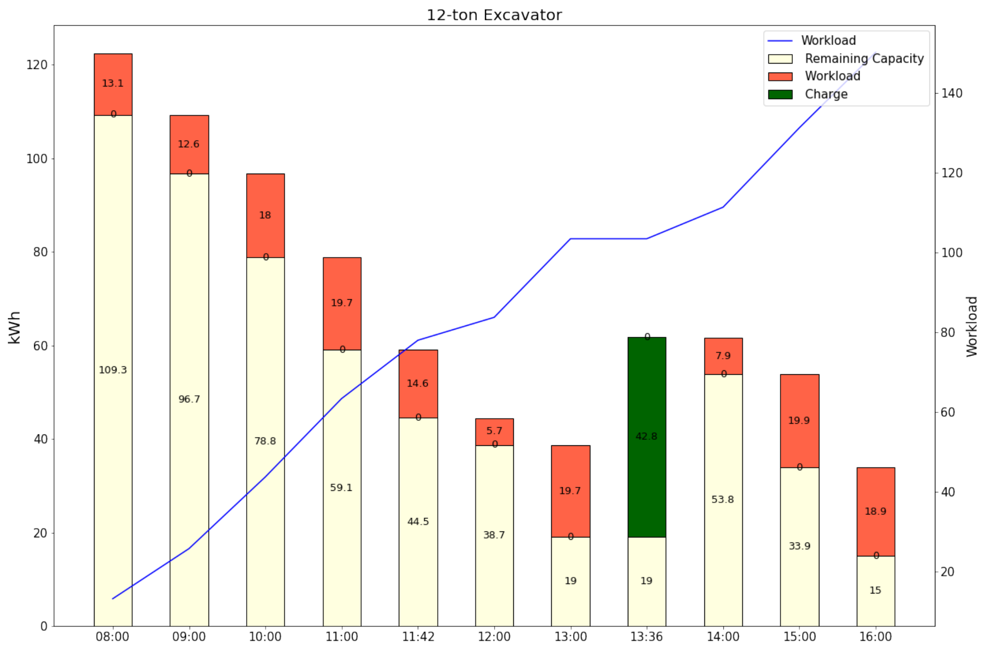
A second optimized schedule was generated using the same trained agent to show how a different approach would generate an even greater total workload of 519. Figure 8.12 shows the second optimized schedule.



(a) BESS Energy Flow



(b) 25-ton Excavator Energy flow



(c) 12-ton Excavator Energy Flow

Figure 8.12: *Second Optimized Schedule for Case Study*

The second optimized schedule shows how the agent is able to maximize the total workloads of both excavators on all non-charging time steps by charging the excavators individually on different time steps. More time is spent charging than the first optimized schedule. However, the earthwork process is not disrupted, as one excavator can continue working while the other is charging. This result shows how the case study process can be improved by charging the machines separately to further accommodate the use of electric excavators and increase productivity.

Furthermore, the sizing method was applied to the case study version of the framework to estimate the minimum required BESS capacity that would generate similar total workloads. The estimation function was configured as explained in Section 7.6.1. The results of estimating the minimum required BESS capacity are shown in Figure 8.13.

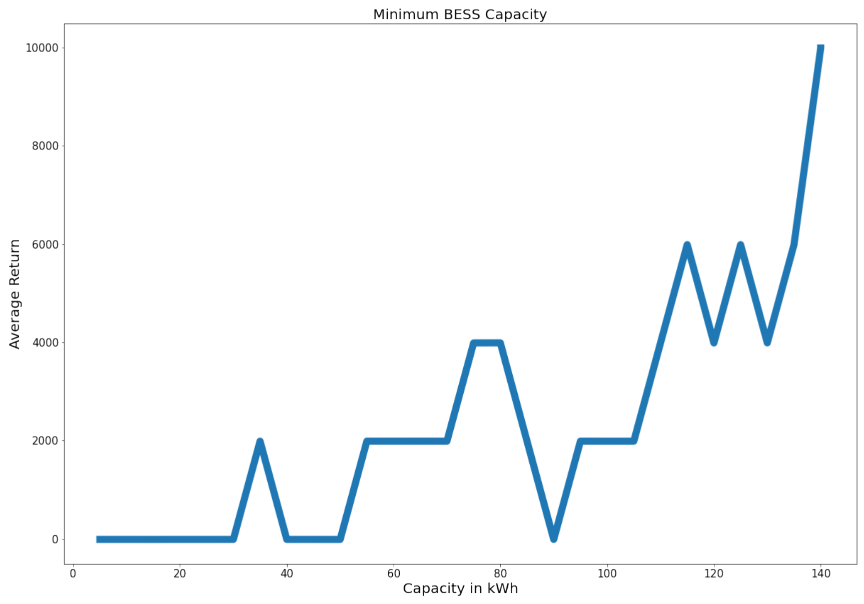


Figure 8.13: *Estimation of the minimum required capacity for the case study example*

The results indicate that the minimum required capacity of a BESS that will produce similar results is $140kWh$. This further undermines the concern of BESS developers about the sizing of future BESS products.

Chapter 9

Conclusion

The deep reinforcement learning framework proposed in this thesis is developed to optimize energy supply schedules for electric earthwork processes where a single BESS is used to charge multiple electric earthmoving machines. The framework is designed to be applicable to several purposeful use cases to provide earthwork professionals with a tool to facilitate the transition from diesel-powered to electric earthwork processes.

The background chapter explains how an earthwork BESS can make electric earthwork processes more feasible by charging electric earthmoving machines efficiently while avoiding unnecessary upgrades to the distribution network.

The literature review identified factors that affect the energy consumption of electric earthmoving machines and addresses previous modeling efforts of diesel-powered earthwork processes due to the lack of standardization, data scarcity, and under-specified influencing factors in the field of electric earthwork research. Moreover, the inability of multi-linear regression methods and discrete event simulation in optimizing electric earthwork processes is emphasized due to their reliance on system identification, detailed data sets, and lack of a generalized response to stochastic scenarios. The proposed methodology is capable of overcoming these challenges by utilizing the model-free features of reinforcement learning in combination with the generalizability of artificial neural networks to optimize decision-making strategies in stochastic electric earthwork scenarios.

Furthermore, the framework generates unique earthwork scenarios to serve as training data for the generalized response based on a set of input values that the user can obtain from case studies, earthwork contractors, and manufacturers of electric earthmoving machines. A trained framework outputs values that represent optimal energy supply decisions for individual machines and the BESS in terms of an optimization goal set by the user. The results show the performance of the framework in achieving multiple optimization goals with varying configurations and input values. The results also demonstrate how this framework can be used to facilitate bidding and reporting processes by generating optimized energy supply schedules and identifying opportunities to increase efficiency.

The optimization goals were identified to answer the main research question on how an earthwork BESS can be operated to optimize the energy supply for electric earthwork processes. The goal of maximizing workloads in a given time frame, together with the goal of minimizing the time required to complete a productivity goal, answers the question of how electric earthwork processes can be optimized to increase efficiency. The goal of minimizing the electricity cost while completing a productivity goal (and charging overnight) answers the question of how electric earthwork processes can be optimized to reduce electricity costs.

The applicability and generalizability of the framework are validated by controlling a battery system in real-time and by estimating the minimum required energy capacity for an earthwork BESS by evaluating ascending values for the maximum energy capacity of a BESS in the environment of a pre-trained version of framework.

The case study explores the impact electrification of earthmoving machines have on earthwork processes by demonstrating how the framework can be used to improve existing earthwork processes to further adopt the use of electric earthmoving machines by estimating future workloads and reducing the time spent charging machines.

The framework was designed with industry application in mind. Therefore, scaleability, modularity, and automation were prioritized when formulating the code. This framework lays the foundations for research aimed at the development of decision support tools to realize efficient electric earthwork processes.

9.1 Discussion and Further Research

The accuracy of the current state of the framework can be significantly improved by better understanding the factors that influence the energy consumption of electric earthmoving machines. Therefore, further research efforts must be made to collect and standardize data from commercially available electric earthmoving machines. This is especially relevant to be able to integrate more earthwork processes, such as grading, spreading, and compacting.

The framework does not address the nonlinear battery dynamics with respect to battery degradation and the effect of temperature on battery cells. Rosewater et al. [104] states that a non-linear energy reservoir model is an underexplored middle ground that may provide improved accuracy with a modest increase in computational complexity. Therefore, a nonlinear energy reservoir model should be considered to improve this framework.

A natural continuation of this project is the integration of temperature management. Specifically, the control of an air conditioning system in the BESS to maintain battery cell temperatures within a safe operating range during charge and discharge cycles. This can be achieved by optimizing precooling schedules based on optimized operation schedules.

Furthermore, multi-agent reinforcement learning, where there is an agent for every earthmoving machine and for the BESS, could potentially improve the performance of the framework. However, this is an underexplored and developing research topic in reinforcement learning where there are few applicable examples.

Appendix A

Appendix 1

A.1 Python Code Example

A.2 Required Python Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import gym
from gym import spaces
import random
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.optimizers import Adam
from rl.agents import DQNAgent, SarsaAgent
from rl.policy import EpsGreedyQPolicy, MaxBoltzmannQPolicy
from rl.memory import SequentialMemory
```

A.3 Construction Scenario Generator

```
def dailywork_train(y, num_ex, num_loader,num_trucks):

    Excavator_dict = {'machine_type':'Excavator',
    'Max_capacity': 300, 'Min_capacity': 30,
    'MaxChargepower': 150, 'Roundtrip': 0.95,
    'Power': 120, 'Priority':float(0),
    'Workload':float(0), 'alreadycharged':float(0),
    'Charge':float(0), 'charging':float(0),
    'StepWork':float(0), 'operator':Operatordict['experienced'],
    'bucketsize': 1.5, 'idle_power':5, 'cycletime': 0.01}

    loader_dict = {'machine_type':'Loader', 'Max_capacity': 150,
    'Min_capacity': 15,'MaxChargepower': 75, 'Roundtrip': 0.95,
    'Power': 50, 'Priority':float(0), 'Workload':float(0),
    'alreadycharged':float(0), 'Charge':float(0),
    'charging':float(0),'StepWork':float(0),
    'operator':Operatordict['experienced'], 'bucketsize': 2,
    'Max_cycle_time':0.05, 'idle_power':5, 'cycletime': 0.01}

    Truck_dict = {'machine_type':'Truck', 'Max_capacity': 180,
    'Min_capacity':18,'MaxChargepower': 75, 'Roundtrip': 0.95,
    'Power': 60,'Unloaded_power': 20, 'Priority':float(0),
    'Workload':float(0), 'alreadycharged':float(0),
    'Charge':float(0), 'charging':float(0), 'StepWork':float(0),
    'operator':Operatordict['experienced'], 'Max_load_capacity':23,
    'load_capacity':0, 'Max_cycle_time':0.5, 'idle_power':1,
    'cycletime': 0.3, 'remaining_cycle_time': 0}

    #Excavators
    for i in range(num_ex):
        Excavator_dict.setdefault('Machine', [])
        Excavator_dict['Machine'].append("excavator"+str(i))

        Excavator_dict.setdefault('SOE', [])
        if y == 1:
            Excavator_dict['SOE'].append(299)
        elif y == 2:
            Excavator_dict['SOE'].append(random.uniform(150,250))
        elif y == 3:
            Excavator_dict['SOE'].append(random.uniform(Excavator_dict
                ['Min_capacity'], 200))
        Excavator_dict.setdefault('Tasks', [])
        Excavator_dict['Tasks'].append(random.randint(300,310))

    excavators=pd.DataFrame(Excavator_dict)
    excavators=excavators.set_index('Machine')



    if num_loader > 0:
        #Loaders
        for i in range(num_loader):
            loader_dict.setdefault('Machine', [])
            loader_dict['Machine'].append("loader"+str(i))

            loader_dict.setdefault('SOE', [])
            if y == 1:
                loader_dict['SOE'].append(125)
            elif y == 2:
                loader_dict['SOE'].append(random.uniform(100, 150))
            elif y == 3:
                loader_dict['SOE'].append(random.uniform(loader_dict
                    ['Min_capacity'], 100))

            loader_dict.setdefault('Tasks', [])
            loader_dict['Tasks'].append(random.randint(150,170))
            #loader_dict['Tasks'].append(150)

        loaders=pd.DataFrame(loader_dict)
        loaders=loaders.set_index('Machine')
    else:
        loaders=pd.DataFrame()

    if num_trucks > 0:
        #Trucks
        for i in range(num_trucks):
            Truck_dict.setdefault('Machine', [])
            Truck_dict['Machine'].append("truck"+str(i))
            Truck_dict.setdefault('SOE', [])
            if y == 1:
                Truck_dict['SOE'].append(Truck_dict['Max_capacity'])
            elif y == 2:
                Truck_dict['SOE'].append(random.uniform(100,
                    Truck_dict['Max_capacity']))
            elif y == 3:
                Truck_dict['SOE'].append(random.uniform(
                    Truck_dict['Min_capacity'], 100))

            Truck_dict.setdefault('Tasks', [])
            Truck_dict['Tasks'].append(random.randint(180,200))
            #Truck_dict['Tasks'].append(100)

        trucks=pd.DataFrame(Truck_dict)
        trucks=trucks.set_index('Machine')
    else:
        trucks=pd.DataFrame()

    dataframes = [trucks,loaders,excavators]
    df = pd.concat(dataframes)
    df['Original_Tasks'] = df['Tasks']
    df= df.reset_index()
    return df



def set_conditions():

    Operatordict = {'newguy': 0.5, 'experienced': 0.7, 'expert': 1}
    Soil_amount = {'rock':200, 'clay': 200, 'sand':100}
    SoilDict = {'rock':0.6, 'clay': 0.8, 'sand':1}
    Soil_power_factor = {'rockhigh':1,'rocklow':0.8, 'clayhigh':
        0.6,'claylow': 0.8, 'sandhigh':0.4, 'sandlow':0.6}
    Soil_power_factor2 = {'rockhigh':0.8,'rocklow':0.5,
        'clayhigh': 0.2,'claylow': 0.5, 'sandhigh':0.2, 'sandlow':0.4}
    return Soil_amount, SoilDict, Soil_power_factor, Operatordict,
        Soil_power_factor2

Soil_amount, SoilDict, Soil_power_factor, Operatordict,
    Soil_power_factor2= set_conditions()


def digging_progress(soil_level, Soil_power_factor, Soil_amount):

    if soil_level <= 0 and soil_level > -Soil_amount['sand']:
        soilstate ='sand'
    elif soil_level <= -Soil_amount['sand'] and soil_level >
        -(Soil_amount['sand']+Soil_amount['clay']):
        soilstate ='clay'
    elif soil_level <= -(Soil_amount['clay']+Soil_amount['sand']):
        soilstate ='rock'
    else:
        soilstate ='rock'

    x = Soil_power_factor[soilstate+'high']
    y = Soil_power_factor[soilstate+'low']

    Current_soil_hardness = random.uniform(x,y)

    return Current_soil_hardness, soilstate
```

A.4 Charging Function

```
def charging_machines(self, df, start_time):
    total_discharge = 0

    for i in df.index:
        if df.at[i, 'SOE'] >= df.at[i, 'Max_capacity']*0.98:
                df.at[i, 'charging']=0
df.at[i, 'Charge'] =0
df.at[i, 'StepWork']=0
chargefromBESS=0

while self.soe > self.minCap:

    if sum(df.charging) == 0:
        break

    for i in df.index:

        if df.at[i,'charging']==1:

            remaining_charge_time= (df.at[i, 'Max_capacity']
                -df.at[i, 'SOE'])/ df.at[i, 'MaxChargepower']

            if self.chargetime < remaining_charge_time:
                chargefromBESS = float(df.at[i,
                    'MaxChargepower']*self.chargetime)
            else:
                chargefromBESS = float(df.at[i,
                'MaxChargepower']*remaining_charge_time)

            if self.soe - chargefromBESS < self.minCap:
                chargefromBESS= chargefromBESS -
                    (chargefromBESS- (self.soe-self.minCap))

            self.soe -= chargefromBESS
            total_discharge += chargefromBESS
            df.at[i, 'SOE'] += chargefromBESS *
                df.at[i, 'Roundtrip']
            df.at[i, 'alreadycharged'] += chargefromBESS
                * df.at[i, 'Roundtrip']
            df.at[i, 'Charge'] += chargefromBESS
                * df.at[i, 'Roundtrip']
            df = work_machine(self, df)



            for i in df.index:
                if df.at[i, 'SOE'] >= df.at[i, 'Max_capacity']*0.98:
                    df.at[i, 'charging']=0

                #Remaining_WL = check_previous_charges(self, df, i)

               # if df.at[i, 'SOE'] > Remaining_WL*1.1:
               #     df.at[i, 'charging']=0



            self.hour += self.chargetime
            if int(self.hour) >= start_time+1:
                break
            if self.soe<= self.minCap:
                break


        for i in df.index:
            df.at[i, 'charging']=0

        self.df=df

    return self.df, self.soe, total_discharge

def set_charge_to_zero(df):
    for i in df.index:
        df.at[i, 'Charge'] =0
        #df.at[i, 'StepWork']=0
    return df

def check_previous_charges(self, df, i):
    Time_spent = int(self.hour)-self.Starto
    Time_remaining = self.maxtimestep - Time_spent

    #if Time_remaining < Time_spent:
    if Time_spent > 2:

        Avg_WL = df.at[i, 'Workload'] /  Time_spent #average workload

        Remaining_WL = Time_remaining * Avg_WL
    else:
        Remaining_WL = 1000

    return Remaining_WL
```

A.5 Workload Functions

```
def work_machine(self, df):
    for i in df.index:
        if df.at[i,'charging']==0:
            if df.at[i, 'machine_type']=='Excavator':

                if self.soil_deponi >= self.max_deponi:
                    #and df[df.remaining_cycle_time == 0].empty
                        == True:
                    Workload = df.at[i,'idle_power'] * self.chargetime
                    time_spent, Workload = MinCap_cond(df, i, Workload)
                else:
                    Current_soil_hardness, soilstate =
                        digging_progress(self.soil_level,
                            Soil_power_factor2, Soil_amount)
                    OEE = random.uniform(0.65,0.83)
                    Workload = df.at[i,'Power'] * df.at[i,'Roundtrip']
                        * self.chargetime  * Current_soil_hardness
                            * OEE

                    time_spent, Workload = MinCap_cond(df, i, Workload)
                    Productivity = (df.at[i,'bucketsize']
                        * SoilDict[soilstate]) *
                        ((self.chargetime*time_spent)/
                            df.at[i,'cycletime']) * OEE
                                * df.at[i,'operator']

                    if self.soil_deponi < self.max_deponi:
                        self.soil_level -= Productivity
                        self.soil_deponi += Productivity

                    elif self.soil_deponi >= self.max_deponi:
                        df, self.soil_level, self.soil_deponi =
                            load_trucks(self, df, Productivity) ##

            elif df.at[i, 'machine_type']=='Loader':
                if self.soil_deponi <= 0:
                    Workload = df.at[i,'idle_power'] * self.chargetime
                    time_spent, Workload = MinCap_cond(df, i, Workload)
                #elif df[df.remaining_cycle_time == 0].empty == True:
                #    Workload = df.at[i,'idle_power'] * self.chargetime
                #    time_spent, Workload = MinCap_cond(df, i, Workload)
                else:
                    Current_soil_hardness, soilstate =
                        digging_progress(self.soil_level,
                            Soil_power_factor2 ,Soil_amount)
                                OEE = random.uniform(0.65,0.83)
                    Workload = df.at[i,'Power'] * df.at[i,'Roundtrip']
                        * self.chargetime  * Current_soil_hardness
                            * OEE

                    time_spent, Workload = MinCap_cond(df, i, Workload)

                    Productivity = (df.at[i,'bucketsize'] *
                        SoilDict[soilstate]) * ((self.chargetime
                            *time_spent) / df.at[i,'cycletime'])
                            * OEE * df.at[i,'operator']
                    #df, self.soil_level, self.soil_deponi =
                        load_trucks(self, df, Productivity)
                    self.soil_level -= Productivity
                    self.soil_deponi += Productivity

            elif df.at[i, 'machine_type']=='Truck':
                if df.at[i,'load_capacity'] < df.at[i,
                    'Max_load_capacity'] * 0.9:
                    Workload = df.at[i,'idle_power'] *
                        self.chargetime

                    time_spent, Workload = MinCap_cond(df,
                        i, Workload) ##

                elif df.at[i,'load_capacity'] >= df.at[i,
                    'Max_load_capacity'] * 0.9:

                    if df.at[i,'remaining_cycle_time'] == 0:
                        df.at[i, 'remaining_cycle_time'] =
                            df.at[i, 'cycletime'] #set full cycle time
                        Workload = df.at[i,'Power'] * df.at[i,
                            'Roundtrip'] * self.chargetime

                        time_spent, Workload = MinCap_cond(df,
                            i, Workload)##

                        df.at[i, 'remaining_cycle_time'] -=
                            self.chargetime * time_spent

                    elif df.at[i, 'remaining_cycle_time'] >
                        self.chargetime: Workload = df.at[i,
                            'Power'] * df.at[i,'Roundtrip']
                                * self.chargetime

                        time_spent, Workload = MinCap_cond(df,
                            i, Workload) ##
                        df.at[i, 'remaining_cycle_time'] -=
                            self.chargetime * time_spent

                    elif df.at[i, 'remaining_cycle_time']
                        <= self.chargetime:
                        Workload = df.at[i,'Power'] * df.at[i,
                            'Roundtrip']* self.chargetime

                        time_spent, Workload = MinCap_cond(df,
                            i, Workload) ##

                        df.at[i, 'remaining_cycle_time'] = 0
                            #self.chargetime * time_spent
                        self.dumped_soil += df.at[i,'load_capacity']
                        df.at[i,'load_capacity'] = 0
                        self.dump_counter +=1

            df.at[i, 'SOE']-= Workload
            #df.at[i, 'Tasks']-= workload
            df.at[i, 'Workload']+= Workload
            df.at[i, 'StepWork'] += Workload

    return df



def MinCap_cond(df, i, Workload):
    if df.at[i, 'SOE']-Workload < df.at[i, 'Min_capacity']:
        time_spent = (Workload -(Workload-
            (df.at[i, 'SOE']-df.at[i, 'Min_capacity'])))
                /Workload
        Workload= Workload -(Workload-(df.at[i, 'SOE']-
            df.at[i, 'Min_capacity']))
    else:
        time_spent =1
    return time_spent, Workload
```

```
def load_trucks (self, df, Productivity):
    if self.trucks == 1:
        Available_trucks = df[df.remaining_cycle_time == 0]
        Available_trucks = Available_trucks.sort_values(by=
            'load_capacity', ascending = False)

        if Available_trucks.empty == False and
            self.soil_deponi > Productivity:
                self.soil_deponi -= Productivity
                truck_name=Available_trucks.iloc[0,0]
                df.at[df[df.Machine == truck_name].index[0],
                    'load_capacity'] += Productivity

        elif Available_trucks.empty == False and
            self.soil_deponi <= Productivity:
            truck_name=Available_trucks.iloc[0,0]
            df.at[df[df.Machine == truck_name].index[0],
                'load_capacity'] += self.soil_deponi
            self.soil_deponi -= self.soil_deponi

        else:# Available_trucks.empty == True:
            if self.soil_deponi > Productivity and
                self.fill_volume < self.max_fill:
                self.soil_deponi -= Productivity/2
                self.fill_volume += Productivity/2
            else:
                self.soil_level -= Productivity/3
                self.soil_deponi += Productivity/3
    else:
        if self.soil_deponi > Productivity and
            self.fill_volume < self.max_fill:
            self.soil_deponi -= Productivity/2
            self.fill_volume += Productivity/2
        else:
            self.soil_level -= Productivity/3
            self.soil_deponi += Productivity/3

    return df, self.soil_level, self.soil_deponi
```

A.6 Machine Selection for Charging

```
def select_machines_for_charging(df,i):
    df.at[i,'charging']=1
    df=sort_priority(df)
    df=df.sort_values(by='Priority', ascending=False)
    test= df['charging']==0
    k=df[test].head(1).index
    df.at[k,'charging']=1
    return df

def can_machine_charge(df, i):
    if df.at[i, 'machine_type']=='Truck' and df.at[i,
        'remaining_cycle_time'] > 0:
        boolo = False
    else:
        test=df.at[i, 'SOE'] < df.at[i, 'Max_capacity']
            *0.98 #and df.at[i, 'Tasks'] > 0
        boolo= bool(test)
    return boolo
```

A.7 Action Dynamics

```
def set_charge_to_zero(df):
    for i in df.index:
        df.at[i, 'Charge'] =0
        #df.at[i, 'StepWork']=0
    return df

def action_reaction2(self, action):
```

```
Machine_actions = self.action_space.n-2
action2= Machine_actions- action

if action < Machine_actions/2 and self.soe >
    self.minCap and can_machine_charge(self.df,
        action)==True and self.discharge_count < 4:
    start_time = int(self.hour)
    self.df=select_machines_for_charging(self.df,
        action)
    self.df, self.soe, total_discharge =
        charging_machines(self, self.df,start_time)
    pre_reward = total_discharge * self.prices[start_time]
    total_charge = 0
    self.discharge_count +=1

elif action > Machine_actions/2 and action < Machine_actions+1 and
        self.soe > self.minCap and can_machine_charge(self.df,
        action2)==True and self.discharge_count < 4:
    start_time = int(self.hour)
    self.df.at[action2,'charging']=1
    self.df, self.soe, total_discharge =charging_machines(self,
    self.df,start_time)
    pre_reward = total_discharge * self.prices[start_time]
    total_charge = 0
    self.discharge_count +=1

elif action == Machine_actions+1 and self.soe < self.maxCap :
    total_charge=0
    start_time = int(self.hour)
    while self.soe < self.maxCap:
        charge = self.grid * self.chargetime
        self.soe += charge * self.RE
        total_charge += charge

        self.df=work_machine(self, self.df)
```

```
                self.hour += self.chargetime
                if self.hour >= start_time+1:
                    break

            #self.df=pd.concat([self.df, completed])
            pre_reward =-total_charge * self.prices[start_time]
            total_discharge=0
            self.df = set_charge_to_zero(self.df)

        else:
            start_time = int(self.hour)
            while int(self.hour) != start_time+1:
                self.df=work_machine(self, self.df)
                self.hour += self.chargetime
                if self.hour >= start_time+1:
                    break
            pre_reward = 0
            total_charge=0
            total_discharge=0
            self.df = set_charge_to_zero(self.df)


        return pre_reward, self.soe, self.df, total_discharge, total_charge
```

A.8 Get Machine States

```
def get_machine_states(df):
    df=df.sort_values(by='Machine', ascending=False)
    MachineStates={}
    for i in df.index:
        SOEname=df.at[i, "Machine"]+'SOE'
        MachineStates.setdefault(SOEname, [])
        MachineStates[SOEname].append(df.at[i, "SOE"])

        Tasksname=df.at[i, "Machine"]+'Workload'
        MachineStates.setdefault(Tasksname, [])
        MachineStates[Tasksname].append(df.at[i, "Workload"])

    data = list(MachineStates.values())

    return np.array(data).ravel()
```

A.9 Work Depenceny Example

```
def work_dependencie(self):
    if self.workDP == 1:
        pass
    else:
        if self.df.at[2, "Tasks"] <= 60:
            #self.df.at[2, "Original_Tasks"]/6:
            self.df.at[0, "Tasks"] += 20
                #self.df.at[0, "hourlydemand"]*2
            self.workDP = 1
        else:
            pass
    return self.df
```

A.10 Priority Function

```
def sort_priority(df):
    for i in df.index:
        SOC = 1-(df.at[i, "SOE"]/df.at[i, 'Max_capacity'])
#percentage SOC the higher it gest the less important
    before subtracting it from one

        if df.at[i, 'machine_type']=='Truck' and df.at[i,
            'remaining_cycle_time'] > 0 :
            df.at[i, "Priority"] = 0
        else:
            df.at[i, "Priority"] = float(SOC)*
                (df.at[i,'MaxChargepower']/
                    df.at[i, 'Max_capacity'])
    return df
```

A.11 DataLogger

```
def get_info_from_worksite(df):

    machinenamesSOE=[]
    machinenamesTasks=[]
    machinenamesWorkload=[]
    MachinenamesStepWork=[]
    MachinenamesCharge=[]
    MachinenamesTotalCharge=[]

    MachineSOE=[]
    MachineTasks=[]
    MachineWorkload=[]
    MachineStepWork=[]
    MachineCharge=[]
    MachineTotalCharge=[]

    for i in df.index:

        machinenamesSOE.append(df.at[i, "Machine"]+'SOE')
        machinenamesTasks.append(df.at[i, "Machine"]+'Tasks')
        machinenamesWorkload.append(df.at[i, "Machine"]+'Workload')
        MachinenamesStepWork.append(df.at[i, "Machine"]+'StepWork')
        MachinenamesCharge.append(df.at[i, "Machine"]+'Charge')
        MachinenamesTotalCharge.append(df.at[i,
            "Machine"]+'alreadycharged')

        MachineSOE.append(df.at[i, "SOE"])
        MachineTasks.append(df.at[i, "Tasks"])
        MachineWorkload.append(df.at[i, "Workload"])
        MachineStepWork.append(df.at[i, "StepWork"])
        MachineCharge.append(df.at[i, "Charge"])
        MachineTotalCharge.append(df.at[i, "alreadycharged"])

    info_worksite=dict(zip(machinenamesSOE, MachineSOE,))
    tasks= dict(zip(machinenamesTasks, MachineTasks))
    work=dict(zip(machinenamesWorkload,MachineWorkload))
    StepWork= dict(zip(MachinenamesStepWork, MachineStepWork))
    Charge=dict(zip(MachinenamesCharge,MachineCharge))
    TotalCharge=dict(zip(MachinenamesTotalCharge,MachineTotalCharge))

    info_worksite.update(tasks)
    info_worksite.update(work)
    #info_worksite.update(StepWork)
    #info_worksite.update(Charge)
    info_worksite.update(TotalCharge)

    return info_worksite
```

A.12 Environment Structure

```
class Batt(gym.Env):

    def __init__(self):

        self.day = []
        for i in range(6000, 7000, 24): #to start everyday at 6
            self.day.append(i)

        #prices
        df2 = pd.read_csv('elspot_2021.csv')
        self.prices = np.array(df2.price)
        self.hour_of_day = np.array(df2.hour_of_day)
        self.weekday = np.array(df2.Day_in_week)

        self.hour = 392#random_hour()
        self.intrahour = []
        self.Starto = self.hour
        self.maxtimestep = 9



        self.trucks = 0#1#0 #with trucks?

        #newnew
        self.soil_deponi = 0
        self.soil_level = 0
        self.dumped_soil  =0
        self.dump_counter = 0
        self.max_deponi = 10000
        self.fill_volume = 0
        self.max_fill = 10000

        #Power outputs
        self.grid = np.random.randint(30, 37)

        ##work dependency##
        self.workDP= 0 #work dependency concept

        #batterypara
        self.capacity =390
        self.maxCap = self.capacity *0.9
        self.minCap = self.capacity*0.2
        self.soe=230
        self.RE = 0.95
        self.charging_cables=2
        self.discharge_rate = 150

        #construction
        self.df = dailywork_train(2,2,0,0)

        #Timesteps
        self.chargetime = 0.1
        self.SOC = []
        self.logger =[]
        self.logger2=[]
        self.additional_actions = 2
        self.dischargecount =0

        #spaces
        self.action_space = spaces.Discrete(len(self.df.index)*2)+
            self.additional_actions)
        self.observation_space = spaces.Box(low=0, high=1,
            shape=(5+(len(self.df)*2), ), dtype=np.float32)

    def get_normalized_states(self):
        norm_time = self.hour_of_day[int(self.hour)]
        norm_weekday = self.weekday[int(self.hour)]
        norm_BC = self.soe
        prices = self.prices[int(self.hour)]
        Soil_lvl = self.soil_level
        #Stockpile = self.soil_deponi
        Dumped = self.dumped_soil
        overview = np.array([norm_time, Dumped, norm_weekday,
            norm_BC, prices])
        #overview = np.array([norm_BC])

        machineStates= get_machine_states(self.df)

        total_states = np.concatenate((overview, machineStates))

        return total_states

    def step(self, action):



        pre_reward, self.soe, self.df, total_discharge,
            total_charge = action_reaction2(self, action)



        self.logger2.append(pre_reward)

        #work_dependencie(self) #WD
        info = {'BC_SOE': self.soe,
                'price': self.prices[int(self.hour)-1],
                'cost': pre_reward,
                'time': self.hour,
                'Charge': total_charge,
                'Discharge': total_discharge,
                'action':action,
                'soil_deponi' : self.soil_deponi,
                'soil_level' : self.soil_level,
                'dumped_soil' : self.dumped_soil,
                'fill' : self.fill_volume,
                'dump_counter' : self.dump_counter
                 }

        info_worksite=get_info_from_worksite(self.df)
        info.update(info_worksite)

        self.logger.append(info)
        #print(action)

        done, reward=reward_system(self, pre_reward, 3)




        return(self.get_normalized_states(), reward, done, info)

    def reset(self):
        self.soe = np.random.randint((self.capacity*0.5),
            (self.capacity*0.6))
        self.hour = 392
        self.Starto=self.hour
        self.logger =[]
        self.logger2=[]
        self.df = dailywork_train(2,2,0,0)
        self.workDP= 0
        self.soil_deponi = 0
        self.soil_level = 0
        self.dumped_soil  =0
        self.dump_counter = 0
        self.fill_volume = 0
        self.dischargecount =0


        return self.get_normalized_states()

    def render(self):
        pass
```

A.13 Reward System

```
def reward_system(self, pre_reward, x):
    if x ==1:
        if int(self.hour)==self.Starto+self.maxtimestep:
            reward = sum(self.df.Workload)
            done=True
        else:
            reward = 0
            done =False



    elif x==2:
        if -self.soil_level >= 800:
            reward = 0
            done = True

        elif int(self.hour) >= self.Starto+10:
            done = True
            reward = -100

        elif int(self.hour) < self.Starto+24:
            reward = pre_reward
            done =False
    elif x ==3:

        if self.dumped_soil >= 300:
            reward = (sum(self.logger2)+100)*10
            done = True
        elif int(self.hour)==self.Starto+24:
            done = True
            reward = 0
        else:
            reward = pre_reward
            done =False
    return done, reward
```

A.14 Train Agent

```
env = Batt()
states =env.observation_space.shape[0]
actions = env.action_space.n

def build_model(states, actions):
    model=Sequential()
    model.add(Flatten(input_shape=(1, states)))
    model.add(Dense(24,activation='relu'))
    model.add(Dense(24,activation='relu'))
    model.add(Dense(actions,activation='linear'))
    return model

def build_agent(model, actions):
    policy = EpsGreedyQPolicy()#MaxBoltzmannQPolicy()

    memory = SequentialMemory(limit=50000, window_length=1)
    dqn = DQNAgent(model=model, memory=memory,
        policy=policy, nb_actions=actions,
        nb_steps_warmup=4000, target_model_update=1e-3)
    return dqn

model =build_model(states, actions)
dqn = build_agent(model,actions)
dqn.compile(Adam(learning_rate=1e-3), metrics=['mae'])
scores=dqn.fit(env, nb_steps=60000, visualize=False, verbose=1)
```

A.15 View Training Returns

```
fig, ax = plt.subplots(figsize=(20, 14))
plt.plot(scores.history['episode_reward'])
ax.set_ylabel('Return',fontsize=20)
ax.set_xlabel('Episode',fontsize=20)
ax.set_title('Results',fontsize=20)
ax.yaxis.set_tick_params(labelsize=15)
ax.xaxis.set_tick_params(labelsize=15)
fig.savefig(fname='RegularResults')
plt.show()
```

A.16 Generate Schedule

```
dqn.test(env, nb_episodes=1, visualize=False)
df = env.logger
df=pd.DataFrame(df)
#df.to_csv('arbitrage.csv')
for i in env.df.Machine:
    #df[i+'SOEb4'] = df[i+'SOE'].shift(1)
    df[i+'Chargo'] = df[i+'alreadycharged'].diff()
    df[i+'Worko'] = df[i+'Workload'].diff()
for i in env.df.Machine:
    df[i+'Chargo'][0] = df[i+'alreadycharged'][0]
    df[i+'Worko'][0] = df[i+'Workload'][0]
df3=df
df.time = df.time - df.time[0]
df.to_csv('arbitrage.csv')
```

```
Testing for 1 episodes ...
Episode 1: reward: 918.627, steps: 8
```

A.17 Plot Optimized Schedule

```
def plot_soil():
    xbaro=df.index#round(df.time,1)

    width = 0.5
    fig, ax = plt.subplots(figsize=(20, 14))



    p1 = ax.bar(xbaro, round(df.soil_deponi,1), width,
        bottom=round(df.dumped_soil,1),   label='soil_deponi'
            ,color='lightyellow',edgecolor = "black")

    p2 = ax.bar(xbaro, round(df.dumped_soil,1), width,
        label='dumped_soil', color='darkgreen',
            edgecolor = "black")

    p3 = ax.plot(xbaro, round(-df.soil_level,1),
        label ='soil_level', color='red')

    #ax2 = ax.twinx()
    #ax2.set_ylabel('soil_level', fontsize =17)
    #ax2.yaxis.set_tick_params(labelsize=15)
    #p3=ax2.plot(xbaro, round(-df.soil_level,1),
        label ='soil_level', color='red')




    ax.set_ylabel('kWh',fontsize=20)
    ax.set_title('soil',fontsize=20)
    ax.yaxis.set_tick_params(labelsize=15)
    ax.set_xticks(xbaro,labels=round(df.time,1),fontsize=15)
    #ax.legend(fontsize=13)
    ax.legend(fontsize=15, handles=p3+[p1, p2], loc='upper␣right')

    ax.bar_label(p1, label_type='center', fontsize=13)
    ax.bar_label(p2, label_type='center',fontsize=13)



    fig.savefig(fname='soil_graph')

    return plt.show()

def plot_bar(machine):
    xbaro=df.index#round(df.time,1)

    width = 0.5
    fig, ax = plt.subplots(figsize=(20, 14))



    p1 = ax.bar(xbaro+0.1, round(df[machine+'SOE'],1),
        width/2,  label=machine+'␣Remaining␣Capacity',
            color='lightyellow',edgecolor = "black")

    p2 = ax.bar(xbaro+0.1, round(df[machine+'Worko'],1),
        width/2, bottom=round(df[machine+'SOE'],1),
            label=machine+'␣Workload',color='tomato',
                edgecolor = "black")

    p3 = ax.bar(xbaro-0.1, round(df[machine+'Chargo'],1),
        width/2, bottom=round(df[machine+'SOE']+
            df[machine+'Worko']-df[machine+'Chargo'],1),
            label=machine+'␣Charge', color='darkgreen',
                edgecolor = "black")

    p4 = ax.bar(xbaro-0.1, round(df[machine+'SOE']+
        df[machine+'Worko']-df[machine+'Chargo'],1),
            width/2,color='lightyellow',edgecolor = "black")

    ax2 = ax.twinx()
    ax2.set_ylabel('Workload', fontsize =17)
    ax2.yaxis.set_tick_params(labelsize=15)
    p5=ax2.plot(xbaro, df[machine+'Workload'], label ='Workload',
        color='red')



    ax.set_ylabel('kWh',fontsize=20)
    ax.set_title(machine,fontsize=20)
    ax.yaxis.set_tick_params(labelsize=15)
    ax.set_xticks(xbaro,labels=round(df.time,1),fontsize=15)
    #ax.legend(fontsize=13)
    ax2.legend(fontsize=15, handles=p5+[p1, p2, p3], loc='upper␣right')

    ax.bar_label(p1, label_type='center', fontsize=13)
    ax.bar_label(p2, label_type='center',fontsize=13)
    ax.bar_label(p3, label_type='center',fontsize=13)
    ax.bar_label(p4, label_type='center',fontsize=13)

    fig.savefig(fname=machine+'MaxWerk')

    return plt.show()

def BESSplot():
    width = 0.5

    fig, ax = plt.subplots(figsize=(20, 14))

    p1 = ax.bar(df.index, round(df['BC_SOE']-df['Charge'],1),
        width,  label='BESS␣Remaining␣Capacity',
            color='lightyellow',edgecolor = "black")
    p2 = ax.bar(df.index, round(df['Charge'],1),
        width, bottom=(df['BC_SOE']-df['Charge']),
            label='BESS␣Charge', color='darkgreen',
                edgecolor = "black", ls='dashed' )
    p3 = ax.bar(df.index, round(df['Discharge'],1),
        width, bottom=round(df['BC_SOE'],1),
            label='BESS␣Discharge',color='tomato',
                edgecolor = "black")
    #p5 = ax.plot(df.time, df['BC_SOE'],
        label ='SOC', color='green')

    ax2 = ax.twinx()
    ax2.set_ylabel('Electricity␣Price', fontsize =17)
    ax2.yaxis.set_tick_params(labelsize=15)
    p4=ax2.plot(df.index, df['price'],
        label ='Electricity␣Price', color='blue')

    ax.set_ylabel('kWh',fontsize=20)
    ax.set_title('BESS',fontsize=20)
    ax.yaxis.set_tick_params(labelsize=15)
    ax.set_xticks(df.index,labels=round(df.time,1)
        ,fontsize=15)
    ax2.legend(fontsize=15, handles=p4+[p1, p2, p3],
        loc='upper␣right')

    ax.bar_label(p1, label_type='center', fontsize=13)
    ax.bar_label(p2, label_type='center',fontsize=13)
    ax.bar_label(p3, label_type='center',fontsize=13)

    fig.savefig(fname='MaxWerkBESS20')




    return plt.show()
```

A.18 Estimate BESS size

```
def estimateBESS():
    global score
    global capacity
    capacity=[]
    score=[]
    Episodes4test = 5
    Factor = 5
    Reward_test= 9000
    env.capacity = 10
    scores=dqn.test(env, nb_episodes=Episodes4test,
        visualize=False)
    AMR = np.mean(scores.history['episode_reward'])

    if AMR < Reward_test:
        while AMR < Reward_test:
            env.capacity += Factor
            scores=dqn.test(env, nb_episodes=Episodes4test,
                visualize=False)
            AMR = np.mean(scores.history['episode_reward'])
            score.append(AMR)
            capacity.append(env.capacity)
    else:
        while AMR > Reward_test:
            env.capacity -= Factor
            scores=dqn.test(env, nb_episodes=Episodes4test,
                visualize=False)
            AMR = np.mean(scores.history['episode_reward'])
            score.append(AMR)
            capacity.append(env.capacity)

        env.capacity += Factor

    return env.capacity
```

A.19 Matlab Environment Code Example

```
class BatteryEnvMatlab(gym.Env):

    def __init__(self):
```

```
self.action_space = spaces.Discrete(3)
    #charge, discharge, and wait



df = pd.read_csv('2019_dt_full.csv')

#a variable with the first hour of each day
    for the environment (used in the reset function)
self.day = []
for i in range(0, 8760-24, 24):
    self.day.append(i)
######

self.hour = 0 #first hour of the year

self.daynum = np.array(df.Daynum)
    #The number of the day of the year we are on
self.prices = np.array(df.price)
    #Electricity prices for each hour

#battery parameters
#self.MAX_charge = 400 #maximum CHARGE
    over a period of one hour.
#self.MAX_discharge =400 #maximum DISCHARGE
    over a period of one hour.
self.Capacity =100  #MAXIMUM CAPACITY,
    battery cannot hold more charge than this in Kwh.

#observation_space variables
self.hour_of_day = np.array(df.hour_of_day)
self.Day_in_month = np.array(df.Day_in_month)
self.month = np.array(df.Month)

#the observation space
self.state = eng.set_param('power_battery/Battery','SOC',
    str(self.Capacity*0.5 ), nargout=0) #State of charge
    in battery, starts with random value.
```

```
        #self.observation_space = get_normalized_states(self)
        self.observation_space = spaces.Box(low=0, high=1, shape=(5, ),
        dtype=np.float32)
        ######
        self.logger=[]


    def get_normalized_states(self):
        norm_time = self.hour_of_day[self.hour]/23
        norm_day = self.Day_in_month[self.hour]/31
        norm_month = self.month[self.hour]/12
        norm_batt = self.state/self.Capacity
        norm_price = self.prices[self.hour]

        return np.array([norm_time, norm_day, norm_month,
            norm_batt, norm_price])
    ######


    def charge_matlab(eng):
        eng.set_param('power_battery/Constant1','Value','0',nargout=0)
            #set to charge
        eng.set_param('power_battery','SimulationCommand','Start',
            nargout=0) #start simulation

        while eng.get_param('power_battery','SimulationStatus')
            =='running': #wait for simulation to complete
            pass
        else:
            pass

        eng.eval('a␣=␣SOE;␣b=␣a.Data(end,:);␣c=␣a.Data(1,:)␣',nargout=0)
            #save initial state of charge and last state of charge
        Last = eng.workspace['b']
        First = eng.workspace['c']
        Charge = Last - First

        return Charge, Last # return what was charged and
            new state of charge.

    def discharge_matlab(eng):
        eng.set_param('power_battery/Constant1','Value','1',nargout=0)
            #set to charge

        eng.set_param('power_battery','SimulationCommand','Start',
            nargout=0) #start simulation

        while eng.get_param('power_battery','SimulationStatus')
            =='running': #wait for simulation to complete
            pass
        else:
            pass

        eng.eval('a␣=␣SOE;␣b=␣a.Data(end,:);␣c=␣a.Data(1,:)␣',nargout=0)
            #save initial state of charge and last state of charge
        Last = eng.workspace['b']
        First = eng.workspace['c']
        Charge = Last - First

        return Charge, Last # return what was charged and
            new state of charge.



    def step(self, action):

        SOC=eng.get_param('power_battery/Battery','SOC', nargout=1)
        SOC=np.float(SOC)

        #Charge
        if action == 0 and SOC < self.Capacity and SOC < 80:
```

```
        charge = charge_matlab(eng)
        charged = charge[0]
        self.state += charged
        pre_reward = -charged * self.prices[self.hour]
            #what we paid for charging
        eng.set_param('power_battery/Battery','SOC', str(charge[1]),

    #Discharge
    elif action == 1 and SOC > 0 and SOC >= 80:
        #if the battery is not empty we are allowed to discharge
        discharge = discharge_matlab(eng)
        charged = discharge[0]
        self.state -= np.abs(charged)
        pre_reward = np.abs(charged) * self.prices[self.hour]
            #what we paid for charging
        eng.set_param('power_battery/Battery','SOC',
            str(discharge[1]), nargout=0)

    #Do Nothing/Hold/Wait
    else:
        charged = 0
        self.state += 0  #we add nothing to the battery
        pre_reward = 0   #do nothing cost nothing and makes nothing.

    reward = pre_reward /(max(self.prices) *self.Capacity)
        #reward normalized

    self.hour+=1  #go to next hour of the day
        (because prices and electricity demand change)



    done = False
    if self.daynum[self.hour] != self.daynum[self.hour-1]:
        #checking if gone through all 24-hours of the day
        done = True

    info = { #not important
```

```
            'SOE': self.state,
            'day': self.daynum[self.hour],
            'charged': charged,
            'hour': self.hour,
            'price': self.prices[self.hour],
            'reward': pre_reward
                }


        self.logger.append(info)




        return self.get_normalized_states(), reward, done, info


    def reset(self):
        self.state = self.Capacity*0.7 #set battery to a random
            state of charge for next episode.
        self.hour = random.choice(self.day)  # randomize to the
            first hour of a different day (for exploration)


        return self.get_normalized_states()

    def render(self):
        pass
```

Bibliography

[1] Energi Norge, Enova, Klimaetaten Oslo Kommune, DNV: Veileder for tilrettelegging av fossilfrie og utslippsfrie løsninger på byggeplassen. (2018)

[2] Huang, Lizhen ; Krigsvoll, Guri ; Johansen, Fred ; Liu, Yongping ; Zhang, Xiaoling: Carbon emission of global construction sector. In: *Renewable and Sustainable Energy Reviews* 81 (2018), S. 1906–1916

[3] Wiik, Marianne Rose K. ; Suul, Jon Are W. ; Sundseth, Kyrre ; Ødegård, Anders ; Mellegård, Sofie E. ; Azrague, Kamal ; Haukaas, Nils-Olav ; Ibsen, Jan I. ; Lekanger, Randi ; Ianssen, Christina: 30 tonns utslippsfri gravemaskin. Teknologistatus, kartlegging og erfaringer. In: *SINTEF Fag* (2018)

[4] Bellona Europe: ZERO EMISSION CONSTRUCTION SITES. (2019)

[5] DNV: PERSPECTIVES ON Zero Emission Construction. (2019)

[6] Roy, Adrien: *Generalizable approaches for tracking, estimating, optimizing, and quantifying uncertainty of fuel use in earthworks operations*, University of Toronto (Canada), Diss., 2020

[7] Liu, Yuanyuan ; Wang, Yuanqing ; Li, Di: Estimation and uncertainty analysis on carbon dioxide emissions from construction phase of real highway projects in China. In: *Journal of cleaner production* 144 (2017), S. 337–346

[8] Saxe, Shoshanna ; Guven, Gursans ; Pereira, Lucas ; Arrigoni, Alessandro ; Opher, Tamar ; Roy, Adrien ; Arceo, Aldrick ; Von Raesfeld, Sofia S. ; Duhamel, Mel ; McCabe, Brenda u. a.: Taxonomy of uncertainty in environmental life cycle assessment of infrastructure projects. In: *Environmental Research Letters* 15 (2020), Nr. 8, S. 083003

[9] Pon Equipment AS: Z-line, an environmental investment. (2022). https://www.pon-cat.com/en-no/pon-equipment/About_us/miljo-og-sikkerhet/z-line

[10] Nordic Booster AS: BoostCharger: Battery - Fast Charging - Mobile. (2022). https://www.nordicbooster.com/products/boostcharger

[11] TEKNISK UKEBLAD: Batteridrevet ladestasjon gjør det mulig å hurtiglade anleggsmaskiner. (2022). https://www.tu.no/artikler/batteridrevet-ladestasjon-gjor-det-mulig-a-hurtiglade-anleggsmaskiner/515213?key=ieXGtpKC

[12] UNITED NATIONS: *The 2030 Agenda for Sustainable Development*. https://sdgs.un.org/2030agendaG, 2015

[13] LEWIS, Michael P.: *Estimating fuel use and emission rates of nonroad diesel construction equipment performing representative duty cycles*. North Carolina State University, 2009

[14] SANDANAYAKE, Malindu ; ZHANG, Guomin ; SETUNGE, Sujeeva: Environmental emissions at foundation construction stage of buildings–Two case studies. In: *Building and Environment* 95 (2016), S. 189–198

[15] ORGANIZATION, World H. u. a.: Ambient air pollution: A global assessment of exposure and burden of disease. (2016)

[16] MARSHALL, David: Collaborative Metropolitan Governance for Social Inclusion: Learning from Experience in Canada. In: *Inclusion, collaboration and urban governance: Challenges in Metropolitan Regions of Brazil and Canada* (2010), S. 251–261

[17] LIPPMANN, Morton: Environmental toxicants: human exposures and their health effects. (2000)

[18] LØKEN, Ingrid ; DNV: Perspectives on zero emission construction. (2019). https://www.klimaoslo.no/wp-content/uploads/sites/88/2019/06/Perspectives-on-zero-emission-construction.pdf

[19] QADRDAN, Meysam ; JENKINS, Nick ; WU, Jianzhong: Smart grid and energy storage. In: *McEvoy's Handbook of Photovoltaics*. Elsevier, 2018, S. 915–928

[20] MANTHIRAM, Arumugam: Materials challenges and opportunities of lithium ion batteries. In: *The Journal of Physical Chemistry Letters* 2 (2011), Nr. 3, S. 176–184

[21] HESSE, Holger C. ; SCHIMPE, Michael ; KUCEVIC, Daniel ; JOSSEN, Andreas: Lithium-ion battery storage for the grid—A review of stationary battery storage system design tailored for applications in modern power grids. In: *Energies* 10 (2017), Nr. 12, S. 2107

[22] LIU, Lollo: *Life Cycle Assessment of a Lithium-Ion Battery pack for Energy storage Systems:-the environmental impact of a grid-connected battery energy storage system*. 2020

[23] MERRILL, Laura C. ; ROSENBERG, Samantha G. ; JUNGJOHANN, Katherine L. ; HARRISON, Katharine L.: Uncovering the Relationship between Aging and Cycling on Lithium Metal Battery Self-Discharge. In: *ACS Applied Energy Materials* 4 (2021), Nr. 8, S. 7589–7598

[24] LIU, Xuan ; LI, Kang ; LI, Xiang: The electrochemical performance and applications of several popular lithium-ion batteries for electric vehicles-a review. In: *Advances in Green Energy Systems and Smart Grid* (2018), S. 201–213

[25] BOWEN, Thomas ; CHERNYAKHOVSKIY, Ilya ; DENHOLM, Paul L.: Grid-Scale Battery Storage: Frequently Asked Questions. (2019)

[26] IRENA: Innovation landscape brief: Utility-scale batteries. (2019)

[27] GELLINGS, Clark W.: The concept of demand-side management for electric utilities. In: *Proceedings of the IEEE* 73 (1985), Nr. 10, S. 1468–1470

[28] TRANI, Marco L. ; BOSSI, Benedetta ; GANGOLELLS, Marta ; CASALS, Miquel: Predicting fuel energy consumption during earthworks. In: *Journal of cleaner production* 112 (2016), S. 3798–3809

[29] LEWIS, Phil ; RASDORF, William: Fuel use and pollutant emissions taxonomy for heavy duty diesel construction equipment. In: *Journal of Management in Engineering* 33 (2017), Nr. 2, S. 04016038

[30] CAO, Tanfeng ; DURBIN, Thomas D. ; RUSSELL, Robert L. ; COCKER III, David R. ; SCORA, George ; MALDONADO, Hector ; JOHNSON, Kent C.: Evaluations of in-use emission factors from off-road construction equipment. In: *Atmospheric environment* 147 (2016), S. 234–245

[31] ANN ARBOR, M ; ARLINGTON, V: User's Guide for the Final NONROAD2005 Model. In: *Environmental Protection Agency: Washington, DC, USA* (2005)

[32] AGGA: NATIONAL GREENHOUSE ACCOUNTS FACTORS. In: *Department of Industry, Science, Energy and Resources* (2021)

[33] TROZZI, Carlo ; LAURETIS, RD: EMEP/EEA air pollutant emission inventory guidebook 2016. In: *European Environment Agency* (2016)

[34] HANDBOOK, Caterpillar P.: Caterpillar. In: *Inc., Peoria, IL* (2017)

[35] KUMATSU: *Fuel Saving Operation Guide.* 2013

[36] JASSIM, Hassanean S. ; LU, Weizhuo ; OLOFSSON, Thomas: Predicting energy consumption and CO2 emissions of excavators in earthwork operations: an artificial neural network model. In: *Sustainability* 9 (2017), Nr. 7, S. 1257

[37] FREY, H C. ; RASDORF, William ; LEWIS, Phil: Comprehensive field study of fuel use and emissions of nonroad diesel construction equipment. In: *Transportation Research Record* 2158 (2010), Nr. 1, S. 69–76

[38] Ahn, Changbum ; Martinez, Julio C. ; Rekapalli, Prasant V. ; Peña-Mora, Feniosky A.: Sustainability analysis of earthmoving operations. In: *Proceedings of the 2009 Winter Simulation Conference (WSC)* IEEE, 2009, S. 2605–2611

[39] Atibila, FA: Effect of Scheduled Inspection and Preventive Maintenance on Exhaust Emissions and Fuel Economy. In: *University of Science and Technology Kumasi* (2011)

[40] Kobbacy, Khairy A. ; Murthy, DN P. u. a.: *Complex system maintenance handbook*. Springer, 2008

[41] Hughes, Khaliah ; Jiang, Xiaochun: Using discrete event simulation to model excavator operator performance. In: *Human Factors and Ergonomics in Manufacturing & Service Industries* 20 (2010), Nr. 5, S. 408–423

[42] Holt, Gary D. ; Edwards, David: Analysis of interrelationships among excavator productivity modifying factors. In: *International Journal of Productivity and Performance Management* (2015)

[43] Frank, Bobbie ; Skogh, Lennart ; Alaküla, Mats: On wheel loader fuel efficiency difference due to operator behaviour distribution. In: *2nd International Commercial Vehicle Technology Symposium, CVT*, 2012, S. 1–18

[44] Carmichael, David G. ; Mustaffa, Nur K.: Emissions and production penalties/bonuses associated with non-standard earthmoving loading policies. In: *Construction Innovation* (2018)

[45] Abbasian-Hosseini, S A. ; Liu, Min ; Leming, Michael: Comparison of least-cost and least-pollution equipment fleet configurations using computer simulation. In: *Journal of Management in Engineering* 31 (2015), Nr. 6, S. 04015003

[46] Burt, Christina N. ; Caccetta, Louis: Match factor for heterogeneous truck and loader fleets. In: *International journal of mining, reclamation and environment* 21 (2007), Nr. 4, S. 262–270

[47] Tang, Pei ; Cass, Darrell ; Mukherjee, Amlan: Investigating the effect of construction management strategies on project greenhouse gas emissions using interactive simulation. In: *Journal of Cleaner Production* 54 (2013), S. 78–88

[48] Kaboli, Alireza S. ; Carmichael, David G.: Truck dispatching and minimum emissions earthmoving. In: *Smart and Sustainable Built Environment* (2014)

[49] Carmichael, David G. ; Williams, Evan H. ; Kaboli, Alireza S.: Minimum operational emissions in earthmoving. In: *Construction Research Congress 2012: Construction Challenges in a Flat World*, 2012, S. 1869–1878

[50] Carmichael, David G. ; Shen, Xuesong ; Peansupap, Vachara: The relationship between heavy equipment cost efficiency and cleaner production in construction. In: *Journal of Cleaner Production* 211 (2019), S. 521–529

[51] Lewis, Phil ; Leming, Michael ; Frey, H C. ; Rasdorf, William: Assessing effects of operational efficiency on pollutant emissions of nonroad diesel construction equipment. In: *Transportation research record* 2233 (2011), Nr. 1, S. 11–18

[52] Oberlender, Garold D. ; Peurifoy, RL: Estimating Construction Costs. In: *Mc Graw–Hill Book Company, New York* (2002)

[53] Ang-Olson, Jeffrey ; Schroeer, Will: Energy efficiency strategies for freight trucking: potential impact on fuel use and greenhouse gas emissions. In: *Transportation Research Record* 1815 (2002), Nr. 1, S. 11–18

[54] Forsythe, Perry ; Ding, Grace: Greenhouse gas emissions from excavation on residential construction sites. In: *Australasian Journal of Construction Economics and Building, The* 14 (2014), Nr. 4, S. 1–10

[55] Li, Hong-xian ; Lei, Zhen: Implementation of Discrete-Event Simulation (DES) in estimating & analyzing CO 2 emission during earthwork of building construction engineering. In: *2010 IEEE 17th International Conference on Industrial Engineering and Engineering Management* IEEE, 2010, S. 87–89

[56] Ding, Yonglian: *Quantifying the impact of traffic-related and driver-related factors on vehicle fuel consumption and emissions*, Virginia Tech, Diss., 2000

[57] Devi, L P. ; Palaniappan, Sivakumar: A study on energy use for excavation and transport of soil during building construction. In: *Journal of cleaner production* 164 (2017), S. 543–556

[58] Hong, Taehoon ; Ji, ChangYoon ; Jang, MinHo ; Park, HyoSeon: Assessment model for energy consumption and greenhouse gas emissions during building construction. In: *Journal of Management in Engineering* 30 (2014), Nr. 2, S. 226–235

[59] Hajji, Apif: The use of construction equipment productivity rate model for estimating fuel use and carbon dioxide (CO2) emissionsCase study: bulldozer, excavator and dump truck. In: *International Journal of Sustainable Engineering* 8 (2015), Nr. 2, S. 111–121

[60] Hajji, Apif M. ; Lewis, Phil: Development of productivity-based estimating tool for energy and air emissions from earthwork construction activities. In: *Smart and Sustainable Built Environment* (2013)

[61] Parsakhoo, A ; Hosseini, S A. ; Jalilvand, Hamid ; Lotfalian, Majid: Physical soil properties and slope treatments effects on hydraulic excavator

productivity for forest road construction. In: *Pakistan Journal of Biological Sciences* 11 (2008), Nr. 11, S. 1422–1428

[62] Pan, Nang-Fei: Assessment of productivity and duration of highway construction activities subject to impact of rain. In: *Expert systems with Applications* 28 (2005), Nr. 2, S. 313–326

[63] Smith, Gary R. ; Hancher, Donn E.: Estimating precipitation impacts for scheduling. In: *Journal of construction engineering and management* 115 (1989), Nr. 4, S. 552–566

[64] Apipattanavis, Somkiat ; Sabol, Kevin ; Molenaar, Keith R. ; Rajagopalan, Balaji ; Xi, Yunping ; Blackard, Ben ; Patil, Shekhar: Integrated framework for quantifying and predicting weather-related highway construction delays. In: *Journal of construction engineering and management* 136 (2010), Nr. 11, S. 1160–1168

[65] El-Rayes, Khaled ; Moselhi, Osama: Impact of rainfall on the productivity of highway construction. In: *Journal of construction engineering and management* 127 (2001), Nr. 2, S. 125–131

[66] Davies, Philip J. ; Emmitt, Stephen ; Firth, Steven K.: On-site energy management challenges and opportunities: a contractor's perspective. In: *Building Research & Information* 41 (2013), Nr. 4, S. 450–468

[67] Bakhiet, Omnia: *Data collection for management of fuel consumption in vehicles and machinery: A study on the challenges and strategic possibilities in the construction industry.* 2017

[68] Ahn, Changbum R. ; Lewis, Phil ; Golparvar-Fard, Mani ; Lee, SangHyun: Integrated framework for estimating, benchmarking, and monitoring pollutant emissions of construction operations. In: *Journal of Construction Engineering and Management* 139 (2013), Nr. 12, S. A4013003

[69] Abolhasani, Saeed ; Frey, H C. ; Kim, Kangwook ; Rasdorf, William ; Lewis, Phil ; Pang, Shih-hao: Real-world in-use activity, fuel use, and emissions for nonroad construction vehicles: a case study for excavators. In: *Journal of the Air & Waste Management Association* 58 (2008), Nr. 8, S. 1033–1046

[70] Frey, H C. ; Kim, Kangwook ; Pang, Shih-Hao ; Rasdorf, William J. ; Lewis, Phil: Characterization of real-world activity, fuel use, and emissions for selected motor graders fueled with petroleum diesel and B20 biodiesel. In: *Journal of the Air & Waste Management Association* 58 (2008), Nr. 10, S. 1274–1287

[71] Barati, K ; Shen, X: Emissions modelling of earthmoving equipment. In: *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction* Bd. 33 IAARC Publications, 2016, S. 1

[72] Heydarian, Arsalan ; Golparvar-Fard, Mani ; Niebles, Juan C.: Automated visual recognition of construction equipment actions using spatio-temporal features and multiple binary support vector machines. In: *Construction research congress 2012: Construction challenges in a flat world*, 2012, S. 889–898

[73] Ahn, Changbum R. ; Lee, SangHyun ; Peña-Mora, Feniosky: Monitoring system for operational efficiency and environmental performance of construction operations using vibration signal analysis. In: *Construction Research Congress 2012: Construction Challenges in a Flat World*, 2012, S. 1879–1888

[74] Yang, Jun ; Park, Man-Woo ; Vela, Patricio A. ; Golparvar-Fard, Mani: Construction performance monitoring via still images, time-lapse photos, and video streams: Now, tomorrow, and the future. In: *Advanced Engineering Informatics* 29 (2015), Nr. 2, S. 211–224

[75] Hajji, Apif M. ; Lewis, Michael P.: How to estimate green house gas (GHG) emissions from an excavator by using CAT's performance chart. In: *AIP Conference Proceedings* Bd. 1887 AIP Publishing LLC, 2017, S. 020047

[76] Heidari, Bardia ; Marr, Linsey C.: Real-time emissions from construction equipment compared with model predictions. In: *Journal of the air & waste management association* 65 (2015), Nr. 2, S. 115–125

[77] Jassim, Hassanean S. ; Lu, Weizhuo ; Olofsson, Thomas: Quantification of energy consumption and carbon dioxide emissions during excavator operations. In: *Workshop of the European Group for Intelligent Computing in Engineering* Springer, 2018, S. 431–453

[78] Martinez, Julio C.: Methodology for conducting discrete-event simulation studies in construction engineering and management. In: *Journal of Construction Engineering and Management* 136 (2010), Nr. 1, S. 3–16

[79] Shannon, Robert E.: Introduction to the art and science of simulation. In: *1998 winter simulation conference. proceedings (cat. no. 98ch36274)* Bd. 1 IEEE, 1998, S. 7–14

[80] Pan, Wenjia u. a.: *The application of simulation methodologies on estimating gas emissions from construction equipment*. University of Alberta, 2011

[81] Zhang, Hong: Discrete-event simulation for estimating emissions from construction processes. In: *Journal of Management in Engineering* 31 (2015), Nr. 2, S. 04014034

[82] Golzarpoor, Hamed ; González, Vicente ; Shahbazpour, Mehdi ; O'Sullivan, Michael: An input-output simulation model for assessing production and environmental waste in construction. In: *Journal of cleaner production* 143 (2017), S. 1094–1104

[83] Golzarpoor, Hamed ; González, Vicente A. ; O'Sullivan, Michael ; Shahbazpour, Mehdi ; Walker, Cameron G. ; Poshdar, Mani: A non-queue-based paradigm in Discrete-Event-Simulation modelling for construction operations. In: *Simulation Modelling Practice and Theory* 77 (2017), S. 49–67

[84] Carmichael, David G. ; Bartlett, Beau J. ; Kaboli, Alireza S.: Surface mining operations: coincident unit cost and emissions. In: *International Journal of Mining, Reclamation and Environment* 28 (2014), Nr. 1, S. 47–65

[85] Kaboli, Alireza S. ; Carmichael, David G.: Optimum scraper load time and fleet size for minimum emissions. In: *International Journal of Construction Management* 14 (2014), Nr. 4, S. 209–226

[86] Zhang, Hong ; Zhai, Dong ; Yang, Ying N.: Simulation-based estimation of environmental pollutions from construction processes. In: *Journal of cleaner production* 76 (2014), S. 85–94

[87] Ahn, Changbum ; Pan, Wenjia ; Lee, SangHyun ; Peña-Mora, Feniosky A.: Lessons learned from utilizing discrete-event simulation modeling for quantifying construction emissions in pre-planning phase. In: *Proceedings of the 2010 Winter Simulation Conference* IEEE, 2010, S. 3170–3176

[88] Riley, Linda A.: Discrete-event simulation optimization: a review of past approaches and propositions for future direction. In: *SummerSim* (2013), S. 47

[89] Nam, Chulu ; Lee, Dongyoun ; Kang, Goune ; Cho, Hunhee ; Kang, Kyung-In: Activity based prediction of on-site CO2 emissions containing uncertainty. In: *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction* Bd. 34 IAARC Publications, 2017, S. 1–8

[90] Shitole, Vivswan ; Louis, Joseph ; Tadepalli, Prasad: Optimizing earth moving operations via reinforcement learning. In: *2019 Winter Simulation Conference (WSC)* IEEE, 2019, S. 2954–2965

[91] Zhang, Wei ; Wang, Jixin ; Liu, Yong ; Gao, Guangzong ; Liang, Siwen ; Ma, Hongfeng: Reinforcement learning-based intelligent energy management architecture for hybrid construction machinery. In: *Applied Energy* 275 (2020), S. 115401

[92] Sutton, Richard S. ; Barto, Andrew G.: *Reinforcement learning: An introduction.* MIT press, 2018

[93] Heaton, Jeff: *Introduction to neural networks with Java.* Heaton Research, Inc., 2008

[94] Dulac-Arnold, Gabriel ; Mankowitz, Daniel ; Hester, Todd: Challenges of real-world reinforcement learning. In: *arXiv preprint arXiv:1904.12901* (2019)

[95] Soman, Ranjith K. ; Molina-Solana, Miguel: Automating look-ahead schedule generation for construction using linked-data based constraint checking and reinforcement learning. In: *Automation in Construction* 134 (2022), S. 104069

[96] Wang, Jixin ; Yang, Zhiyu ; Liu, Shaokang ; Zhang, Qingyang ; Han, Yunwu: A comprehensive overview of hybrid construction machinery. In: *Advances in Mechanical Engineering* 8 (2016), Nr. 3, S. 1687814016636809

[97] Chang, Fangyuan ; Chen, Tao ; Su, Wencong ; Alsafasfeh, Qais: Control of battery charging based on reinforcement learning and long short-term memory networks. In: *Computers and Electrical Engineering* 85 (2020), S. 106670

[98] Kwon, Kyung-bin ; Zhu, Hao: Reinforcement Learning Based Optimal Battery Control Under Cycle-based Degradation Cost. In: *arXiv preprint arXiv:2108.02374* (2021)

[99] Sui, Yu ; Song, Shiming: A multi-agent reinforcement learning framework for lithium-ion battery scheduling problems. In: *Energies* 13 (2020), Nr. 8, S. 1982

[100] Li, Jianwei ; Wang, Hanxiao ; He, Hongwen ; Wei, Zhongbao ; Yang, Qingqing ; Igic, Petar: Battery optimal sizing under a synergistic framework with DQN based power managements for the fuel cell hybrid powertrain. In: *IEEE Transactions on Transportation Electrification* (2021)

[101] Wang, Shuoyao ; Bi, Suzhi ; Zhang, Yingjun A.: Reinforcement learning for real-time pricing and scheduling control in EV charging stations. In: *IEEE Transactions on Industrial Informatics* 17 (2019), Nr. 2, S. 849–859

[102] Kaggle: *Kaggle Machine Learning and Data Science Survey.* https://www.kaggle.com/c/kaggle-survey-2019, 2019

[103] IRENA: Time-of-use tariffs (BRIEF, INNOVATION LANDSCAPE). (2019)

[104] Rosewater, David M. ; Copp, David A. ; Nguyen, Tu A. ; Byrne, Raymond H. ; Santoso, Surya: Battery energy storage models for optimal control. In: *IEEE Access* 7 (2019), S. 178357–178391

[105] OpenAI: *Gym.* https://github.com/openai/gym, 2022

[106] Mnih, Volodymyr ; Kavukcuoglu, Koray ; Silver, David ; Rusu, Andrei A. ; Veness, Joel ; Bellemare, Marc G. ; Graves, Alex ; Riedmiller, Martin ; Fidjeland, Andreas K. ; Ostrovski, Georg u. a.: Human-level control through deep reinforcement learning. In: *nature* 518 (2015), Nr. 7540, S. 529–533

[107] Kingma, Diederik P. ; Ba, Jimmy: Adam: A method for stochastic optimization. In: *arXiv preprint arXiv:1412.6980* (2014)

[108] Tremblay, Olivier ; Dessaint, Louis-A ; Dekkiche, Abdel-Illah: A generic battery model for the dynamic simulation of hybrid electric vehicles. In: *2007 IEEE Vehicle Power and Propulsion Conference* Ieee, 2007, S. 284–289

[109] Nord Pool: *System Price Curve Data.* https://www.nordpoolgroup.com/en/elspot-price-curves/, 2021

[110] Energi Norge Bellona, DNV Enova: FOSSIL- OG UTSLIPPSFRIE BYGGEPLASSER Rapport. (2019)

[111] Miljodirektoratet: *Utslipp av klimagasser i kommuner.* https://www.miljodirektoratet.no/tjenester/klimagassutslipp-kommuner/?area=426&sector=7, 2022

[112] Iacobuta, Gabriela ; Dubash, Navroz K. ; Upadhyaya, Prabhat ; Deribe, Mekdelawit ; Höhne, Niklas: National climate change mitigation legislation, strategy and targets: a global update. In: *Climate policy* 18 (2018), Nr. 9, S. 1114–1132

[113] Seo, Min-Seop ; Kim, Taeyeon ; Hong, Goopyo ; Kim, Hyungkeun: On-site measurements of CO2 emissions during the construction phase of a building complex. In: *Energies* 9 (2016), Nr. 8, S. 599

[114] Sandanayake, Malindu ; Zhang, Guomin ; Setunge, Sujeeva ; Li, Chun-Qing ; Fang, Jun: Models and method for estimation and comparison of direct emissions in building construction in Australia and a case study. In: *Energy and Buildings* 126 (2016), S. 128–138

[115] Barandica, Jesús M ; Fernández-Sánchez, Gonzalo ; Berzosa, Álvaro ; Delgado, Juan A. ; Acosta, Francisco J.: Applying life cycle thinking to reduce greenhouse gas emissions from road projects. In: *Journal of cleaner production* 57 (2013), S. 79–91

[116] Carmichael, David G. ; Malcolm, Cassandra J. ; Balatbat, Maria C.: Carbon abatement and its cost in construction activities. In: *Construction Research Congress 2014: Construction in a Global Network*, 2014, S. 534–543

[117] Hajji, Apif M. ; Muladi ; Larasati, Aisyah: 'ENPROD'MODEL–estimating the energy impact of the use of heavy duty construction equipment

by using productivity rate. In: *AIP Conference Proceedings* Bd. 1778 AIP Publishing LLC, 2016, S. 030008

[118] Vazquez-Canteli, Jose R. ; Dey, Sourav ; Henze, Gregor ; Nagy, Zoltan: CityLearn: Standardizing Research in Multi-Agent Reinforcement Learning for Demand Response and Urban Energy Management. In: *arXiv preprint arXiv:2012.10504* (2020)