המכללה האקדמית
להנדסה בתל אביב

A management system for charging Electric Vehicles to avoid overloading Electric Grid in residential buildings

A project report submitted toward the degree of
Master of Science in Systems Engineering

By

Daniel Tubiana

This project was carried out in the M.Sc. program of
Systems Engineering

Under the supervision of Dr. Mendel Shalom

Abstract:

With the expected rise of electric vehicles (EVs) globally, there emerges a challenge concerning the coordination between EV charging and the power grid, especially during peak hours. Unregulated charging can result in grid congestion, potentially harming the electrical infrastructure, causing voltage instability, and indirectly increasing pollution. This case study focuses on designing a management system for EV charging, in residential buildings with multiple parking lots, to alleviate future grid congestion. Using Systems Engineering methodologies, this project contributes significantly to various stakeholders, including electricity providers, environmental organizations, electricity consumers and EV users.

Deep reinforcement learning (DRL) algorithms, specifically A2C and PPO, were investigated for their capability in managing EV charging schedules. These algorithms were chosen based on their potential to balance the increasing EV charging demand with grid constraints. To support system capabilities, an SQL database was designed to store data for billing, facilitating operations research, and optimizing electric vehicle fleets among other potential applications. Data simulation played a pivotal role in approximating real-world scenarios, simulated data representing crucial factors like user behavior, electricity usage in buildings, and EV characteristics. After establishing the simulated data, an environment was developed to simulate a 24-hour scheduling process for using reinforcement learning algorithms. Both the A2C and PPO algorithms were tested in this designed environment to assess their effectiveness.

The study reveals that A2C, in both 100% and 78% adoption scenarios, outperformed PPO across all metrics. However, neither algorithm comprehensively addressed the scheduling problem within the provided constraints, especially when it comes to maintaining a consistent power profile. Initial episodes for both algorithms showed spikes in power, indicating an inclination towards immediate rewards, contrary to the expectation of a balanced reward realization throughout episodes.

In conclusion, while A2C demonstrated superior adaptability, there remains a necessity to formulate a more sophisticated reward system and to engage in extensive hyperparameter optimization for both algorithms. This study emphasizes the capabilities of Systems Engineering to tackle real-world grid overload issues, setting the foundation for future research in this area. Subsequent investigations should prioritize refining the algorithms used, delving into more advanced algorithmic models, and integrating the multiple reward functions to yield effective solutions.

Acknowledgements:

First, I would like to thank my supervisor Dr. Shalom Mendel for his guidance and support along this journey. He has been a source of inspiration and encouragement for me, and I have learned a lot from his expertise and experience. I am grateful for his constructive feedback and patience throughout this process.

I would also like to thank all the lecturers and staff of the Systems Engineering program for enabling me to explore and learn this field in an extensive manner. They have shared their knowledge and insights with us and have created a stimulating and enriching environment for learning. I appreciate their dedication and professionalism.

Table of Contents

1 List of abbreviations and notations:

- EV – Electric Vehicle
- RL – Reinforcement learning
- DRL – Deep Reinforcement learning
- PV – Photo voltaic
- TEMP - Test and Evaluation Master Plan
- COT - Cost of Test
- V2G – Vehicle to grid method
- DDQN – Double Deep Q-Network
- DDPG – Deep Deterministic Policy Gradient
- P-DQN – Parameterized Deep Q-Network
- SOC – State of Charge
- GA – Genetic algorithm
- IEA – International Energy Agency
- L1, L2, L3 – charge levels AC and DC charging methods
- DSO – Distribution System Operator
- ENonD – Energy Needed on next Departure
- DB - Database
- EDD – Estimated Drive Distance
- KDE – Kernal Density Estimator
- IEC - Israeli Electricity Company
- Agent: The decision-maker or the learner.
- Environment: Everything outside the agent.
- Action (a): A set of all possible moves the agent can make.
- State (S): A scenario that the agent can observe.
- Reward (R): Feedback from the environment.
- R_{t+1} is the reward as the next time step.
- Policy (π): The strategy that the agent employs to determine the next action based on the current state.
- Value (V): The expected long-term return with discount, as opposed to the short-term reward R.
- Vπ(s) is defined as the expected long-term return of the current state under policy π.
- $Q(s,a)$ Q-value or action-value: Q-value is similar to Value, except that it takes an extra parameter, the current action a.
- G_t is the cumulative reward.
- γ is the discount factor
- $\pi_\theta(a|s)$: Policy function, $\pi: S \rightarrow A$
- $V_\phi{}^\pi(s)$: State-value function under policy π, the expected return from state s
- $Q^\pi(s,a)$: Action-value function under policy π, the expected return from taking action a in state s
- $A^\pi(s,a)$: Advantage function under policy π, the advantage of taking action a in state s over the average action.
- MC – monte Carlo learning method.
- TD – Temporal Difference learning method.
- RSD – relative standard deviation

2 List of figures:

3 List of tables:

4 Introduction:

In recent years, the shift towards a more sustainable transportation landscape has gained significant momentum, with electric vehicles (EVs) emerging as a preferred alternative to traditional gasoline-powered vehicles. Countries worldwide, and Israel one of many, have been encouraging the adoption of EVs as a part of their green initiatives, often supplemented by government subsidies. Such endeavors emerge from the broader objective to reduce carbon footprints, decrease greenhouse gas emissions, and promote a cleaner environment.

However, with progress come challenges. The surging growth of the EV market brings forth a crucial concern: the potential strain on our existing electrical grids. As more EVs take to the roads, the demand for charging – especially during peak hours – can pose significant challenges to electrical infrastructures. Unregulated charging can lead to problems such as grid overloading, voltage instability, and energy losses, which could, in turn, reduce the lifespan of critical components like transformers and transmission lines[1]. Moreover, the secondary consequences, including potential damage to home appliances due to voltage instability and elevated pollution levels during peak demand times, cannot be ignored.

Such challenges necessitate a well-coordinated, systems engineering approach. A methodical response that can balance the evolving needs of EV users with the practical constraints of the electrical grid is essential. The stakeholders of this solution span a broad spectrum, including national governments, electricity suppliers, environmental advocacy groups, EV manufacturers, and of course, the end-users.

This project is set against this backdrop. Its core objective revolves around designing a robust management system to oversee the charging of EVs, particularly in residential areas with multiple parking facilities. By leveraging advanced techniques like deep reinforcement learning (DRL) algorithms, the aim is to develop a mechanism that can efficiently schedule EV charging queues, adhering to grid constraints and individual user preferences. The overarching goal is twofold: ensure a seamless EV charging experience for users while safeguarding the integrity and stability of the electrical grid. In achieving this, the project underscores the critical role of systems engineering in addressing pressing real-world challenges, epitomizing interdisciplinary collaboration's potential to shape a sustainable future.

4.1 The need for the system and stakeholders:

4.1.1 The need for the system:

In line with the global and local trend of EV adoption to reduce emissions, forecasts show that the global EV fleet will reach 230 million vehicles in 2030 [2], and a few hundred thousand EVs in the next decade in Israel [3]. The charging of EVs is supplied by the electric grid, but it is not coordinated with it. Lack of coordination with the electric grid will lead to high demand on the grid during peak hours. The high demand for electricity will cause damage to the electrical infrastructure, such as reduced lifespan of transformers and transmission lines, and secondary damage to home appliances due to voltage instability and energy losses [1]. Additionally, high pollution is expected as a result of the high demand for electricity, with peak demand expected late in the afternoon and early in the evening [3].

4.1.2 Stakeholders:

- States & countries
- Electricity suppliers
- Environmental Protection organizations
- Electric vehicles manufacturers
- Customers of electric vehicles

5 Objective:

The purpose of this project/study is to design a management system for charging electric vehicles in order to prevent overloading the electricity grid in residential buildings with multiple parking lots. The main objective is to avoid grid congestion by controlling the charging request queue and regulating the charging power based on grid constraints and user preferences.

6 The context and contribution to Systems Engineering:

The power grid is a multifaceted entity, operated and maintained by thousands of dedicated professionals. Systems Engineering plays a pivotal role in managing, maintaining, and devising solutions for this intricate system. As the world witnesses a surge in the adoption of EVs, there arises a pressing need to ensure that the existing electric infrastructure can accommodate this shift without faltering. This is where the project comes into play, aiming to directly benefit the Electricity Authority, electricity providers, the nation at large, environmental organizations, everyday electricity consumers, and, of course, EV users. The primary objective of this project is to delve deep into the influence and performance of deep reinforcement learning (DRL) algorithms on the scheduling of EV charging queue. This is all part of a larger endeavor to develop a comprehensive management system for charging EVs, ensuring that the electric grid in residential buildings remains stable and free from overloading.

In the grand scheme of sustainable transportation, this project stands as a testament to the potential of Systems Engineering in addressing real-world challenges. By employing deep reinforcement learning (RL) algorithms, the project seeks to establish a harmonious balance between the increasing demand for EV charging and the limitations of the existing electric grid. The end goal is to create a user-centric system that not only prioritizes the needs of EV users but also ensures the longevity and stability of the power grid. Such initiatives not only pave the way for a more sustainable future but also underscore the importance of interdisciplinary collaboration in solving complex problems.

6.1 Needs analysis:

6.1.1 Increased EV Adoption:

With the global and local trend leaning towards the adoption of EVs to mitigate emissions, there's an anticipated surge in the number of EVs. Forecasts indicate that the global EV fleet will reach 230 million by 2030, and Israel alone expects a few hundred thousand EVs in the next decade. This rapid growth necessitates a robust system to manage the charging demands of these vehicles.

6.1.2 Uncoordinated Charging:

Currently, the charging of EVs is powered by the electric grid, but there's no coordination between the two. This lack of synchronization can lead to unregulated charging, especially during peak hours. A system is required that can seamlessly integrate EV charging with the electric grid's operations.

6.1.3 Protection of Electrical Infrastructure:

Unregulated and high demand on the electric grid, especially during peak hours, can inflict significant damage to the electrical infrastructure. This includes a reduced lifespan for transformers and transmission lines. There's a pressing need for a system that can distribute the charging load, ensuring the longevity and health of the electrical components.

6.1.4 Home Appliance Safety:

The secondary repercussions of an overloaded grid include potential damage to home appliances due to voltage instability and energy losses. A system that ensures stable voltage and minimizes energy losses will not only safeguard the grid but also protect household appliances.

6.1.5 Environmental Concerns:

High electricity demand, especially during peak hours in the late afternoon and early evening, can lead to increased pollution. This contradicts the primary goal of EV adoption, which is to reduce emissions. A system that can manage and spread out the EV charging demand, especially during these peak hours, is essential to ensure that the environmental benefits of EVs are fully realized.

6.2 Concept Exploration:

When it comes to grid load balancing for EV charging, several strategies stand out as potential solutions. Firstly, leveraging advanced optimization and scheduling algorithms can ensure efficient use of resources. A dynamic electricity pricing mechanism, which adapts to the grid's load and prevailing demand, can act as a demand-side management tool. Integrating renewable energy sources, such as Photovoltaic Cells (PV) and Wind turbines, not only diversifies the energy mix but also promotes green energy consumption. Moreover, energy storage solutions, like advanced battery systems, and Vehicle to Grid (V2G), offer a cushion during peak loads or when renewable sources aren't generating. On the infrastructure side, refining the charging systems to counteract issues like phase unbalance, voltage deviations, and power harmonics is of paramount importance. Importantly, an engineered solution may often merge several of these tactics, ensuring holistic and effective grid management.

6.3 Concept Definition:

This phase encompasses the analysis of alternatives, functional architecture, and physical architecture. Alternatives were assessed between centralized and decentralized methods to address the issue. The centralized approach was selected. Within this centralized framework, two optimization methods were analyzed: reinforcement learning and meta-heuristic methods, such as the Genetic Algorithm. Reinforcement learning was chosen because literature suggests that the RL method can be scalable and deployed for large EV fleets.

The needs and objectives of the systems served as general requirements, from which specific system requirements were derived.

6.3.1 General requirements:

1. The system shall manage and optimize the charging process of EVs to prevent grid overload.
2. The system shall schedule each EV charging to user preferences and needs.
3. The system shall rely on historical data of evs and power limits.
4. Every EV fleet which consumes power from distribution transformer and connected to system shall be connected to Hub that controls charging process.
5. The system shall enable users to input their Arrival and Departure times, and planned travel distance.
6. The mobile application shall send all the users' information and preference to cloud for next day scheduling.
7. The Hub shall support connection to cloud and enable bi-directional data transfer.
8. Hub unit shall collect data from building electricity meter and from chargers.
9. Cloud unit shall perform all computation of next day scheduling.

10. The system shall allow users to advance in line in case of a request to advanced.
11. The systems shall not significantly harm or delay users who choose not to advance in the queue.

6.3.2 Functional Architecture:

In Figure 1 Functional diagram of system, The operation of the system is described as follows: There are two external inputs to the system. The first is user preferences, such as arrival time, departure time, and estimated drive distance for the next departure. The second input is the connection of the EV to the charger. Once the inputs are received, the data is sent to the cloud for data collection. Following data collection, decision-making is conducted in the cloud to determine when each EV will charge and how much energy will be provided. After the decision is made, the scheduling and charge profile are sent to the user's mobile app for approval. Users have the ability to request a different charge profile, but the number of times they can request a new profile is limited to prevent endless requests. Once the final profile decision is made, the cloud sends the data to the Hub, which then updates the charges for the next charging session.

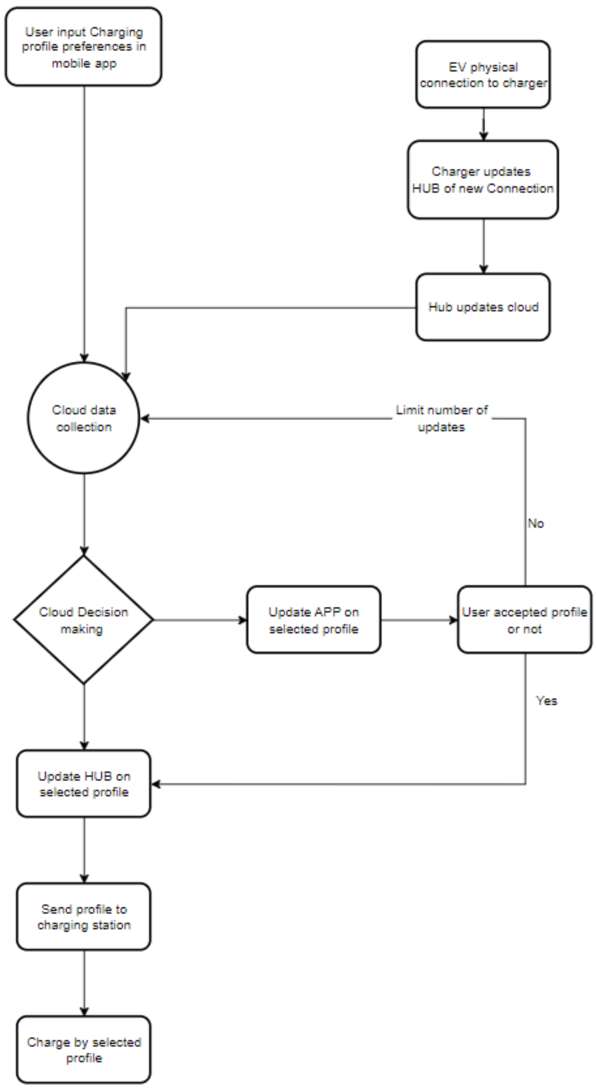

User input Charging profile preferences in mobile app
EV physical connection to charger
Charger updates HUB of new Connection
Hub updates cloud
Cloud data collection
Limit number of updates
No
Cloud Decision making
Update APP on selected profile
User accepted profile or not
Yes
Update HUB on selected profile
Send profile to charging station
Charge by selected profile

Figure 1 Functional diagram of system

6.3.3 Physical Architecture

The system composed from four main components as shown in Figure 2- Physical Architecture of the system, the main components are Smart hub, Cloud, EV Charger, and mobile app. Components are explained in detail in the next chapters.

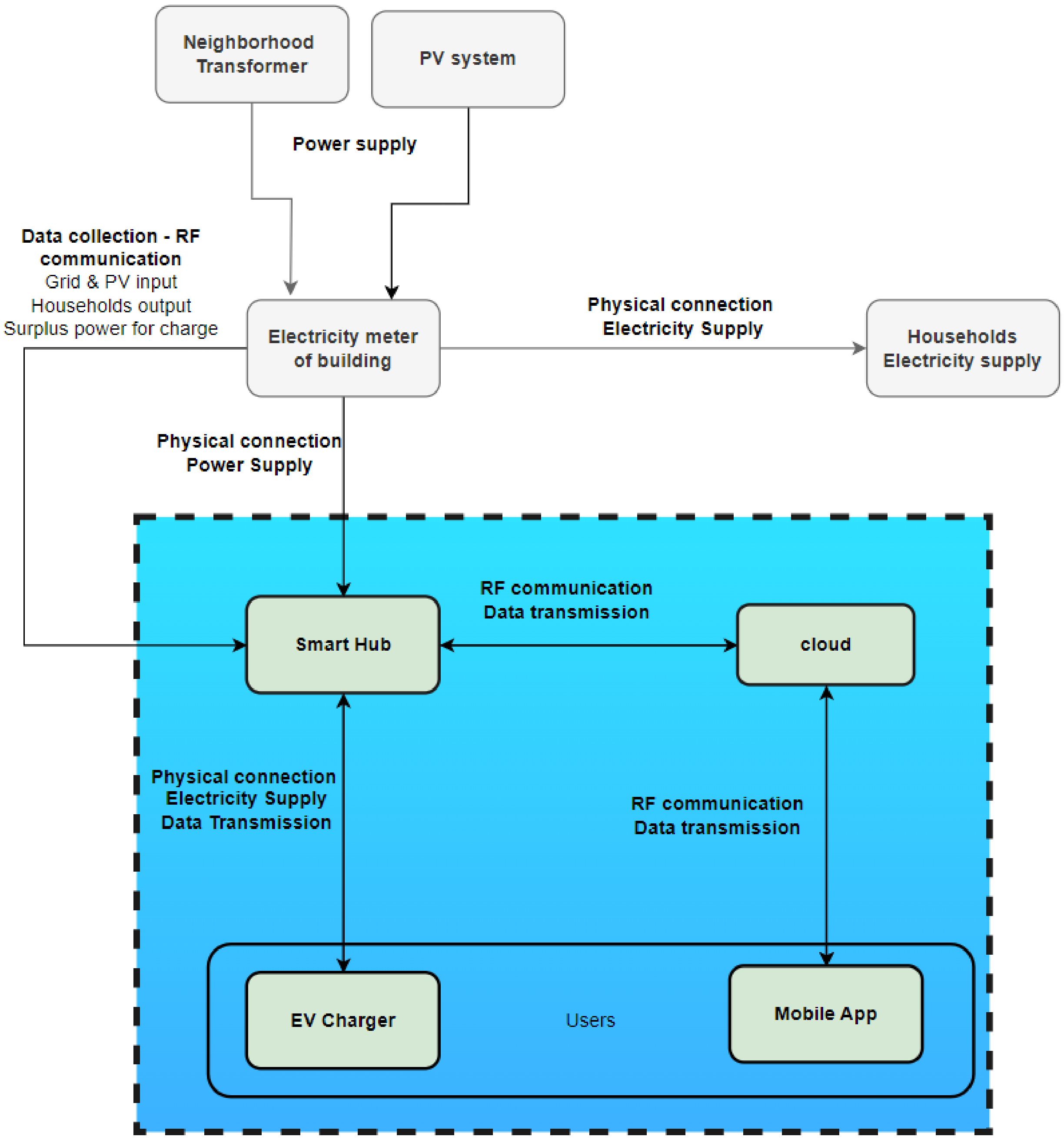


Figure 2- Physical Architecture of the system

6.4 Advanced Development:

6.4.1 Risk management

This step focuses on the identification and reduction of development risks and the formulation of system design specifications. Risk management is carried out by analyzing each component of the system and assessing potential outcomes. During this phase, several risks were identified that are related to the scope of this work: comparing the performance of two different algorithms to schedule EV charging queues and control charging loads to prevent grid overload.

Table 1 -Risks and mitigations

Risk	mitigation
Agent charge decision is not applicable – Charging EV when it is not at home (did not arrive or departure)	Add in the environment condition to check if the EV is in home and if it not, do not charge
EV was not charged to required level	This risk will not be addressed in this phase
User preferences was changed: arrival time or departure time is changed	Change user inputs and run algorithm to reschedule charging – This risk will not be addressed in this phase and the solution will not be tested
Total charge power is above surplus power	This risk will not be addressed in this phase

6.4.2 System specifications

System specification was reminded in high level since the project was oriented to perform a case study of algorithms performance scheduling charge queue and managing grid power load, system high level and shown above in Figure 2- Physical Architecture of the system

1. HUB (Smart Hub):

Purpose: Act as an intermediary device bridging the communication between the mobile app, EV chargers, and the cloud infrastructure.

Main Functions:

- Directly connect and control the charging process of the EVs based on commands received.

- Interface with the cloud for bi-directional data transfer, ensuring updated algorithms outputs and user inputs are correctly relayed.

- Support detection and verification of EV presence to avoid charging anomalies.

Connectivity:

- Provide wired connectivity to seamlessly interface with EV chargers.

- Secure cloud connectivity capabilities to ensure data integrity and protection.

2. Cloud Computing Unit:

Purpose: Host the core algorithms responsible for optimizing EV charging schedules and provide storage for historical data of EVs and grid power limits.

Main Functions:

- Perform computations for the next-day charging scheduling.
- Store user preferences and historical data securely.
- Distribute updated charging algorithms to the HUB as needed.

Infrastructure:

- Scalable server architecture to handle varying loads and number of users.
- Redundancy in place to ensure high availability and minimize downtime.
- Adequate security measures to safeguard user data and maintain data integrity.

3. Mobile App:

Purpose: Allow users to input their preferences, view charging schedules, and receive updates about their EV charging status.

Main Functions:

- Enable users to set their arrival, departure times, and planned travel distance.
- Relay user information and preferences to the cloud for next-day scheduling.
- Provide real-time notifications and updates regarding the charging process.
- Possess a user-friendly interface, ensuring ease of operation.

4. EV Chargers:

Purpose: Physically charge the EVs based on commands received from the HUB.

Main Functions:

- Accept and implement charging instructions relayed from the HUB.
- Integrate built-in safety features to prevent overcharging or any form of damage to the EVs.
- Communicate with the HUB to provide status updates and any potential issues during charging.

Connectivity:

- Seamless interface capabilities with the HUB to ensure uninterrupted charging sessions.

6.5 Engineering Design:

6.5.1 Building blocks and system decomposition

As described In concept definition the core of the EV charging management system comprises four building blocks:

- Hub

The hub, designed to seamlessly integrate with the EV charging infrastructure and cloud systems, should encompass advanced communication modules for dual interfacing: wired connections like Ethernet for high-speed data transfers with chargers, and wireless modules such as Wi-Fi or 4G/5G for cloud connectivity. Embedded with a robust microcontroller or microprocessor, the hub should have the computational capability to interpret, preprocess, and relay data. It must incorporate secure and encrypted communication protocols to ensure data integrity and protection against unauthorized access. Additionally, the hub should have firmware that's remotely updatable to synchronize with evolving cloud algorithms and charger technologies, as well as ports or interfaces that facilitate easy physical integration with diverse charger models.

- Cloud Infrastructure

The cloud infrastructure for the EV charging management system should consist of scalable compute resources capable of running complex optimization algorithms in real-time. This requires robust servers with high computational capacity, ideally using container orchestration tools for scalability and resilience. The infrastructure should also incorporate a database system, such as a relational database solution, to store and manage historical data of EVs, user preferences, and power limits. Secure API endpoints need to be established to facilitate smooth data transfer between the mobile application and the cloud, ensuring encryption and authentication mechanisms are in place for data integrity and user privacy. Additionally, redundancy measures, including backup and disaster recovery solutions, should be integrated to ensure the continuous operation of the system.

- User Interface (Mobile Application)

The mobile app, pivotal for user engagement in the EV charging management system, should feature an intuitive user interface, allowing users to effortlessly input preferences like Arrival-Departure times and planned travel distance. Integrated with robust API calls, the app should ensure seamless and encrypted communication with the cloud infrastructure for real-time data exchange. Push notifications should be enabled, updating users on their charging status, queue advancements, or any critical alerts. The app should also incorporate secure user authentication protocols, such as biometric recognition or two-factor authentication, to safeguard user data and preferences. Lastly, an embedded help or tutorial section, coupled with responsive customer support integration, can ensure users have a streamlined and trouble-free experience.

- EV chargers

The chargers, as pivotal endpoints in the EV management system, should be engineered to function autonomously based on commands from the hub. Equipped with integrated communication interfaces, they should be capable of interpreting and executing commands received from the hub in real-time, such as starting or stopping the charging process, modulating charging rates, or reporting any anomalies. These chargers should also feature built-in sensors to monitor real-time metrics like current, voltage, and temperature, subsequently relaying this data back to the hub for continuous system feedback. To ensure reliable operation, the chargers must possess fail-safe mechanisms, enabling them to revert to a safe mode in case of command conflicts or discrepancies, ensuring user safety and grid stability.

6.5.2 Requirements analysis

1. Identification:

The provided requirements have been outlined as follows:

- Optimize EV charging.
- Schedule based on user preferences and needs.
- Utilize historical data of EVs and power limits.
- Ensure every EV fleet user connects to a hub via mobile application.
- User input for Arrival-Departure times and planned travel distance.
- Use of cloud for next-day scheduling.
- Bi-directional data transfer from the hub.
- Provision for users to advance in the charging queue without significant harm or delay to others.

2. Clarification:

- Optimize EV Charging: "optimize" means charging most of cars in the time window of 24 hours and charge each EV to required charge level without load thr grid during charge.

- Historical Data: Historical data referred to surplus power which derives from total power input from grid and PV cells after the required energy deliver to households, another part of historical data is EVs data: Arrival time, departure time, Energy needed on next departure, and EV characteristics.

- Advance in Charging Queue: User request for advance should be used in emergency cases, where departure time should occur before the real departure time, and significant harm or delay is quantified by that the other users should not influenced form user request for advance in the queue.

3. Classification:

Functional Requirements:

- Optimizing EV charging.
- Scheduling based on user input and historical data.
- Bi-directional data transfer from Hub to cloud and vice versa.
- Provision for queue advancement.
- Utilizing historical data.
- Connecting every EV fleet to a hub via Chargers.
- Cloud infrastructure for computation.
- Hub-to-cloud communication.

Non-functional Requirements:

- User interface for providing input and receiving updates.
- No harm or delay users who choose not to advance in the queue.

4. Prioritization:

High Priority:

- Optimizing EV charging.
- Scheduling based on user preferences and needs.
- Utilizing historical data of EVs and power limits.

Medium Priority:

- Bi-directional data transfer from the hub.
- Cloud-based computation for next-day scheduling.

Low Priority:

- User's ability to advance in the charging queue without significantly harming or delaying others.

5. Validation:

- Feasibility: Given current technology and grid infrastructure, is it achievable to optimize EV charging as described? Is real-time data transfer between the hub and cloud consistently reliable?
    - In literature review the feasibility of optimizing grid load in different techniques is available
    - real time data transfer is also feasible and achievable without additional development
- Relevance: Will utilizing historical data significantly improve scheduling effectiveness? Does the majority of the user base require next-day scheduling?
    - Utilizing historical data will improve systems performance since the algorithms in the code of the systems are based on historical data.
    - Most of the users in the case of "business as usual" are scheduling their arrival and departure times.

6.6 Integration & Evaluation:

For the Integration and Evaluation phase of the EV charging system, the strategy will commence with the guidance of the Test and Evaluation Master Plan (TEMP). This ensures a structured approach, aligned with the unique requirements of the system. Key to this stage is the acquisition and calibration of test equipment specifically tailored to the charging infrastructure, ensuring the evaluation captures real-world scenarios like phase unbalance, voltage deviations, and power harmonics. Initial focus will be on integrating individual charging subsystems; this involves ensuring the chargers react optimally to commands received from the hub and validating their performance against user-input data from the mobile app. Once individual subsystems demonstrate satisfactory operation, total system integration will ensue, connecting the cloud, hub, mobile application, and chargers into one harmonized entity. The culminating step will involve comprehensive operational tests and evaluations, assessing not just the basic functionalities but also the system's adherence to our predefined general requirements, ensuring seamless scheduling, user preference accommodation, and grid load optimization.

6.7 Production:

In the Production phase, the main emphasis will be on the manufacturing of chargers and hubs, given their bespoke nature, while synergizing them with off-the-shelf products for other system components. Adopting Systems Engineering (SE) principles within our factory setup ensures that the production processes remain streamlined and integrated. By emphasizing concurrent engineering, we'll facilitate simultaneous product design and process development, boosting efficiency and reducing time-to-market. The transition from development to production will be orchestrated through a robust production Configuration Management system, keeping a keen eye on the Cost of Test (COT) to ensure optimal resource utilization. As for production operations, the organizational structure will be delineated to prioritize both component manufacture and system acceptance. This hierarchy ensures that while individual components like chargers and hubs are produced to exact specifications, there's an overarching layer supervising the integration and acceptance of the entire system, ensuring every unit rolling out aligns perfectly with our set requirements and quality benchmarks.

6.8 Operation & Support:

To ensure the seamless functionality of the EV charging system, a methodical approach to its operation and support is vital. Initial installation and testing are meticulously planned to be non-disruptive, allowing users and connected infrastructure to operate without unnecessary interruptions. Post-installation, the focus shifts to consistent in-service support and maintenance. As technology progresses and new optimization strategies are developed, major software upgrades will be periodically rolled out. Before any system-wide deployment, these upgrades undergo rigorous testing to guarantee compatibility and enhanced performance. Field services play a pivotal role, acting as the primary sources of operational knowledge, swiftly addressing any on-ground issues, and facilitating feedback loops for continuous improvement. Finally, as components reach the end of their life cycles, a structured system disposal process ensures that they are retired in an environmentally friendly and secure manner, minimizing wastage and potential security risks.

7 Literature review:

7.1 Related work:

Smart charging systems cover numerous topics from systems economics, optimization methods to prevent grid load, use of renewable energies such as wind and solar energy to increase the available power during the day and even store it and use during high demand hours, Hardware implementations and improvements to maximize charging power without damaging the battery during the charging process and draw energy from EV to the grid in case of high load. The review of related works in this study centers on optimization methods for charging scheduling techniques and their algorithms to prevent grid overload and regulating the charging power based on grid constraints and user preferences.

In [4] the authors focused on applying different Reinforcement learning (RL) methods such as Double Deep Q-network (DDQN), Deep Deterministic Policy Gradient (DDPG), Parameterized Deep Q-Network (P-DQN) and the study objective was to increase PV self-consumption of the energy system, and achieve as high State of Charge (SOC) at the EV departure as possible. The different RL techniques were compared to other control techniques such as Role-based control and Model predictive control. The simulated system in this paper was composed of two energy sources Grid and PV, two energy consumers the building and EV. Nevertheless, the simulation of this case demonstrated a simplistic system since only one EV was simulated. The research conclusion highlighted the potential of reinforcement learning methodology for efficient online implementations and its suitability for future energy systems that are characterized by an abundance of data, growing complexity, and scalability. However, the existing research has two main problems, the choice of the Mode, where simultaneous supply to EVs from multiple sources is prohibited and the algorithms were tested on a single EV rather than a fleet of EVs.

The use of RL for smart charging is widely used as optimization algorithm to prevent grid overload, in [5] the research objective was to build a centralized smart charging coordination system for a scalable EV fleet based on a single-agent RL algorithm. The researchers used a Deep Q-learning algorithm to schedule the charging queue. To deploy one RL-Agent, the system was composed from a top-level environment and several sub environments, one for each EV. The results of this research show that the Reinforcement Learning-based charging coordination system performed very well in reducing peak loads and shifting energy demand to off-peak periods. Compared with an uncontrolled charging strategy, the Reinforcement Learning algorithm reduces the variance of the total load by 65%. Overall, this study demonstrates that smart charging strategies based on reinforcement learning can effectively address the problem of grid overload caused by the uncontrolled charging of electric vehicles. This article demonstrates an algorithm with feasibility to implant on a real system, however, this work has some limitations, EV fleet should be more heterogeneous, and the charging power was set for the lowest level at 3.7KW while EVs support more charging levels.

In [6] a Genetic Algorithm (GA) was used to optimize the charging process for electric vehicles. GA is a metaheuristic optimization-based approach that can obtain a flat load curve by compressing the level of power fluctuation and find optimal solution in low number of iterations. The researchers also combined Vehicle to Grid (V2G) facilitation which can draw energy from EV to grid in case of high load and can reduce electricity cost, in addition in low demand hours EV will be charged again. Although the algorithm demonstrated promising results, there are several unresolved concerns that the researchers have yet to address. These concerns primarily pertain to system scalability and the potential for exclusion of EVs with the highest (SOC) from the charging queue.

Most of the studies in this field are theoretical, in contrast the researchers in [7] were implanted a real system as part of case study to explore and evaluate the smart charging technology and the development of a scalable EV-to-building integration platform. They used real-time data to test the algorithm performance and evaluated the financial viability of implementing smart charging technology. The financial analysis focused on two topics: installation costs and operation costs of the system. The result shows installation costs can be reduced by 13% and 15% when load management technology is introduced for L2 chargers and EV supply equipment, and only for L2s, respectively. During the tests, a 37% reduction in operating costs was achieved, and a 29% reduction in peak charging load demand for a building with on-site PV generation compared to the other phase of the tests for system without any on-site generation facility. This case study justifies the importance and influence of smart load management in the financial domain. Smart charging enables cost reduction in infrastructure and saves costs by using renewable energy sources.

7.2 EV adoption rate forecast:

The International Energy Agency (IEA) forecasts that the global EV fleet will reach 145 million vehicles in 2030, which will represent 7% of the road vehicle fleet. However, if governments accelerate efforts to reach climate goals, the EV market could be significantly larger. In a sustainable development scenario, the global EV fleet will reach 230 million vehicles in 2030, 12% of the road vehicle fleet [2]. Another survey predicts that the number of light-duty EVs and their charging plugs will multiply to over 300 million and 175 million, respectively, worldwide by 2035. High adoption rates are expected in cities and neighborhoods with high socioeconomic levels [8].
Forecasts of Israeli Electricity authority and the Central Bureau expects a 300 thousands EVs in 2025 and 950 thousands in 2030. These numbers are based on the center of forecast, and the expected growth rate is 3% per year of new cars in Israel [3]. The forecast was based on a cumulative number of electric vehicles in proportion to the rate of entry of new vehicles for the forecast of 2025 and 2030 based on the IEA publication [2].

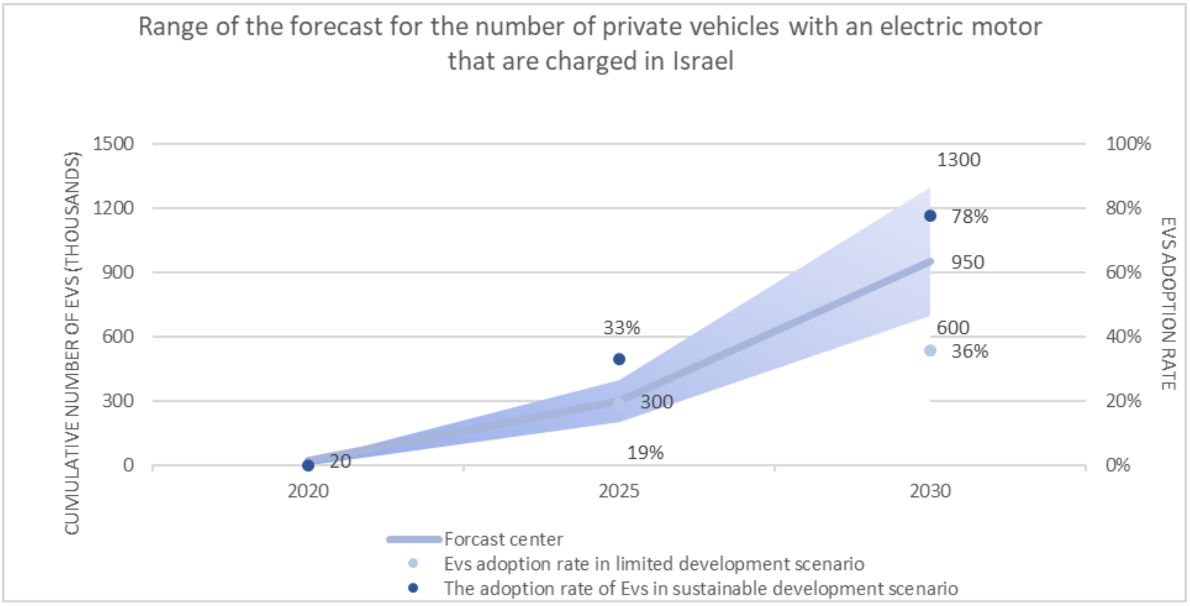


Figure 3 Range of forecast for the number of private vehicles with an electric motor that is charged in Israel taken from [4]

7.3 EVs charging influence on grid and methods to minimize effects:

High EV adoption rate and direct connection to grid without coordination could lead to grid problems such as overloading, voltage instability, voltage sag, energy losses, harmonics, frequency variation, and consequently, reduction in supply quality. These can damage household appliances and shorten transformer lifespan and transmission lines. A few methods proposed to solve these problems, such as Decentralized / Centralized coordination, use of renewable energies and different optimization methods such as heuristic methods and real time balancing of the load [1]. Decentralized coordination allows EV users to choose charging times to reduce load variability but can still overload lines and transformers. Centralized coordination has electricity suppliers manage charging times and offer low-cost electricity during off-peak hours. Renewable energy (wind, PV cells) can offer lower-cost electricity to be stored in EVs battery or designated power storage systems for use during peak hours. But, in case the renewables are in isolated structures, they are far more vulnerable to unplanned energy variances than interconnected structures [1].

7.4 Electric Grid effects and emissions pollution:

The objective of demand management and charging management is to reduce or smooth the peak of electricity consumption during rush hours. The management of demand and charging power takes place by changing the start time of charging, power management, and electricity tariff change. Demand management itself is not an optimal solution to reduce the electricity demand, but also changing charging access and charging behavior throughout all day helps to reduce the load [8]. in addition, it has been found that Centralized coordination is best optimal management strategy to solve issues of EVs integration into grid. The Solver/tools used in Centralized coordination are: Convex optimization, Quadratic optimization, Dynamic optimization, Meta-heuristic method, Fuzzy logic, and Artificial immune system. [1]

7.5 Electric Grid effects and emissions pollution:

EV charging, despite its goal of reducing emissions, is not coordinated with the grid. Powell et al. [8] found that high EV adoption increases annual electricity consumption by 0.11% per 1% increase, reaching 11% at full adoption. Adequate infrastructure is needed for multi-parking residential buildings to support this demand. Early identification of load on the grid is crucial for planning and reducing risks. In their study four charging scenarios were modeled based on location and time of charging. EV charging is much more environmentally friendly and may significantly reduce pollutant emissions into the air compared to internal combustion engines. The researchers found that the grid emissions will reduce by 4 times the operational emissions of power grid compared with the average internal combustion engine vehicle.

7.6 Charging types:

There are 3 types of charging, Level 1 and Level 2 based on AC, and Level 3 based on DC, each type has advantages and disadvantages, in Table 2 - Types of Charging stations and the power of the station charge types presented. The lower charging levels (1&2) offer affordable prices but slower charging times to EVs and vice versa on Level 3 charging. **Error! Reference source not found.**.

Table 2 - Types of Charging stations and the power of the station

Type of station	**Single Phase – Level 1**	**Three Phase -Level 2**		**DCFC – Level 3**	
Station Power	3.7kW (1x16A)	11kW (3x16A)	22kW (3x32A)	50kW (3x80A)	150kW (3x250A)

$$C_{\Delta t} = B_C \cdot SOC * P_L$$

Equation 1 - Charge time calculation

- B_c - Total battery capacity in [KWh]
- SOC - State of Charge in % from total capacity which is 100%
- P_L – Power level in [kW]

8 Methods:

8.1 Approach and Method of Analysis:

This case study will adopt a quantitative approach by implementing two algorithms to address grid overload, control the charging request queue, and regulate charging power based on grid constraints and user preferences. This implementation will yield both numerical and categorical data, which will be analyzed through a comparative evaluation of the algorithmic results. The algorithms will be compared and evaluated based on several parameters. These parameters include efficiency, measured by the time of convergence, and required resources, scalability and adaptability to new data or parameters; count of deviations, including "not enough energy" for EV before departure, the number of events the "power limits exceeded"; and the load variance.

8.2 Tools:

- Data simulation
- Algorithms implementation – Python and supported libraires.
- Algorithms evaluation – Python and statistical libraries for operational research to compare results for tests to see if there is statistical significance between algorithms.
- Database – SQL for data storage and access for algorithms.

8.3 Work Process:

8.3.1 Database design

The SQL Database was designed to store both billing and operational data of systems. This facilitates operations research, leveraging the data in the schema. Utilizing this data for operations research offers a plethora of potential applications. For instance:

1. The data can be used to forecast charge loads and offer insights for algorithm improvements.
2. Energy management and infrastructure enhancements aim to save costs.
3. Electric Vehicle Fleet Optimization: By analyzing data from the 'ev' and 'user_ev_connection' tables, electric vehicle fleets can be optimized for various objectives, such as maximizing revenue, minimizing charging time, or reducing greenhouse gas emissions.
4. Environmental Impact Analysis: Data from various tables can be analyzed to assess the environmental impact of electric vehicle charging and the use of renewable energy sources.
5. Risk Analysis: Operations research techniques can be employed to evaluate potential risks and uncertainties in the electric vehicle charging process. This might encompass evaluating the effects of fluctuating electricity prices, the availability of solar power, or unforeseen shifts in user behavior.

DB tables and values description:

Database table schema is shown in Figure 4 - Database Schema, the code for DB creation is shown in appendix 12.1.

Electricity Company Tariff Table

This table stores all the necessary data for electricity pricing from the grid for billing, electricity pricing is affected by season and time of day since the electric demand changes during the day and session, by setting lower prices out of demand hours electricity company can encourage electricity users to change their behavior.

Electricity company ID: Represents company id, in some countries called Distribution system operator (DSO)

Month: Represents the month to which the tariff applies.

hour: Represents the hour of the day to which the tariff applies.

price: The price of electricity tariff for the specified month and hour.

Solar panels Tariff Table

The electricity company sets solar panels tariff, and the price depends on the rated PV total output.
solar panel ID: representing the unique ID of the solar panel.

price: The price of solar panel tariff.

Building electricity meter Table

This table represents all the inputs or electricity to building from Grid and PV, and also the output power where is composed from tow consumers households and charging system.

Building ID: Representing the unique ID of the building.

timestamp: The timestamp of the electricity meter reading.

Grid power input: The amount of power input from the grid to the building.

Solar power (PV) input: The amount of power input from solar panels (PVs) to the building.

Households power consumption: The amount of power consumed by households in the building.

Charge power output: The amount of power output used for charging electric vehicles.

User Table

This table stores data of user, the data of the user is subject to changes since users can change their driving behavior during time.

User ID: Representing the unique ID of the user.

timestamp: The timestamp related to the user data entry at some time point during usage of the system.

Estimated arrival time: Estimated time of user's arrival.

Estimated departure time: Estimated time of user's departure.

Estimated drive distance: Estimated drive distance of the user.

EV Table

This table stores all the EV data on specific time points.

EV ID: Representing the unique ID of the electric vehicle (EV).

timestamp: The timestamp related to the EV data entry at some time point during usage of the system.

EV brand model: The brand and model of the EV.

EV efficiency: The efficiency of the EV, how much energy is needed for distance [Wh/ km] and it uses to calculate the Energy Needed on next Departure (ENonD).

EV battery capacity: The battery capacity of the EV, it is used to calculate SOC during charge process.

Energy Needed on next Departure: Represents the Energy that needed on the next departure of EV.

SOC: State of Charge (SOC) of the EV battery.

EV charger Table

This table stores all the data during charging of each charger.

Charger ID: Representing the unique ID of the EV charger.

timestamp: The timestamp related to the EV charger sample time.

Output power: The output power of the EV charger during the charging process.

User charge session-user Table

This table is a connection table between user data to user-charge-session to allow recording of several sessions for each user.

Session user id: Representing the unique ID of the session-user relationship.

Session id: Referencing the session ID from user-charge-session table.

User id: Referencing the user ID from user_ table.

User charge session-EV Table

This table is a connection table between EV data to user-charge-session to allow recording of several sessions for each EV.

Session EV id: Representing the unique ID of the session-EV relationship.

Session id: Referencing the session ID from user-charge-session table.

EV id: Referencing the EV ID from EV table.

User charge session – EV charger Table

This table is a connection table between EV Charger to user-charge-session to allow recording of several sessions for each Charger.

Session EV charger id: Representing the unique ID of the session-EV charger relationship.

Session id: Referencing the session ID from user-charge-session table.

EV charger id: Referencing the EV charger ID from EV charger table.

User charge session Table

This table records and organizes all the data that related to charge session which includes User, EV, Charger, and all additional data to for calculating cumulative power of charge and cumulative price of charge session.

Session ID: Representing the unique ID of the charge session.

Electricity company ID: Referencing the electricity company ID from Electricity Company Tariff table.

Solar panel ID: Referencing the solar panel ID from Solar panels Tariff table.

Building ID: Referencing the building ID from Building electricity meter table.

Connection time: Timestamp when the EV was connected to the charger.

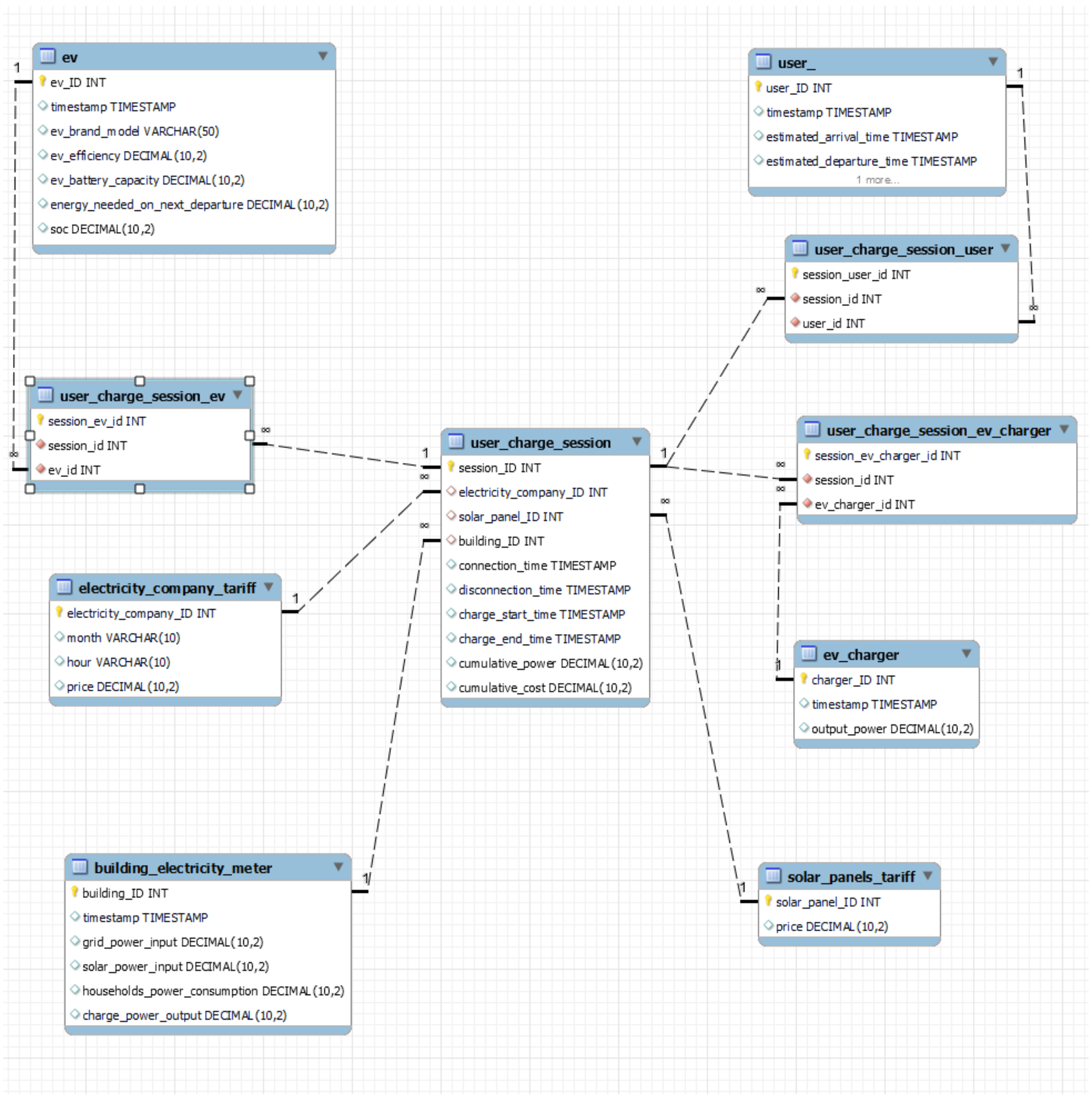
disconnection time: Timestamp when the EV was disconnected from the charger.

Charge start time: Timestamp when the charging session started.

Charge end time: Timestamp when the charging session ended.

Cumulative power: Cumulative power consumed during the charging session.

Cumulative cost: Cumulative cost of electricity during the charging session.


ev
ev_ID INT
timestamp TIMESTAMP
ev_brand_model VARCHAR(50)
ev_efficiency DECIMAL(10,2)
ev_battery_capacity DECIMAL(10,2)
energy_needed_on_next_departure DECIMAL(10,2)
soc DECIMAL(10,2)
user_
user_ID INT
timestamp TIMESTAMP
estimated_arrival_time TIMESTAMP
estimated_departure_time TIMESTAMP
1 more...
user_charge_session_user
session_user_id INT
session_id INT
user_id INT
user_charge_session_ev
session_ev_id INT
session_id INT
ev_id INT
user_charge_session
session_ID INT
electricity_company_ID INT
solar_panel_ID INT
building_ID INT
connection_time TIMESTAMP
disconnection_time TIMESTAMP
charge_start_time TIMESTAMP
charge_end_time TIMESTAMP
cumulative_power DECIMAL(10,2)
cumulative_cost DECIMAL(10,2)
user_charge_session_ev_charger
session_ev_charger_id INT
session_id INT
ev_charger_id INT
electricity_company_tariff
electricity_company_ID INT
month VARCHAR(10)
hour VARCHAR(10)
price DECIMAL(10,2)
ev_charger
charger_ID INT
timestamp TIMESTAMP
output_power DECIMAL(10,2)
building_electricity_meter
building_ID INT
timestamp TIMESTAMP
grid_power_input DECIMAL(10,2)
solar_power_input DECIMAL(10,2)
households_power_consumption DECIMAL(10,2)
charge_power_output DECIMAL(10,2)
solar_panels_tariff
solar_panel_ID INT
price DECIMAL(10,2)
1
∞
1
∞
1
∞
1
∞
1
∞
1
∞
1
∞
1
∞
Figure 4 - Database Schema

8.3.2 Data Simulation:

In order to find an optimal solution to the problem and evaluate system performance, data is a crucial component of this system simulation. The goal of data simulation is to generate a dataset that closely mirrors real-world data. This simulation aims to create 5 distinct tables out of the 7 available in the database. Not all tables were utilized in the algorithms, and some were specifically designed to simulate data intended for database loading, such as electricity pricing from the Grid and PV. The tables created for the algorithms include the user table, building electricity meter, and EV characteristics. The EV charger and User-charge-session tables were not simulated, as they are the outputs of the algorithms and represent system outputs.

Simulation requirements:
Simulation requirements were derived from [3], some of the requirements was hardened to extreme conditions.

- Distribution transformer of building with 126 apartments or private houses supply 630 [KVA].
- The area of building is wealthy, and each family have 1.5, and total of Vehicles is 189.
- EVs Rate of EVs in building in phase of algorithms development should be 100% which it is 189 EVs
- EVs Rate of EVs in building in phase of algorithms evaluating performance should be 78%, based on adoption rate of EVs in sustainable development scenario, which it is a 148 EVs.
- All EVs users need to charge their EVs in a timeframe of 24 hours.

The simulation process is divided into five steps, steps are described below:

<u>User Data Simulation:</u> This section of the script simulates data for EV users. It includes details such as the user ID, time of the charge request, and estimated driving distance (EDD). For each user, the script also determines the energy required for the next departure (ENonD) by considering the estimated driving distance and the EV's efficiency. This data is then stored in a data frame named 'df_user'.

<u>EV Data Generation</u>: This section of the script simulates data for various types of EVs, using a list of the 8 best-selling cars in Israel. For each EV, the script generates a random state of charge (SOC) and determines the energy required for the next departure (ENonD). This data is then stored in a data frame named 'df_ev_sampled'.

<u>Electricity Price Simulation</u>: This part of the script simulates the price of electricity from the grid for different months and hours of the day. It also assumes a fixed price for electricity from photovoltaic (PV) panels.

<u>Building Electricity Meter Simulation</u>: This section of the script simulates the solar power input, grid power input, and household electricity load for a building. It employs an irradiation dataset to determine the power output of the PV panels at various times throughout the day. The grid's maximum power input is established based on the transformer's capacity and the power factor. The household electricity consumption is derived using a dataset specific to household power usage. All this data is then stored in a data frame named 'df_building'.

<u>Data Export</u>: Finally, the script exports the simulated data to Excel files. The exported datasets include the user and EV data, electricity price data, PV price data, and building power data.

1. **User Table:**

   1.1. **User ID**

   The range of numbers starts at 0 and ends at $N_{EVs} - 1$.

   1.2. **Time of request:**

   This value was assumed to align with the arrival time and was therefore not included in the dataset.

   1.3. **Arrival and departure:**

   Arrival and departure data were sourced from [9]. In the dataset, 'arrival' and 'departure' are represented as 'start plugin' and 'end plug-out' parameters, respectively.
   Arrival time defined in the data set as arrival time and departure time defined as time until departure (TuD)

   During data preprocessing, the aforementioned parameters were converted to DateTime format. Subsequently, each column was transformed based on the day of the week and the 'start plugin' and 'end plug-out' parameters. This led to the creation of four distinct groups: 'start plugin weekday', 'start plugin weekend', 'end plug-out weekday', and 'end plug-out weekend'.

   To determine the best-fit distribution for each group, the Python Fitter library was utilized. While the Fitter library tested 110 known distributions and returned the top five based on the lowest Sum-Square-Error (SSE) between the actual data distribution and the fitted Probability Density Function (PDF), the Gaussian Kernel Density Estimation (KDE) method was also evaluated. Upon comparison, the Gaussian KDE method yielded superior results for all data distributions and was thus selected for distribution determination.

   For departure times, the distribution wasn't directly determined. This is because, in certain instances, the time difference between arrival and departure was negative. After determining the distribution for each function, data sampling was conducted, and the difference between the values was calculated. Randomly selecting departure times could result in times earlier in the day than the arrival times. To avoid negative 'time until departure' values, the difference between departure and arrival times was calculated, yielding the 'time until departure'.

   To ensure that the 'time until departure' was never less than one hour, an additional operation was performed, increasing each sample by one hour. Following these steps, the 'time until departure' distribution was fitted using the Gaussian KDE method.

   Results:

The best fit for arrival times on weekdays was found by using the Gaussian KDE method:

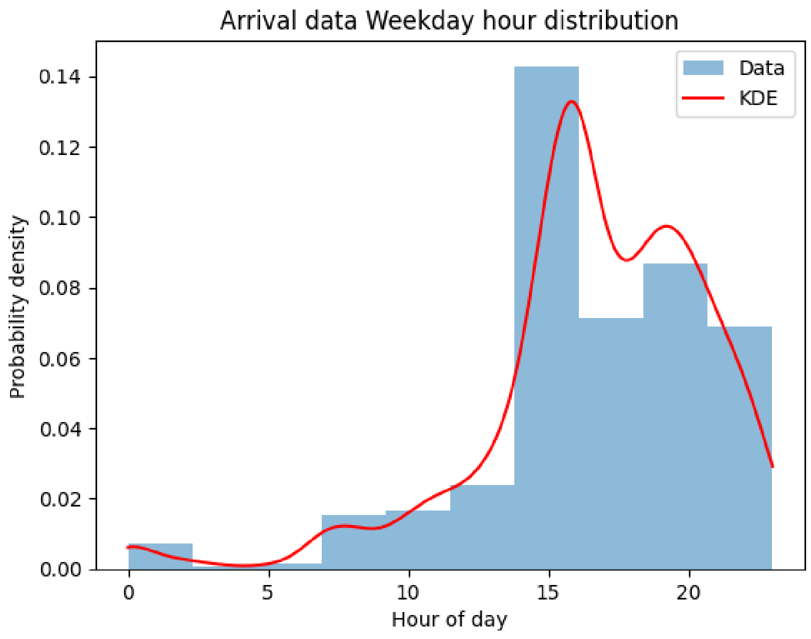


Figure 5 - Arrival time distribution during weekdays

The best fit for arrival times weekends was found by using the Gaussian KDE method:

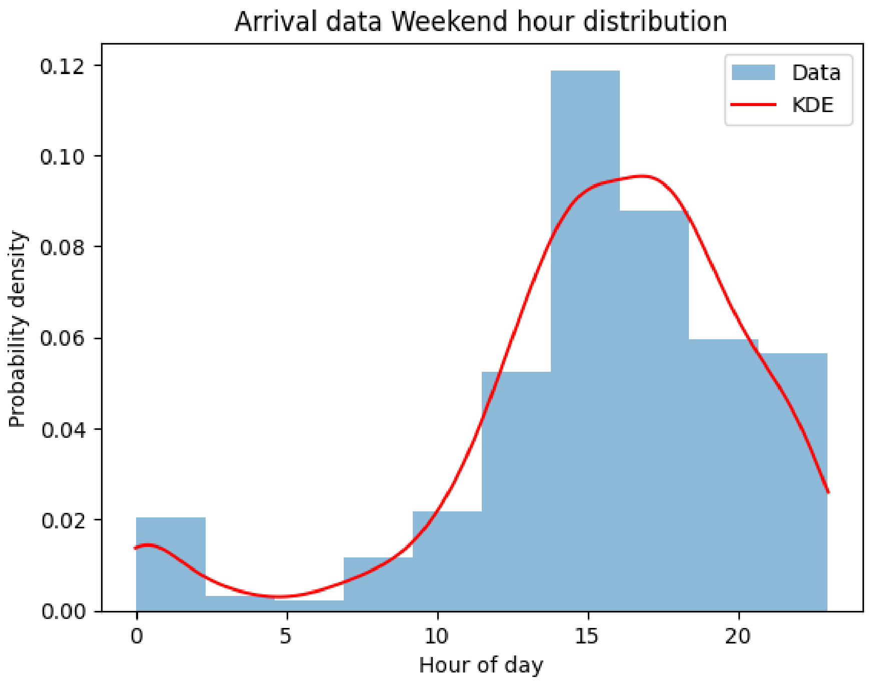


Figure 6 - - Arrival time distribution during weekends

Wait time – Time until departure:

The best fit for Wait time weekday until departure (Time until Departure – Tud) was found by using the Gaussian KDE method:

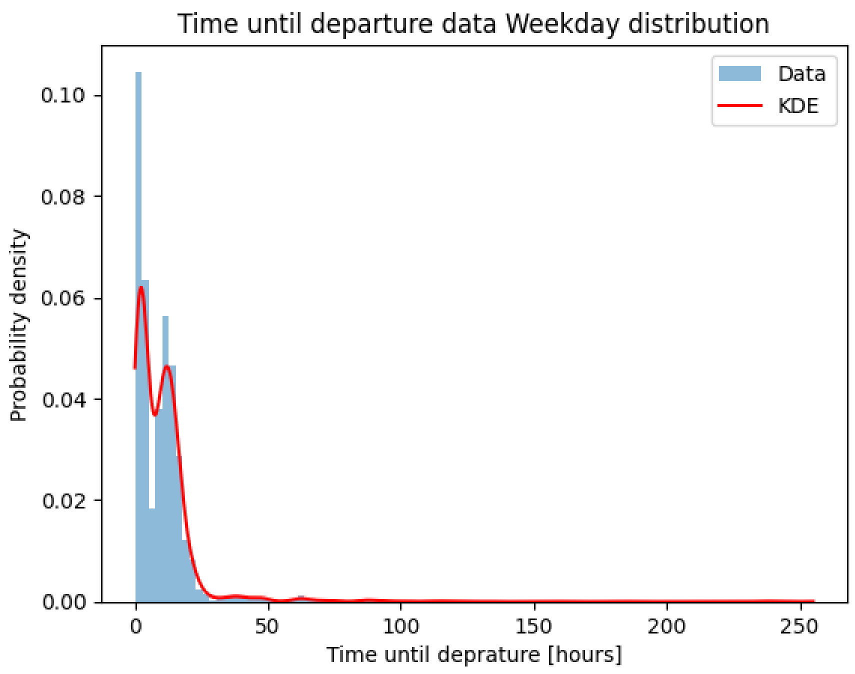


Figure 7 -Time until departure distribution during weekdays

The best fit for Wait time weekend until departure was found by using the Gaussian KDE method:

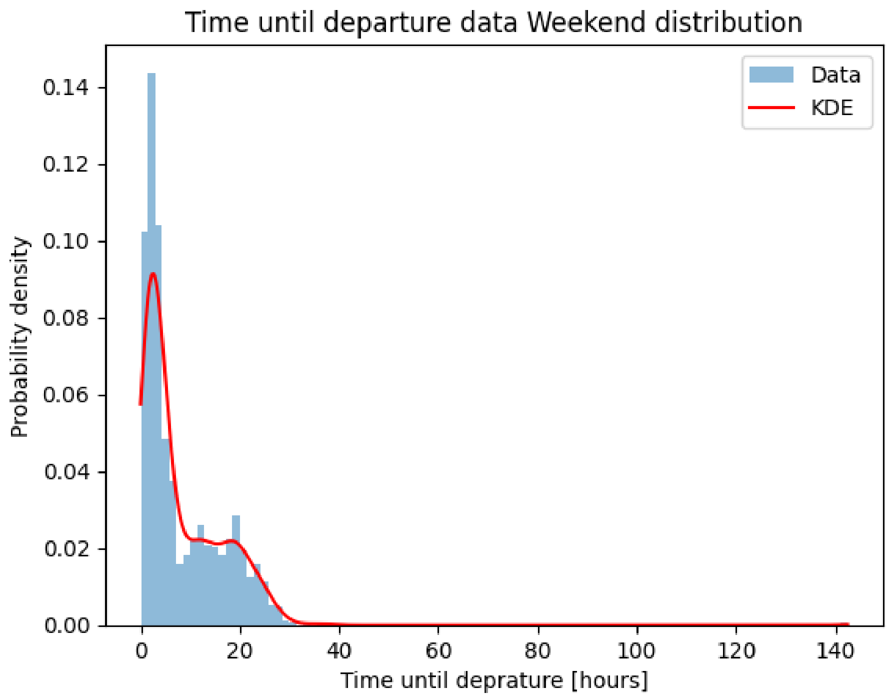
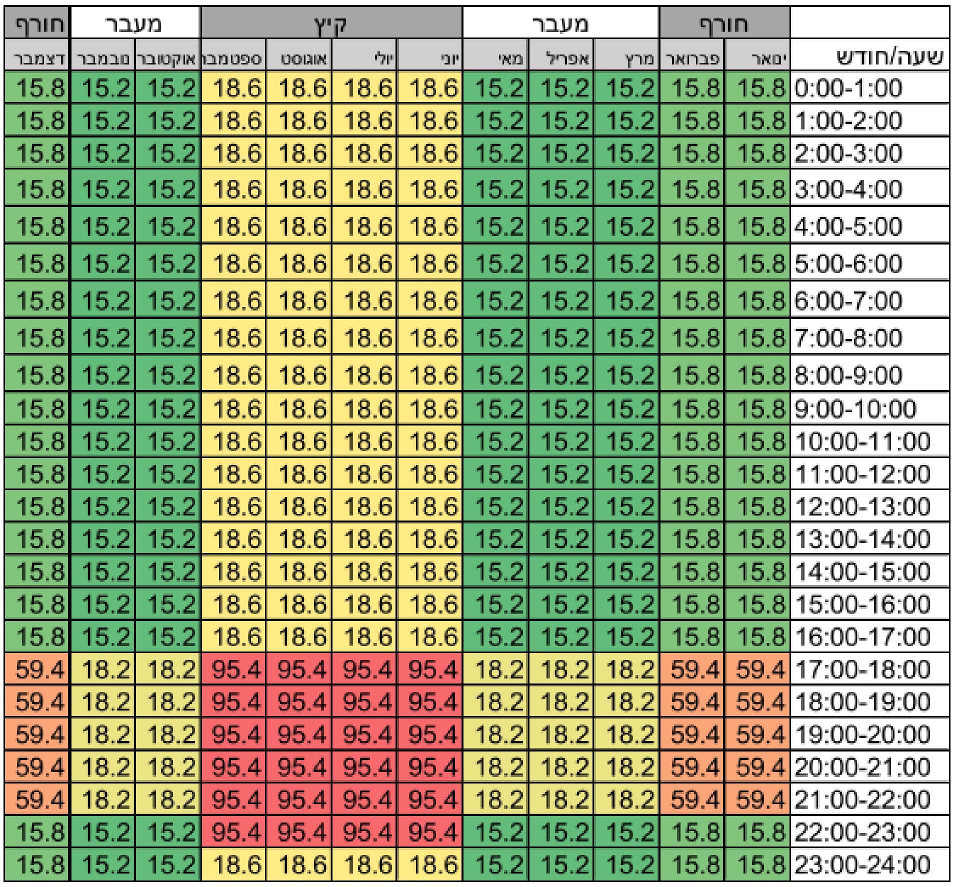


Figure 8 -Arrival time distribution during weekend

2. **Electricity price GRID:**

The pricing of electricity was taken from [3] paper and shown in the Figure 9- Electricity pricing below, the pricing are depends on the season, and time of day.



חורף	מעבר		קיץ				מעבר			חורף		
דצמבר	נובמבר	אוקטובר	ספטמבר	אוגוסט	יולי	יוני	מאי	אפריל	מרץ	פברואר	ינואר	שעה/חודש
15.8	15.2	15.2	18.6	18.6	18.6	18.6	15.2	15.2	15.2	15.8	15.8	0:00-1:00
15.8	15.2	15.2	18.6	18.6	18.6	18.6	15.2	15.2	15.2	15.8	15.8	1:00-2:00
15.8	15.2	15.2	18.6	18.6	18.6	18.6	15.2	15.2	15.2	15.8	15.8	2:00-3:00
15.8	15.2	15.2	18.6	18.6	18.6	18.6	15.2	15.2	15.2	15.8	15.8	3:00-4:00
15.8	15.2	15.2	18.6	18.6	18.6	18.6	15.2	15.2	15.2	15.8	15.8	4:00-5:00
15.8	15.2	15.2	18.6	18.6	18.6	18.6	15.2	15.2	15.2	15.8	15.8	5:00-6:00
15.8	15.2	15.2	18.6	18.6	18.6	18.6	15.2	15.2	15.2	15.8	15.8	6:00-7:00
15.8	15.2	15.2	18.6	18.6	18.6	18.6	15.2	15.2	15.2	15.8	15.8	7:00-8:00
15.8	15.2	15.2	18.6	18.6	18.6	18.6	15.2	15.2	15.2	15.8	15.8	8:00-9:00
15.8	15.2	15.2	18.6	18.6	18.6	18.6	15.2	15.2	15.2	15.8	15.8	9:00-10:00
15.8	15.2	15.2	18.6	18.6	18.6	18.6	15.2	15.2	15.2	15.8	15.8	10:00-11:00
15.8	15.2	15.2	18.6	18.6	18.6	18.6	15.2	15.2	15.2	15.8	15.8	11:00-12:00
15.8	15.2	15.2	18.6	18.6	18.6	18.6	15.2	15.2	15.2	15.8	15.8	12:00-13:00
15.8	15.2	15.2	18.6	18.6	18.6	18.6	15.2	15.2	15.2	15.8	15.8	13:00-14:00
15.8	15.2	15.2	18.6	18.6	18.6	18.6	15.2	15.2	15.2	15.8	15.8	14:00-15:00
15.8	15.2	15.2	18.6	18.6	18.6	18.6	15.2	15.2	15.2	15.8	15.8	15:00-16:00
15.8	15.2	15.2	18.6	18.6	18.6	18.6	15.2	15.2	15.2	15.8	15.8	16:00-17:00
59.4	18.2	18.2	95.4	95.4	95.4	95.4	18.2	18.2	18.2	59.4	59.4	17:00-18:00
59.4	18.2	18.2	95.4	95.4	95.4	95.4	18.2	18.2	18.2	59.4	59.4	18:00-19:00
59.4	18.2	18.2	95.4	95.4	95.4	95.4	18.2	18.2	18.2	59.4	59.4	19:00-20:00
59.4	18.2	18.2	95.4	95.4	95.4	95.4	18.2	18.2	18.2	59.4	59.4	20:00-21:00
59.4	18.2	18.2	95.4	95.4	95.4	95.4	18.2	18.2	18.2	59.4	59.4	21:00-22:00
15.8	15.2	15.2	95.4	95.4	95.4	95.4	15.2	15.2	15.2	15.8	15.8	22:00-23:00
15.8	15.2	15.2	18.6	18.6	18.6	18.6	15.2	15.2	15.2	15.8	15.8	23:00-24:00

Figure 9- Electricity pricing

3. **Electricity price PV:**

PV electricity price for PV was taken from IEC pricing for PV systems.

3.1. The price of electricity from the grid is 0.48 NIS for systems with a capacity of up to 15[KW]

3.2. The price of electricity from the grid is 0.45 NIS for systems with capacity from 15[KW] up to 100[KW].

4. **Building electricity meter:**

4.1. Solar power input

Solar power input data was sourced from [10]. Utilizing solar power as a renewable energy source can increase the available power from the grid, enhancing power utility without overloading the grid and improving electricity pricing throughout the day. The dataset used in this study spans from January 5, 2023, at 00:00:00 (midnight) to December 5, 2023, at 23:50:00. This data originates from the Beit Dagan radiation station, a reputable source for environmental radiation data under the Israeli meteorological service.
The data was preprocessed and analyzed to determine the mean power output from the PV system. Date values were converted to DateTime format, and the data was grouped by the hour. Direct radiation values were aggregated to ascertain the mean power output of the PV system. As illustrated in Figure 10 - Radiation vs hour of day radiation correlates with daylight hours. The peak radiation in this dataset occurs at 13:00 .Solar power output is influenced by irradiation, which measures the power received from the sun per unit area, expressed in Watts per square meter (W/m^2) [11].

$$E = A \cdot r \cdot H \cdot PR$$

Equation 2- Energy calculation form PV

Where:

- E is energy [KWh]
- A is the total solar panel area $[m^2]$
- r is solar panel efficiency (%) is 20% [12] :
- H is the average solar irradiation $[\frac{W}{m^2}]$
- PR is the Performance Ratio, the coefficient for losses, in this case, the value of PR is 0.75 for a default value.

To calculate the mean Power per hour the following formula is used: $P = \frac{E}{T}$

To calculate Solar power the following parameters were assumed:

$$PV_{area} = 50\ [m^2]$$

$$PV_{efficany} = 0.2\ [\%]$$

$$Perforamce\ ratio = 0.75\ [\%]$$

And Equation 2- Energy calculation form PV was used to calculate the power output.

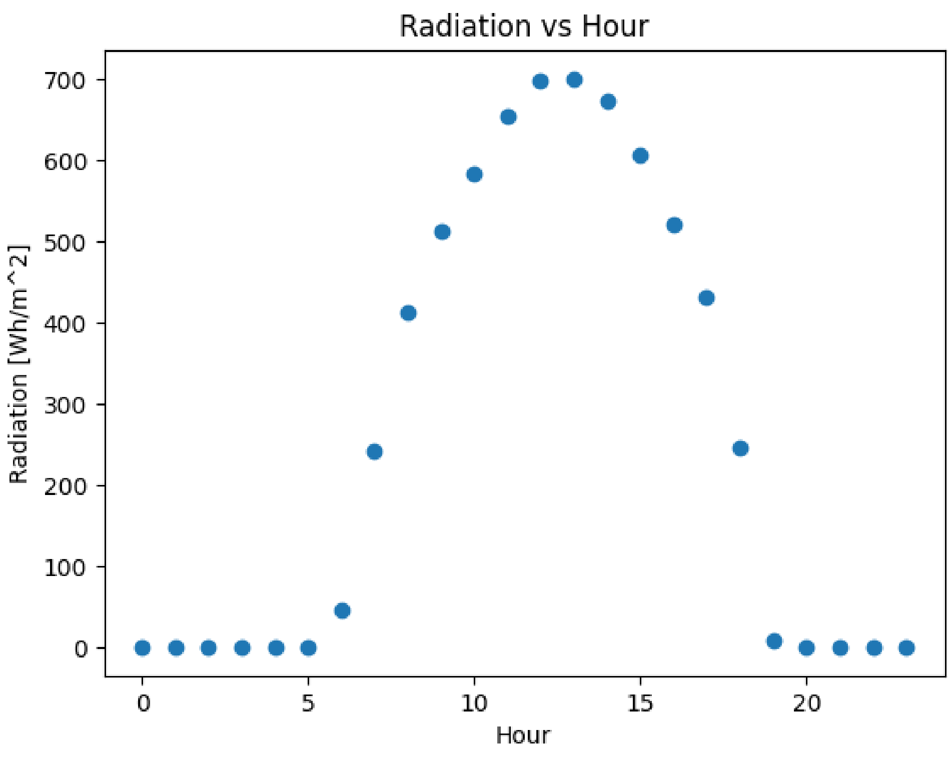


Figure 10 - Radiation vs hour of day

In Figure 11 - Power input from PV to building during day the power input from PV system to the building is shown

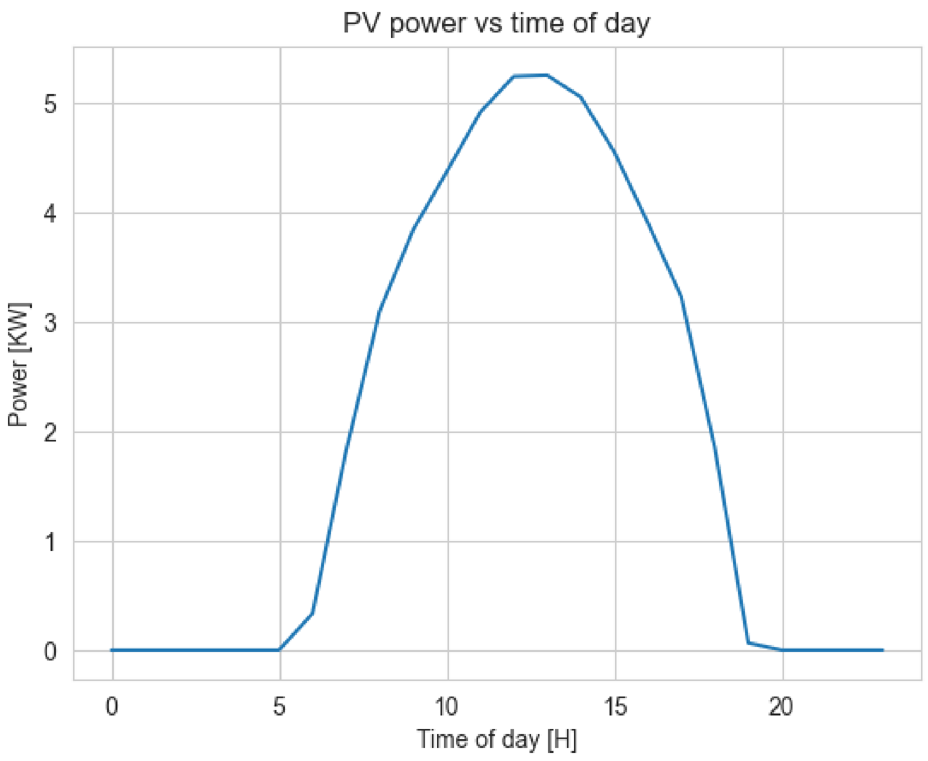


Figure 11 - Power input from PV to building during day

4.2. Grid power input

Grid power is considered as the maximum power allowed of the typical transformer, as noted in the [3]distribution transformer rated power is 630 [KVA] which supplies electricity for 126 households, the real power allowed is 630 * power factor, power factor assumed to be 0.85 whereas power factor varies from 0.7 to 0.9 in residential buildings [13].

4.3. Households' electricity Load:

Households load data was taken from [14] , Data preprocessing involves preparing raw data for further analysis by cleaning, transforming, and organizing it. In this case, the first step is to drop any rows or records that contain missing or null values (NA values). Subsequently, a computation is carried out to represent the active energy consumed every minute by household electrical equipment that isn't measured in sub-metering 1, 2, or 3. The formula employed is as follows:

$$Global\ active\ power = active_{power} \cdot \frac{1000}{60} - sub_{metering_1} - sub_{metering_2} - sub_{metering_2}$$

Equation 3- Global Active power calculation

The active power is multiplied by 1,000 to convert it from kilowatts (kW) to watts (W). It is then divided by 60 to represent the value in watt-hours per minute. From this total, the values of sub_metering_1, sub_metering_2, and sub_metering_3 are subtracted. These values correspond to the energy consumption measured in different rooms of the house. This subtraction isolates the active energy consumed by electrical equipment not covered by these sub-metering's.
The final value indicates the active energy consumed every minute, in watt-hours, by the household's electrical equipment not tracked by sub-metering 1, 2, and 3. After these calculations, the Global active power is converted back to kW to represent the total power consumed throughout the day.
To simulate the power consumption of multiple households, the vector containing household power data is multiplied by the number of households, estimated power consumption of households is presented in Figure 12 - Households electricity load during day

4.4. Surplus Power:

Surplus power is calculated from all the power sources that mentioned above and calculated by the following Equation 4 - surplus power calculation and it's presented in Figure 13- Surplus power profile during day :

$$Surplus\ power = PWR_{grid} + PWR_{PV} - PWR_{consmption\ Household}$$

Equation 4 - surplus power calculation

Where:

- PWR_{grid} is the input power from the grid into building.
- PWR_{PV} is the input power from the PV system into building.
- $PWR_{consumption\ Household}$ is the power that is consumed by households.

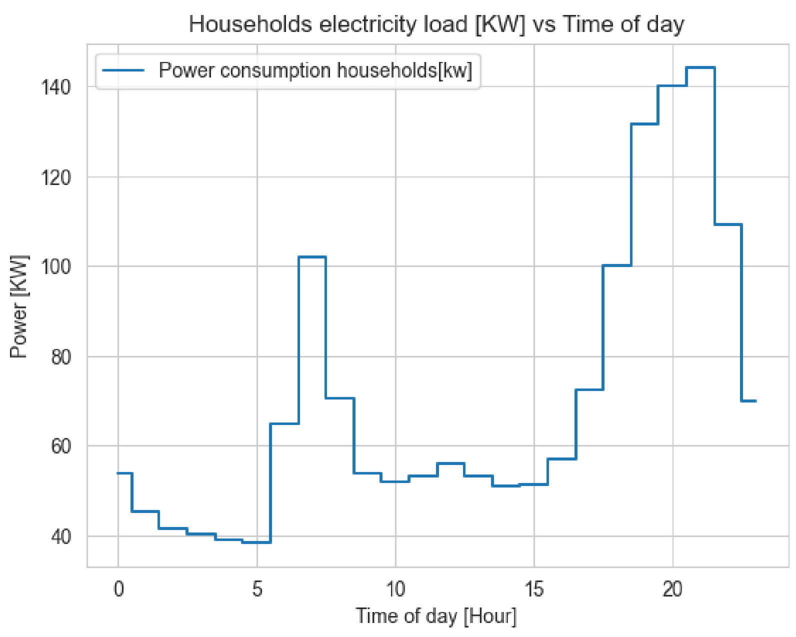


Figure 12 - Households electricity load during day

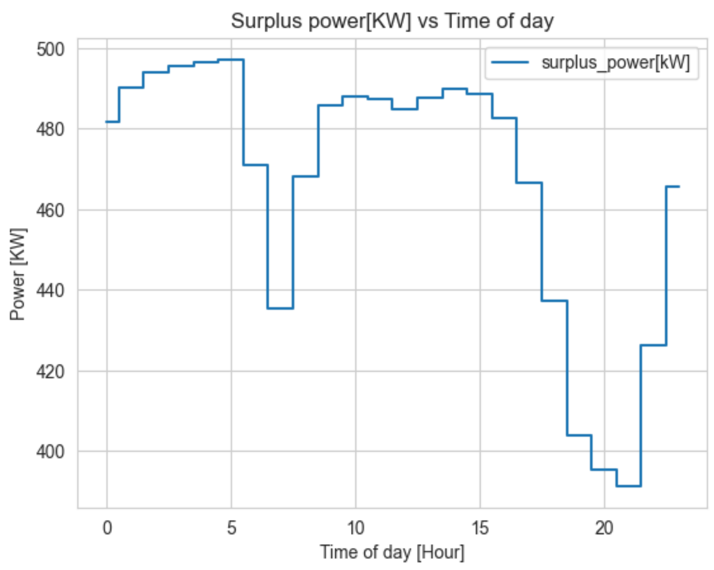


Figure 13- Surplus power profile during day

5. EV:

5.1. EV ID

EV ID aligned to User-ID from user table for tables merging on this key.

The following data fields were taken from the 10th most common EV in Israel Icar magazine [15], the ninth and tenth cars were dropped since those cars do not support charging in 11 [KW], the generated dataset contains the following fields are: Brand and model, EV efficiency, EV Battery capacity. EV characteristics as taken from [16], EVs characteristics includes Battery capacity, and EV efficiency

5.2. EV SOC

was assumed to be distributed as uniform distribution and randomly created with uniform distribution, the minimal SOC value was set to 15% and the maximal value was set to 50%.

5.3. Estimated Driving Distance EDD:

The driving distance is utilized to calculate the SOC (State of Charge). By randomly selecting a car and using its energy consumption, the SOC can be determined.
Drawing from the simulation parameters of [3] and data from the Israeli Central Bureau of Statistics, the estimated driving distance is set at 51 km per day. While the type of distribution, standard deviations, or variance was not provided, it was assumed that the standard deviation (STD) is 11 km, and the driving distance follows a normal distribution.

5.4. Needed on Next Departure ENonD:

The energy required on the next departure is determined based on the estimated driving distance and the efficiency of the EV, as each EV has a unique energy consumption per kilometer.

$$ENonD = \frac{EED[km] \cdot EV_{efficancy} \left[\frac{Wh}{km}\right]}{1000}$$

Equation 5 - Calculation of Energy needed on next departure

6. Data preprocessing for algorithms input:

During data preprocessing, before the data was loaded into the algorithms, it was discovered that some entries did not align with the simulation requirements, EVs data should contain 189 EVs to simulate extreme case were all families in building have EV and need to charge their EV during the next 24 hours, the reason for that the sum of arrival time and TUD was larger than the time frame. To prevent that case a large data set was created with 2000 evs, and data stratifying was used to maintain dataset statistical characteristics.
In data preprocessing the following operations were done:
EVs dataset:

- TUD values were rounded.
- All Evs with Arrival time + TUD above 24 hours were dropped.
- Non relevant columns were dropped and only the following columns were exported 'Arrival time', 'TuD', 'Battery capacity', 'ENonD', 'SOC'.

Building power dataset:

- PV power was converted to KW.
- Columns names renamed.
- Surplus power was calculated from residual power of grid and PV input and household consumption.

7. **Power need per hour:**

To confirm that the required total power is available during the charging session, ENonD was aggregated by the hour of arrival time. It is evident that the power needed over a 24-hour period does not exceed the surplus power, indicating that the algorithm can effectively address the problem. Energy needed per hour is shown in Figure 14 -Energy needed per hour, the blue line represents the energy needed per arrival hour, and the orange line represents the surplus power.

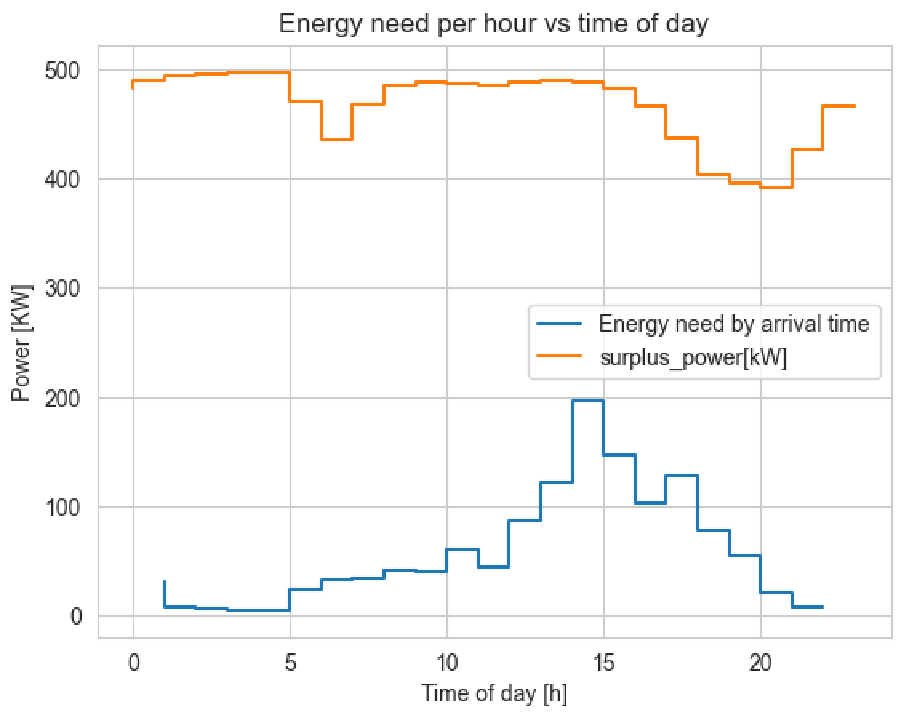


Figure 14 -Energy needed per hour

8.3.3 Algorithms research:

Reinforcement Learning (RL) is a type of machine learning where an agent learns to make decisions by interacting with its environment. In RL, an agent takes actions in an environment to achieve a goal. The agent receives rewards or penalties in return for its actions and aims to maximize the total reward.[17]

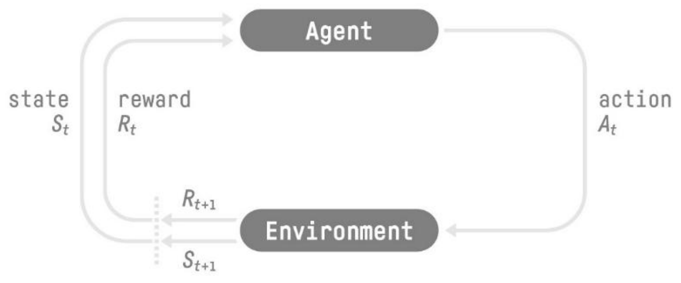


Figure 15- The RL process

The RL process is a loop that outputs a sequence of state, action, reward, and next state, RL process is shown in Figure 15- The RL process and the picture was taken from [17].

RL process steps in general:
1. the agent observes the current state of the environment.

2. The agent takes action.

3. The environment transitions to a new state.

4. The agent receives a reward or penalty based on his actions.

5. The agent updates its knowledge with the new experience.

6. The process is repeated until the agent achieves the goal, or a maximum number of steps is reached.

The key components of RL are:

- Agent: The decision-maker or the learner.
- Environment: Everything outside the agent.
- Action (a): A set of all possible moves the agent can make.
- State (S): A scenario that the agent can observe.
- Reward (R): Feedback from the environment.
- Policy (π): The strategy that the agent employs to determine the next action based on the current state.
- Value (V): The expected long-term return with discount, as opposed to the short-term reward R. Vπ(s) is defined as the expected long-term return of the current state under policy π.
- Q-value or action-value (Q): Q-value is similar to Value, except that it takes an extra parameter, the current action a. $Q_{\pi}\ (s, a)$ refers to the long-term return of the current state s, taking action a under policy π.

The RL process can be summarized by the following Equation 6- transition dynamics of an environment ,this equation is the transition dynamics of an environment:

$$S_{t+1} = f(S_t, A_t)$$

Equation 6- transition dynamics of an environment

The goal of the agent is to maximize the expected cumulative reward. The cumulative reward from time step t is given by the following Equation 7 - Cumulative reward

$$G_t = \sum_{k=0}^{\infty} \gamma^{\mathrm{k}} \cdot R_{t+k+1}$$

Equation 7 - Cumulative reward

Where:

G_t is the cumulative reward.

R_{t+1} is the reward as the next time step.

γ is the discount factor.

Observations or States represent the information that our agent receives from the environment.
Observation: This is the agent's perception of the environment at a given time step. It's the information the agent receives from the environment after taking an action. However, an observation does not necessarily provide complete information about the state of the environment. For example, in a game of poker, an agent's observation might include its own cards and the bets of other players, but not the cards of other players.
State: This is a complete description of the condition of the environment at a given time. It represents all the information necessary to determine what happens next in the environment. In fully observable environments, the state is fully determined by observation. However, in partially observable environments, the agent might not have access to the complete state of the environment.
In the context of this project, the agent has access to the entire state space, which includes parameters such as Electric Vehicle (EV) characteristics, power limits, and the time of day. In this scenario, there is no hidden information, meaning that the agent has complete visibility into all aspects of the environment.

Reinforcement Learning (RL) accommodates various types of tasks, including Episodic and Continuing tasks. In Episodic tasks, there are distinct starting and ending points. For example, an electric vehicle (EV) charging system might operate on an episodic task basis, beginning at time 0 and concluding at time 24 to schedule EV charging. On the other hand, Continuing tasks do not have a terminal state. In these tasks, the agent continuously interacts with the environment, selecting the optimal action for the subsequent state. An example of this could be an agent interacting with a time-series environment, where the agent's actions are based on ongoing observations and do not have a predefined endpoint.

The exploration/exploitation trade-off in reinforcement learning refers to the dilemma an agent faces when deciding between exploiting current knowledge for immediate reward or exploring new actions for potential future gains. Exploration, crucial in early learning stages, helps the agent discover rewarding actions, while exploitation allows the agent to maximize rewards based on existing knowledge.

However, balancing exploration and exploitation is challenging as they can be in conflict. Over-exploration may cause missed immediate rewards, while excessive exploitation can overlook potentially better actions. Strategies like ε-greedy, SoftMax, and upper confidence bound (UCB) address this trade-off by balancing immediate rewards and potential future gains. The optimal balance often depends on the problem's specific characteristics, such as environment complexity and task time horizon.

Value-based methods in reinforcement learning focus on determining the value of each state or action, typically through a value function. The goal is to find the optimal value function, which provides the expected long-term return for each state or action. The policy is then derived from this value function, usually by choosing the action that maximizes the value function at each state. On the other hand, policy-based methods directly learn the policy function without the need for a value function. The policy function, in this case, is mapping from states to actions. These methods can handle high-dimensional action spaces and can learn stochastic policies. They are typically implemented using policy gradient methods, where the policy is updated iteratively in the direction that improves the expected return.

Monte Carlo (MC) and Temporal Difference (TD) Learning are two different strategies on how to train our value function or our policy function. Both of them use experience to solve the RL problem. MC methods in reinforcement learning involve learning from complete episodes of interaction with the environment. They estimate the value of a state or action by averaging the returns observed after visiting that state or action. MC methods can only be applied to episodic tasks, where all episodes terminate, as they require the final outcome of an episode to calculate the returns. TD methods, on the other hand, are a combination of Monte Carlo ideas and dynamic programming ideas. They learn directly from episodes of experience without requiring the final outcome of an episode like MC methods. TD methods update estimates based on other learned estimates, bootstrapping from the current estimate of the value function. This allows them to learn online and update their estimates at each time step, making them suitable for continuous tasks.

Deep Reinforcement learning

Deep Reinforcement Learning (DRL) is a subfield of artificial intelligence that combines deep learning and reinforcement learning. It involves the use of neural networks to approximate functions in reinforcement learning problems, enabling the handling of large, complex state spaces. In traditional reinforcement learning, an agent learns to make decisions by interacting with its environment. It learns a policy, which is a mapping from states to actions, that maximizes the expected cumulative reward. However, traditional reinforcement learning methods often struggle with environments that have large state or action spaces, or where the state or action spaces are continuous rather than discrete.[18]

Deep reinforcement learning addresses these challenges by using deep learning techniques. In particular, it uses neural networks to represent the policy or the value function, which is a function that estimates the expected cumulative reward for each state or state-action pair. The neural network can learn to make accurate estimates even in environments with large, complex, or continuous state or action spaces.

Deep reinforcement learning has been used to achieve state-of-the-art results in a variety of challenging tasks. For example, it has been used to train agents that can play video games at a superhuman level, and to control robots in complex, real-world environments.

However, deep reinforcement learning also has its challenges. Training deep reinforcement learning agents can be computationally intensive and may require a large amount of data. Furthermore, the training process can be unstable and sensitive to the choice of hyperparameters.

Advantage Actor Critic (A2C) method:

The Actor-Critic method, also known as Advantage Actor-Critic (A2C), is a type of policy gradient method in reinforcement learning. It combines the strengths of both value-based and policy-based methods. The "Actor" refers to the policy part of the model, which proposes a set of possible actions given a state. The "Critic" evaluates the action taken by the actor based on the value function.[19]

The A2C method is used to reduce variance which received in Reinforce algorithm, stochasticity of environment and stochasticity of the policy occur as result different trajectories can lead to different returns, which can lead to high variance. The solution to reducing the variance of the Reinforce algorithm and training our agent faster and better is to use a combination of Policy-Based and Value-Based methods: the Actor-Critic method. In architecture the architecture of the neural networks is shown.

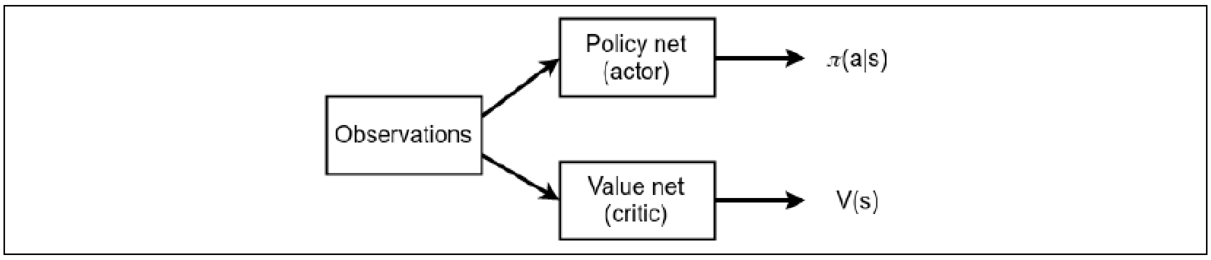


Figure 16 - A2C neural networks architecture

A2C definitions and notations:

1. S: State space

2. A: Action space

3. r : Reward function

4. γ : Discount factor

5. $\pi_\theta(a|s)$: Policy function, π: $S \to A$

6. $V_\phi{}^\pi(s)$: State-value function under policy π, the expected return from state s – **Critic**

7. $Q^\pi(s, a)$: Action-value function under policy π, the expected return from taking action a in state s - **Actor**

8. $A^{\pi}(s,a)$: Advantage function under policy π, the advantage of taking action a in state s over the average action

Advantage Function:

The advantage function $A^{\pi}(s,a)$ is the core concept of A2C. It is defined as the difference between the action-value function $Q^{\pi}(s,a)$ and the state-value function $V^{\pi}(s)$:

$$A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s)$$

Equation 8 - Advantage function

By definition, the advantage function measures how much better or worse an action a is compared to the average action in state s under policy π. If $A^{\pi}(s,a) > 0$, action a is considered better than the average, while if $A^{\pi}(s,a) < 0$, action a is considered worse.

A2C algorithm:

The A2C algorithm iteratively improves the policy by performing the following steps:

1. **Actor Step**: The Actor samples an action a from the current policy π given the current state s.

2. **Environment Step**: The action a is executed in the environment, resulting in the next state s' and the reward r.

3. **Critic Step**: The Critic evaluates the state-value function $V^{\pi}(s)$ and the advantage function $A^{\pi}(s,a)$.

4. **Update Step**: The Actor updates its policy using the advantage function, and the Critic updates the state-value function using the reward and the estimated value of the next state.
The policy update in A2C is done using policy gradients, a method that adjusts the policy parameters in the direction that maximizes expected return. In the A2C algorithm, the policy gradient is modified to use the advantage function $A^{\pi}(s,a)$ instead of the action-value function $Q^{\pi}(s,a)$, which reduces the variance of the gradient estimate and improves learning efficiency. The policy update step can be written as follows:

$$\theta \leftarrow \theta + \alpha \nabla_{\theta} log \pi(a|s;\theta) A^{\pi}(s,a)$$

Equation 9 - Policy update function

where:
- θ are the policy parameters,

- α is the learning rate,

- $\nabla_{\theta}\ log\ \pi(a|s;\theta)$ is the policy gradient, and

- $A^{\pi}(s,a)$ is the advantage function.

The Critic updates the state-value function using TD learning, a method that bootstraps the value function using the reward and the estimated value of the next state. The state-value function update step can be written as follows:

$$V(s) \leftarrow V(s) + \beta\,(r + \gamma\, V(s') - V(s))$$

Equation 10- state-value function update

where:
- β is the learning rate for the Critic,

- r is the reward,

- γ is the discount factor,

- $V(s')$ is the estimated value of the next state

- $V(s)$ is the current estimated value of the state.

Actor-Critic methods, especially the Advantage Actor-Critic (A2C) variant, have proven to be effective and versatile approaches to reinforcement learning tasks. By combining the strengths of both policy-based and value-based methods and introducing the concept of an advantage function, A2C offers a balanced and efficient approach towards reinforcement learning. Future work could explore further enhancements to the Actor-Critic framework, such as improving the estimation of the advantage function or integrating function approximation methods for handling large state and action spaces.

Proximal Policy Optimization (PPO):

In the rapidly evolving field of reinforcement learning, Proximal Policy Optimization (PPO) has emerged as a leading algorithm that combines the benefits of policy gradient methods with the efficiency and robustness of advanced optimization techniques. The PPO algorithm, proposed by OpenAI, is known for its simplicity, ease of implementation, and effectiveness across a wide range of tasks. PPO is a policy optimization method designed to address these challenges. It strikes a balance between sample complexity (how many samples it needs to learn), ease of implementation, and computational cost. PPO optimizes the policy in a more stable way by limiting the step size in policy space, which leads to more robust and efficient learning.[20]

PPO definitions and notations:

1. S: State space

2. A: Action space

3. r : Reward function

4. γ : Discount factor

5. π: Policy function parameterized by $\theta, \pi_\theta : S \rightarrow A$

6. $V_\phi(s)$: State-value function under policy π_θ , parameterized by ϕ, the expected return from state s.

PPO objective function:

PPO introduces a novel objective function for policy optimization. Instead of directly optimizing the expected return, PPO optimizes a surrogate objective function that encourages exploration and limits the change in policy at each update.

The PPO objective function is defined as:

$$L^{CLIP}(\theta) \;=\; \hat{E}_t\; [\, min(\, r_t(\theta)\hat{A}_t\,, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\; \hat{A}_t]$$

Equation 11 - PPO objective function

Where:

- $r_t(\theta) = r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ is the likelihood ratio,
- $\widehat{A_t}$ is the estimated advantage at time t,
- $CLIP(r_t(\theta), 1-\epsilon, 1+\epsilon)$ is a clipped version of the likelihood ratio, which limits the step size in policy space,
- $\epsilon\ is\ a\ hyperparameter\ that\ controls\ the\ cliping\ range$

PPO algorithm:

The PPO algorithm iteratively improves the policy by performing the following steps:

1. **Data collection:** The agent interacts with the environment following the current policy π_θ and collects trajectory data (state, action, reward).
2. **Advantage Estimation**: The agent computes the advantage function $\widehat{A_t}$ using Generalized Advantage Estimation (GAE)
3. **Policy Optimization**: The agent optimizes the policy π_θ by maximizing the PPO objective function using gradient ascent. The policy parameters θ are updated using the Adam optimizer.
4. **Value Function Update**: The agent updates the value function parameters ϕ using the TD error as the loss function.

The policy optimization step can be written as follows:

$$\theta \leftarrow \theta \;+ \alpha\; \nabla_\theta\; L^{CLIP}(\theta)$$

Equation 12 - PPO Policy update function

Where:

- α is the learning rate,
- $\nabla_\theta L^{CLIP}(\theta)$ is the gradient of the PPO objective function.

The value function update step can be written as follows:

$$\phi \leftarrow \phi \;- \beta\; \nabla_\phi\; L^{VF}(\phi)$$

Equation 13 - PPO value function update

Where:

- β is the learning rate for the value function,
- $\nabla_{\phi} L^{VF}(\phi)$ the gradient of the value function loss

PPO has proven to be a highly effective reinforcement learning algorithm that balances the trade-off between sample complexity, ease of implementation, and computational cost. It achieves this by introducing a novel objective function that limits the change in policy at each update, leading to more stable and efficient learning. Future research could explore further enhancements to the PPO algorithm, such as incorporating more sophisticated exploration strategies or developing more efficient optimization methods.

8.3.4 Reinforcement Learning environment Development:

To execute advanced reinforcement learning strategies on a custom environment, the Stable Baselines3 toolkit [21], built on PyTorch, was used. It offers a comprehensive suite of reinforcement learning algorithms. The custom environment class primarily comprises three essential functions: `init`, `reset`, and `step`. By leveraging these functions, the algorithms can simulate and optimize the EV charging process during their training phase. Notably, the structure of this environment is based on the Gym infrastructure, a project from the Farma foundation, which was developed by OpenAI.

Two distinct environments were constructed: `EVChargingEnv_continuous` and `EVChargingEnv_discrete`. The motivation behind developing these two separate environments was to assess the impact of reward structures on agent training and prediction processes, as well as to facilitate the fine-tuning of rewards.

Both `EVChargingEnv_continuous` and `EVChargingEnv_discrete` are subclasses of the Gym's `Env` class, a widely accepted standard for reinforcement learning environments. The purpose of this environment is to simulate a system where multiple electric vehicles are charging. The primary objective is the optimal management of the charging process, which initiates at time 0 of a day and concludes at 24 hours. Throughout this process, the system charges the EVs based on the actions prescribed by the agent. At every step within this environment, the agent receives observations composed of three elements: the EVs and their specifics, surplus power, and the time of day. Environment code attached to this paper and can be found in appendix 12.2

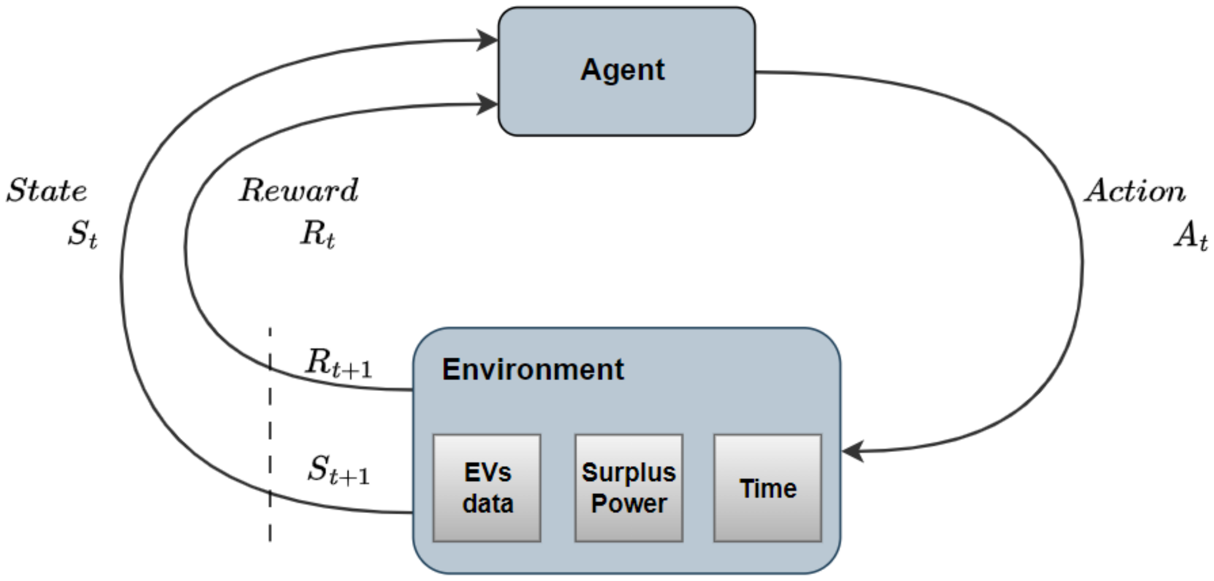


Figure 17 - The RL process in the developed environment

Initialization (init method):

The environment is initialized using a data frame of EVs and a surplus power vector, which indicates the power available for charging the EVs. The action space is a MultiDiscrete space, with each EV having three potential actions. These actions correspond to charging rates of (0, 3.7, 11) [KW]. The MultiDiscrete space was selected because it allows each EV to receive a unique action based on the agent's decision. In contrast, other action spaces would result in all EVs receiving identical actions, which might hinder the charging process. Such uniform actions could lead to conflicts arising from not adhering to system constraints. These constraints encompass charging exclusively within the permissible timeframe (i.e., when the EV is present in the building), charging below the maximum allowed load, and not exceeding the desired state of charge (SOC) threshold. The observation space is structured as a dictionary and includes the keys 'evs', 'surplus power', and 'time'.

- ‘evs’ is the dataset of all evs that loaded to class and contains the ‘arrival time’, ‘TUD’, ‘Battery Capacity’, ‘ENonD’ and ‘SOC’.
- ‘Surplus Power’ is the data vector of max power that is allowed in each hour during the day.
- ‘time’ its environment variable and it is the time of day that represents the current time to the agent.

Also, the init function contains the following variables:

- terminated: Termination refers to the episode ending after reaching a terminal state that is defined as part of the environment definition.
- truncated: Truncation refers to the episode ending after an externally defined condition (that is outside the scope of the Markov Decision Process). Not in use in the environment.
- Total reward: stores the cumulative rewards of each episode.

Reset (reset method):

This method resets the environment to its initial state. It deep copies the initial state of the EVs, resets the time to 0, resets terminated to False and total_reward to 0, and constructs an initial observation from these values.

Step (step method):

In this method, actions dictate how the EVs are charged. The agent outputs actions corresponding to three charging levels: 0, 3.7, and 11 [KW]. These levels represent widely accepted charging rates for EVs, and the majority of the market's EVs support these rates. During each step, the state of charge (SOC) values are adjusted based on the charging power actions. Also, the Time Until departure (TUD) is updated, decreasing by one hour with every step. By the conclusion of each step, the environment returns its observation, reward, termination status, truncation, and additional information. If the time reaches 24, the episode concludes.

Reward shaping and design employ a spectrum of values, ranging from negative to zero, to guide agent towards desirable behavior. The design of rewards, especially the use of negative to zero rewards, plays a crucial role in shaping the behavior of the agent. By using negative rewards, the agent can avoid undesirable behaviors, negative rewards act as penalties for actions that are undesirable or harmful. By assigning a negative reward to such actions, the agent is discouraged from repeating them. Using negative rewards found more efficient and helped algorithms to converge faster to enhance agent exploration zero rewards given and it was not overly penalized for those actions and can explore how to charge EVs. Also the technique of negative rewards used by [5] to increase the algorithms performance.

For each charging action, a reward is assigned based on the following constraints within the Discrete environment:

1. If a charging action occurs during a prohibited time and is above zero, the agent receives a penalty.

2. If a charging action is executed during a prohibited time but is equal to zero, the agent is awarded a positive reward.

3. When a charging action takes place during an allowed time, the reward is set to zero.

Charge Time reward:

$$charge\_time\_reward = \begin{cases} -Constant, & if\big((arrival\ time < time) or (TuD \le 0)\big)\ and\ (charge\ action > 0) \\ Constant, & if\ (arrival\ time < time)\ or\ (TuD \le 0)\ and\ Ccharge\ action = 0) \\ 0\,, & if\ (arrival\ time \ge time) and\ (TuD > 0)\ and\ (charge\ action > 0) \end{cases}$$

Equation 14 - Charge time reward function

In the continues environment the same conditions were used to give penalty on charge out of the allowed hours.

Power reward:
In the discreate environment power reward has two states and it given as penalty or zero in case the total charge power is below surplus power zero reward is given, and in case the total charge power reached above the allowed power a negative reward is given.

$$power\ reward = \begin{cases} -Constant, & \sum_{k=0}^{n\_evs} charge\ action\_k > power\ allowed \\ 0, & \sum_{k=0}^{n\_evs} charge\ action\_k \le power\ allowed \end{cases}$$

Equation 15 - Power reward discrete function

In the continues environment Power reward function calculates the ratio of total charge actions from power limit, if the there is a negative deviation, where the total charge power is above the surplus power a negative reward is given in case the total charge power is below the surplus power clip function is applied and the value of the reward of zero. In Figure 18 - Power reward continues function is shown, the exponent constant is 5 and the penalty constant is 1.

$$power\ reward = 1 - e^{\left(-\exp const\ pwr \cdot \frac{power\ limit\ [time] - \sum_{k=0}^{n_{evs}} charge\ action_k}{power\ limit[time]}\right)}$$

$$power\ reward\ = penalty\ const\ pwr \cdot\ clip(power\ reward, \{-10^6, 0\})$$

Equation 16 - Power reward continues function

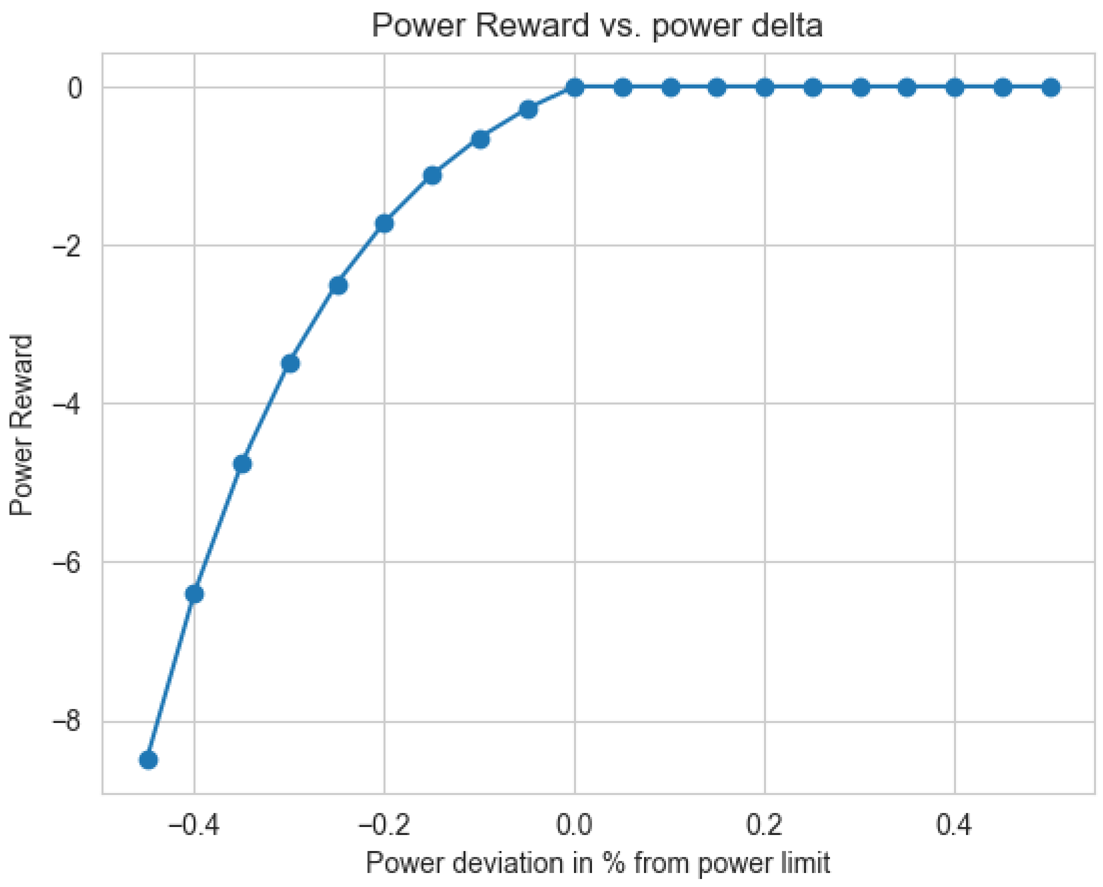


Figure 18 - Power reward continues function

SOC reward

At the end of episode SOC levels of the EVs are evaluated and rewards are given. Min SOC level was set to 0.2, and ENonD soc depends on the EVs requirement.

In The discrete environment, the cases were split to two, SOC values above SOC requirement are not rewarded, and SOC values below SOC requirement are penalized with constant value.

$$SOC\ reward = \begin{cases} 0, & if\ SOC \geq ENonD_{soc} + Min_{SOC} \\ -constant, & else \end{cases}$$

Equation 17 - SOC reward discreate function

In the continues environment SOC value reward given by the deviation from SOC requirement which it is the $ENonD_{SOC} + Min_{SOC}$, in case SOC is below the required SOC level negative reward is given to encourage agent to charge EVs more , in case the SOC value is above the required soc level reward value is set to zero with clip function. In SOC reward function is shown, the exponent constant is 3 and the penalty constant is 1, and Min soc set for 0.2 and ENonD was set to 0.

$$SOC\ reward = 1 - e^{(\exp const\ pwr \cdot \frac{(ENonD_{SOC} + Min_{soc}) - EV\ SOC}{ENonD_{SOC} + Min_{SOC}})}$$

$$SOC\ reward\ = penalty\ const\ SOC \cdot\ clip(SOC\ reward, \{-10^6, 0\})$$

Equation 18 - SOC reward continues function

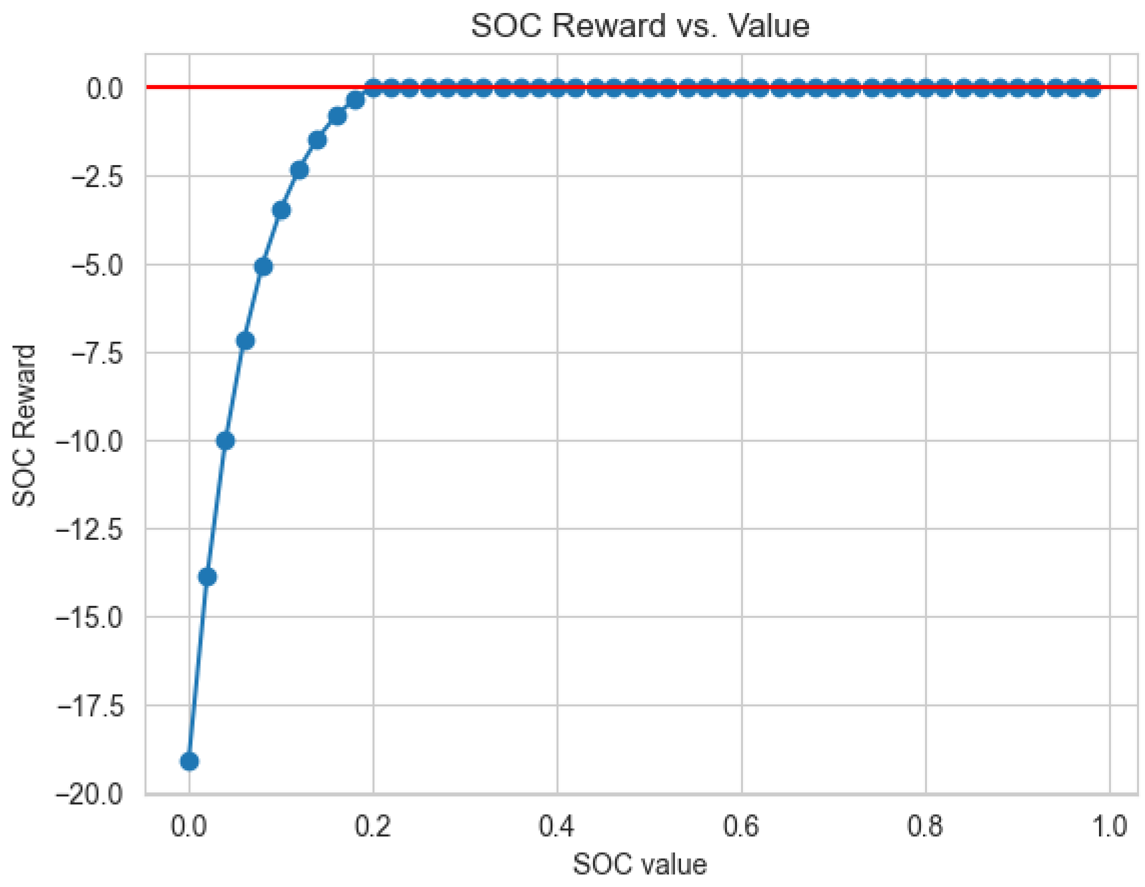


Figure 19 -SOC reward continues function

8.3.5 Algorithm Inputs and Outputs for Grid Overload Scheduling.

In training, the algorithms utilized a train data set for input, while a test data set was used during prediction to test agent's policy. While such overfitting might make the agent optimal for particular paths, it may not guarantee optimal performance in other scenarios, by using test data set it can help us to evaluate agent policy and performance. The outputs generated by the algorithms during this phase were crucial in designing and fine-tuning reward functions. These functions are vital in shaping agent performance and ensuring that the algorithms are precisely tailored to tackle the problem at hand.

Two main tasks were done in this step:
- Find optimal values for reward functions.
- Test different configurations of environment and how they affect algorithm performance.

1. **Finding optimal values for reward functions:**

In this phase, dozens of experiments were conducted to identify the optimal values for calibrating rewards given to the agent, enabling it to generate the best possible solution.

Initially, a discrete environment was employed, largely because altering values in this setting is more straightforward and offers a clearer rationale. During this phase, it became evident that using a discrete environment for agent training was less efficient and more time-consuming, primarily due to the sparsity of the rewards. Despite the shortcomings of this environment, it proved valuable in establishing baseline metrics to initiate tests and calibrations for the continuous environment.

Afterwards, additional experiments were carried out in the continuous environment to refine the reward structures and attain superior outcomes. Table 3 - Ten representative runs for reward shaping showcases ten representative runs from dozens runs that were used to find optimal values, detailing the constant values associated with reward functions. All tests were conducted using the A2C algorithm, chosen due to its quicker convergence time. The stop criteria for training were determined by the first instance when both policy loss and value loss fell below 0.01 and shown in Figure 20-Example of Policy and value networks convergence. During this phase, it was observed that both the magnitude and ratio of rewards play significant roles. Minor alterations in these values can lead to suboptimal results, such as many EVs not charging adequately or exhibiting excessive charge rates.

The investigated values during this step were:

- Mean Power delta – This represents the average difference between the charge power and the surplus power. Positive and higher values are preferred because reduced charge values exert less strain on the grid.
- The count of cases "Not Enough Energy" – how much deviations there was from target: $Min\ Soc\ +\ ENonD \leq\ SOC$
- In this scenario, while lower values are still preferable, maintaining a reduced charging power is of greater importance. As a result, it isn't essential to always opt for the lowest values.
- The number of cases SOC smaller than ENonD defined as zero, models run results that exceed from this limit failed.

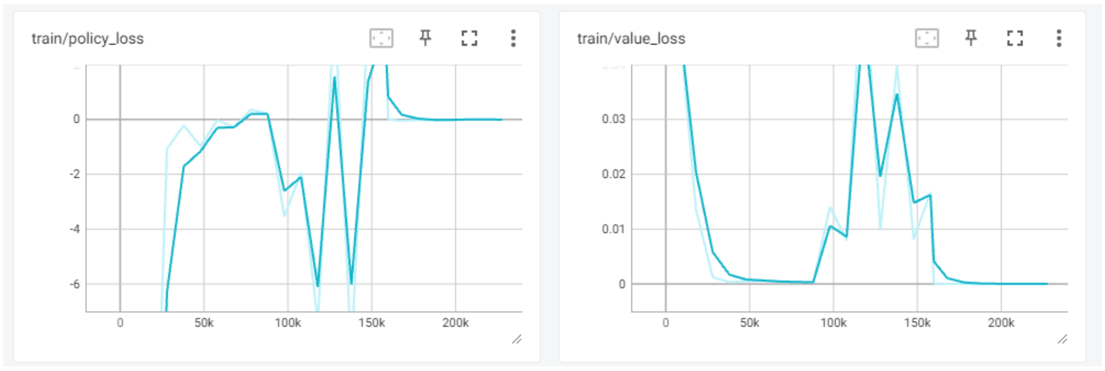


Figure 20-Example of Policy and value networks convergence

Table 3 - Ten representative runs for reward shaping

Reward function	Charge time reward			Power reward		SOC reward	
Run name	Prohibited time, charge above 0	Prohibited time, charge equal 0	Allowed time	Exponent constant	Penalty const	Exponent constant	Penalty const
First Run	-50	-50	0	5	8	8	2
Second run	-50	-50	0	5	2	5	1
Third run	0	0	0	5	2	5	1
Fourth run	0	0	0	6	3	5	1
Fifth run	0	0	0	6	3	5	2
Sixth run	0	0	0	8	3	5	2
Seventh run	0	0	0	4	2	5	2
Eight run	0	0	0	6	3.5	5	1
Ninth run	0	0	0	6	3	5	1.5
Tenth run	0	0	0	2	1	2	2
A2C_chargeTime01_	-50	1	0	2	2	5	2

2. <u>**Test different configurations of environment and how they affect algorithm performance:**</u>
   The aim of these experiments is to assess the agent's performance across varied environmental settings and to determine if a particular configuration might significantly enhance the agent's performance. The test cases for these configurations, along with their underlying rationale, are described below:

- <u>Unclip only Power reward function:</u>
  Unclipping the reward function can offer a broader range of rewards, which can provide the agent with more granular feedback about its actions. Specifically, by removing the power reward clip, it aimed to understand how the agent capitalizes on the additional reward when the total charge power is lower than the surplus power. This unclipped feedback might drive the agent to discover more nuanced strategies and behaviors that might otherwise be missed in a clipped reward scenario.

- <u>Unclip only SOC reward function:</u>
  Unclipping the reward function can offer a broader range of rewards, which can provide the agent with more granular feedback about its actions. Specifically, by removing the power reward clip, it aimed to understand how the agent capitalizes on the additional reward when the total SOC is higher than required SOC. This unclipped feedback might drive the agent to discover more nuanced strategies and behaviors that might otherwise be missed in a clipped reward scenario.

- <u>Give reward on SOC in each step and not just at the end of the episode:</u>
  In initial development of the environment it decided to give agent reward at the end episode, the case of giving reward to agent during episode is not tested and this case can influence on agent decision is tested.

8.3.6 Running Algorithms on Simulated Data for Grid Overload Analysis and algorithms comparison:

The process of running the algorithms started with the initiation of the environment and the loading of the datasets containing EVs and the surplus power. After data initialization, the next step was employing the environment as vectorized environments. Utilizing vectorized environments offers the advantage of parallel execution, enhancing efficiency through faster data collection. This parallelism not only ensures a quicker data gathering process but also enriches the diversity of experiences. By running multiple environments simultaneously, the agent is exposed to a varied set of experiences. This broader experience base can foster enhanced exploration and potentially faster convergence since the agent processes a more diverse set of experiences within each batch. Furthermore, diversifying experiences can stabilize the learning trajectory. This minimizes the chances of the agent getting trapped in local optima or being swayed by a series of exceptionally good or bad experiences. On the technical front, vectorized environments optimize the utilization of multi-core CPUs and GPUs, promoting more effective training.

Transitioning to the evaluation phase, the model selected to gauge algorithmic performance was labeled as 'tenth_run_A2C_chargeTime01' from the Table 3 - Ten representative runs for reward shaping. This model emerged as the frontrunner in reward fine-tuning tests and outperformed in comparisons with diverse environmental configurations.

For the termination of the algorithmic process, the stopping criteria were set based on the first plateau observed in the average reward per episode, lasting for at least 100K steps. This metric, derived as the output of the algorithm run, computes the mean reward over 100 consecutive runs. Additionally, a secondary stopping condition ensured a minimum of 1 million runs."

Two main scenarios were tested as defined in data simulation requirements:

1. Compare between A2C to PPO algorithm in case of 100% of vehicles in the building are EVs.
2. Compare between A2C to PPO algorithm in case of 78% of vehicles in the building are EVs, which this case represents the forecasts of [2], for sustainable development scenario of adoption rate.

8.3.7 Employ operational research methods to evaluate and compare the results obtained from the analysis.

The tools that build for in this step used in evaluation of results of all the experiments, the tools was written to analyze all results and output the results as pictures and as excel table. The following metrics was calculated:

- Mean total reward - is calculated from all environments output of total reward.
- Std total reward- based on the output of all the environments output the STD calculated.
- Mean power delta - This metric used to find what is the mean distance between surplus power to charge power, negative values indicates that the charge power exceeds above surplus power, where positive and larger values are required to maintain the grid unloaded.
- Charge power variance – used to find what are the variation of power during charging.
- The count of cases “Not Enough Energy” – how much EVs were not charge to required level: $Min\,Soc\ +\ ENonD \le\ SOC$
- SOC smaller than ENonD – This metric counts the number of cars whose SOC is below ENonD. This criterion is crucial to ensure that all vehicles depart with sufficient energy for their next ride, as defined.
- Mean SOC deviation - This parameter measures the deviation in battery percentages for vehicles that haven't attained the necessary charging level.

9 Results and Discussion:

9.1 Reward parameters calibration:

This chapter analyzes the results of phase chapter “Algorithm Inputs and Outputs for Grid Overload Scheduling.”

9.1.1 Finding optimal values for reward functions:

In Table 4- Result reward values tuning presented, this table shows the outcomes Table 3 - Ten representative runs for reward shaping.

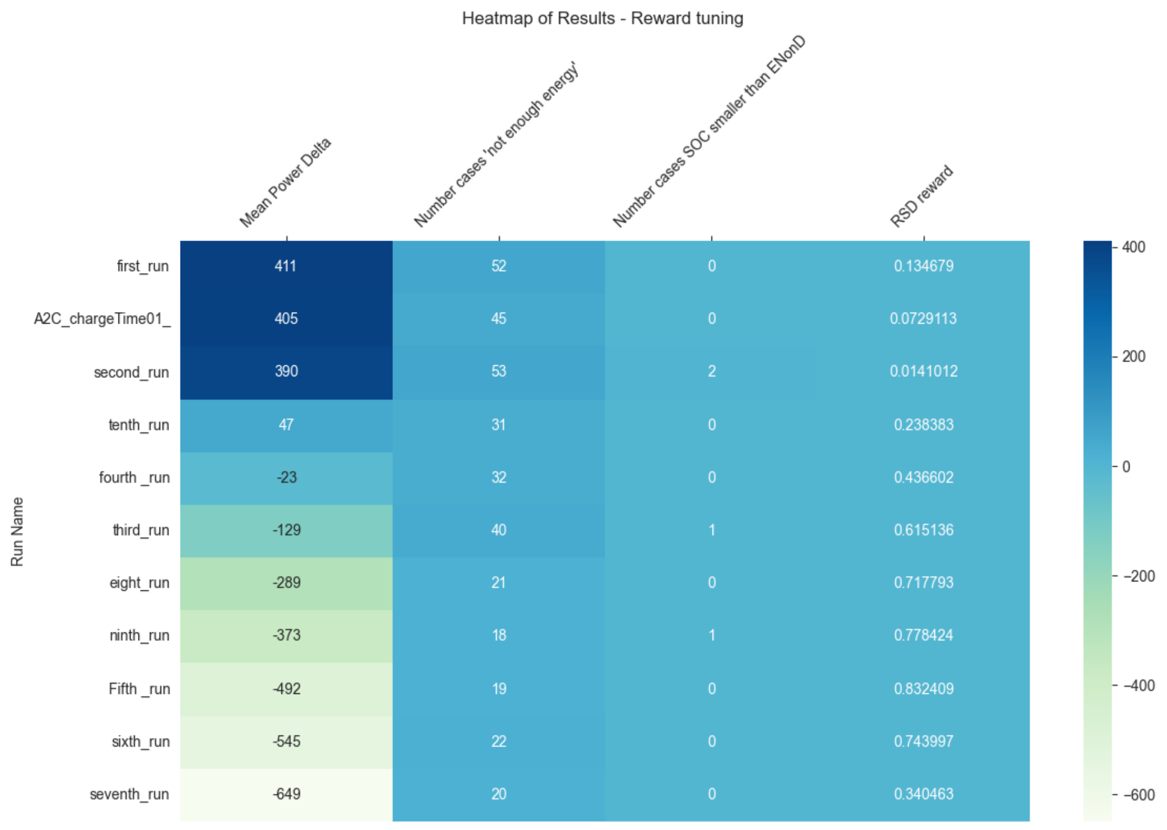
Table 4- Result reward values tuning

Run Name	**Mean total reward**	**STD total reward**	**Mean Power Delta**	**Charge power variance**	**Number cases 'not enough energy'**	**Number cases SOC smaller than ENonD**	**Mean soc deviation**
first_run	**1950**	**263**	**411.05**	**33963**	**52**	**0**	**0.275**
second_run	**32382**	**457**	**390.69**	**41504**	**53**	**2**	**0.280**
third_run	**-234**	**144**	**-129.47**	**1855**	**40**	**1**	**0.212**
fourth _run	**-113**	**49**	**-23.81**	**2808**	**32**	**0**	**0.169**
Fifth _run	**-156027**	**129878**	**-492.23**	**4569**	**19**	**0**	**0.101**
sixth_run	**-4394779**	**3269702**	**-545.41**	**1421**	**22**	**0**	**0.116**
seventh_run	**-23591**	**8032**	**-649.13**	**15933**	**20**	**0**	**0.106**
eight_run	**-8611**	**6181**	**-289.63**	**1389**	**21**	**0**	**0.111**
ninth_run	**-22732**	**17695**	**-373.68**	**2978**	**18**	**1**	**0.095**
tenth_run	**-4**	**1**	**47.09**	**9679**	**31**	**0**	**0.164**
A2C_chargeTime01_	**2605**	**190**	**405.50**	**50308**	**45**	**0**	**0.238**

To determine the most accurate environment based on the defined reward values, the absolute Relative Standard Deviation (RSD) was computed for both Mean Total Reward and Standard Deviation (STD) Total Reward. A model with the lowest RSD suggests that the agent takes consistent actions across different environments, and the reward system provides clear guidance for action selection. The values of mean delta power were ranked in descending order to identify runs with the highest values, implying adequate power reserves for EV charging. This ranking also considered the ratio of instances labeled as "Not Enough Energy." A higher result for the mean power reward indicates a greater buffer between the charge power and surplus power.

In Figure 21 Analyzed results of fine tuning process**,** the analyzed results shown, and it can be seen that the highest mean power delta is on first run but in “first run” results the Number of cases with ‘not enough energy’ is higher than   “A2C_ChargeTime01_” and the RSD reward is also higher which indicates on less stable case.

In alternative scenarios, the observed metrics indicated suboptimal outcomes. As a result, the "A2C_chargeTime01_" iteration was identified as the most suitable for the next experimental evaluations.



Run Name	Mean Power Delta	Number cases 'not enough energy'	Number cases SOC smaller than ENonD	RSD reward
first_run	411	52	0	0.134679
A2C_chargeTime01_	405	45	0	0.0729113
second_run	390	53	2	0.0141012
tenth_run	47	31	0	0.238383
fourth _run	-23	32	0	0.436602
third_run	-129	40	1	0.615136
eight_run	-289	21	0	0.717793
ninth_run	-373	18	1	0.778424
Fifth _run	-492	19	0	0.832409
sixth_run	-545	22	0	0.743997
seventh_run	-649	20	0	0.340463

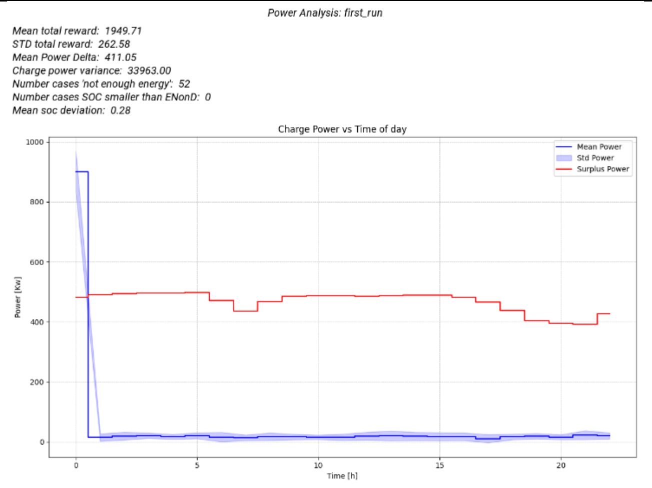
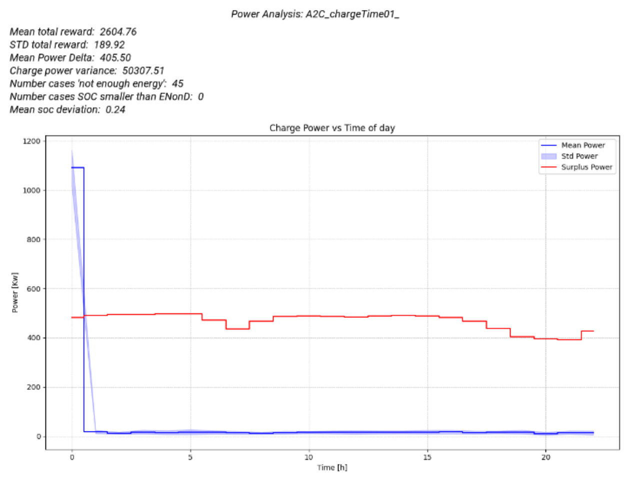
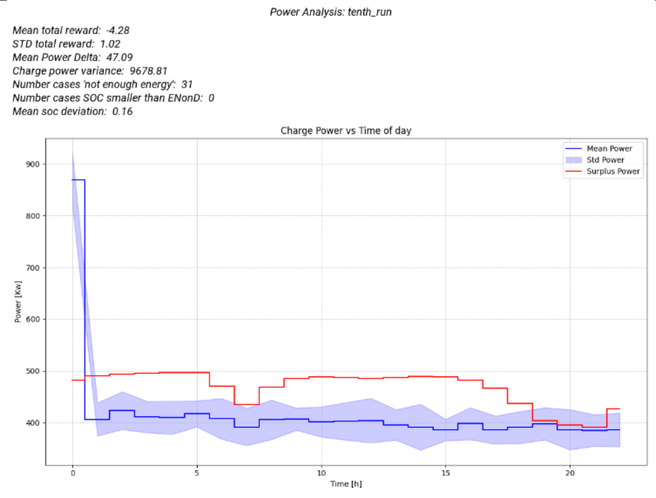
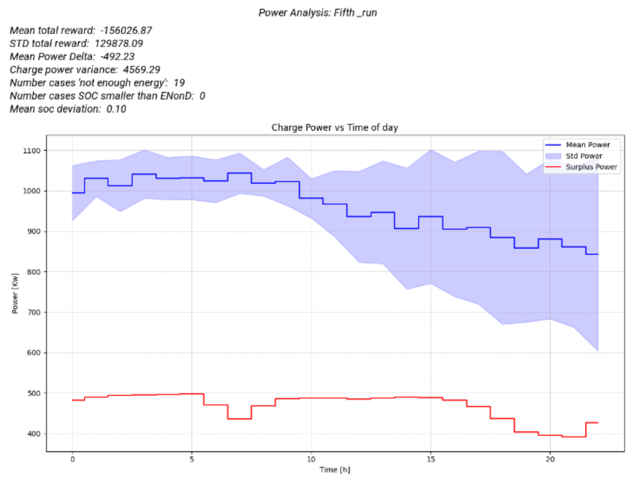
Figure 21 Analyzed results of fine tuning process

Charge profiles for representative runs are displayed in Table 5 - Charge profiles results from reward tuning **.**A visual comparison between these profiles elucidates the differences in outcomes. In the associated figures, the x-axis represents time (in hours) throughout the day, while the y-axis depicts power in [KW]. The surplus power is illustrated by the red line, whereas the blue line showcases the total charging power at each hour. Encircling the blue line is a light blue envelope, which represents the standard deviation (STD) of the power across all environments.

Both “A2C_chargeTime” and “first run” exhibit similar behavior and patterns; however, the latter displays a slightly elevated standard deviation. The “Tenth run” bears resemblance in shape to the previous two but has a more substantial charging power, closely mirroring the surplus power. Despite this, its performance does not necessarily outshine the others.

In the "Fifth run", the average power surpasses the surplus power by 492 [KW], and there's a noticeable divergence in the standard deviation of the charging power towards the episode's conclusion.

Table 5 - Charge profiles results from reward tuning

Charge profile figures	
First Run	A2C chargeTime01
 *Power Analysis: first_run* *Mean total reward: 1949.71* *STD total reward: 262.58* *Mean Power Delta: 411.05* *Charge power variance: 33963.00* *Number cases 'not enough energy': 52* *Number cases SOC smaller than ENonD: 0* *Mean soc deviation: 0.28*	 *Power Analysis: A2C_chargeTime01_* *Mean total reward: 2604.76* *STD total reward: 189.92* *Mean Power Delta: 405.50* *Charge power variance: 50307.51* *Number cases 'not enough energy': 45* *Number cases SOC smaller than ENonD: 0* *Mean soc deviation: 0.24*
Tenth run charge	Fifth run
 *Power Analysis: tenth_run* *Mean total reward: -4.28* *STD total reward: 1.02* *Mean Power Delta: 47.09* *Charge power variance: 9678.81* *Number cases 'not enough energy': 31* *Number cases SOC smaller than ENonD: 0* *Mean soc deviation: 0.16*	 *Power Analysis: Fifth _run* *Mean total reward: -156026.87* *STD total reward: 129878.09* *Mean Power Delta: -492.23* *Charge power variance: 4569.29* *Number cases 'not enough energy': 19* *Number cases SOC smaller than ENonD: 0* *Mean soc deviation: 0.10*

9.1.2 Test different configurations of environment and how they affect algorithm performance:

As described in chapter 8.3.5, efforts were made in this step to identify an optimal environment configuration by providing the agent with rewards under varying setups. The observations indicated that the original configurations, as initially conceived for the environment, produced superior outcomes relative to other configurations.

A comparative analysis of the mean power delta exhibited consistent patterns, with the most notable deviation being a 27[KW] difference between the “SOC reward unclip” and “Power reward unclip” configurations. For the metric "Number of cases 'not enough energy'", the results largely aligned, except in the scenario of "power reward unclip", which charged an average of 10 more than other configurations. Under the metric "Number of cases where SOC is smaller than ENonD", the "SOC reward unclip" failed to charge 2 EVs to the minimal SOC level required for the subsequent journey. For the RSD reward, the "A2C_chargeTime01_" configuration registered the lowest RSD.

Given these findings, the decision was made to select "A2C_chargeTime01_" for the final evaluation of the models.

Table 6- Run results environment configurations

Run Name	Mean total reward	STD total reward	Mean Power Delta	Charge power variance	Number cases 'not enough energy'	Number cases SOC smaller than ENonD	Mean soc deviation
A2C_chargeTime01_	2604.76	189.92	405	50307	45	0	0.24
Power reward unclip	-654.59	131.77	380	63586	37	0	0.20
SOC reward unclip	-6076.72	1008.31	407	45744	47	2	0.25
SOC reward during episode	-6491.26	975.59	388	50210	48	0	0.25

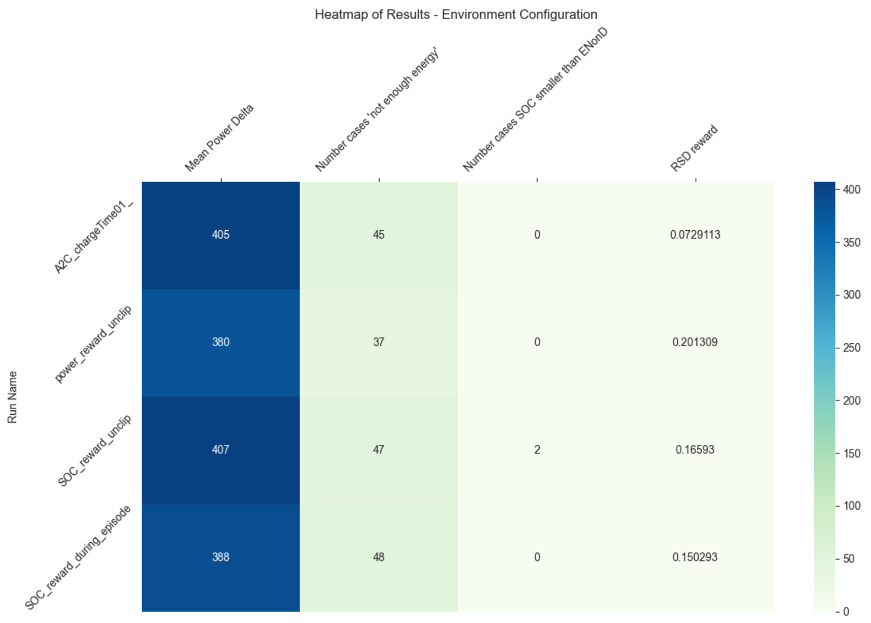
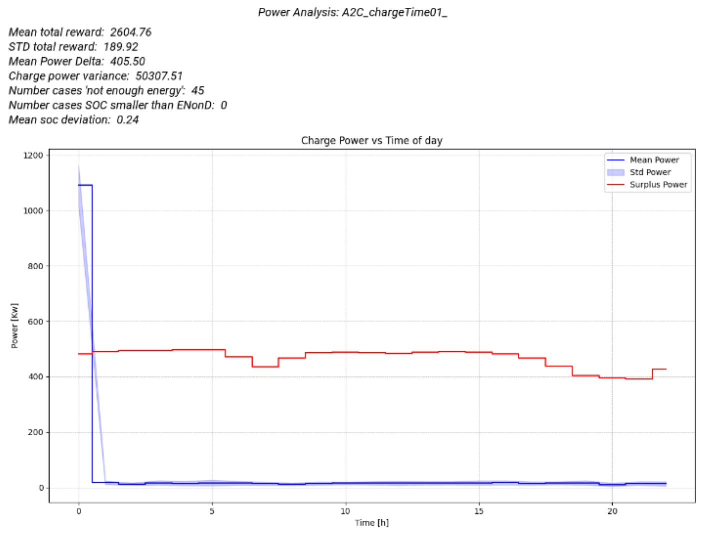
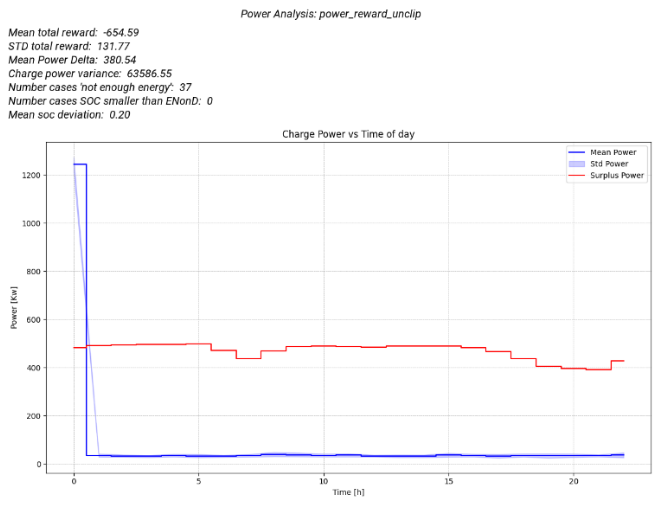
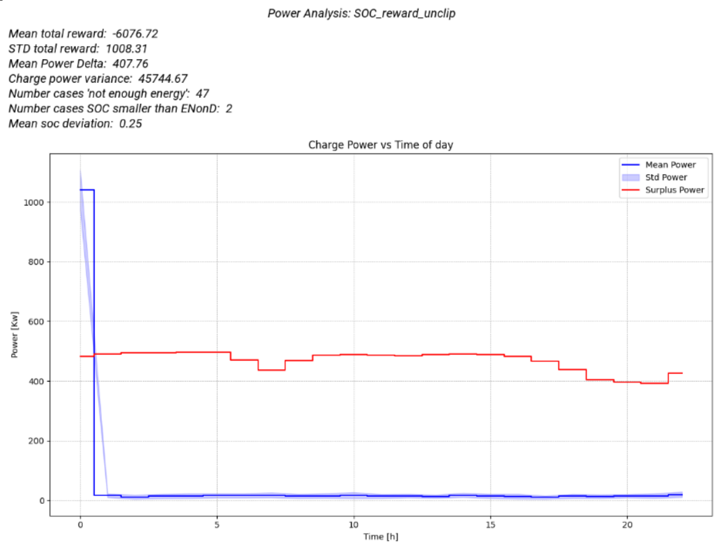
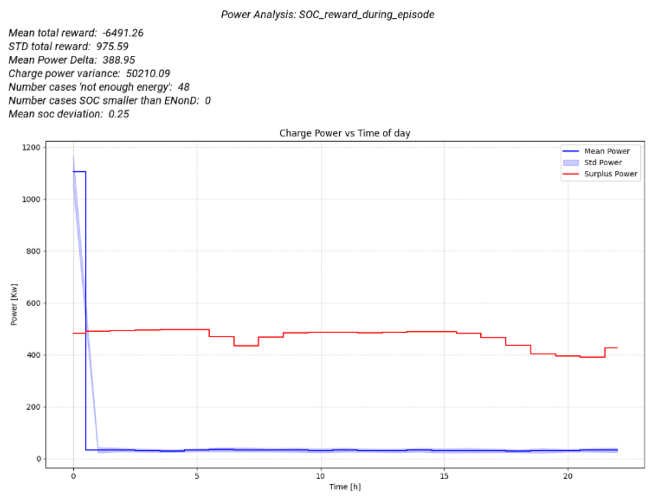


Figure 22 - Analyzed results of environments configurions

In Table 7- Charge profiles results from Environments configurations, the profiles of all four configurations are presented. Upon inspection, all four configurations exhibit similar behaviors with only subtle variations that are not immediately discernible.

Table 7- Charge profiles results from Environments configurations

Charge profile figures	
A2C chargeTime01 - charge profile	Power reward unclip
 *Power Analysis: A2C_chargeTime01_* *Mean total reward: 2604.76* *STD total reward: 189.92* *Mean Power Delta: 405.50* *Charge power variance: 50307.51* *Number cases 'not enough energy': 45* *Number cases SOC smaller than ENonD: 0* *Mean soc deviation: 0.24* 	 *Power Analysis: power_reward_unclip* *Mean total reward: -654.59* *STD total reward: 131.77* *Mean Power Delta: 380.54* *Charge power variance: 63586.55* *Number cases 'not enough energy': 37* *Number cases SOC smaller than ENonD: 0* *Mean soc deviation: 0.20*
SOC reward unclip	SOC reward during episode
 *Power Analysis: SOC_reward_unclip* *Mean total reward: -6076.72* *STD total reward: 1008.31* *Mean Power Delta: 407.76* *Charge power variance: 45744.67* *Number cases 'not enough energy': 47* *Number cases SOC smaller than ENonD: 2* *Mean soc deviation: 0.25* 	 *Power Analysis: SOC_reward_during_episode* *Mean total reward: -6491.26* *STD total reward: 975.59* *Mean Power Delta: 388.95* *Charge power variance: 50210.09* *Number cases 'not enough energy': 48* *Number cases SOC smaller than ENonD: 0* *Mean soc deviation: 0.25*

9.2 Running Algorithms on Simulated Data for Grid Overload Analysis and algorithms comparison:

In the final evaluation of the algorithms, all models were executed up to 1,358,000 steps. When comparing the performance of A2C to PPO, the former outperformed the latter by a factor of 3.44 times in the case of 100% adoption, and 3.53 times for a 78% adoption rate. Run times of the different runs shown in Table 8 -Algorithms run times. There was a difference in the setting of the hyperparameters of the algorithms: A2C was updated every 5 steps, whereas PPO was updated every 512 steps, even though the default value for PPO was set at 2048 steps. The value of n_steps was adjusted to 512 to expedite the process. However, this speed-up came with a trade-off: there were noticeable oscillations in the time/fps metric, which represents the number of frames per second, inclusive of the time taken for gradient updates. Figure 23 - Learning process time/fps illustrates an example of these oscillations, The gray line represents the output of the PPO algorithm, while the orange line depicts the output of the A2C algorithm.

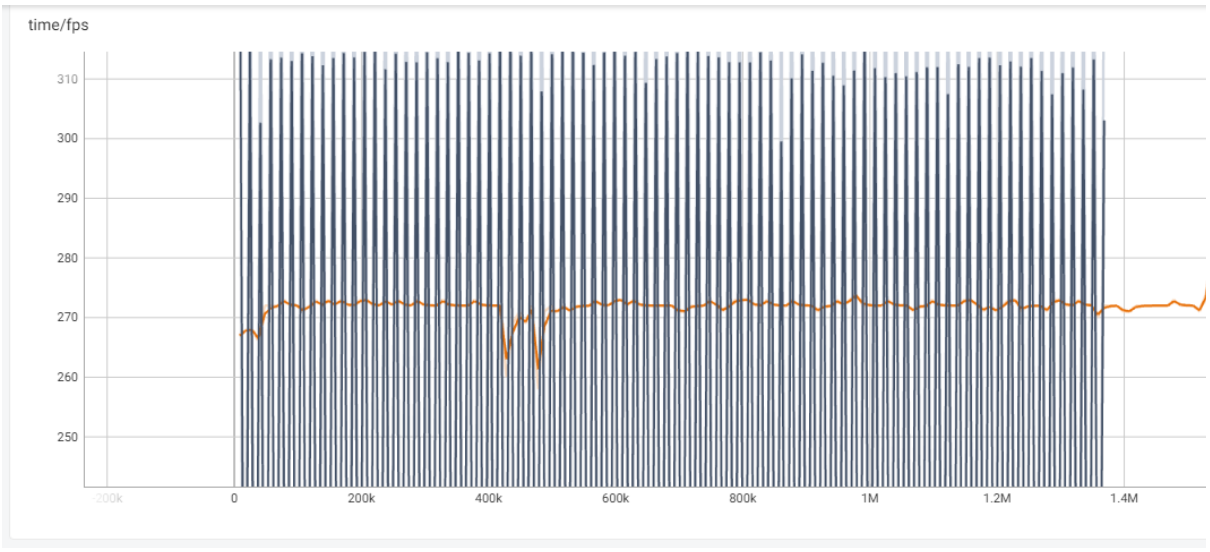


Figure 23 - Learning process time/fps

Table 8 -Algorithms run times

Run name	Run Time [h]
chargeTime_100_A2C	1.383
chargeTime_100_PPO	4.761
chargeTime_78_A2C	1.066
chargeTime_78_PPO	3.763

9.2.1 Evaluate and compare algorithms on case 100% of vehicles are EVs in the building:

For the case of 100% adoption, the A2C algorithm outperformed and yielded superior results across all metrics. However, neither the A2C nor the PPO algorithms were successful in addressing the scheduling problem under the given constraints.

in Figure 24 - Analyzed results of 100% adoption rate scenario it can be seen that A2C was achieved better results in all the metrics but the results in the figure are not revel the exception from the power limit and In Table 9 - Charge profiles results of Algorithms in 100% adoption the all profile shown. It's evident that A2C performed consistently with previous runs, whereas PPO consistently exceeded the surplus power throughout the episode, exhibiting a high standard deviation in charge power.

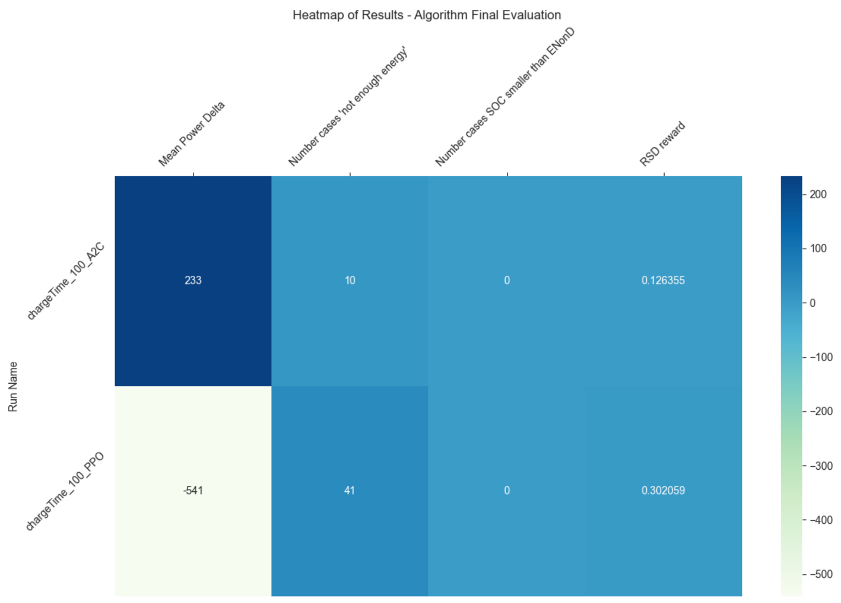
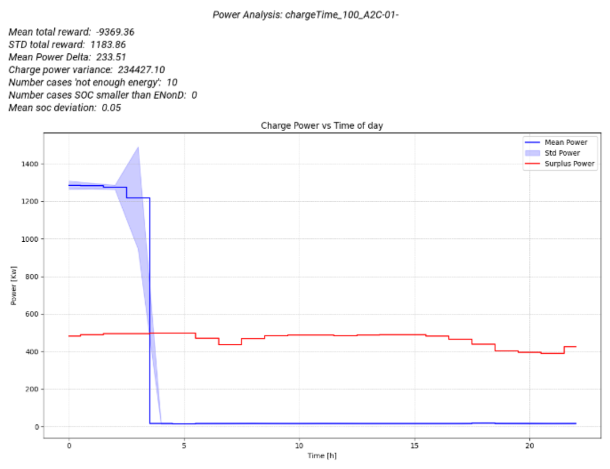
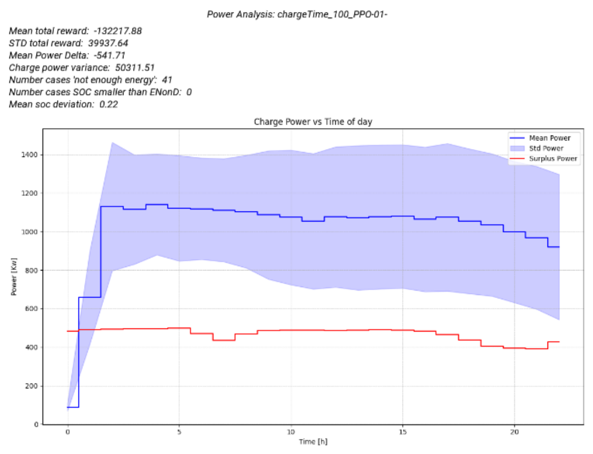


Figure 24 - Analyzed results of 100% adoption rate scenario

Table 9 - Charge profiles results of Algorithms in 100% adoption

A2C charge profile at 100% adoption	PPO charge profile at 100% adoption
 Power Analysis: chargeTime_100_A2C-01- Mean total reward: -9369.36 STD total reward: 1183.86 Mean Power Delta: 233.51 Charge power variance: 234427.10 Number cases 'not enough energy': 10 Number cases SOC smaller than ENonD: 0 Mean soc deviation: 0.05	 Power Analysis: chargeTime_100_PPO-01- Mean total reward: -132217.88 STD total reward: 39937.64 Mean Power Delta: -541.71 Charge power variance: 50311.51 Number cases 'not enough energy': 41 Number cases SOC smaller than ENonD: 0 Mean soc deviation: 0.22

9.2.2 Evaluate and compare algorithms on case 78% EVs of vehicles are in the building:

In the scenario with a 78% adoption rate, it was observed that A2C exceeded the surplus power at the start of the episode, which then reduced to a minimal power level. The PPO algorithm behaved in the same manner as it did in the 100% adoption scenario, consistently exceeding the permissible limit throughout the entire episode, with a high standard deviation in power.

In Table 10 - Charge profiles results of Algorithms in 78% adoption **,**the power profiles of the algorithms at 78% adoption are displayed. It's evident that A2C performed consistently with previous runs, whereas PPO consistently exceeded the surplus power throughout the episode, exhibiting a high standard deviation in charge power.

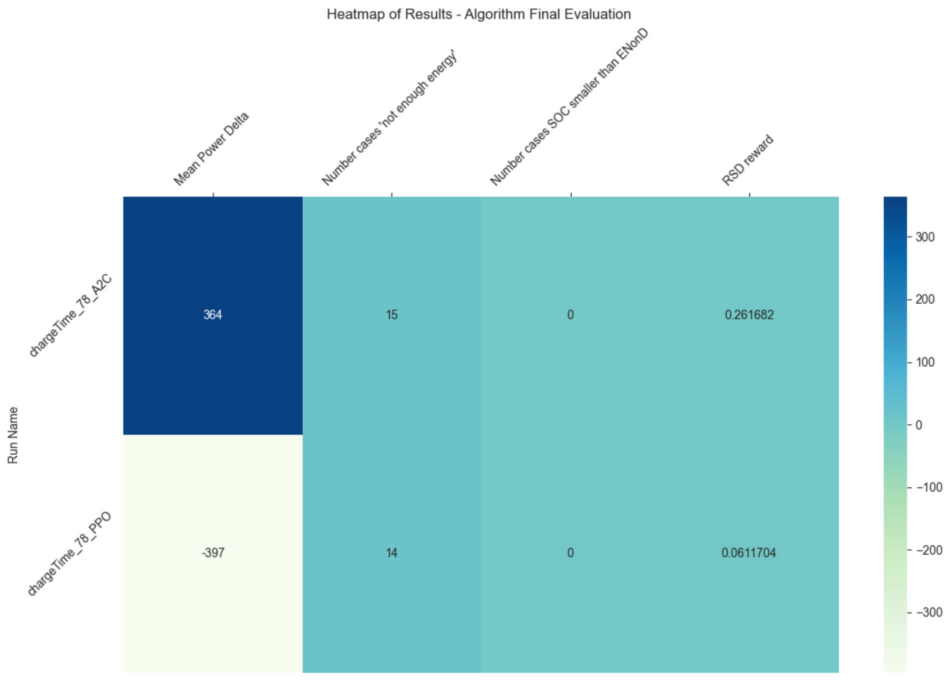
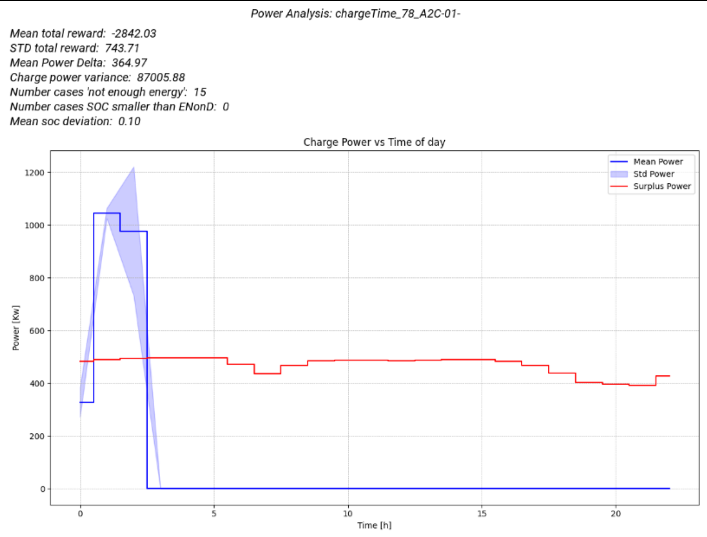
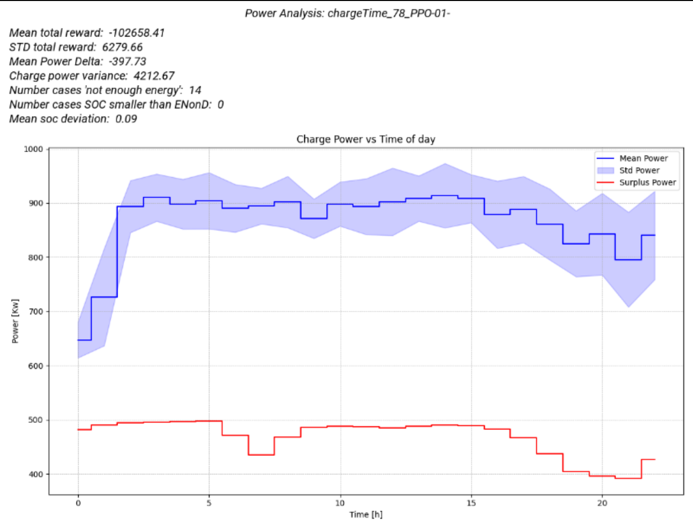


Figure 25 - Analyzed results of 78% adoption rate scenario

Table 10 - Charge profiles results of Algorithms in 78% adoption

A2C charge profile at 78% adoption	PPO charge profile at 78% adoption
 Power Analysis: chargeTime_78_A2C-01- Mean total reward: -2842.03 STD total reward: 743.71 Mean Power Delta: 364.97 Charge power variance: 87005.88 Number cases 'not enough energy': 15 Number cases SOC smaller than ENonD: 0 Mean soc deviation: 0.10	 Power Analysis: chargeTime_78_PPO-01- Mean total reward: -102658.41 STD total reward: 6279.66 Mean Power Delta: -397.73 Charge power variance: 4212.67 Number cases 'not enough energy': 14 Number cases SOC smaller than ENonD: 0 Mean soc deviation: 0.09

9.2.3 Discussion:

In the dynamic sphere of power systems management, Systems Engineering emerges as an indispensable asset, intricately weaving together various components for seamless operations. The power grid, a vast network sustained by countless professionals, embodies the complexity synonymous with modern engineering challenges. With the escalating global interest in electric vehicles (EVs), the urgency to evolve and adapt our existing electric infrastructure has never been more tangible. Enter this project, which aspires to address these nuances head-on, targeting a broad beneficiary base—from the Electricity Authority, power suppliers, the broader community, environmental organizations, to everyday grid consumers and the burgeoning EV community. At its core, the initiative seeks to probe the efficacy of deep reinforcement learning (DRL) algorithms in dictating the scheduling patterns of EV charging queues. This commitment aligns with a greater vision: devising a robust system to manage EV charging, ensuring that our residential electrical grid remains resilient against potential overloads.

The analytical voyage exploring grid overload scheduling predominantly hinged on the A2C and PPO algorithms. Insights indicated that both algorithms demonstrated strengths in specific domains, they didn't holistically address the intricate challenges presented by the established environment across the considered adoption scenarios. Their capabilities in charging a vast majority of EVs were evident, yet their adherence to strict power constraints remained less than ideal. From an engineering standpoint, the importance of preserving a consistent power profile is paramount, and herein lies an area where the algorithms could be optimized.

Throughout the various project phases—ranging from reward calibrations to conclusive evaluations—there persisted a theme: the unpredictability of the models. The aspirational state of consistently staying below the surplus power throughout episodes was not achieved. This stability is essential in real-world grid operations to ensure balanced power distribution and avoid potential grid complications.

A closer examination of charge power behavior throughout evaluations unveiled an intriguing anomaly. The beginning of episodes witnessed an unusually elevated charge power, which progressively reduced as the episode continued. This pattern suggests a tendency of the agent to prioritize late rewards.

It's worth noting that the A2C algorithm stood as the primary focal point during the initial stages of environment development and assessments. PPO's integration came predominantly during the concluding evaluations. Such a phased methodology could inherently shape the outcomes and comparability, considering the unique operational attributes associated with each algorithm. Notably, in all tested scenarios, neither algorithm comprehensively addressed the scheduling problem; while managing to charge most EVs, they didn't consistently adhere to power constraints or achieve a consistent power profile.

9.2.4 Work limitations:

A dataset comprising EV arrivals and departures, Battery SoC, will be generated specifically for this research. It is important to note that the generated data will not be real-world data, and the distribution of data will approximate real-world data.

10 Conclusions:

The examination of grid overload scheduling with respect to the A2C and PPO algorithms has shed light on their respective capabilities and limitations with the developed environment. The results highlight the superior performance of A2C, yet it's evident that neither algorithm, in its current form, entirely addresses the complexities of the problem. The complex challenges posed by grid overload scheduling require sophisticated solutions, and although the algorithms showed promise, they did not fully achieve the set benchmarks.

Given the high complexity of this problem, it is necessary to delve deeper into the configurations and nuances of the algorithms. The observed behaviors may suggest that the agent could potentially be entangled in policy overfitting. Addressing this would require more rigorous training and perhaps introducing regularizations to prevent overfitting. Consolidating multiple reward functions into a singular, cohesive reward function might enhance the agent's understanding and performance. While A2C showed a more promising trajectory, further refinement is necessary. Both algorithms would benefit from an extensive hyperparameter optimization phase, which has not yet been undertaken.

With the inherent complexities of the problem acknowledged, there's a palpable need for hyperparameter optimization, ensuring the algorithms are better tuned to the specific challenges of grid overload scheduling. As the results indicate potential areas of policy overfitting, future attempts should consider mechanisms to neutralize this, perhaps by introducing varied training scenarios or regularization techniques. Efforts should also be concentrated on streamlining the reward mechanisms, potentially converging multiple reward functions into a unified framework to enhance clarity and focus during training. Beyond A2C and PPO, the exploration of hybrid or more advanced algorithmic models may present more effective solutions for the scheduling challenges at hand. This research lays a solid groundwork for future engineering pursuits in the sphere of grid overload scheduling, offering crucial technical insights and potential paths forward.

11 Bibliography:

[1] M. T. Hussain, D. N. Bin Sulaiman, M. S. Hussain, and M. Jabir, “Optimal Management strategies to solve issues of grid having Electric Vehicles (EV): A review,” *J. Energy Storage*, vol. 33, p. 102114, Jan. 2021, doi: 10.1016/j.est.2020.102114.

[2] IEA, “Global EV Outlook 2021 - Accelerating ambitions despite the pandemic,” *Global EV Outlook 2021*. p. 101, 2021. [Online]. Available: https://iea.blob.core.windows.net/assets/ed5f4484-f556-4110-8c5c-4ede8bcba637/GlobalEVOutlook2021.pdf

[3] Electricity Authority of Israel, “The challenges of the electricity sector and the regulation required for the integration of electric vehicles in Israel.” 2022.

[4] M. Dorokhova, Y. Martinson, C. Ballif, and N. Wyrsch, “Deep reinforcement learning control of electric vehicle charging in the presence of photovoltaic generation,” *Appl. Energy*, vol. 301, p. 117504, Nov. 2021, doi: 10.1016/j.apenergy.2021.117504.

[5] F. Tuchnitz, N. Ebell, J. Schlund, and M. Pruckner, “Development and Evaluation of a Smart Charging Strategy for an Electric Vehicle Fleet Based on Reinforcement Learning,” *Applied Energy*, vol. 285. 2021. doi: 10.1016/j.apenergy.2020.116382.

[6] S. Abdullah-Al-Nahid, T. A. Khan, M. A. Taseen, T. Jamal, and T. Aziz, “A novel consumer-friendly electric vehicle charging scheme with vehicle to grid provision supported by genetic algorithm based optimization,” *J. Energy Storage*, vol. 50, p. 104655, Jun. 2022, doi: 10.1016/j.est.2022.104655.

[7] W. B. Heredia, K. Chaudhari, A. Meintz, M. Jun, and S. Pless, “Evaluation of smart charging for electric vehicle-to-building integration: A case study,” *Appl. Energy*, vol. 266, p. 114803, May 2020, doi: 10.1016/j.apenergy.2020.114803.

[8] S. Powell, G. V. Cezar, L. Min, I. M. L. Azevedo, and R. Rajagopal, “Charging infrastructure access and operation to reduce the grid impacts of deep electric vehicle adoption,” *Nat. Energy*, vol. 7, no. 10, pp. 932–945, Sep. 2022, doi: 10.1038/s41560-022-01105-7.

[9] Å. L. Sørensen, K. B. Lindberg, I. Sartori, and I. Andresen, “Analysis of residential EV energy flexibility potential based on real-world charging reports and smart meter data,” *Energy Build.*, vol. 241, p. 110923, Jun. 2021, doi: 10.1016/j.enbuild.2021.110923.

[10] “Israel Meteorological Service.” https://ims.gov.il/he/data_gov

[11] Enviromental Protection Agency, “Green Power Equivalency Calculator.” https://www.epa.gov/green-power-markets/green-power-equivalency-calculator-calculations-and-references

[12] C. for S. Systems, “Photovoltaic energy,” *CSS07-08*. Center for Sustainable Systems, University of Michigan. 2022. [Online]. Available: https://css.umich.edu/publications/factsheets/energy/photovoltaic-energy-factsheet#:~:text=Though most commercial panels have,cells with efficiencies approaching 50%25.&text=Assuming intermediate efficiency%2C PV covering,electricity to meet national dema

[13] S. Mugo, “Power Factor: Determining how Much Electricity Your Power System Consumes,” 2022, [Online]. Available: https://eepower.com/technical-articles/power-factor-determining-how-much-electricity-your-power-system-consumes/

[14] A. Hebrail, Georges and Berard, “Individual household electric power consumption,” 2012. https://doi.org/10.24432/C58K54

[15] N. Rhein, “10th most common ev in Isreal,” *Icar*. https://www.icar.co.il/-רכב
חשמלי/אחת_מכל_10_מכוניות_שנמכרו_בישראל_היתה_חשמלית/

[16] “Electric Vehicle Database.” https://ev-database.org/

[17] R. S. Sutton and A. G. Barto, “Reinforcement Learning, In The Lancet (Vol. 258). https://doi.org/10.1016/S0140-6736(51)92942-XtonBartoSecondBook,” *The Lancet*, vol. 258, no. 6685. pp. 675–676, 1998.

[18] O. Simonini, Thomas and Sanseviero, “The Hugging Face Deep Reinforcement Learning Class,” *GitHub*, 2023. https://github.com/huggingface/deep-rl-class

[19] K. K. Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, “Asynchronous Methods for Deep Reinforcement Learning.” 2016. [Online]. Available: https://arxiv.org/abs/1602.01783v2

[20] O. K. John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, “Proximal Policy Optimization Algorithms.” 2017. [Online]. Available: https://arxiv.org/abs/1707.06347

[21] A. R. and A. H. and A. G. and A. K. and M. E. and N. Dormann, “Stable-Baselines3: Reliable Reinforcement Learning Implementations.” Journal of Machine Learning Research, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html

12 Appendix:

12.1 DB scheme code:

```
CREATE DATABASE chargeSystemDB;

USE chargeSystemDB;

CREATE TABLE Electricity_Company_Tariff (

  electricity_company_ID INT PRIMARY KEY,

  month VARCHAR(10),

  hour VARCHAR(10),

  price DECIMAL(10, 2)

);

CREATE TABLE Solar_panels_Tariff (

  solar_panel_ID INT PRIMARY KEY,

  price DECIMAL(10, 2)

);

CREATE TABLE Building_electricity_meter (

  building_ID INT PRIMARY KEY,

  timestamp TIMESTAMP,

  grid_power_input DECIMAL(10, 2),

  solar_power_input DECIMAL(10, 2),

  households_power_consumption DECIMAL(10, 2),

  charge_power_output DECIMAL(10, 2)

);

CREATE TABLE user_ (

  user_ID INT PRIMARY KEY,

  timestamp TIMESTAMP,

  estimated_arrival_time TIMESTAMP,

  estimated_departure_time TIMESTAMP,

  estimated_drive_distance DECIMAL(10, 2)

);
```

```
CREATE TABLE ev (
  ev_ID INT PRIMARY KEY,
  timestamp TIMESTAMP,
  ev_brand_model VARCHAR(50),
  ev_efficiency DECIMAL(10, 2),
  ev_battery_capacity DECIMAL(10, 2),
  energy_needed_on_next_departure DECIMAL(10, 2),
  soc DECIMAL(10, 2)
);
CREATE TABLE ev_charger (
  charger_ID INT PRIMARY KEY,
  timestamp TIMESTAMP,
  output_power DECIMAL(10, 2)
);
CREATE TABLE user_charge_session (
    session_ID INT PRIMARY KEY,
    electricity_company_ID INT,
    solar_panel_ID INT,
    building_ID INT,
    connection_time TIMESTAMP,
    disconnection_time TIMESTAMP,
    charge_start_time TIMESTAMP,
    charge_end_time TIMESTAMP,
    cumulative_power DECIMAL(10,2),
    cumulative_cost DECIMAL(10,2),
    FOREIGN KEY (electricity_company_ID) REFERENCES Electricity_Company_Tariff(electricity_company_ID),
    FOREIGN KEY (solar_panel_ID) REFERENCES Solar_panels_Tariff(solar_panel_ID),
    FOREIGN KEY (building_ID) REFERENCES Building_electricity_meter(building_ID)
);
```

```
CREATE TABLE user_charge_session_user (

  session_user_id INT PRIMARY KEY AUTO_INCREMENT,

  session_id INT NOT NULL,

  user_id INT NOT NULL,

  FOREIGN KEY (session_id) REFERENCES user_charge_session(session_id),

  FOREIGN KEY (user_id) REFERENCES user_(user_id)

);

CREATE TABLE user_charge_session_ev (

  session_ev_id INT PRIMARY KEY AUTO_INCREMENT,

  session_id INT NOT NULL,

  ev_id INT NOT NULL,

  FOREIGN KEY (session_id) REFERENCES user_charge_session(session_id),

  FOREIGN KEY (ev_id) REFERENCES ev(ev_id)

);

CREATE TABLE user_charge_session_ev_charger (

  session_ev_charger_id INT PRIMARY KEY AUTO_INCREMENT,

  session_id INT NOT NULL,

  ev_charger_id INT NOT NULL,

  FOREIGN KEY (session_id) REFERENCES user_charge_session(session_id),

  FOREIGN KEY (ev_charger_id) REFERENCES ev_charger(charger_ID)

);
```

12.2 Environment code:

```
import gymnasium as gym
from gymnasium import spaces
import numpy as np
import copy


class EVChargingEnv_continues(gym.Env):
    def __init__(self, evs, power_limit):
        super(EVChargingEnv_continues, self).__init__()

        """This function intended to initialize environment variables
and load data set

         # Action space is multi-discrete, in this case, there are 3
actions per EV, and actions could be different
        # from each other
        # all the algorithms that supports MultiDiscrete action space
support this environment

         # In observation_space 'evs; high limit value is assumed to
300
        # and on power limit / power_allowed high should not reach
above 1000
        """
        self.time = None
        self.total_reward = None

        self.terminated = False
        self.truncated = False

        self.evs_const = evs  # evs dataset
        self.evs = None  # used in reset() to load initial dataset
        self.power_limit = power_limit  # Power limits is surplus power
that system can use to charge evs

        self.action_space = spaces.MultiDiscrete([3] * len(evs))
        ev = evs[0]
        num_keys = len(ev.keys())
        NUM_OF_FEATURES = num_keys

        self.observation_space = spaces.Dict({
            'evs': spaces.Box(low=0, high=300, shape=(len(evs),
NUM_OF_FEATURES)),
            'power_allowed': spaces.Box(low=0, high=1000, shape=(24,)),
            'time': spaces.Discrete(24)
        })

    def reset(self, **kwargs):
        """ Reset function is intended to reset all environment
variables to enable initial
         state for observation and learning of the agent """

        self.terminated = False
        self.truncated = False
        self.time = 0
        self.total_reward = 0  # cumulative reward
        # self.evs = self.evs_const
        self.evs = copy.deepcopy(self.evs_const)
        info = {}

        evs_obs = np.array([[ev['Arrival_time[h]'], ev['TuD (int)'],
ev['Battery capacity [KWh]'],
                             ev['ENonD'], ev['SOC']] for ev in
self.evs], dtype=np.float32)
        power_allowed_obs = np.array(self.power_limit,
dtype=np.float32)
        time_obs = self.time

        observation = {'evs': evs_obs, 'power_allowed':
power_allowed_obs, 'time': time_obs}

        episode_info = {"total_reward": self.total_reward,
"observation": observation}
        info = {"episode": episode_info}  # Update the info dictionary
with the episode information

        return (observation, info)

    def step(self, actions):
        """ Step function intended to execute agent actions:
        in this case the actions that return are 0,1,2, and they
converted to charge rate
        this function also responsible to update SOC, TUD, and rewards
        at the end of the step the function return the new state St+1
to agent """

        info = {}  # Additional information for debugging

        charge_actions = [0, 3.7, 11]  # Charge levels that are allowed
        charge = np.zeros(len(actions))
        # fill charge vector in power values based on actions
        for j, action in enumerate(actions):
            charge[j] = charge_actions[action]

        # Immediate rewards - per step
        charge_time_reward = 0
        SOC_reward = 0
        power_reward = 0

        # Update the SOC of each EV in self.evs
        for idx, ev in enumerate(self.evs):

            '''1. Time constraints reward'''

            if (ev['Arrival_time[h]'] < self.time) | (ev['TuD (int)']
== 0):
                if charge_actions[actions[idx]] > 0:

                    #charge[idx] = 0 #
                    charge_time_reward -= 50
                else:
                    charge_time_reward += 1
                # pass

            elif (ev['Arrival_time[h]'] >= self.time) & (ev['TuD
(int)'] > 0):
                # Update charge levels based on the selected actions
                ev['SOC'] += charge_actions[actions[idx]] / ev['Battery
capacity [KWh]']
                charge_time_reward += 0

            # Update TUD
            if ev['TuD (int)'] > 0:
                ev['TuD (int)'] -= 1

            elif ev['TuD (int)'] <= 0:
                pass

        """ 2. Power constraint"""
        exp_const_pwr = 2
        penalty_const_pwr = 2

        power_reward = 1 - np.exp(
            -exp_const_pwr * (self.power_limit[self.time] -
charge.sum()) / self.power_limit[self.time]) #* penalty_const_pwr

        power_reward = penalty_const_pwr * (np.clip(power_reward,
a_min=-1000000, a_max=0))

        """3 SOC during episode"""
        ''' for idx, ev in enumerate(self.evs):
            ENonD_SOC = ev['ENonD'] / ev['Battery capacity [KWh]']
            min_SOC = 0.2
            exp_const_soc = 5
            penalty_const_soc = 2

            temp_reward_soc = 1 - np.exp(
                exp_const_soc * (((ENonD_SOC + min_SOC) - ev['SOC']) /
(ENonD_SOC + min_SOC)))

            SOC_reward += penalty_const_soc * (np.clip(temp_reward_soc,
a_min=-1000000, a_max=0))

        # Update the time and check if the episode is done'''
        self.time += 1

        if self.time == 24:
            self.terminated = True

            for idx, ev in enumerate(self.evs):
                """ 3. SOC level constraint give reward only on
departure """
                ENonD_SOC = ev['ENonD'] / ev['Battery capacity [KWh]']

                min_SOC = 0.2
                exp_const_soc = 5
                penalty_const_soc = 2

                temp_reward_soc = 1 - np.exp(exp_const_soc *
(((ENonD_SOC + min_SOC) - ev['SOC']) / (ENonD_SOC + min_SOC))) #*
penalty_const_soc

                #SOC_reward += temp_reward_soc
                SOC_reward += penalty_const_soc *
(np.clip(temp_reward_soc, a_min=-1000000, a_max=0))


            evs_obs = np.array(
                [[ev['Arrival_time[h]'], ev['TuD (int)'], ev['Battery
capacity [KWh]'], ev['ENonD'], ev['SOC']] for ev
                 in self.evs], dtype=np.float32)
            power_allowed_obs = np.array(self.power_limit,
dtype=np.float32)
            time_obs = self.time
            observation = {'evs': evs_obs, 'power_allowed':
power_allowed_obs, 'time': time_obs}

        reward = (charge_time_reward + power_reward + SOC_reward)

        # Create the observation for the new time step
        evs_obs = np.array(
            [[ev['Arrival_time[h]'], ev['TuD (int)'], ev['Battery
capacity [KWh]'], ev['ENonD'], ev['SOC']] for ev in
             self.evs], dtype=np.float32)

        power_allowed_obs = np.array(self.power_limit,
dtype=np.float32)
        time_obs = self.time
        observation = {'evs': evs_obs, 'power_allowed':
power_allowed_obs, 'time': time_obs}

        # info output data

        self.total_reward += reward

        info['total_reward'] = self.total_reward
        info['observation'] = observation
        info['charge_pwr'] = np.sum(charge)

        return observation, reward, self.terminated, self.truncated,
info


# Define a wrapper function that takes 'evs' and 'power_limit' as
arguments and returns the environment instance
def make_ev_charging_env(evs, power_limit):
    return EVChargingEnv_continues(evs=evs, power_limit=power_limit)
```

13 תקציר:

עם העלייה הצפויה של כלי הרכב החשמליים ברחבי העולם, נוצר אתגר שאר נוגע לתיאום וניהול לתיאום בין טעינת רכבים חשמליים ורשת החשמל, במיוחד בשעות השיא. חוסר תיאום בין טעינת רכבים חשמליים ורשמת החשמל עלול לגרום לעומסים ברשת, פגשיה בתשתיות החשמל, אי יציבות במתחי ההספקה ולהגברת פליטת המזהמים בעקיפין. חקר מקרים זה מתמקד בתכנון מערכת ניהול ושליטה בטעינת רכבים חשמלים בבתי מגורים בבנייני מגורים עם מספר חניות רב, בכדי להקל על עומסי רשת עתידיים. שימוש במתודולוגיות מעולם הנדסת המערכות, פרויקט זה תורם באופן משמעותי לבעלי עניין שונים, כולל ספקי חשמל, ארגונים סביבתיים צרכני החשמל ובעלי הרכבים החשמליים.

אלגוריתמי Deep Reinforcement learning, וביחוד A2C ו-PPO נחקרו ונבחנו על יכולתם לנהל את תזמון הטעינה של הרכבים החשמליים. אלגוריתמים אלו נבחרו על סמך הפוטנציאל שלהם לעמוד באילוצי הרשת וצורכי המשתמשים. בכדי לתמוך ביכולות המערכת, תוכנן מסד נתונים על בסיס SQL לאחסון הנתונים לצורך תהליך החיוב, ביצוע חקר ביצועים, תפעול ואופטימיזציה של ציי רכב חשמליים לבין יישומים פוטנציאליים אחרים. בכדי לקרב ולבחון את הבעיה באופן ממשי ותואם לעולם האמיתי בוצעה סימולציה של הנתונים אשר מיצגים בקירוב את התנהגות המשתמשים, אופן וכמות החשמל שבשימוש בבניין, ואת מאפייני הרכבים החשמליים. בנוסף בכדי לאפשר שימוש באלגוריתמים השונים פותחה סביבה המדמה תהליך תזמון הטעינה למשך 24 שעות, ושני האלגוריתמים הנ"ל נבדקו בסביבה שפותחה במיוחד בכדי להעריך את יעילותם.

ממצאי המחקר מראים שאלגוריתם A2C בשני התרחישים השונים 100% ו-78% אימוץ של רכבים חשמליים עלה בביצועיו על PPO בכל המדדים שנבחנו. עם זאת אף לא אחד מהאלגוריתמים לא פתר את הבעיה באופן מלא במסגרת האילוצים של הרשת והמשתמשים, במיוחד כשאר נבחן פרופיל הטעינה הכולל. בשלבים הראשונים של התהליך (השעות הראשונות) שני האלגוריתמים הראו עלייה משמעותית מאוד בהספק הנצרך בכדי להטעין את כל צי הרכב, דבר אשר הצביע על נטייה לאיסוף תגמול מידיי, בניגוד לתפיסה בה האלגוריתמים יאספו פרסים לאורך כל התהליך.

לסיכום, בעוד אלגוריתם A2C הוכיח יכולת הסתגלות טובה יותר, נותר צורך לגבש מערכת תגמול מתוחכמת יותר ולעסוק באופטימיזציה נרחבת של ההיפר-פרמטרים עבור שני האלגוריתמים. מחקר זה מדגיש את היכולות של הנדסת מערכות להתמודד עם בעיות של עומס יתר על רשתות החשמל בעולם האמיתי, מה שמציב את הבסיס למחקר עתידי בתחום זה. עבודות המשך צריכות לתעדף את חידוד האלגוריתמים שבהם נעשה שימוש, התעמקות במודלים אלגוריתמיים מתקדמים יותר, ושילוב פונקציות התגמול המרובות לאחת בכדי להניב פתרונות יעילים.