Reinforcement Learning-Driven Local Transactive Energy Market for Distributed Energy Resources

by

Shida Zhang

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Software and Intelligent Systems

Department of Electrical and Computer Engineering
University of Alberta

Abstract

Technological breakthroughs in renewable power generation, battery storage, electric mobility, and advanced data logistics are changing the electric grid. The huge influx of distributed energy resources (DERs), while important to curb carbon emissions, is not without consequences. The highly intermittent nature of renewable energy resources (RES), combined with the decreased visibility of DERs (from the system operator's perspective), makes it increasingly difficult for the grid utility companies to balance generation with loads over time. If this trend continues, then phenomena such as voltage fluctuations, reverse power flow, and degraded power quality are expected to increase in frequency, bringing higher costs to the customers while decreasing quality of service they receive.

To overcome these challenges, research have been underway to find alternative demand management strategies that can better integrate DERs. Transactive Energy (TE) is a framework that promises to achieve flexible, robust, and adaptive energy management systems that properly integrate DERs. Local energy markets are emerging as a tool for coordinating generation, storage, and consumption of energy from distributed resources. In combination with automation, they promise to provide an effective energy management framework that is fair and brings system-level savings. This thesis presents work towards advancing the practical implementation of economy focused TE systems. Specifically, on multi-agent systems with learning energy trading agents that exploits the cooperative-competitive nature of auction markets to dynamically balance of supply and demand in a local community.

A TE simulator is first developed to complete the subsequent tasks. Next, the

relationship between double auction market properties and reinforcement learning is examined. This is a critical step, as a poorly designed market may yield unintended behavior of market participants. Results show that the market must be truthful and weakly budget balanced in order for the agents to develop behaviours that reflect price theory, which is a necessary condition to generate strong and relevant reward signals.

Since the price generation process in the local energy market is fundamentally different from contemporary pricing schemes (such as time-of-use), a mathematical model that aggregates and converts individual agent policies to a global pricing scheme is created so that some properties of pricing schemes can be compared directly. In this study, it can be shown that the TE price model is far more responsive and relevant compared to time-of-use, which is suggests that agent behaviours can be more accurate and efficient when used for load shaping. Furthermore, significant bill reductions are achieved when compared to net billing. The community as a whole experienced a bill reduction of 35.9%, and the mean and median of individual bill reductions are 74.51% and 38.8%, respectively.

Finally, the dynamic power balancing aspect of the proposed TE system is studied with the integration of battery energy storage. A test circuit was created so that voltage violations exist, but cannot be eliminated or reduced via self sufficiency alone. By combining local energy trading with a battery storage, all of the voltage violations are completely eliminated.

Preface

The research presented in this thesis was performed under the supervision of Dr. Petr Musilek and the co-supervision of Dr. Mustafa Gul.

A version of Chapter 4 has been published in Energy and AI as "Reinforcement Learning-Driven Local Transactive Energy Market for Distributed Energy Resources" [1]. Daniel May developed the game theoretic models used in 4.4 and 4.5. I was responsible for the rest of the study, which includes major areas of concept formulation, agent design, market design and implementation, data collection and analysis, and manuscript composition. Dr. Petr Musilek was the supervisory author of this article and was involved throughout the project, and aided in the initial formulation of the topic, and the manuscript composition and refinement process.

The contents of Section 4.6.2 has not been published. It is meant to address a reviewer comment for the aforementioned article. This content was not a necessary addition to the article at the time of publication.

The contents of Section 4.6.3 has been published in Energies as "The Impact of Battery Storage on Power Flow and Economy in an Automated Transactive Energy Market" [2]. I conceived and designed the concept formulation, market design, battery model and control scheme, agent design, data collection and analysis, and manuscript composition. Dr. Petr Musilek was the supervisory author of this article and was involved throughout the project, and aided in the initial formulation of the topic, and the manuscript composition and refinement process.

Acknowledgements

I would like to thank my supervisor, Dr. Petr Musilek, for providing support, guidance, and feedback throughout my time as his student.

I would like to thank Dr. Mustafa Gul for your role as my co-supervisor, and for providing critical feedback for several publications.

I would like to thank my team mates, Daniel May and Peter Atrazhev, for existing.

Table of Contents

List of Tables

List of Figures

Abbreviations

ACE Agent-based computational economics.

AI Artificial intelligence.

ALEX Autonomous Local Energy Exchange.

ANN Artificial neural network.

API Application programming interface.

BB Budget balancing.

BESS Battery energy storage system.

DER Distributed energy resource.

DL Deep learning.

DP dynamic pricing.

DR Demand response.

DRL Deep reinforcement learning.

EE Economic efficiency.

EMH Efficient Market Hypothesis.

EV Electric vehicle.

GWAC GridWise Architecture Council.

HVAC heating, ventilation, and air conditioning.

IR Individual rationality.

LEM Local energy market.

ML Machine learning.

MPC Model predictive control.

NIST National Institute of Standards and Technology.

NN Neural network.

OPF Optimal power flow.

RES Renewable energy source.

RL Reinforcement learning.

ROI Return on investment.

RTP Real-time pricing.

TCL Thermostatically controlled load.

TF Truthfulness.

TOU Time-of-use.

T-REX Transactive Renewable Energy Exchange.

TC Transactive control.

TE Transactive energy.

Chapter 1

Introduction

1.1 Motivation

Technological breakthroughs in renewable power generation, battery storage, electric mobility, and advanced data logistics are changing the electric grid. The huge influx of distributed energy resources (DERs), while important to curb carbon emissions, is not without consequences. The highly intermittent nature of renewable energy sources (RES), combined with the decreased visibility of DERs (from the system operator's perspective), makes it increasingly difficult for the grid utility companies to balance generation with loads at the right time. If this trend continues, then phenomena such as voltage fluctuations, reverse power flow, and degraded power quality [3–5] are expected to increase in frequency, bringing higher costs to the customers while decreasing quality of life.

To overcome these challenges, works have been underway to find alternative demand management strategies that can better integrate DERs. Transactive Energy (TE) is one such framework that promises to achieve flexible, robust, and adaptive energy management systems that properly integrates DERs. The GridWise Architecture Council (GWAC) defines TE as "a set of economic and control mechanisms that allows the dynamic balance of supply and demand across the entire electrical infrastructure using value as a key operational parameter." [6]. This high level definition provides some clues as to how TE should work, but leaves the specific implementation

details to system designers.

Currently, one of the most popular forms of TE implementation is transactive control (TC), which aims to generate fine-grained price and/or control signals that can reach optimal power flow. However, as we will see in literature review presented in Chapter 2, this approach has the same limitations as existing demand management schemes. An alternative form of TE is the economy-based approach, which does not aim for optimal powerflow, but simply observes it as a consequence of efficient market transactions. This form of TE is the focus of this thesis.

1.2 Objectives

The ultimate objective of this thesis is to answer the following questions:

- What market mechanism and market properties will allow a set of independent reinforcement learning (RL) agents to trade energy in a way that reflects price theory?
- Can optimal, or near optimal power flow be achieved as a consequence of efficient market transactions?

The following tasks are completed as part of the process to answering these research questions:

- Developed a economy focused transactive energy simulator that enabled the completion of subsequent tasks.
- Designed and implemented methods that allow agents to trade energy in a double auction market.
- Examined the relationships between double auction market properties and reinforcement learning, specifically:
  - Investigated how the market properties affect RL policy development.

  - Determined the strength of each property required for RL policy to reflect price theory.

- Inspected the developed RL policies, and devised methods to make comparisons to contemporary demand response pricing scheme, specifically:
  - Created a generalized math model to aggregate and convert individual agent policies to a global pricing scheme.
  - Compared responsiveness, relevancy, and bill cost against net billing and time-of-use.

- Studied the effects of integrating battery storage in the local energy market, with a focus on changes to power flow:
  - Designed and implemented a rule-based battery management agent that maximizes market interactions.
  - Contrasted the resulting power flow with and without local energy trading.

1.3 Thesis Layout

This thesis is composed of five chapters. Chapter 2 presents relevant background, such as demand management techniques, transactive energy systems, agent-based economics, and reinforcement learning. Chapter 3 describes Transactive Renewable Energy Exchange (T-REX), the simulation software developed to perform economy focused TE simulations. Chapter 4 introduces Autonomous Local Energy Exchange (ALEX), the framework for studying market-based coordination in a behind-the-meter community. A series of experiments are performed using ALEX to answer the research questions posed in Section 1.2. Conclusions and future work are discussed in Chapter 5.

Chapter 2

Background

This section provides the necessary background on related topics and concepts that are used throughout the thesis. It also offers a literature review relevant to the presented research.

2.1 Energy and Markets

2.1.1 Demand Side Management

The addition of DERs, especially at grid edge, creates difficult problems for the electric grid. The foremost problem is the decreased visibility into the behaviour of these DERs, which in turn makes load balancing much more difficult. Improper load management can lead to phenomena that can affect everyday lives, such as voltage fluctuations, reverse power flow, and degraded power quality [3–5]. Utility companies use demand response (DR) techniques to alleviate some of these issues by trying to align customer and/or DER behaviours closer to ideal load curves.

Demand response methods can be divided into two categories: direct, and indirect. Direct DR refers to methods where the electric utility company or grid operator is given direct control over heavy appliances, such HVACs, to increase or decrease the load on the grid as needed. Direct DR is limited in scope due to the large infrastructure overhead [7–9]. It is also considered more invasive, and tends to create more friction between the utility company and customers, such as when turning up

thermostats during heatwaves to counter decreased line capacity. Indirect DR, on the other hand, tries to uses incentive signals to influence customer behaviour, with the hope that enough customers will respond to make a difference. Popular methods include time-of-use (TOU) pricing, critical peak pricing, real-time pricing (RTP), and dynamic pricing (DP) [7]. Unfortunately, a survey conducted by Chen et al. [8] concludes that customers generally prioritize comfort over responding to price signals. This means that the response rate will be lower when a high level of response is needed the most, rendering indirect DR largely ineffective.

2.1.2 Transactive Energy

Given the challenges and limitations facing DR, interest in transactive energy (TE) has been increasing. The GridWise Architecture Council (GWAC) officially defines TE as "a system of economic and control mechanisms that allows the dynamic balance of supply and demand across the entire electrical infrastructure using value as a key operational parameter" [6].

A large body of research exists on TE in the form of transactive control (TC), which uses the TE framework to generate a combination of direct and indirect control signals to reach optimal power flow (OPF). Typically, transactive control schedules are calculated via a utility function in conjunction with forecasts and expert-designed behavioral models [7–9]. Some notable TC approaches include:

- Works by Hu et al. [10], who proposed an aggregator-based optimization approach that generates charging/discharging schedules for electric vehicles (EVs). A simulation was performed on a Danish distribution network to show the decreased frequency of line congestion and voltage violation.

- Nazir et al. [11] used an aggregator-based model incorporated into a model predictive control (MPC) framework to both calculate optimal price signals and to control thermostatically controlled loads (TCLs) and storage devices to

decrease power oscillations at substation feeders.

- Soarez et al. [12] introduced a comparable aggregator-based approach using a dual decomposition algorithm. To validate the efficacy of the algorithm, they performed a field test involving six houses. Despite the promising simulation results, the field test was inconclusive due to the lack of flexibility at the test site.

A smart grid with a large number of independently operating DERs can be modelled as a non-stationary stochastic system. Since TC schemes are generally derived from deterministic models, optimal control cannot be guaranteed when the outputs are used on stochastic systems. This uncertainty of the load response to the control signal itself is the same challenge that indirect DR still faces [7, 8]. Although certain aspects of these challenges have been modeled by Yu et al. [13], Huang et al. [14], and Ferreira et al. [15] in an effort to find a solution, the fundamental weaknesses of schedule-based DR remain.

2.1.3 Agent-Based Computational Economics

TC largely focuses on creating schedules to reach optimal power flow, and relegates price signals as a control proxy that must be followed. The focus on making customers follow a strict schedule is one of the factors contributing to the failure of TC and other indirect DR approaches. Findings by Chen et al. [8] back up this claim, and suggest that automated responses will be the key to making DR and DR-like programs successful.

Unlike TC, economy-based TE does not create schedules that must be strictly followed to reach optimal power flow. However, power flow improvements may be observed as a result of efficient market transactions. This approach is more inline with the definition of TE, which suggests the use of economic principles to dynamically balance supply and demand. This also makes it possible to use the agent-based

computational economics (ACE) framework for studying and discovering alternative TE approaches, especially ones that are fully automated.

ACE is used in the study of open-ended dynamic systems of interacting agents, where agents may be defined as an independent entity capable of affecting the trajectory of outcomes for the system they inhabit. Real-world economies exhibit five essential properties [16, 17]:

1. The world consists of heterogeneous interacting participants;
2. The world dynamics are driven by successive interactions of participants;
3. Participant decisions take into account previous actions and potential future actions by other participants;
4. All participants are locally constructive;
5. Actions taken by participants at any given time affect future local states.

These properties imply that real-world economies can be modeled as locally-constructive sequential games.

2.1.4 Net Billing Vs. Net Metering

This section clarifies the distinction between net billing and net metering. In certain jurisdictions, such as Alberta, Canada, the electricity market is "unbundled". In simple terms, electric utilities are only in charge of building and operating the infrastructure (wires), and a multitude of retailers (which cannot be the same entity as the electric utility company) are allowed to sell electricity to end users, with almost complete freedom to set the rate of electricity. Customer bills are therefore also separated into two main components: infrastructure (transmission and distribution, or T&D fees[1], which can have a fixed component and a variable component), and

[1]T&D fees are not the only non-energy fees. Other fees can include administration, riders, property taxes, "adjustments", etc. In most cases, T&D fees makes up the vast majority of of the fees, and is therefore simpler and good enough to ignore the others.

energy. Under net metering, any electricity that flows into the meter (loads) incurs both energy and T&D costs, and any electricity that flows out of the meter (generation) has both energy and T&D costs deducted, either as credits or cash. Net billing is the same for loads, but only the energy component is deducted for generation. One way to avoid this infrastructure cost is to install both the solar panel and a battery behind the meter to minimize the amount of energy flowing out of the meter, effectively making net billing into net metering. The advantage of net billing is the socialized cost of infrastructure, which is more evenly divided amongst all customers. In contrast, net metering tends to shift these costs onto the segment of the population who cannot afford their own solar (this is called uneconomic bypass, a well known and often criticized problem). The disadvantage of net billing is that the return on investment (ROI) can be significantly longer due to less bill deductions. From this perspective, net billing is a more fair baseline. It also provides more opportunities for community-based energy management, such as through local energy markets described in Chapter 4.

2.1.5 Efficient Market Hypothesis

The Efficient Market Hypothesis (EMH) [18] states that the price of assets reflects all available information. EMH is usually applied to security markets, but should be applicable capital markets in general. It is believed that, in an efficient capital market, new information spreads quickly, and is then incorporated into the price with little to no delay. Real world markets that are operated by humans are not perfectly efficient, and the actors are not fully rational, which led to criticisms of the theory [19].

2.1.6 Double Auction

Double auction [20, 21] is a type of market that facilitates the process of buying and selling goods between multiple buyers and multiple sellers. A well known double auction market is the stock exchange.

For each auction round, potential sellers must submit their desired sell price (ask, or S) to the auctioneer. Likewise, potential buyers must submit their desired purchase price (bid, or B). The auctioneer then uses some settlement mechanism to find a settlement price, p, that maximizes the quantity of goods exchanged. There exists many types of settlement mechanisms [22], but all can be completely described by four properties [23]:

1. **Individual rationality (IR)**, which states that a participant will not join an auction where they will lose money.

2. **Economic efficiency (EE)**. In an economically efficient system, at the end of all trading, the highest bidder should be the winner and receive the item(s).

3. **Budget balancing (BB)**. Budget balancing comes in a strong variant (SBB) and a weak variant (WBB). In a SBB system, all money transacted go from the buyer to the seller. In a WBB system, a portion of the money transacted goes to the auctioneer.

4. **Truthfulness (TF)**. The dominant strategy for participants in a truthful market is report prices at what they believe should be the true value of the item to be exchanged. i.e., there is no incentive to lie.

According to the Myerson–Satterthwaite theorem [23], an ideal double auction market satisfies all four properties, but it is not possible in practice, even for markets with one buyer, one seller, and one item. Therefore, a practical market design must first determine the required properties, then make compromises for the rest. A market mechanism study is carried out in Section 4.6.1.

2.2 Artificial Intelligence and Machine Learning

2.2.1 Artificial Intelligence

Artificial Intelligence (AI) [24] is a field of computer science/engineering that focuses on creating intelligent machines, especially through computer software. In essence, AI takes some input data through one or many computational models to produce outputs that are useful. Today, the applications of AI is numerous, and include (but not exclusive to) search engines, product recommendations, speech recognition, playing competitive games (chess, go, StarCraft, etc.), image/feature recognition, autopilot, stock trading, and so on. The process for the computational models used in AI has evolved over time. Early on, AI applications used rule based systems that are highly situation specific, and required immense expert knowledge to design and implement. Later, fuzzy systems and similar augmentations simplified the process with more generalized rules. Then, simple machine learning techniques were introduced to extract knowledge from raw data. Today, the combination of sophisticated machine learning techniques, massive data sets, and powerful computer hardware is the primary way to build models for AI applications.

2.2.2 Machine Learning

Machine Learning (ML) is a branch of artificial intelligence in which algorithms are used to learn from data to create or improve a model (or models) for use in AI applications. ML can be categorized into three types: supervised learning, unsupervised learning, and semi-supervised learning, with the main difference being the amount of human influence on training data.

Supervised learning uses datasets that are completely labeled for training. Problems that uses supervised learning can be further broken down into classification, which attempts to assign data presented into a category or several categories, and regression, which is used to understand relationships between labels. Application

of supervised learning include computer vision (to accurately identify objects in an image. For example, in self driving vehicles), spam detection, sentiment analysis, predictive analysis, etc. K-fold cross validation is typically used to ensure that the trained model is generalizable and does not over- or underfit to the training data. The major drawback of supervised learning is the time and effort required by humans to label data, which, as the amount of data increases, can quickly become expensive.

Unlike supervised learning, **unsupervised learning** uses data that is completely unlabelled, and instead relies on the algorithm to identify important patterns and relationships (i.e, clustering). Clustering algorithms are mainly used to group data with similar features or patterns. Common application of clustering are: pattern recognition, anomaly detection, recommendation engines, etc. Outputs from unsupervised learning algorithms may be used as one of the steps for deep learning. For example, the data clusters may be treated as labels to further process the data through supervised learning algorithms. Similarly, dimensionality reduction (principal component analysis, autoencoders, etc.) can be used to create new representations of the data that may be more computationally efficient for training. Although using unlabelled data with unsupervised learning may be faster to start compared to supervised learning, the outputs are often less accurate, and may require extensive post-analysis to fully understand the results.

Semi-supervised learning uses a small set of labelled data and a larger set of unlabelled data for training. Unlike unsupervised learning, which could create clusters that are difficult for humans to understand, semi-supervised learning uses the labelled data to help label the unlabelled data. This approach attempts to strike a "happy medium" between supervised and unsupervised learning, and can solve the problem of either not having enough labelled data, or the potential high cost of labelling sufficient amounts of data.

As computers are expected to increasingly perform complex tasks in place of humans, Goodfellow et al. contend that "machine learning is the only viable approach to

building AI systems that can operate in complicated real-world environments." [24]. Two sub-fields of ML, deep learning and reinforcement learning (which can be combined into deep reinforcement learning), have recently been highly successful at training AI to perform tasks in complex, real-world environments.

2.2.3 Neural Networks

Neural networks (NNs), or artificial neural networks (ANNs), are a sub-field of machine learning. As the name suggests, neural networks are inspired by the human brain, and have structures that mimic the way signals are passed between biological neurons.

The base unit of a neural network is a node. Each node contains an activation function, which typically produces an output when some input threshold is reached. Connections between neurons can be augmented by weights. Multiple neurons can form a layer that takes multiple inputs and produces multiple outputs, and layers of neurons can be connected to form a network. The depth of a neural network is usually defined by the hidden layers (layers that are not the input or output of the network), and a network consisting of more than two hidden layers can be considered a deep neural network. Figure 2.1 shows the classic representation of a neural network.

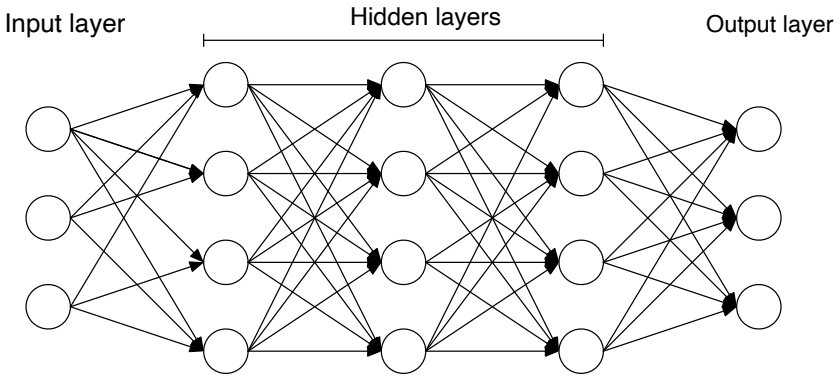

Figure 2.1: The classic representation of an "deep" neural network. More specifically, this is a feed-forward network, as the signals flow in one direction only, from input to output.

Neural networks are trained via supervised learning using back propagation. In basic terms, for each set of inputs that goes through the network, an error is produced by comparing the output to the known output. A backpropagation algorithm [25, 26] is then used to adjust the network weights to decrease this error.

An important property of neural networks is that they can be universal approximators [27], even for networks with a single hidden layer. Essentially, a number of neurons, each representing a simple math function, can be combined using arithmetic manipulation to form a more complex function. Therefore, any function can be accurately approximated as the number of neurons approach infinity. The combination the universal approximator theorem, more sophisticated network architectures, and the explosion of computational power, allowed neural networks and deep learning to achieve much success in many fields in recent years.

2.2.4 Deep Learning

Deep learning (DL) is a sub-field of neural networks that focuses on deep neural networks. One of the earliest and seminal works that demonstrated the usefulness of deep neural networks was published in 1989, in which LeCun et al. [28] used a neural network to recognize handwritten zip codes provided by USPS. It was demonstrated that a single network can learn the entire recognition operation, from image normalization, to character classification. More importantly, this demonstration hinted at the central principal of deep learning - complex concepts can be broken down into multiple simple concepts that build on each other, aka. learning hierarchical representations [29].

Figure 2.2 shows a more clear illustration of deep learning on image recognition, and how each subsequent layer learns new features from the outputs of the previous layer. A big advantage of feature learning over many hidden layers is that this process is unsupervised, making the data labelling requirement much less intensive. For example, only the objects need to be labelled for object recognition. There is no need

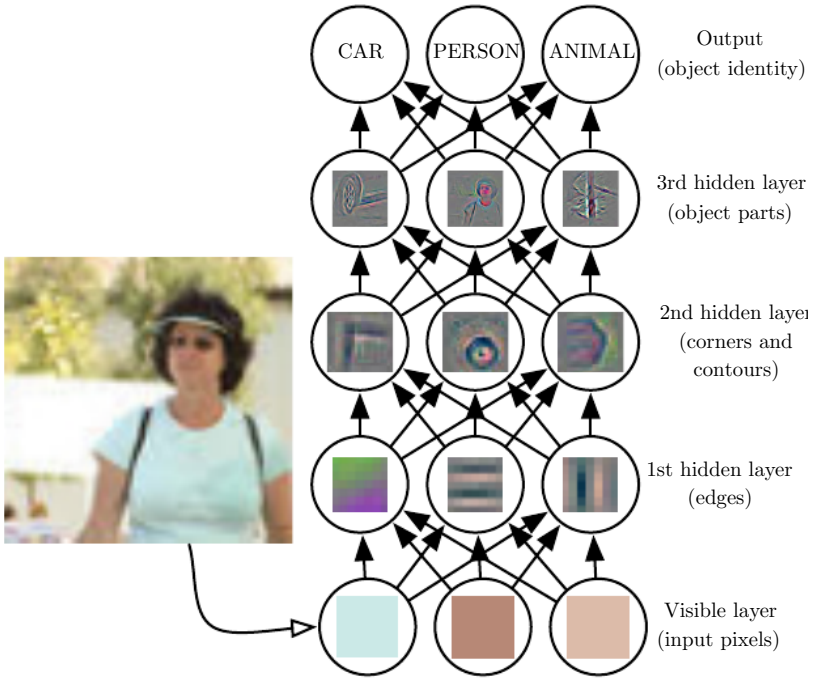

Figure 2.2: An illustration of deep learning. Originally from "Deep Learning" [24], pg. 6

for the network designer to manually label extract what they think the important features are needed in order for the network to learn. Over the years, deep learning models, such as ResNet [30], have achieved error rates that outperform human accuracy.

Despite these impressive feats, deep learning is notoriously "data-hungry". As illustrated by Alom et al. [31] in Figure 2.3, traditional machine learning approaches outperforms deep learning for smaller datasets. As the amount of data increases, DL performance drastically increases traditional approaches plateau. This property can be considered a double-edged sword, as it may potentially lock out the use of DL in industries where data generation or collection is minimal or restricted. Solving this problem is currently an active area of research.

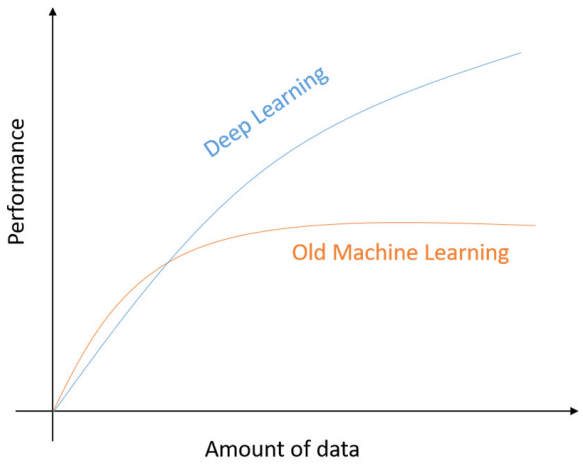

Figure 2.3: Deep learning performance vs. amount of data. Originally from "A State-of-the-Art Survey on Deep Learning Theory and Architectures" [31], pg. 7

2.2.5 Reinforcement Learning

There have been numerous approaches used to address the optimization problems inherent to demand side management [7–9], including mixed-integer programming, stochastic programming, and dynamic programming. After reinforcement learning (RL) demonstrated great competence in partially observable, stochastic game environments [32–34], it also gained popularity as a control method for DR [35]. The learned policy can substitute the solution of the equivalent optimization problem at each time step. As a result, RL approaches can be more computationally efficient at scale, when compared to conventional optimization methods.

At a high level, RL is a type of machine learning in which an entity, referred to as an agent, learns through trial-and-error interaction with its environment [36]. An agent performs actions and is provided feedback on the impact of these actions through a scalar value known as reward r_t. The goal is to maximize the sum of future rewards known collectively as the return $G_t = \sum_{t=0}^{\infty} r_t$. In addition to rewards, the agent also collects information about the environment through observations o_t. Through rewards and observations, agents learn policy $\pi(o_t)$ that guides their actions. To formulate this policy, agents keep tabs on the value of states, $V_t(s) = \mathbf{E}[G_t|S_t = s]$,

or quality of state action pairs, $Q_t(s, a) = \mathbf{E}[G_t|S_t = s, A_t = a]$.

Since RL agents learn through trial and error, they extract information about their environment through the feedback of their actions. Since agents seek to maximize return, they must have a mechanism to gain information about their environment; this is known as exploration. The balance between exploring the environment and exploiting information already known to the agent is of pivotal significance across RL learning algorithms. A popular way to create an exploratory drive is to inject a random action selection with a certain probability ϵ, this is known as *ϵ-greedy* exploration.

Figure 2.4: A typical representation of a reinforcement learning loop. In the RL setting, an agent learns to maximize the return G by interacting with its environment through actions a while receiving observations of the environmental state s. G is commonly defined as the expected, discounted cumulant of future reward R

$$G_t = \sum_{i=t}^{T} \gamma^{(i-t)} R_i, \ \forall \gamma \in [0...1], \tag{2.1}$$

where γ is the discount factor. The expected value of G_t, given the current state s_t or the current state-action tuple (s_t, a_t), is referred to as the state value

$$V(s_t) = \mathbb{E}\big(G_t|_{s_t,\pi(s_t)}\big), \tag{2.2}$$

or action value

$$Q(s_t, a_t) = \mathbb{E}\big(G_t|_{s_t,a_t}\big), \tag{2.3}$$

respectively. An RL agent acts according to a policy π

$$\pi : S \times A \to [0...1]. \tag{2.4}$$

Policy is a (probabilistic) mapping of the state space S on action space A, i.e.

$$\sum_a \pi(a, s_t) = 1. \tag{2.5}$$

This allows the definition of the state value V as action value Q weighted by π

$$V(s_t) = \frac{1}{n_a} \sum_a \pi(a, s_t) Q(s_t, a). \tag{2.6}$$

This system of equations (2.1-2.6) is sufficient to broadly classify all RL algorithms along two axes: the learned function and the relation between the target and behavior policy. According to the learned function, RL algorithms can be classified as policy-gradient methods and value-based methods. The policy-gradient methods directly learn the policy π, while the value-based methods learn estimations for either V or Q, and employ a fixed mapping of these values to π. RL algorithms can also be classified into on-policy and off-policy methods, by comparing their target and exploratory behavior. An on-policy RL algorithm explores the environment with the same policy that is optimized, while an off-policy algorithm explores the environment with a behavioral policy $b \neq \pi$.

2.2.6 Multi-Armed Bandit

A multi-armed bandit [37] is a type of problem where an actor chooses actions in a way that maximizes returns. The agent acts on partial information known about each action, and gains more knowledge about each action as they are taken. Multi-armed bandit is special case of reinforcement learning where the next state does not depend on the action taken in the previous step, and exemplifies the exploration-exploitation trade-off. The classic example of a multi-arm bandit is a gambler in front of a row of slot machines, where each slot machine may have a different payoff. At every step, gambler must decide whether to continue with the same machine, use the machine that is known to give the highest payoffs so far, or try a new machine. Any time a machine is used, information about the machine is updated, which helps the bandit make the next action. A form of multi-armed bandit is used in Section 4.6.1.

2.2.7 Q-Learning

Q-learning is a well-established, value-based, off-policy algorithm. It is named as such because the learning function calculates the quality of each state-action combination. It learns the greedy policy, a deterministic policy that always picks a corresponding to the largest Q, by following behavioral policy b. A popular choice for b is the ϵ-greedy policy, which takes a random action with probability ϵ and otherwise follows the greedy policy, as follows.

$$\pi(s) = \begin{cases} \text{random}_a Q(s,a) & \text{if } \epsilon \\ \text{argmax}_a Q(s,a) & \text{otherwise} \end{cases} \tag{2.7}$$

The $Q(s,a)$ values are usually stored in a lookup table. Value updates follow the Markov Decision Process (MDP), which allows $Q(s,a)$ to be expressed recursively as the Bellman's equation:

$$Q(s,a) = r(s,a) + \gamma \text{max}_{a'}(Q(s',a')) \tag{2.8}$$

The Q-Learning rule can then be written as:

$$Q^{\text{new}}(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha(r_t + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t)) \tag{2.9}$$

where α is the learning rate, r_t is the reward, and γ is the discount factor.

Q-learning is a relatively well-understood RL algorithm, and commonly used in the related literature reviewed in section 2.2.9. The Q-function has strong convergence criteria, as long as both ϵ and α are annealed towards 0 at infinity. However, the tabular-Q has two major weaknesses. First, the state table is inefficient, and only suitable for a small set of discrete states, whereas state representations for real world problems are far more complex. Second, unseen states are not updated, even if some properties of nearby states are shared. These weaknesses can apply to other tabular methods as well. While quantization of the state-action combinations may seem like a reasonable solution, it results in inefficient learning, largely due to the curse of

dimensionality. Hence, the most the real solution to these weaknesses is to use deep neural networks to approximate the state table.

2.2.8 Deep Reinforcement Learning

As previously mentioned, traditional reinforcement learning algorithms are generally unsuitable for solving real world problems. This is largely due to the large number of state representations, and the inefficiencies of tabular methods. The real solution to this problem is to replace the state-action table with a function approximator, usually with a deep neural network (hence, deep reinforcement learning). The Deep Q-networks algorithm (DQN) [38] is the first to successfully demonstrate the use of a deep reinforcement learning (DRL) to learn control policies directly from high-dimensional inputs. By combining Q-Learning, convolutional neural networks, and experience replay, the agent learned to play a few Atari 2600 games at superhuman-level performance. This work kick started a wave of DRL research, and since then, multiple extensions have been proposed to improve efficiency and performance (Double DQN [39], prioritized experience replay [40], dueling DQN [41], Distributional Q-Learning [42], etc.) Hessel et al. [43] combined six extensions (Rainbow) to achieve the same performance as the benchmarks faster, and better performance overall. DRL has since demonstrated superhuman performance in more complex domains [33, 34, 44], and continues to solidify itself as one of the best candidate to perform highly complex tasks in partially observable, stochastic environments, such as real-time double auction markets.

2.2.9 RL for Local Energy Markets

Several articles investigate the combination of RL and DP for centralized control. Notably, Kim et al. [45], and Lu et al. [46] develop RL-based approaches for DP from the perspective of a service retailer. Both articles address difficulties in predicting participant response to a pricing schedule by mitigating reliance on accurate cus-

tomer side information. A Markov decision process is formulated based on customer behaviour models and preferences. A Q-learning agent is trained to simultaneously minimize customer costs and maximize the service provider benefit. The two articles differ in the formulation of the reward function, which is generally a major influencing factor in RL algorithms. Lu et al. [46] use a weighted sum of retailer and customers, while Kim et al. [45] use a modelled utility function. Although both proposed approaches successfully implement a non-scheduling-based DP strategy, they still rely on modeling consumer behavior and preferences via utility functions.

Zhang et al. [47] train a RL agent to manage a community-shared battery and trade its resources in economy-maximizing fashion on a TE market. As such, the reward function is the economic performance of the battery. The authors show that even considering the running costs of the battery, positive economic benefits can be achieved.

Xiao et al. [48] investigate optimized trading between a large number of interconnected microgrids using a deep Q-network (DQN) based RL agent. Similarly to Kim et al. [45], a utility function is used to determine the reward. As expected, the more sophisticated DQN algorithm outperforms the benchmark hotbooting Q-learning algorithm.

Foruzan et al [49] investigate the behavior of self-interested Q-learning agents, exchanging energy within a micro grid over a LEM. The agent's goal is to maximize its own profit. Managed DERs include battery energy storage systems, solar rooftop, wind and diesel generators. The participant's stochastic behavior is approximated using random models. The authors perform an in-depth hyperparameter study of the RL algorithm with regards to return, self-sufficiency and fairness metrics and investigate several different micro grid configurations.

Zhou et al. [50] combine fuzzy rule-based systems with Q-learning to train agents to exchange energy resources over a peer-to-peer LEM setup whose pricing is directly tied to the ratio of supply and demand. They investigate the performance of several

community configurations with ranging number of battery energy storage systems and renewable generation assets. They show that such a system setup generally achieves lower bills than TOU and net-billing baselines.

Chen et al. [51] employ a DQN variant to automate the interactions of prosumers equipped with battery energy storage system in a LEM. The RL agents' action space consists of four distinct, discrete actions covering buy/sell and charge/discharge. Their learned policy outperforms an intuitive, rule-based strategy and a pure random policy equivalent to a zero-intelligence agent, originally proposed by Ghode et al. [52] as a baseline for agent competence in automated markets. In another article, Chen et al. [53] investigate the function of Q-learning based energy brokers as LEM consensus mechanism for settlements. The agent's reward is its profit. The authors perform several ablation and sensitivity studies and show that this the brokers efficiently learn how to maximize their own profit and the market's efficiency.

Kim et al. [54] extend a DQN variant designed for stock trading applications to manage a household's participation in a peer-to-peer energy exchange. They set up experiments including several different rate schemes and demonstrate that their developed agent outperforms the simplified versions in a set of experiments in terms of loss minimization and revenue maximization.

Bose et al. [55] focus on the emerging participant interaction within a fixed LEM setup under differing levels of DER penetration. They demonstrate that RL-based agents in such a setup can lead to the emergence of partial energy self-sufficiency, and further show that the degree of such self-sufficiency and the complexity of agent interactions depends on the degree of DER penetration within the LEM.

Mengelkamp et al. [56] study three different extensions of the Erev-Roth RL algorithm applied to automate LEM participation. They find that the extensions further increase the self-sufficiency of the LEM when compared to the original Erev-Roth algorithm proposed by Erev et al. [57].

Mengelkamp et al. [58] compare a peer-to-peer LEM against a closed book, double-

auction LEM with settlement rounds. They compare the performance of zero-intelligence agents and "intelligent" agents adopted from Nicolaisen et al. [59] on both LEM designs. They conclude that all market scenarios offer similar economic advantages, with the peer-to-peer LEM used by intelligent agents slightly outperforming the remaining variants. However, they also note that using the same strategy on a different market results in different price trends and conclude that the agent strategy and market design need to be co-developed to guarantee the system's performance.

Chapter 3

T-REX Simulation Software

3.1 Introduction

Most modern engineering solutions start with simulations in software. However, the simulators for power engineering have not evolved much over the years, and are inadequate for holistically studying TE in depth. A report by NIST [60] details four prominent TE simulators designed for the NIST Transactive Energy Modelling and Simulation Challenge for the Smart Grid, and summarizes the pros and cons of each. In addition, several other simulators exist that function in a similar vein: FNCS [61], TESP [62], IESM [63], SEPSS [64], C2WT-TE [65], and TE-SAT [66]. In general, these simulators primarily focus on solving for optimal power flow, and aim to produce external control signals that can steer human behaviours or devices to reach this state (aka. transactive control). Regardless of the complexity or detail of the signals generated this way, fundamentally speaking, this is not too different from existing DR techniques, and therefore still does not solve the problems with scalability or low customer participation rates.

One other obvious weakness of this collection of TE simulators is that, by primarily focusing on optimal power flow, the economic aspect is overlooked. As a reminder, TE is defined "a system of economic and control mechanisms that allows the dynamic balance of supply and demand across the entire electrical infrastructure using value as a key operational parameter" [6]. Although a small number of economy focused

TE simulators (and related research) exist [59, 67], they do not offer the flexibility required to easily integrate software defined markets with modern machine learning frameworks that can be used to create powerful agents. In order to conduct the research objectives for this thesis, such a simulator first had to be created. This chapter summarizes the simulator design and goes over some key topics, from the high level architecture, to implementations of some lower level functions. The latest stable release of the simulator is available on GitHub [68].

3.2 Design Requirements and Constraints

The end goal of the simulator is to enable a group of learning agents to manage DERs, using markets as a coordination mechanism. The name, T-REX, short for Transactive Renewable Energy Exchange, is carried over from an earlier design concept that had to be abandoned, because it was impossible to practically implement.

To study as many combinations of markets and agents as possible, the simulator must be modular. The simulation environment should also be as realistic as possible, and include real world limitations if necessary. In order to achieve these goals, the following requirements should be met at a minimum:

1. It must not be purely focused on solving for optimal power flow.
2. It must integrate well with modern ML frameworks, such as Tensorflow and PyTorch.
3. It should be based on the principles of agent-based economics.
4. Systems designed in the simulator should be deployable with minimal changes.

The last point is especially important, as practical TE systems may differ from theoretical designs and are often difficult to implement [12]. By adding the constraints for deployment, systems designed in T-REX should be more practical and easier to deploy.

When considering practical constraints, the biggest source is arguably the life cycle cost of the system. This usually involves the costs of developing, purchasing, installing, operating, and maintaining the hardware and software. Certain trade offs must be made in order to minimize this cost. These trade offs can greatly influence the overall system architecture.

3.3 System Architecture

This section provides a high level overview of the current system architecture. A simplified architecture diagram is shown in Figure 3.1, and an overview of a typical event flow diagram is shown in Figure 3.2.

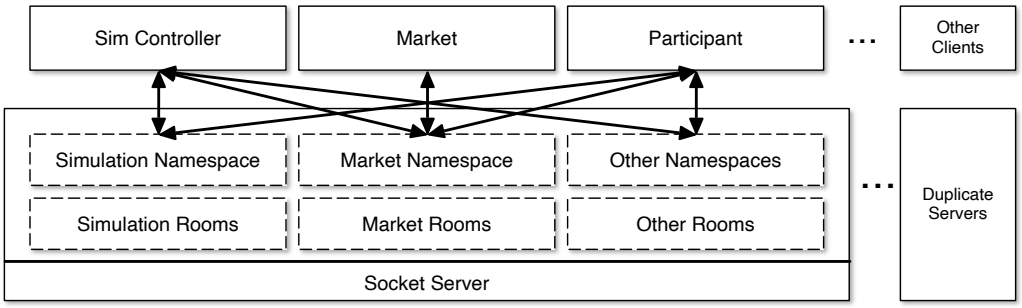

Figure 3.1: Simplified T-REX V3 Architecture Diagram

The event flow diagram shows the sequence of events that occur for a typical time step. As part of the design goals, the vast majority of the event sequencing is kept identical between simulation and real-time (deployment) mode. The major difference between the two modes is timing control. In deployment mode, timing is based on a global clock, and the market will advance when a set interval lapses, regardless of whether participants are able to complete their actions during this time. In simulation mode, the timing is controlled by the simulation controller. The simulation controller ensures that all agents have completed all of their actions before directing the market to advance. Beyond timing control, the simulation controller is also in charge of monitoring the health of all other modules, as well as controlling the train-

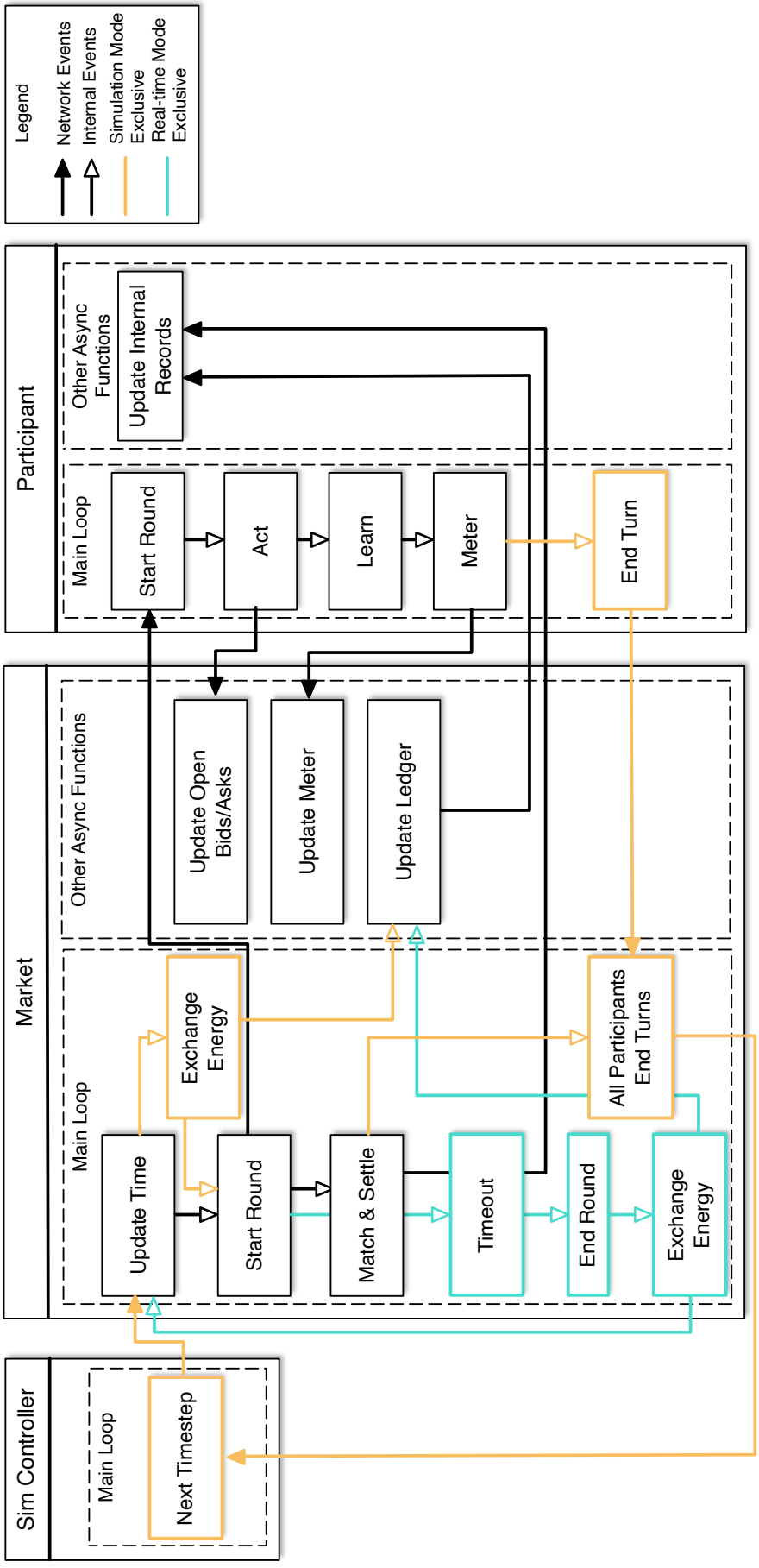

Figure 3.2: Overview of T-REX V3 Event Flow Diagram

ing curriculum. For example, signaling hyperparameter changes as the simulation progresses. More efficient agent designs results in shorter simulation times, and as a consequence, can allow tighter timings when operating in deployment mode, making sub-minute transactions viable (in contrast to the industry standard 15-minute trading intervals).

3.3.1 Networking and Scalability

One of the major bottlenecks of operating TE systems is the large amounts of data that need to be collected, processed, and transmitted. The amount of data available grows exponentially as we move towards the grid edge, making networking a critical cost factor when considering TE at distribution scale. Dedicated fiber networks are nearly non-existent on the distribution scale due to the prohibitively high cost, and consumer broadband networks may not be reliable enough to transmit time sensitive, high-frequency data required for TE, especially in rural areas.

An alternative that is both low cost and highly reliable is to build dedicated wireless mesh networks using LoRaWAN, which has become the industry standard. The 900MHz configuration is especially enticing, as it can transmit over several kilometers with greater penetrability over dense material. The drawback is the much lower bandwidth, which makes it unsuitable for centralized TE over large, densely populated areas. It was decided that decentralized TE over smaller, densely populated areas is more suitable for this particular setup, and may be most effective application of distribution level TE.

To further increase ease of development and deployment, the rest of the system is build on top of a hardware agnostic network software stack, `socket.io` [69]. `Socket.io` is widely used in applications and services where a large number of users must be served in parallel, and can scale well for decentralized TE. This is one of the reasons that `socket.io` is used instead of more traditional Python parallel computing libraries, such as Ray. Since the first alpha in early 2018, T-REX has not

encountered any scaling issues thus far. Most of the performance constraints come from the CPU core count limitations for multiprocessing, and physical media write speeds when logging large amounts of data.

3.3.2 Servers and Clients

A typical T-REX environment consists of one `socket.io` server and multiple clients, as shown in Figure 3.1. Client modules, and the interactions between them is purely through `socket.io`'s messaging API. Currently, three main types of clients are implemented, as described below:

- Participant modules, which are in charge of energy trading and managing energy resources that are directly accessible. Participants are, for example, households and self-driving EVs.
- Non-participant modules, e.g., the TE market. The market facilitates the discovery and exchange of energy between participants.
- Simulation-only modules, e.g., the simulation controller. The simulation controller augments the deployment environment to form a simulation model. It can also perform advanced functions such as training curricula for ML applications.

With a few restrictions pertaining to the simulation controller, the number of modules of each type is unlimited. The functions are also not restricted to the list described above. For example, a power flow co-simulator can be added to calculate power flow in real-time based on the market transactions.

3.3.3 Bridge Server

Building T-REX on top of `socket.io` makes it easy to run T-REX simulations on one computer, or multiple computers across a local network or the internet. However, sometimes it is not possible to directly use `socket.io` as the communication

protocol. For example, some LoRa-based modules cannot communicate over TCP/IP, and instead rely on MQTT for messaging. Bridge servers are designed to translate `socket.io` events and messages to a comparable format in other communication protocols, with a relatively low computational cost. The architecture diagram for a T-REX system using bridge servers is shown in Figure 3.3.

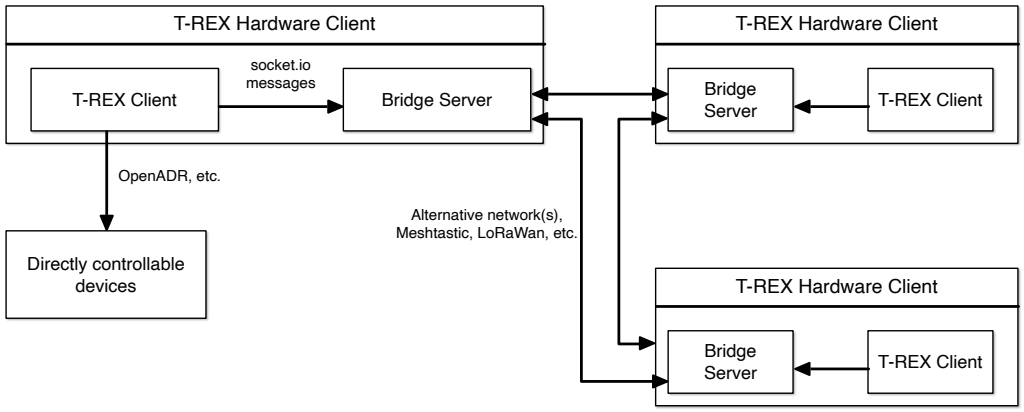

Figure 3.3: T-REX Bridge Architecture Diagram

As the architecture diagram shows, the typical application of bridge servers is when connecting multiple, single client machines. The single, centralized `socket.io` server shown in Figure 3.1, is replaced by a bridge server running concurrently on each machine. Aside from enabling more networking options, bridge servers is another modular design concept that makes sure T-REX remains internally consistent, so that the same code can run in any hardware configuration. Like other specialized features, bridge servers can be easily enabled or disabled through the configuration file.

3.4 T-REX Software Functions

3.4.1 Configuration File

T-REX simulations are configured using a single, json format file. To remain consistent, this file is also used for other co-simulation or post-simulation pipelines. Not all

parameters will be used for all types of applications, but as much relevant information should be included as possible for record keeping and repeatability. As the simulator is under constant development, new features may be introduced, and old features may be modified or deprecated. In order to keep track of the feature changes, a version check is performed by the simulator prior to initialization.

Depending on base configuration file and the type of simulation to be performed, the T-REX launcher may further modify the configurations to be simulation specific. For example, a baseline simulation will always be performed for only one episode, using the built-in baseline agent for all traders. Similarly, validation runs will use the same trader type as the simulation run, but the learning flags will always be "false". For record keeping purposes, the original and modified configurations will be stored inside the simulation database.

The rest of this section explains the purposes of some typical configurations in the style of a code documentation. The complete configuration file used is available in Appendix A.

3.4.2 Study Parameters

Parameters used to perform specific studies are configured under the "study" section. An example of the section that includes all the configurable parameters are as follows.

```
"study": {
    "name": "descriptive_name",
    "description": "long_description",
    "start_datetime": "YYYY-MM-DD hh:mm:ss",
    "start_datetime_sequence": "sequential",
    "timezone": "America/Vancouver",
    "days": 1,
    "time_step_size": 3600,
```

```
    "generations": 10
}
```

Study Name {`name`}

The name of the study to be performed should be short and descriptive. The study name is also used to create database, table, and directory names for the simulation database, and subdirectories in the simulation root folder. While the data created by T-REX will be saved to the database, some exceptions, such as TensorFlow weights or TensorBoard data must be saved in the file system. Having a consistent naming scheme helps with organization, and makes automated tasks that require access to data from multiple locations much easier.

Study Description {`description`}

An optional, long description of the study can be stored in this section to expand upon the study name. The text in this section is not used anywhere else, and is purely for note keeping.

Start Time and Date {`start_datetime`}

The start time and date dictates the beginning of the data stream for each episode. Wherever possible, any human readable time strings must be encoded in the ISO 8601 format, i.e. YYYY-MM-DD hh:mm:ss, alongside a timezone. This is required so that T-REX can accurately convert time from a human readable format to UNIX timestamps for internal use. This also removes any potential "false gaps" or time repetitions caused by daylight savings changes, leap seconds, etc.

For most situation, starting each episode at the same point in time is sufficient. However, there are certain situations where different starting points for each episode may be required. T-REX allows this to be configured in two ways using a list of timestamps. Note that "episodes" and "generations" are currently used interchangeably.

1. If the list contains only two elements, i.e. $[dt_1, dt_2]$, a list of timestamps equalling to the number of generations will be created, using dt_1 and dt_2 as range limits.

2. If the list contains more than two elements, i.e. $[dt_1, dt_2, ..., dt_n]$, a list of start times will be created in one of the following ways:

   (a) If the number of elements is greater than the number of generations, the list is truncated so that its length is equal to the number of generations

   (b) If the number of elements is less than the number of generations, the list is repeated until the number of elements is equal to or longer than the number of generations, then truncated to the number of generations if necessary

   (c) If the number of elements is equal to the number of generations, no further modifications are necessary

Once this list of start times are created, the simulation controller will transmit the corresponding start times for each episode.

Start Time Sequence {`start_datetime_sequence`}

This parameter is only used if `start_datetime` is a list containing multiple timestamps. If the text is set to *sequential*, then the start times of each episode will follow the list sequence. If the text is set to *shuffled*, then the list will be shuffled once. The default option is *sequential*.

Time Zone {`timezone`}

Time zones must be declared alongside a human readable time string, following the format defined in the IANA time zone database, i.e. "America/Vancouver". This allows accurate conversions to UNIX timestamps for internal use, and removes the most common sources of "false gaps" created by day light savings, leap seconds, etc.

Number of Days to be Simulated per Generation {`days`}

This parameter defines the number of days that should be simulated for each episode. Since T-REX uses one-minute time intervals by default for all internal operations, decimal days are accepted, and will be rounded to the nearest number of minutes.

Number of Seconds per Step {`time_step_size`}

Timing control in T-REX is globally aligned with the data time intervals in energy profiles. By default, T-REX expects energy readings in one-minute intervals. However, data from external sources often have lower time resolutions (5, 15, and 60 minute intervals are common). Although a data processor is available to convert this data to one-minute intervals, the conversion process is time consuming. Furthermore, simulation and training efficiency may be negatively affected due to the repeat presentation of the same data. This parameter allows the default time interval to be manually adjusted to align with the timestamps in existing profiles. Currently, it is up to the user to ensure that all the energy profiles used in the simulation have the same time interval. If a check is implemented to automate this process in the future, this parameter will be removed.

Number of Generations to be Simulated {`generations`}

The definition of an episode for reinforcement learning [36] is used here. In the context T-REX, a generation and an episode is synonymous. A generation is

Data Locations {`profiles_db_location`}

```
"study": {
    "profiles_db_location": "postgresql://un:pws@localhost/profiles",
    "output_db_location": "postgresql://un:pw@localhost",
    "sim_root": "/home/trex/sim/"
}
```

As T-REX works mostly with time-series data in a time-sequential nature, a relational database (i.e. PostgreSQL, TimescaleDB, SQLite, etc.) is used to provide quick and efficient access to energy profiles. Additional tools are available to convert data from outside vendors to be suitable for immediate use by T-REX. The tools can be found on GitHub [70]. Similar to the profile database, most data generated by simulations is stored in a relational database. Only the base location is necessary here, as a database will be created and used for each simulation based on the study name.

The example given uses the simple authentication method, where the username and password are given in the same string in plain text. This is only recommended for low security situations [1]. Follow the documentation provided by the database engine of your choice if more security is required.

Although T-REX tries to make the most use of relational databases, there are certain situations where the file system must be used, i.e. TensorFlow saved weights or TensorBoard files. These files are stored under the directory defined by `sim_root`. For organization reasons, a subdirectory that is the same as the study name will be created and used for each simulation. If this parameter is undefined, then the T-REX Python working directory will be used.

3.4.3 Server Parameters

```
"server": {
    "host": "localhost",
    "port": "5100"
}
```

In order to prevent cross-talk between simulations, a unique combination of `socket.io` server address and port should exist for each running T-REX instance. If the server

[1] To make configurations more easily publishable, this parameter, along with certain others, can be stored in a separate file. See the updated documentations on GitHub for details on how to do this

address is undefined, then *"localhost"* will be used by default. If a T-REX simulation is performed on multiple machines, then the server address should be set to the IP address of the machine that the server is running on to enable network access. As of configuration version 3.7.0, defining a unique port each running simulation is optional, but still recommended. In the event that two configuration files share the same port, the T-REX launcher will attempt to detect if any defined port is already in use, and increment the port number until a free one is found.

3.4.4 Training Parameters

Agent training is an important part of T-REX, and the reason for its creation. In some reinforcement learning environments, hyperparameter adjustments are usually hidden within the agents themselves, and could make inspection difficult if one is unfamiliar with the underlying code. Great effort has been put into T-REX to expose these hyperparameters in the configuration file in an easily understandable manner. As of configuration version 3.7.0, two major features are available: hyperparameter search, and training curriculum.

Hyperparameter Search {`hyperparameters`}

```
"training": {
    "hyperparameters": {
        "alpha_critic": {"start": 0.1, "stop": 0.2, "num": 2},
        "alpha_actor": 0.1
    }
}
```

Hyperparameter search is one of the most crucial and time consuming tasks in machine learning, as a well tuned set of hyperparameters can drastically improve performance. This is usually accomplished via grid search. The T-REX runner creates a number of simulations based on combinations of all elements in all hyperparameters

to be searched, then performs them in parallel.

The hyperparameter to search, range of search, and resolution are defined using the same keyword arguments for `np.linspace`, rounded to four decimal places. Only hyperparameters that are defined for the agent will be tuned. In the above example, the list of *alpha_critic* to be searched is $[0.1, 0.2]$. Since *alpha_actor* is set to a single value, any agents that have *alpha_actor* as a hyperparameter will have the value overwritten as 0.1, unless otherwise defined in the training curriculum.

Training Curriculum {`curriculum`}

```
"training": {
    "curriculum": {
        "0": {
            "learning": false
        },
        "1": {
            "learning": true,
            "exploration_factor": 0.5
        },
        "50": {
            "anneal": {
                "exploration_factor": [0.01, "subtract", 0.05],
                "alpha_actor": [0.98, "multiply", 1e-7]}
        }
    }
}
```

In addition to hyperparameter search, agent performance can be influenced by changing hyperparameters over time, such as annealing learning rates. As previously mentioned, training curriculum is often embedded inside the agents, which adds lay-

ers of obscurity and makes inspections difficult. T-REX allows the user to program the training curriculum in a clear manner in the configuration file. The simulation controller and the training controller will then instruct the agents to make the appropriate changes during run time.

The training curriculum can be programmed in two layers. The first layer denotes the episode that any changes should be made, and the second layer is what and how the changes should be made. Any changes made is carried through to all subsequent generations until another change to the same parameter/hyperparameter is made later on. The episode number must be an integer string to conform to `JSON` formatting.

In the example above, learning is turned off for all agents in episode 0 to establish a random policy baseline. Learning is turned on again in episode 1 and exploration factor is set to 0.5. As training continues, it may make sense to start annealing some hyperparameters. Episode 50 shows two ways of annealing. Exploration factor is annealed by subtracting 0.01 every episode starting from episode 50 until it reaches 0.05. Similarly, *alpha_actor* is annealed by multiplying by 0.98 every episode starting from episode 50 until it reaches $1e-7$. This gives the user an easy way to implement both a linear and an exponential decay curve.

3.4.5 Market Parameters

```
"market": {
    "id": "M3B1",
    "type": "MicroTE3B",
    "close_steps": 2
    }
}
```

One of the focuses of T-REX is to enable market mechanisms to be written in code. Unlike market designs written in prose, markets written in code leave no room for different interpretations, which is critical for training RL agents. The market section

of the configuration is the place for setting market parameters, so that the market module can assemble and link submodules appropriately during run time.

The top level parameters are for the main market. A unique market ID is required to run in deployment mode, but is also recommended for simulation mode for clarity. If the ID is not set, then the T-REX runner will create an ID based on the simulation type. In the example above, MicroTE3B is used as the main market mechanism, which is the double auction based transactive energy market. The market specific parameter, *close_steps*, is set to 2 to indicate that settlements occur 2 rounds before delivery.

Default Retail Parameters {grid}

```
"market": {
    "grid": {
        "price": 0.069,
        "fee_ratio": 1.1
    }
}
```

The current implementations assume that local energy markets are imperfect and cannot fully balance energy using local DERs alone. As a consequence, the electricity grid must be relied upon to compensate for these imperfections by either sinking the excess energy, or supplying any shortages. These energy compensations follow net billing rules by default, and is implemented under the *grid* section.

In the net billing setting, energy from the grid is billed as the sum of energy price plus fees, and energy going into the grid is paid as the energy price only. Fees consists of many components, such as transmission, distribution, delivery, riders, adjustments, property taxes, etc. Although the exact fee components can be can be implemented, often times it is sufficient to calculate the sum of fees as a ratio of the base energy price. This implementation is shown in the example above.

If fees are not defined, then the net metering setting is assumed to be in effect, and the fees are included in the base energy price.

Time-of-use Schedule {tou}

```
"market": {
    "tou": {
        "5,6,7,8,9,10": {
            "7,8,9,10,17,18": 0.144,
            "11,12,13,14,15,16": 0.208
        },
    "1,2,3,4,11,12": {
        "11,12,13,14,15,16": 0.144,
        "7,8,9,10,17,18": 0.208
        }
    }
}
```

The default retail pricing can be further augmented to be time-of-use. Time-of-use schedules is defined using a double nested dictionary, with the first layer of keys setting the months, and second layer of keys setting the hours. The example above shows a how a time-of-use schedule similar to Ontario's implementation can be set up. The first half follows the summer schedule, which means that it is in effect from May to October. Peak rate is between 11AM and 4PM, and the rate is $0.208/kWh. Mid-peak rate is from 7AM to 10AM and 5PM to 6PM, and the rate is $0.144/kWh. The hours undefined is assumed to be off-peak, and uses the base energy price defined in the *grid* section as the rate. The second half follows the winter schedule, which is in effect November to April. The peak rate and mid-peak rate are switched from the summer schedule for the winter.

3.4.6 Participant Parameters

The parameters for each participant in the simulation must be configured individually. The configurable parameters include, but are not limited to: participant type, energy profiles to use, DERs attached, and agent related parameters.

Participant Type `type`

```
"participants": {
    "R5": {
        "type": "Residential"
    }
}
```

The type of participant (i.e. residential, commercial, farm, hospital, etc.) must be properly identified. In some market configurations, different energy rates and fees may be a applied depending on the participant types. For example, a *commercial* type may incur demand charges, whereas a *residential* type will not. Differentiating the types of participant could also influence the types of strategies that agents will develop. For example, agents for commercial customers may adopt a more peak-shaving focused strategy to avoid demand charges by bidding at a higher price than residential customers, thus creating more price incentive for selling to large, peaky loads. Participant type differentiation also allows markets with priority levels to be designed and studied, especially when mixed participant type trading is allowed. For example, if a hospital and a store bids at the same price, then the hospital may be prioritized.

Trader Parameters {`trader`}

```
"participants": {
    "R5": {
        "trader": {
```

```
            "type": "qlearn_bandit",
            "bid_price": 0.07,
            "ask_price": 0.14,
            "learning": true,
            "reward_function": "net_profit",
            "learning_rate": 0.1,
            "exploration_factor": 0.1,
            "track_metrics": true,
            "use_synthetic_profile": "test_profile_1kw_constant_gl"
        }
    }
}
```

The trader section defines the agent type used, as well as any hyperparameters associated with it. In general, agent types state the type of learning algorithms used, but may also be a description of its purpose (e.g., "basic_trader" is an agent that acts in accordance to net metering). T-REX agents are implemented as self-contained modules, and must exist in the "_agent/traders" directory. The module name must be identical (case sensitive) to the trader type to be usable.

Similar to agents, reward functions are also implemented as independent modules, and must exist in the "_agent/rewards" directory to be usable. Since rewards play a large role in reinforcement learning, separating the reward calculations away from the core functionalities of the agent makes it easy to pair different learning algorithms with different reward functions to see the outcomes. Currently, two reward functions are available as part of the T-REX package: net profit, and relative advantage. Net profit calculates the net profit of transactions occurred the last round, and relative advantage calculates the difference between net profit and the equivalent situation without the local energy market.

Aside from trader type, any other variable can be configured in this section as long as it exists as a keyword argument for the agent itself. Any hyperparameter that need to be tuned through hyperparameter search must also be set with a default value here. These variables will be passed into the trader module as keyword arguments during initialization.

Two special parameters to note are *track_metrics* and *use_synthetic_profile*. T-REX has a built-in system that allows user-defined metrics to be streamed to the T-REX database system in real-time, which is useful for analysis during simulations. This option lets the user choose whether to save those metrics on an agent-by-agent basis. The use of a standard SQL database for storage makes it easy to develop third party tools, such as real-time visualizations.

use_synthetic_profile is an option that allows an energy profile that is different from the participant ID to be used. This is useful in early testing, as synthetic profiles are more predictable and less noisy compared to real profiles, and makes algorithm performance evaluations easier.

Profile Augmentations

```
"participants": {
    "R5": {
        "load": {
            "scale": 1
        },
        "generation": {
            "scale": 10
        }
    }
}
```

Scaling the energy profiles is an easy way to augment the overall system supply/de-

mand without additional computational resources. Currently, it is possible to scale the amplitude of both the load and generation profiles of individual participants. Profile scaling happens at run time, and is applied after the load and generation of the relevant timestamp is read from the database.

Energy Storage Device `storage`

```
"participants": {
    "R5": {
        "storage": {
            "type": "Bess",
            "capacity": 10000,
            "power": 5000,
            "efficiency": 0.95,
            "monthly_sdr": 0.05
        }
    }
}
```

Storage devices are implemented in a similar way to traders, as independent modules. The storage *type* must be a Python module that exists in the "_devices" directory in order to be usable. The rest of the parameters are passed into the storage device module as keyword arguments, if applicable. All storage modules, regardless of type, are controlled the same way via scheduling. This is both a simplification of the control system that is easily usable by the agents, and an attempt at abstracting and unifying the control interface of devices from different vendors.

The above example is the configurations for a battery energy storage system (BESS). These parameters are derived using data sheets provided by BESS vendors, such as Tesla and Siemens, and should be a fair representation of BESS systems available on the market at the moment of writing. The battery's usable capacity and

continuous power ratings are defined in units of watt hours and watts, respectively.

For simplicity, batteries are currently designed to operate on a programmable schedule. The current design paradigm allows agents to schedule a fixed amount of energy to be charged or discharged over some interval in the future, and is assumed to be at constant power. The schedule can be freely adjusted until one time step before execution. The energy quantity to be scheduled will be automatically capped by the battery power limit and the predicted state of charge at the end of the scheduled interval.

The existing battery module uses an ideal battery model to calculate the state of charge based on the total energy charged or discharged over a certain time interval. This largely corresponds to the linear region of a real battery. It is assumed that the battery management system will keep the battery operating in this region, and therefore the ideal model approximation is sufficiently accurate for most cases. Additional battery modules that use more accurate charge and discharge curves can be easily implemented following the template in the code repository.

The last two options shown in the example are battery efficiency and self discharge rates. These parameters make the battery behaviour more realistic and can also affect economic calculations, which may affect agent behaviour. The battery efficiency is set as one-way efficiency, and can be squared to obtain the round-trip efficiency. Note that since T-REX measures energy at the meter, battery efficiency directly affects the state of charge (SoC) after each battery action. When efficiency is less than 1, battery needs to discharge more than measured at the meter. Similarly, battery SoC increases less than measured at the meter for charging.

Self discharge rate is usually represented as a percentage decrease per month. This is internally converted to Wh loss per minute, using 28 days/month[2] for convenience. Since the SDR is small, this approximation is acceptable.

[2]28 days/month is exactly 4 weeks, which is computationally simple

3.5 Using T-REX

3.5.1 Running a Simulation

Assuming the project directory is set up correctly, running a T-REX simulation is as simple as executing main.py, which typically looks as follows:

```
if __name__ == '__main__':
    from _utils.runner.runner import Runner
    runner = Runner(config='config_name', resume=False, purge=True)
    simulations = {
        'baseline',
        'training',
        'validation'
    }
    runner.run(simulations)
```

The configuration files to be used must be under the "_configs" folder in the working directory. The configuration name must be the same as the file name (case sensitive). Simulation lists the different types of simulations that can be run in parallel. The T-REX runner will modify the configuration file for each simulation type during simulation startup according to the type of simulation. In general, it is recommended to perform both base line and training at minimum. As of version 3.7.3, the runner automatically sets a number of parallel simulation to be launched based on the number of CPU cores available and the number of subprocesses used for each simulation. At the same time, the simulation controller has been modified so that it is able to reuse the existing clients for a different simulation once the current set is complete. This is especially useful for hyperparameter search, where previously a large number of parallel simulations would launch and cause the operating system to thread overflow.

3.5.2 Output Data

T-REX simulations operate on the principles of agent-based economics, and therefore primarily focus on market interactions. Every simulation will produce a set of market transactions with the following headers:

- Transaction ID
- Transaction Quantity (Wh)
- Seller ID
- Buyer ID
- Energy source
- Settled sell price
- Settled buy price
- Ask fees
- Bid fees
- Energy creation time
- Energy purchase time
- Energy consumption time

Each row of information is produced using a combination of trading data and metered data. The entire data set is stored in a relational database, such as PostgreSQL. So far, this set of information has been sufficient for a variety of post-processors to extract/recreate data for analysis. Minimizing the amount of data required is overall beneficial for the system, and can result in decreased network and storage usage, and increased data privacy.

3.5.3 Simulation Controller

The event flow diagram in Figure 3.2 shows two modes of operation: real-time mode, and simulation mode. In real-time mode, the market has control of advancing system timing based on a predefined interval, which is 60 seconds by default. However, advancing simulations at a set time interval is not ideal. Setting the interval longer is safer, but can create unnecessary wait times that prolong simulations. Setting the interval shorter may shorten simulation time, but may fail to handle edge cases where some processes take longer than expected, potentially causing the entire simulation to fail. More precise timing control results in faster and more reliable simulations, and is one of the primary reasons for creation of the simulation controller.

In essence, the simulation controller inserts itself into the market timing loop, checks the statuses of all connected clients, and only allows the simulation to continue if everything reports normal. The simulation controller may also issue commands at certain checkpoints (for example, changing the training curriculum). Status checking happens in two ways: first, all connected clients are required to report a status update at certain checkpoints. Second, the simulation controller performs periodic status checks as a background process. The simulation will be held until every client successfully reports reaching the required checkpoint. If the simulation fails to continue after five minutes (in real time), then the background status check loop will begin to broadcast special messages at fixed intervals to attempt to revive the simulation. If the simulation fails to revive after five minutes, then a system status will be printed out at five minute intervals (while still broadcasting the revival messages), to help developers find potential failure points.

3.5.4 Data Processors and Extensions

A set of post-processors has been created to extract and reconstruct simulation data for analysis, and used throughout this thesis [70]. Functions include extracting transaction summaries for specific time intervals, recreating load profiles from market

transactions, etc. With a few exceptions, these post-processors are highly specialized, and therefore are kept separately from the core. Update and maintenance should be done by individual contributors. Individual contributors may also submit an request to make their tools visible on an index page in the T-REX GitHub Wiki.

Data Ingest Pipeline

The data ingest pipeline is one of the most critical preprocessing steps to ensure that T-REX simulations can be executed quickly and consistently, regardless of the the modules used to assemble the simulation[3].

Since different data vendors store data in different formats, it is necessary to convert them to a common format for later access. We have currently made two data ingestors available, for eGauge and the SunDance [71] data set. As we work with data from more vendors, the appropriate data ingestors will be created and released.

Regardless of the original data format, the current ingest pipeline follows these steps:

1. Study the original data format

2. If the original data is readable as a table format, continue to the next step. Otherwise, create code that first converts the data into a readable table format.

3. Analyze timestamp sequence, and convert to UNIX timestamp format by converting the time to UTC+timezone and removing daylight savings. This ensures that time is continuous and at fixed intervals.

4. Import and process the data, add "generation" and "consumption" fields with corresponding values.

5. Record the converted data to a relational database, such as PostgreSQL.

[3]Assuming that custom modules follow the templates provided in the source code

6. Write metadata regarding each imported profile into the corresponding table(s). Metadata may include timezone, geographic coordinates, profile statistics, etc.

Synthetic Profile Generators

Even though T-REX is designed to work with real data, it is sometime advantageous to work with synthetic energy profiles with well known properties to remove potential sources of noise when developing new learning algorithms.

Currently, three types of commonly used profiles can be generated: flat, cosine, and square. To generate any profile, the following inputs are required:

- Profile start date/time as a string, formatted as "YYYY-MM-DD hh:mm:ss"
- Profile end date/time as a string, formatted as "YYYY-MM-DD hh:mm:ss"
- Profile time zone in IANA time zone format, e.g., "America/Vancouver"
- Peak power in watts

For the cosine and square profiles, two additional inputs may be used to set the period and offset. By default, the offset is zero and the period is 1440. As the profiles are generated in one minute intervals, a period of 1440 means that the profile pattern is repeated daily. The peak power is used to set the magnitude of the profile in watt hours. For a one minute interval profile, the power is divided by 60.

Statistical Measures of Profiles

Although the data ingest pipeline introduced previously attempts to create relevant metadata, it is sometimes the case that additional data or metadata is required after the fact. Typically, the way to deal with this requirement is to do this data processing during run-time, since the additional data may only be needed for experimental agents, and the relatively low cost computation time is not worth the trade off of using more data storage until much later in time. However, there are cases where

processing on-the-fly is not possible. Normalizing the energy profiles as an observation is one such case, which requires all rows of data to be read to generate the relevant statistics. This functionality is not available to any of the existing T-REX clients, since they are only designed to process incoming data streams.

One way of producing normalized and directly comparable observations from sets of data with vastly different magnitudes is with standard scores, or z-scores. The z-score measures the distance between the original value and the mean of the data in units of standard deviation, and can be calculated as

$$z = \frac{x - \mu}{\sigma} \tag{3.1}$$

where μ is the mean of the data, and σ is the standard deviation of the data.

The data processors provided in the TREX-Tools repository can calculate the statistical measures of a given energy profile, then store them in the "statistics" table in the profiles database. PostgreSQL has built-in functions for calculating the mean and standard deviation, ensuring minimal time cost even for extremely large number of rows of data. These statistical measures allows the z-score of each measurement to be calculated at run time. Calculating z-scores at run time is overall more efficient than pre-calculating them, as the math is simple and computationally cheap. Furthermore, it avoids the need to completely recalculate the z-scores when new data is appended, which can be time consuming and takes a lot of storage.

3.5.5 Power Flow Simulations

Since the main focus of T-REX is market based TE, power flow simulations is not integral to the simulation pipeline, and is instead implemented as a separate module for post-simulation analysis. Currently, power flow analysis is done using OpenDSS [72], a free and open source tool created and maintained by EPRI. OpenDSS was chosen because it is one of the few free power flow simulators capable of solving North American unbalanced distribution circuits, has multiple Python interfaces [73, 74], and can

achieve results nearly identical to commercial software [75].

To perform post-simulation power flow analysis, a data processor is first used to convert market transactions recorded on the database into the appropriate time-series load profiles for OpenDSS. Then, a separate script loads up the matching circuit model and performs time-series power flow analysis. Finally, a few plotters can be used to visualize the results for inspection. This workflow is used in Section 4.6.3.

As previously mentioned, power flow co-simulations is a secondary option in T-REX. The primary reason behind this design decision is that acquiring accurate power flow data in deployment, on-site, in real-time is only feasible in rare circumstances. Therefore, it is highly difficult to directly deploy TE solutions that rely on data from power flow co-simulators. Despite this drawback, however, have the option of acquiring power flow data through co-simulations is still a useful way to compare performance or study system sensitivity in the presence/absence of data. In order to add power flow co-simulations, the "powerflow" section of the configuration file must be filled. This will launch a power flow client alongside the other clients, and enable relevant events and messages on the server. An example of the configuration section is as follows.

```
"powerflow": {
    "circuit_model": "557_NALV_RES",
    "node_map":{
        "P_x": "N0",
        "P_y": "N1"
        }
    }
```

Currently, the name of the circuit model and a node map are required. The number of nodes in the circuit model should be greater than or equal to the number of participants, and a default load or load profile should be present for each node. The

node map defines the physical location of each participant on the circuit model, and is needed even if the participant name can be found in the circuit model.

Chapter 4

ALEX: Autonomous Local Energy Exchange

4.1 Introduction

The previous chapter described T-REX as the simulation software to study TE systems. This chapter introduces ALEX, which is an implementation of a TE system for studying specific system behaviours. ALEX is intended to be used as a framework to validate whether decentralized agents can, through the use of a partially observable market mechanism, achieve DER coordination on an arbitrary electrical system. The outcomes of these studies may help find solutions that address critical weaknesses of existing DR approaches, such as the need to model customer and system specific utility functions, or the relatively rigid schedules that do not adjust well to differences between predicted and actual behaviours.

4.2 Core Concept

Conceptually, ALEX is structured as a behind-the-meter community. This community operates in two layers. The inside layer consists of all the participants, which are allowed to exchange energy with each other over a local energy market. Energy rates for energy exchanged this way are determined through the local market mechanism. The outside layer is the interface between the community and the rest of the grid. Energy exchanged this way uses standard retail rates as determined by the electricity

retailer or the electric utility company, depending on the jurisdiction.

To further implement ALEX, the following assumptions have been made:

- Participants are self-interested and, therefore, prioritize their own economic well-being and comfort in the decision making process.
- Participants are willing to automate some, or all decision making regarding interactions with indirect DR measures (e.g., using RL agents).
- Each participating unit is equipped with a smart meter, and has collected a sufficient amount of high-resolution historical data to train the RL agents.
- The electricity grid that the community is connected to is an infinite bus. This means that it can supply or sink the deficient/excess energy from the community without causing violations.

4.3 Market Mechanism

Like any market-based approach, it is critical for the the market used in ALEX to accurately reflect price theory. Price theory states that the price of any specific good or service only comes from the balance of supply and demand at the time of exchange. To accomplish this, it is reasonable to start with a well known market design, and make adjustments to adapt to ALEX specific considerations:

1. **Suitability for electricity grids with high penetration of DER and RES.** This means that, from a high level perspective, a market (or a collection of markets) must be able to effectively target localization and the intermittent nature of RES.
2. **Technical constraints and requirements of deployment:** Data acquisition, transportation, storage, and overall lifetime cost must be minimized to be viable for deployment.

3. **Machine learning considerations for agents:** Related to the point above, artificial intelligence will play an important role in trading and managing of energy resources. For this reason, the market should be conducive to machine learning (especially reinforcement learning). One way to achieve this is to compose the market with a small set of explicit rules, with no exceptions, no loopholes, and no room for different interpretations.The rules should also provide enough flexibility to offer large action spaces. The Market should provide strong feedback signals for learning.

The market implemented is based on double auctions [20, 21], which is a well known, widely used design that provides the scalability and efficiency required. The following is an description of the market rules for ALEX:

1. Rates for energy exchanged with the grid is done through net billing. Using retail electricity prices in Alberta as a reference, buying energy from the grid costs \$0.1449/kWh, and selling earns \$0.069/kWh[1].

2. Auctions settle for energy to be delivered during the one-round period from the end of the current round. However, the delivery period can be adjusted during run-time for future design explorations.

3. During the current round, participants can submit bids and asks for energy to be delivered during or beyond the next delivery period. Not joining the local energy market means defaulting to the retail electricity prices.

4. Bids/asks are settled pairwise, with bids sorted from the highest to lowest, and asks in reverse to ensure pareto equality.

5. Bid/ask quantities can be partially settled[2].

[1] As mentioned in Chapter 3, the grid price model can and should be adjusted to best reflect the jurisdiction where ALEX will be deployed

[2] The market is capable of settling both integer and floating point quantities, but the implementation is dependendent on the next rule.

6. A bid/ask quantity must be an integer multiple of 1 Wh. This is in consideration of hardware integration, to allow direct use of the watt-pulse function of most smart-meters[3].

7. During the delivery period, if a seller is in short supply, it must financially compensate for the shortage by buying the difference from the grid at retail prices. If batteries are available, the seller has the option to first compensate by discharging its batteries for all or part of the shortage during this period.

8. During the delivery period, if a buyer settled for more energy than used, the buyer must still pay the seller for the unused energy at the settlement price.

The major difference between this version of the market and a standard double auction market is that pool settlements is replaced by pairwise settlements. This borrows from peer-to-peer markets and allows the reward signals received by agents to be more personalized, and should help develop policies that are better adapted to each participant. Also note that specific mechanisms for determining settlement prices are intentionally left out, as the suitability of settlement mechanisms is a major study performed using ALEX. As will be shown later, subtle differences in the settlement mechanism can have drastic impacts on the policies developed by the agents and overall market behaviour, and therefore should be empirically studied for every market design.

4.4 Market Interaction

This section provides a more detailed look at how agents interact with the market. Currently, the market operates with fixed settlement frequency Δt at fixed intervals $[t, t + \Delta t)$. During each interval, participants can communicate their intention to trade energy by submitting bids

$$\text{bid}_t = (q_t^{\text{bid}}, p_t^{\text{bid}}), q \in [0...q_{\text{max}}^{\text{bid}}], p \in [p_{\text{min}}, ...p_{\text{max}}], \tag{4.1}$$

[3]Future variations may allow the use of floating point values, depending on hardware capabilities.

and asks

$$\text{ask}_t = (q_t^{\text{ask}}, p_t^{\text{ask}}), q \in [0...q_{\text{max}}^{\text{ask}}], p \in [p_{\text{min}}, ...p_{\text{max}}], \tag{4.2}$$

where bid_t and ask_t communicate the intention to buy or sell energy, respectively.

Bids and asks are represented by tuples consisting of the desired quantity q and desired price p of energy to be exchanged. Quantities are expressed in Wh as integers. The settlement signal m_t is represented as a list of tuples containing the settled quantities, which the agents may use to calculate a reward at the end of each round.

Participants only receive relevant information about their settlements and therefore, do not have access to information on the behaviour of other participants. This omission is intentionally implemented in order to avoid targeted exploitation.

$$m_t = (q_t^{\text{settlement}}, p_t^{\text{settlement}}) \tag{4.3}$$

In contrast to most other DR approaches, which use price signals as a form of open-loop control, the market design for ALEX is effectively a closed-loop control system. In this market, participant actions directly affect the price signals in real-time, and the new price, in turn, determine the subsequent actions. For a fully automated system, closed-loop control is proven to be superior.

After energy exchanges within the community are concluded, the grid (as an infinite bus) is used to compensate for any surpluses or shortages using net billing. Surplus energy is sold to the grid at price $p_{\text{sell}}^{\text{grid}}$ and deficient energy is purchased from the grid for price $p_{\text{buy}}^{\text{grid}}$, including fees. This setup grants the community as a whole to effectively avoid a portion of the fees by increasing self sufficiency. The natural bounds of the local market prices then become

$$p_{\text{min}} = p_{\text{sell}}^{\text{grid}} <= p_{\text{market}} <= p_{\text{buy}}^{\text{grid}} = p_{\text{max}}, \tag{4.4}$$

For a graphical overview, Figure 4.1 shows a sequence diagram of the interactions between the server, market, and agents in T-REX.

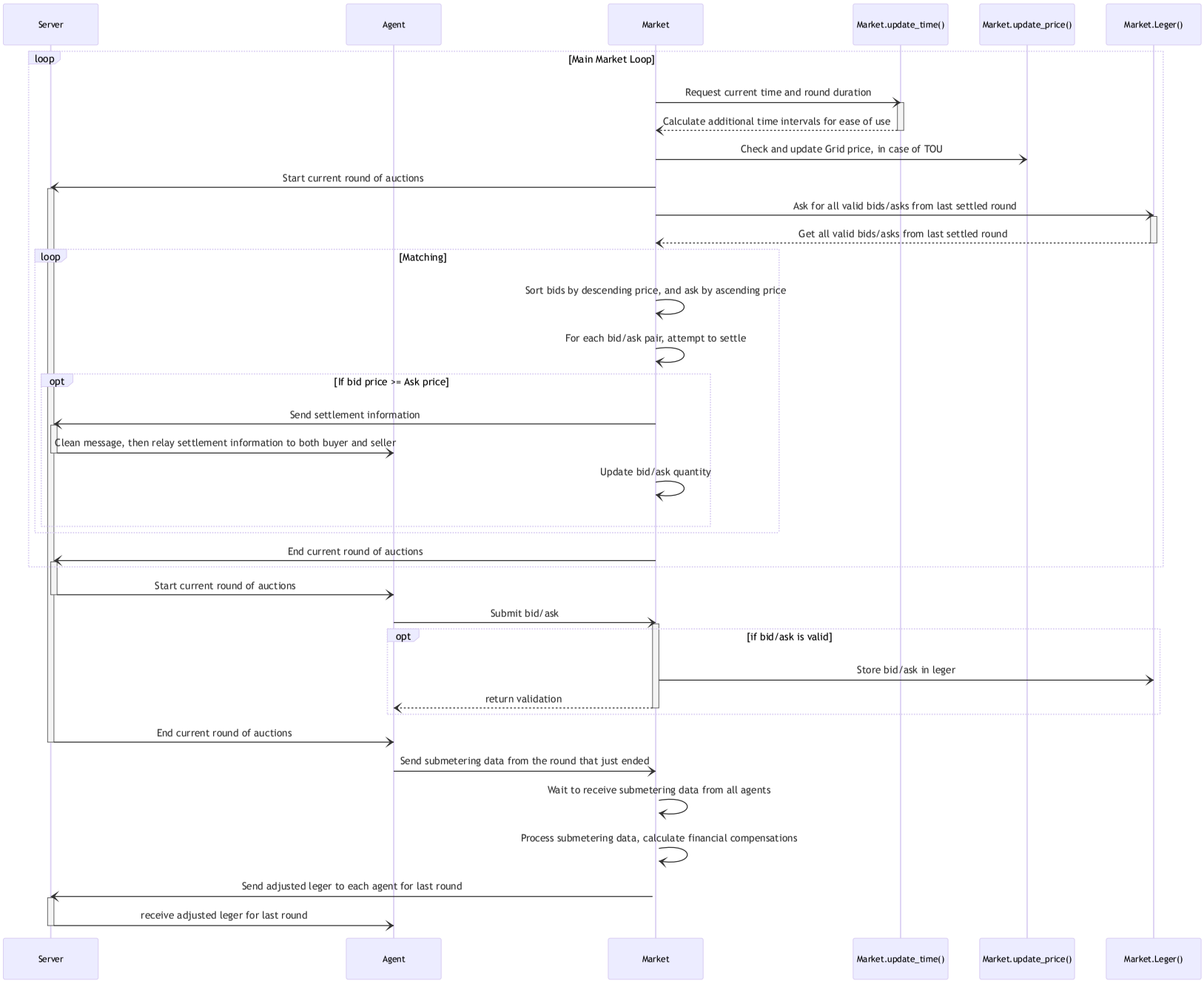

Figure 4.1: Sequence diagram of ALEX market interactions

4.5 ALEX as a Stochastic Game

Although T-REX is designed to emphasize empirical analysis in accordance to the principles of agent-based computational economics, it may still be useful to model ALEX as a discounted stochastic game to allow game theory to be used for theoretical analysis to supplement the empirical results.

$$\Gamma \coloneqq (n, L, S, A, P, R) \;\; \forall t \in [0...T], \lambda \in (0...1), \tag{4.5}$$

where n is the number of players, L is the list players of length $|L| = n$, S is the state space, A is the action space, P represents the state transition probabilities, R is the reward function, t is the current time step over the modelling period $[0...T]$, and λ is the discount factor.

Both S and A can be decomposed into n individual components S^i and A^i, as shown below

$$S = S^1 \times ... \times S^n, \tag{4.6}$$

$$A = A^1 \times ... \times A^n. \tag{4.7}$$

Superscript i refers to a specific individual L^i, while the subscript is reserved for time t. The action space A is separated in notation from a specific set of actions a_t at time step t, as in the commonly used RL nomenclature [36].

State transition probabilities are defined for any set of actions a_t taken at time step t, as follows

$$\forall a_t : \; P(S_{t+1}|S_t, a_t) \coloneqq S_t \rightarrow S_{t+1} \tag{4.8}$$

The reward or payoff in the stochastic game at time step t is defined analogously to the RL setting, by

$$R_t \coloneqq S \times A \rightarrow r, \tag{4.9}$$

which maps from (S_t, a_t) to a real number $r \in \mathbb{R}$. Similarly, each agent aims to maximize their own return G_t. Thus, all participants use their individually developed policy π^i, to determine action set a_t^i based on observations from S_t^i.

At each time step, all agents can interact with the market by submitting bids (4.1) and asks (4.2). This leads to the following definition of action

$$a_t^i = (\text{bid}_t^i, \text{ask}_t^i, e_t^i), \tag{4.10}$$

where the additional parameter, e_t^i, is contingency for expansion of the model, e.g., to define nonmarket actions, such as battery management and/or thermal load control.

Finally, the state observations for each agent are defined as follows

$$S^i = (d_t^i, g_t^i, m_{t-1}^i), \tag{4.11}$$

where d_t^i and g_t^i are, respectively, the load demand and generation at time t, and m_{t-1}^i are settlements received at time $t-1$.

Note that the transition probabilities P result from the collective actions of all agents. However, due to the pairwise settlement mechanism, market design, and the observation space, P is not fully accessible to L^i. This ensures that the developed model is a truly stochastic game. At least one stable Nash equilibrium is guaranteed to exist within Γ, as long as n, S and A are finite. This condition can be guaranteed by limiting prices p to a reasonable degree of accuracy (e.g., 4 or 5 significant digits commonly used in banking). A is logically bounded by the condition previously defined by (4.4). As a result, S must also be finite, therefore guaranteeing at least one stable Nash equilibrium for each instance of ALEX.

4.6 ALEX Experiments

This section details the three experiments performed under ALEX to progressively answer the research questions outlined in Chapter 1. As a reminder, the objectives for each experiment are as follows:

1. The first experiment studies the market properties required for the reinforcement learning agents to learn trading policies that can properly reflect price theory.

2. The second experiment compares economic advantages against time-of-use pricing and net billing by mathematically modelling an equivalent global price scheme for ALEX.

3. The last experiment adds energy storage as load shaping devices. More advanced agents are used to perform both trading and storage control. The resulting power flow will be compared to the scenario where agents follow a commonly used greedy policy, which only focuses on achieving self sufficiency.

4.6.1 Determining the Optimal Settlement Mechanism

One key takeaway in Chapter 2 is that automation is critical in increasing the effectiveness of DR when high penetration levels of DERs are involved. This is one of the reasons why ALEX is designed to be fully automated. Because humans have no input into any control decisions, it is paramount for the agents to learn to make the right decisions to adapt to human behaviour and market behaviour.

Predicting human behaviour via load profiles is a well studied topic, and relatively simple for modern machine learning techniques, especially since the data stream is fully observable. On the other hand, agent-to-market behaviour is more complex. Depending on the implementation, the environment (market) is more likely to be only partially observable, and when only a handful of other competing agents are involved, their individual behaviours can have drastic influences on the environment. Finally, as agents learn to compete against each other, the same action performed likely will not end in the same state, making the environment nonstationary and even more complex to learn.

All these complexities can be solved with more sophisticated machine learning

techniques. However, a prerequisite condition must be fulfilled: any interactions between agents through the market must reflect price theory. Therefore, the focus of this experiment is to find the market mechanisms required for reinforcement learning agents to develop policies that properly and truly reflect price theory.

Experimental Design

The market rules described earlier in Section 4.3 is used as the the starting point for the settlement mechanism designs to be studied. For this experiment, three different mechanisms are designed for testing, and are described as follows:

1. Average-Price (M1): Trades are settled if the bid price is greater than or equal to ask price. The settlement price is the average of the bid and ask prices.

2. Exact-Match (M2): Sellers and buyers can choose bid and ask prices from a list of 100 available prices. Trades are settled if and only if the bid price equals the ask price.

3. Action-Price (M3): Trades are settled if the bid price is greater than or equal to the ask price. The buyer buys from the auctioneer at the bid price, and the seller sells to the auctioneer at the ask price.

Because the market in ALEX is based on double auction, it can be completely described by the same properties described in Section 2.1.6. Use reinforcement learning agents as the only participants in this market simplifies the parameter testing process, as both individual rationality and economic efficiency can be guaranteed[4]. This reduces the number of candidate designs to be tested from 16 to 4 for complete coverage, since only budget balancing and truthfulness are relevant in differentiating the settlement mechanisms. The properties for the settlement mechanisms tested for this experiment are summarized in Table 4.1. Note that a market with weak budget

[4]Assuming that the agent is designed to maximize the users' financial gain and comfort.

balancing and no truthfulness is strictly worse than M1, and therefore excluded from the experiment.

	Market Property Settings			
Mechanism	**Individual rationality**	**Economic efficiency**	**Budget balancing**	**Truthfulness**
M1	Yes	Yes	Strong	False
M2	Yes	Yes	Strong	True
M3	Yes	Yes	Weak	True

Table 4.1: Settlement mechanism properties

Methods and Procedure

Three ratios of supply/demand are evaluated for all the settlement mechanisms tested: over-supply (10:1), over-demand (1:10), and equal (1:1). The emerging market behaviour for bid, ask, and settlement prices as a result of policies developed by the agents are plotted as probability distributions. The market that reflects the law of supply and demand will be considered the most fitting candidate for subsequent work, and is expected to produce the following results:

- Over-supply: The sellers should compete for demand, and drive ask prices low. Bid prices should be slightly higher than ask prices.
- Over-demand: The buyers should compete for supply, and drive bid prices high. Ask prices should be slightly lower than bid prices.
- Equal supply and demand: The bid and ask prices should converge around the middle of the available price range. Bid prices should be somewhat higher than this value, and ask prices should be somewhat lower than this value.

In order to avoid the emergence monopolistic behaviour by any agent, at least two buyers ($d^i > g^i$) and two sellers ($d^i < g^i$) must exist to maintain competition on both

sides of the market. For this reason, a set of $n = 4$ learning participants is used. Since the task for this experiment is only to find equilibrium pricing, only steady-state (flat, time-invariant) energy profiles are employed, with the supply or demand scales adjusted to match the ratios previously given.

Reducing the task to only finding equilibrium pricing collapses the observation space for each agent to a single point and fixes q^i_{bid} or q^i_{ask} to the residual load, which means that only price policy needs to be learned. From the view of individual agents, the experiment becomes a partially observable, nonstationary multi-armed bandit, where the number of arms corresponds to a number of discrete price actions $|p|$. For these experiments, a set of 100 discrete price points between \$0.07 and \$0.14 (both inclusive) are available for the agents to choose from. Independent tabular Q-learning is used for learning, with α set to 0.1, γ to 0.98, and ϵ to 0.1. This setup maintains loose convergence guarantees despite the properties of the resulting environment [76].

The reward function used for learning is the net profit of the last delivery interval, which is calculated as *total profit - total cost*. The sources of total profit/cost can each be broken down into three components: local market, the grid, and financial compensations. Each reward component is derived from the ledger module that keeps track of successful transactions. For example, equations 4.12 and 4.12 show the calculations for the profit and cost from successful transactions on the local market.

$$\text{LMProfit}_t = \sum_{i=1}^{N} q_i^{\text{ask}} \times p_i^{\text{ask}} \tag{4.12}$$

$$\text{LMCost}_t = \sum_{i=1}^{N} q_i^{\text{bid}} \times p_i^{\text{bid}} \tag{4.13}$$

Where t is the time interval, N is the number of transactions recorded, q is the quantity of energy transacted in Wh, and p is the settlement price in \$/kWh.

The grid and financial compensation components can be calculated in a similar way. The specific implementation of reward calculations can be found in the source code.

Results and Discussion

The results for the experiments are shown in Figures 4.2 through 4.4.

For settlement mechanism M1, we can observe that the bid, ask, and settlement prices closely converge near the low end of the price range for the over-supply scenario, and near the high end of the price range for the over-demand scenario. These observations fit the first and second conditions described previously. However, the balanced supply to demand scenario shows that the bid and ask prices diverge to the limits of the price range, and therefore does not fulfil the third condition. This phenomenon is likely attributed to the lack of truthfulness of this market design. Since the settlement price is the mean of the each bid/ask pair, as long as a trade is settled, both parties benefit more than the expectation (i.e., buyer pays less than the bid price, and seller makes more than the ask price). Because of this, the agents have no incentive to truthfully report what they believe is the true value of the asset to be traded, and will instead lie to maximize the chances to making a settlement. This result suggests that truthfulness is a necessary market property.

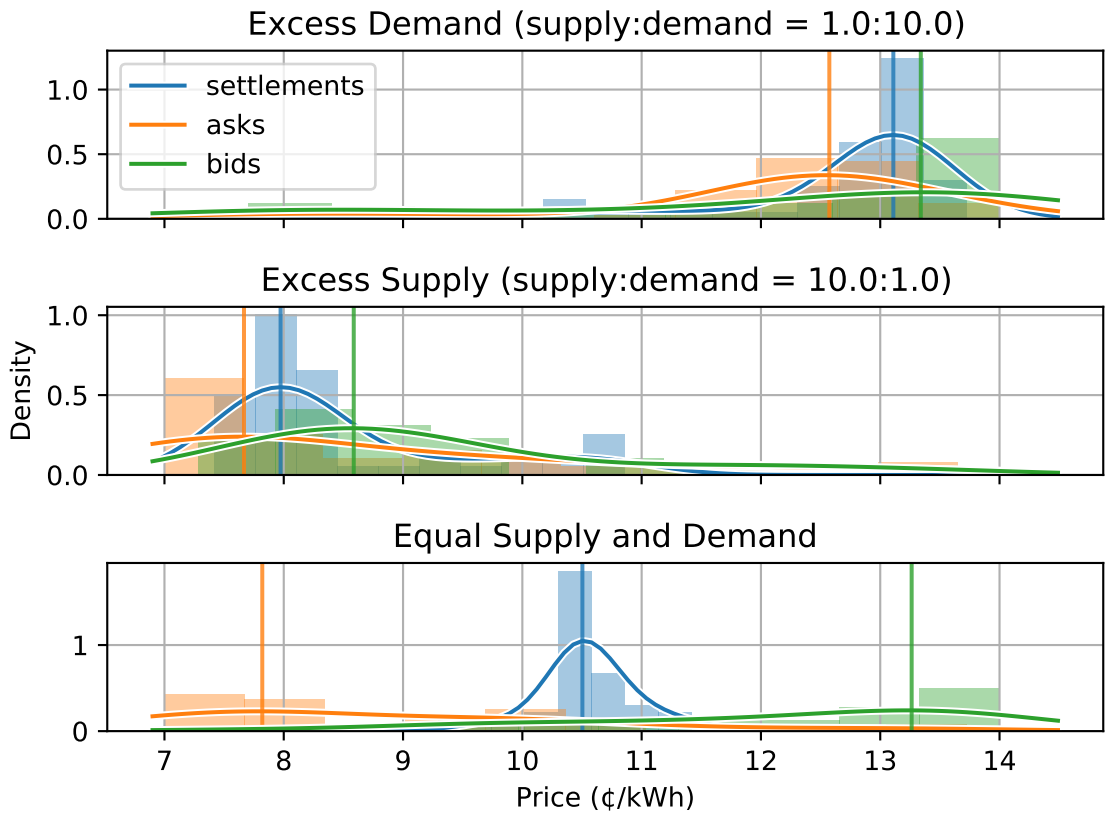

Figure 4.2: Validation policies for bid, ask, and resulting settlement prices for agents operating under M1 for episodes 70 to 100. The histograms show the probabilities of the discrete action policies for the agents and the resulting settlement prices. Modes of the prices are highlighted and shown by the vertical dashed lines. Probability density functions of the histograms are overlaid on top, which are approximated with the Gaussian KDE function in the scikit-learn Python package with default parameters.

Fig. 4.3 shows the results for settlement mechanism M2. The density plots for the prices appear very flat, with no clear indications of convergence. These plots suggest that this market design does not meet any of the aforementioned criteria for reflecting price theory, which is especially apparent for the balanced scenario where the average bid and ask prices are on the opposite sides of where they are expected to be. One possible explanation is that the strong budget balancing drastically decreases the number of successful settlements, which in turn increases the sparsity of rewards, making the training time given insufficient. Another possibility may be a much simpler explanation: M2 satisfies the conditions to be an ideal double auction market.

As a reminder, the Myerson–Satterthwaite theorem [23] states that a perfect double auction is impossible to practically implement, even for markets with one buyer, one seller, and one item. The results for M2 is empirical proof of this theorem, and also suggests that budget balancing should be weakened.

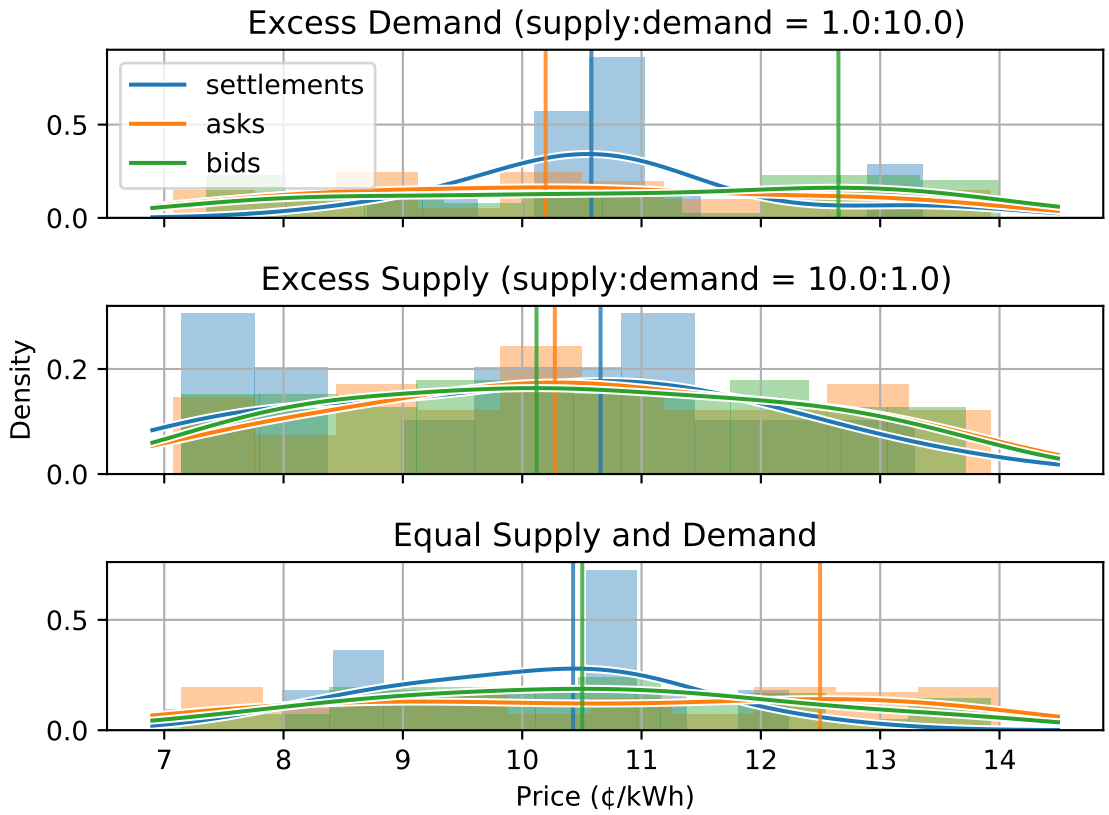

Figure 4.3: Validation policies for bid, ask, and resulting settlement prices for agents operating under M2 for episodes 70 to 100. The histograms show the probabilities of the discrete action policies for the agents and the resulting settlement prices. Modes of the prices are highlighted and shown by the vertical dashed lines. Probability density functions of the histograms are overlaid on top, which are approximated with the Gaussian KDE function in the scikit-learn Python package with default parameters.

Fig. 4.4 shows the results for settlement mechanism M3. Similar to M1, the prices converge towards the low end of the price range for the over-supply scenario, and the high end of the price range for the over-demand scenario. However, the average bid, ask, and settlement prices are clustered closer together compared to M1. Furthermore, in contrast to M1, price convergence can be observed for the balanced supply-demand

scenario as well, with the prices clustered closely near the middle of the available price range. Clearly, this market design satisfies all the criteria for reflecting price theory, and suggests that a double auction market most suitable for reinforcement learning should be truthful and weakly budget balanced. This particular market design is used for subsequent experiments conducted in this chapter.

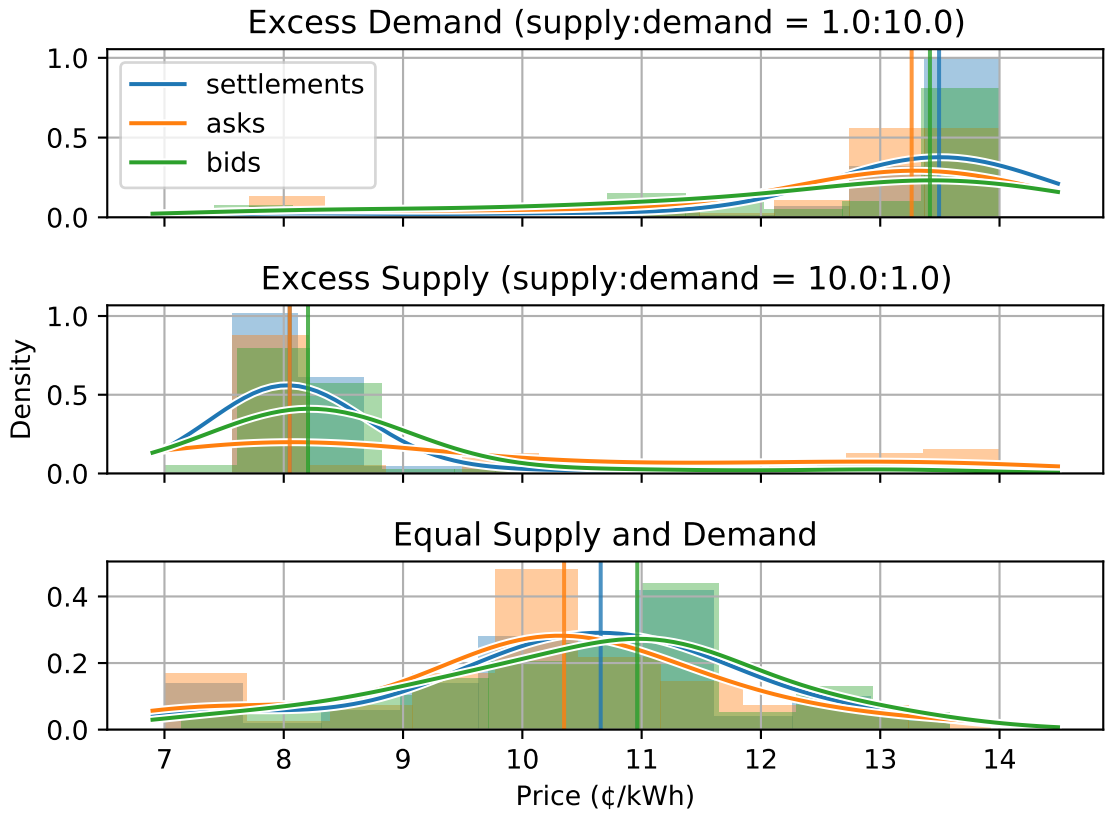

Figure 4.4: Validation policies for bid, ask, and resulting settlement prices for agents operating under M3 for episodes 70 to 100. The histograms show the probabilities of the discrete action policies for the agents and the resulting settlement prices. Modes of the prices are highlighted and shown by the vertical dashed lines. Probability density functions of the histograms are overlaid on top, which are approximated with the Gaussian KDE function in the scikit-learn Python package with default parameters.

Achieving full automation is critical in increasing the effectiveness of demand response. A prerequisite condition to achieving this goal in ALEX is that any interactions between agents through the market must reflect price theory. The goal of the experiments performed in this section is to empirically determine the properties of a

double auction market required to fulfil this condition.

The results of the three performed experiments make it clear that the market must be truthful and weakly budget balanced. Truthfulness is necessary to prevent the agents from advantaging themselves by lying, and to ensure that the emerging behaviour truly reflects the law of supply and demand. Weakening budget balancing is both necessary to fulfil the Myerson–Satterthwaite theorem, as well as to produce a stronger, denser reward signal that makes agent training time reasonable. Reinforcement agents deployed in this market should maximize the value exchanged between participants to guarantee both individual rationality and economic efficiency. Although a stronger budget balancing cannot be achieved, it is notable that many notable real world double auction markets, such as the stock exchange, are also weakly budget balanced. These markets ultimately need to make a small profit from conduction transactions in order to maintain market operations.

4.6.2 Evaluating Fitness for Demand Response

The previous experiment determined the market properties necessary for reinforcement learning agents to develop trading policies that reflect price theory. The goal of this section is to design a simple experiment that makes the resulting behaviour more easily comparable to more contemporary pricing schemes. Three criteria will be investigated to evaluate the fitness of ALEX for demand response: responsiveness, relevancy, and economic performance.

Experimental Design

Recall that the previous experiments focuses on deducing the statistical equilibrium pricing from agent behaviours. Different equilibrium prices can be deduced by training agents in different stateless environments, each with a unique ratio of supply to demand. These stateless environments can be thought of as a snapshot in time. Given enough snapshots, an approximation of agent behaviour in a stateful environment can

be constructed without needing to perform a time-series simulation with time-varying energy profiles. At the time of writing, deep reinforcement learning agents specific to ALEX are still being developed by other members of the research lab, making stateful simulation more difficult, slower, and more noisy in comparison.

Evaluating the fitness of ALEX for demand response is performed in three steps. The first step is to find more equilibrium prices with more supply to demand ratios on market M3 by following the same procedures in the previous experiment. This creates additional data points for more accurate interpolation. Next, the equilibrium prices for all the available supply to demand ratios are interpolated to produce a generalized mathematical model of the price behaviour. Finally, the pricing model is applied to a virtual community, and the responsiveness, relevancy, and the resulting economic performance are evaluated and compared against net billing and time-of-use.

The virtual community is modelled as a North American low voltage circuit consistent of ten residential participants [77] situated behind an community meter upstream from node R1. A corresponding number of energy profiles were randomly selected from the openly available SunDance data set [71, 78], then sequentially to the participants. The selected profile IDs and the corresponding node are as follows: 10011 (R5), 1001625 (R6), 1002714 (R7), 10068 (R8), 100703 (R9), 1001420 (R10), 1003173 (R11), 1001230 (R12), 100114 (R13), 100196 (R14). Because the virtual community consists of prosumers who can generate their own energy using rooftop solar, a single summer day (June 1, 2015) is used to illustrate the widest range of supply-to-demand ratios. A plot of the aggregate values of supply and demand of the community for this day is shown in Figure 4.5, illustrating the wide range of supply-to-demand ratios experienced throughout the day.

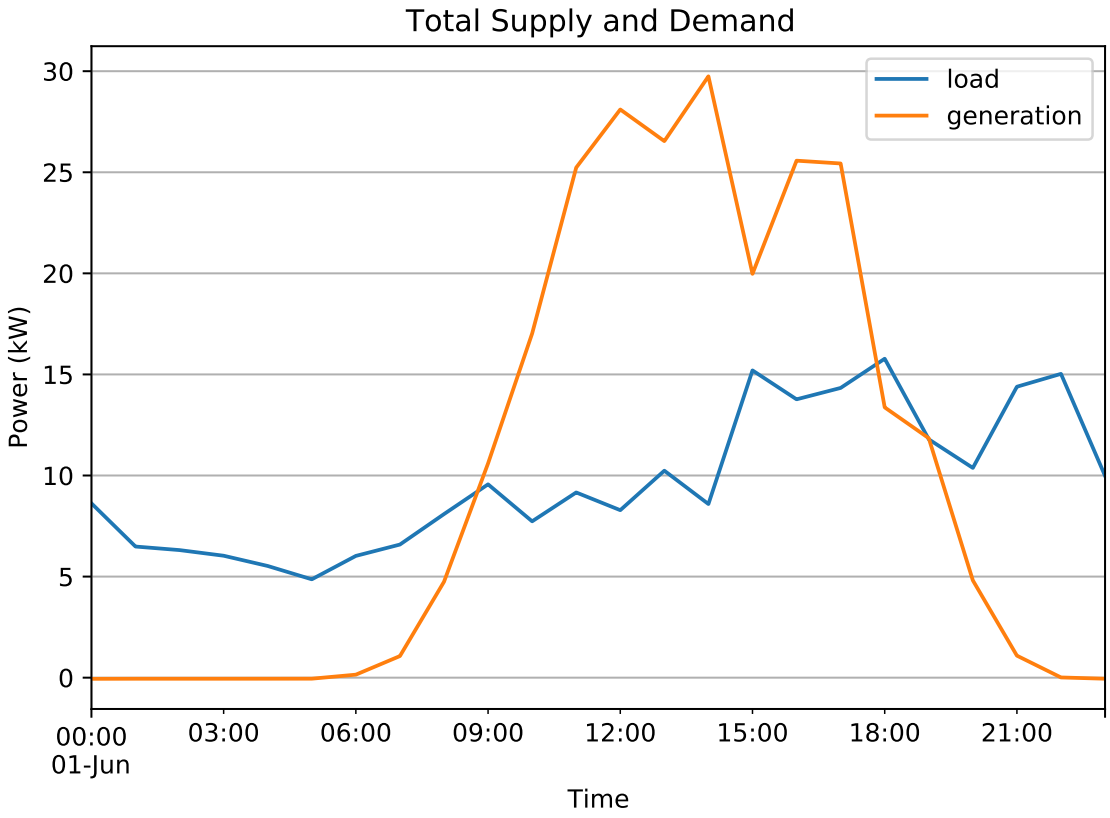

Figure 4.5: Total supply and demand profile of the residential community test over one summer day in June 1, 2015

Results and Discussion

The simulations for M3 from the previous experiment is extended, and also supplemented by four additional experiments using the following supply-to-demand ratios: 1:1.5, 1:2, 1.5:1, 2:1. To help accelerate convergence, exploration factor and learning rate are annealed starting from episode 100 with a multiplier of 0.98 applied at the beginning of each episode. The expanded distribution plots for the price behaviours of all seven supply-to-demand ratios are shown in Figure 4.6. The histograms show the probabilities of the discrete bid, ask, and resulting settlement prices. Modes of the prices are highlighted and shown by the vertical lines. Probability density functions of the histograms are overlaid on top, which are approximated with the Gaussian KDE function in the scikit-learn Python package with default parameters.

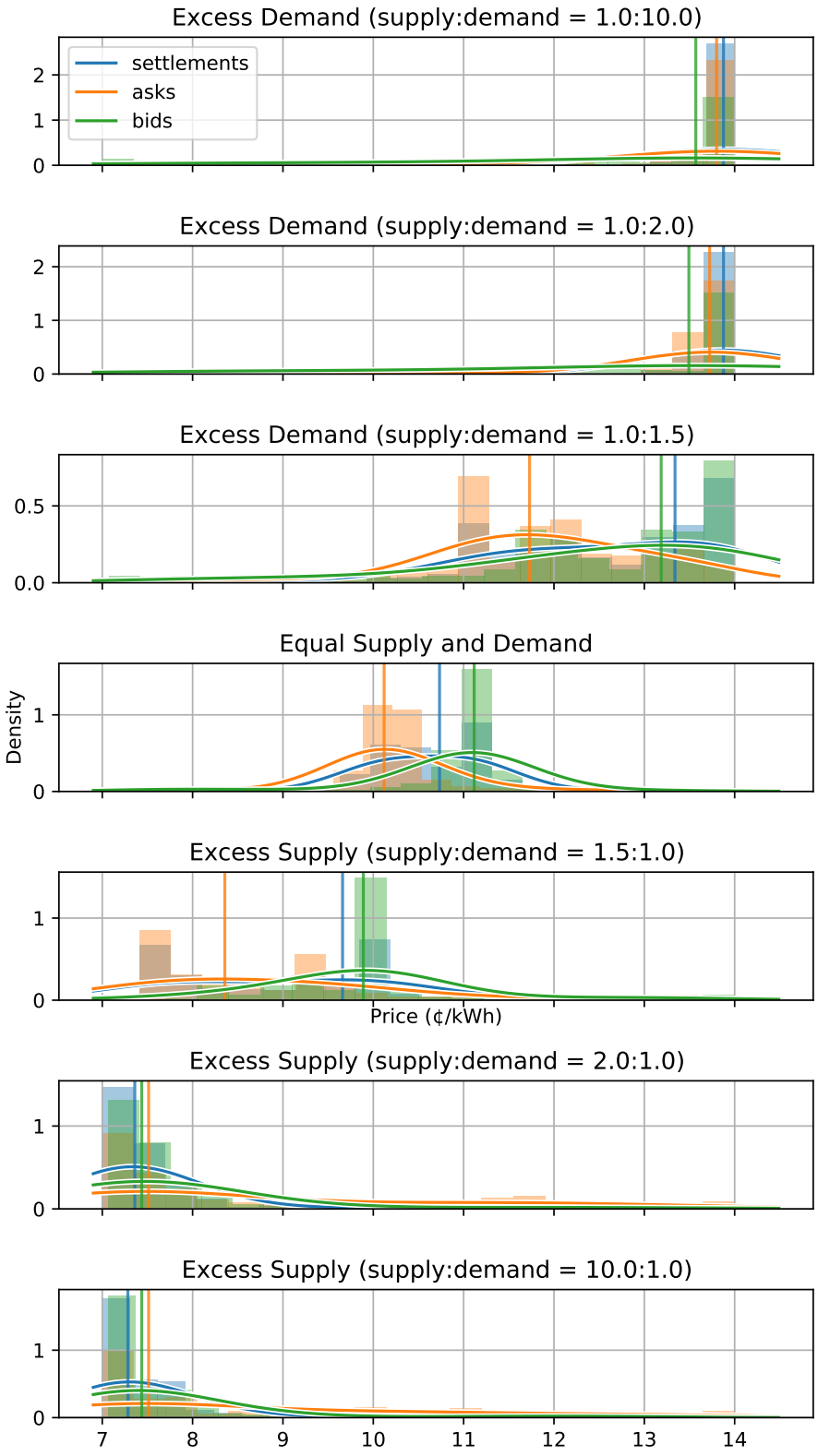

Figure 4.6: Expanded policies for bid, ask, and resulting settlement prices for four agents operating under market M3.

Interpolating over the means of the bid and ask prices for each supply-to-demand ratio produces the equation for the generalized mathematical model of the price curve (equation 4.14). A plot of this model is showin in Figure 4.7

$$P(s,d) = \begin{cases} P^{\text{grid}}_{\text{NB,load}} & P(s,d) \geq P^{\text{grid}}_{\text{NB,buy}} \\ P^{\text{grid}}_{\text{NB,gen}} & P(s,d) \leq P^{\text{grid}}_{\text{NB,sell}} \\ -0.0254\frac{s}{d} + 0.1426 C^{\text{TOU}}_{H,M,L} & \text{if buying} \\ -0.0280\frac{s}{d} + 0.1299 C^{\text{TOU}}_{H,M,L} & \text{if selling,} \end{cases} \tag{4.14}$$

where s is energy supply, d is energy demand, $P^{\text{grid}}_{\text{NB,load}}$ is price of electricity when buying electricity from the grid under net billing, $P^{\text{grid}}_{\text{NB,gen}}$ is price of electricity when selling electricity to the grid under net billing, and $C^{\text{TOU}}_{H,M,L}$ are the adjustment factors when TOU is used for energy transacted with the grid.

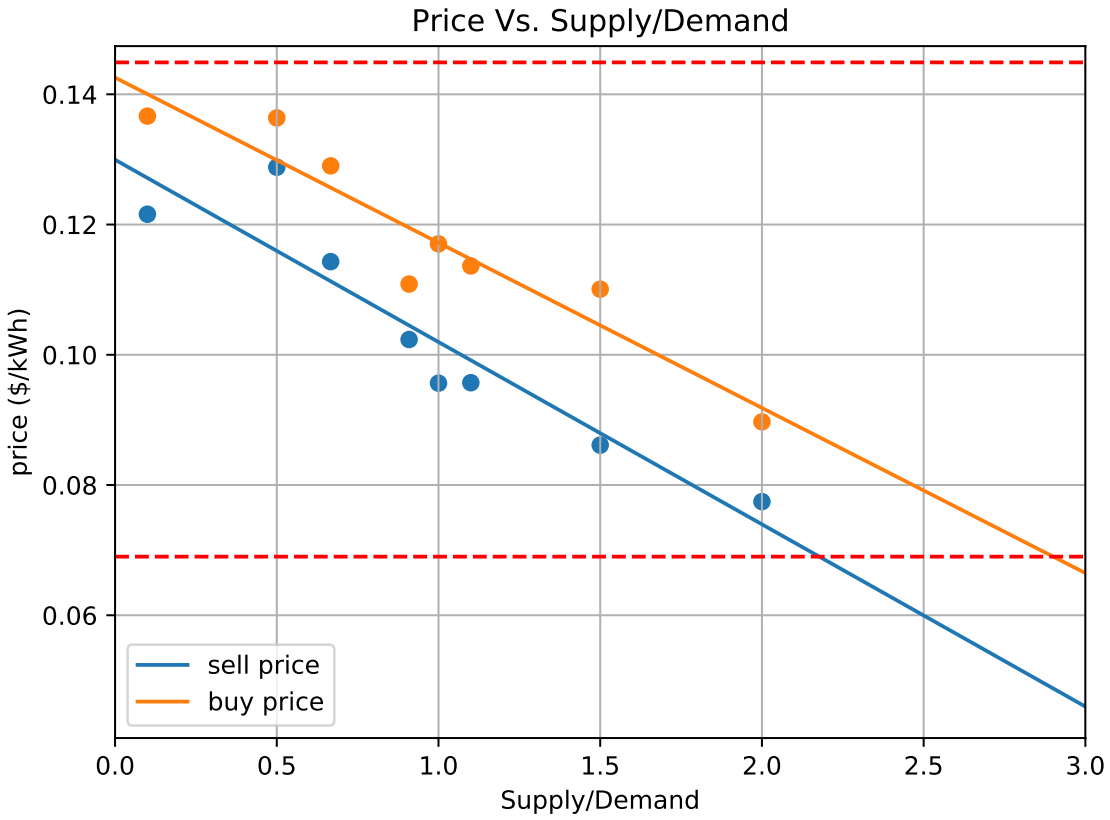

Figure 4.7: System-wide pricing model developed for the virtual community. The dotted lines show the price boundaries defined by (4.4). Linear regression between the price points leads to a well-fit mathematical model.

A supply-to-demand dependent pricing schedule for the local market can be obtained by applying equation 4.14 to the aggregate energy profiles for all participants in a given time interval. The aggregate load and generation profiles for the virtual community is show in Figure 4.5. The resulting price schedule for the virtual community is shown in Figure 4.8. As a reminder, agents cannot observe the system-wide supply-to-demand and will develop personalized price curves. Therefore, the analysis performed here should only be used for initial exploration.

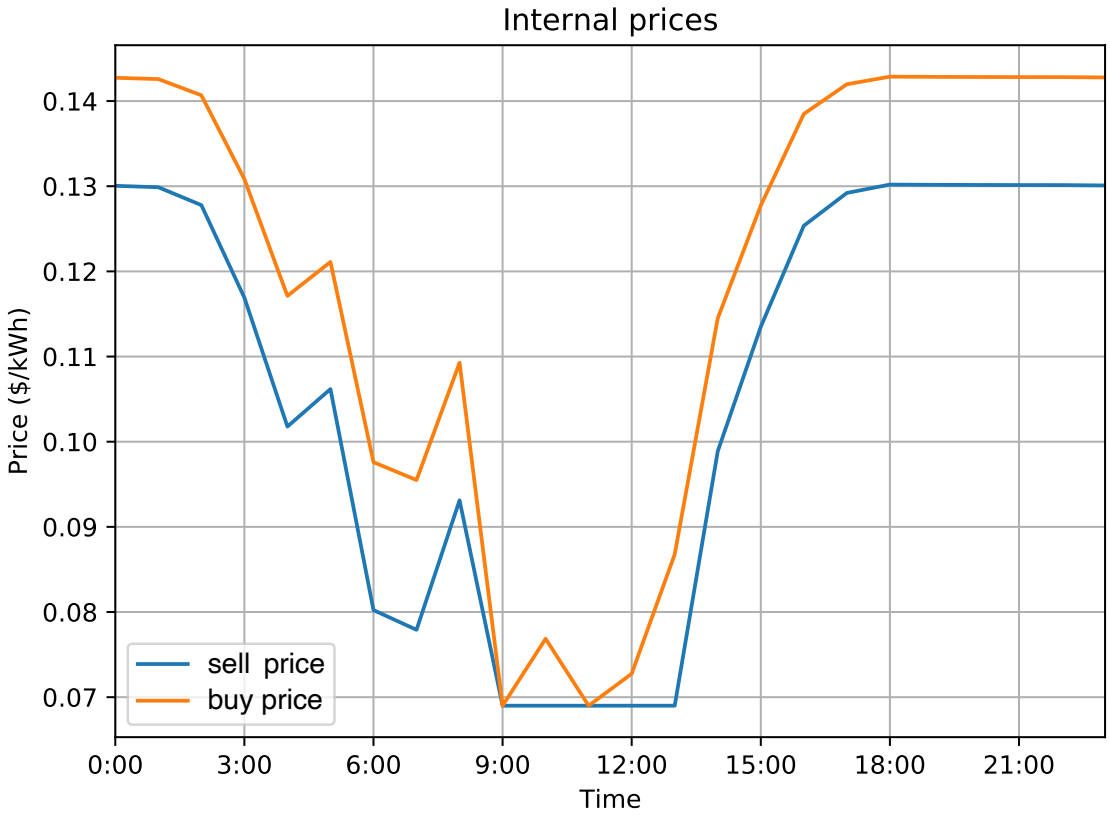

Figure 4.8: Internal prices of ALEX used to conduct transactions

Not surprisingly, the resulting price schedule determined using the model corresponds exactly to the ratio of supply and demand of the community throughout the day. For example, at midnight, when solar generation is nil, the price for selling energy to a peer is \$0.1449, which is the same for the buyer as if purchasing energy from the grid. Later in the morning, at 7:00AM, when solar energy becomes available,

the price for selling to peers lowers accordingly. Starting at around 9:00AM, when generation is significantly higher than demand, the price for selling to peers drops to $0.069, which is the same as selling to the grid under net billing. Prices then increase later in the day as local generation decreases and demand increases.

A time-varying and supply-to-demand dependent price schedule is similar to the philosophy behind the development of TOU. However, TOU derives its prices from the time-varying supply/demand of the entire grid instead of individual areas. Therefore, the community pricing schedule should be compared to TOU to quantify their relative performance. Ontario TOU is used as a benchmark in Canada and is often referenced by utility companies in jurisdictions without TOU, such as Alberta.

Fig. 4.9 displays the two pricing schedules, showing the stark contrast between their shapes. Whereas the local energy price decreases toward noon due to the increase in generation, TOU price increases, which suggests that there is more load than generation during this period. This is likely due to the fact that more commercial and industrial loads exist during the day, which are the type of loads putting the most strain on the grid. However, since TOU is targeted towards residential customers, it also suggests TOU tries to shift the responsibility of decreasing loads onto residential customers. Since TOU is not highly correlated to residential customer behaviour, this may explain the lack of participation in TOU mentioned in Chapter 2. It also suggests the need for very localized, highly relevant pricing signals to increase the efficiency of managing DERs and the overall system.

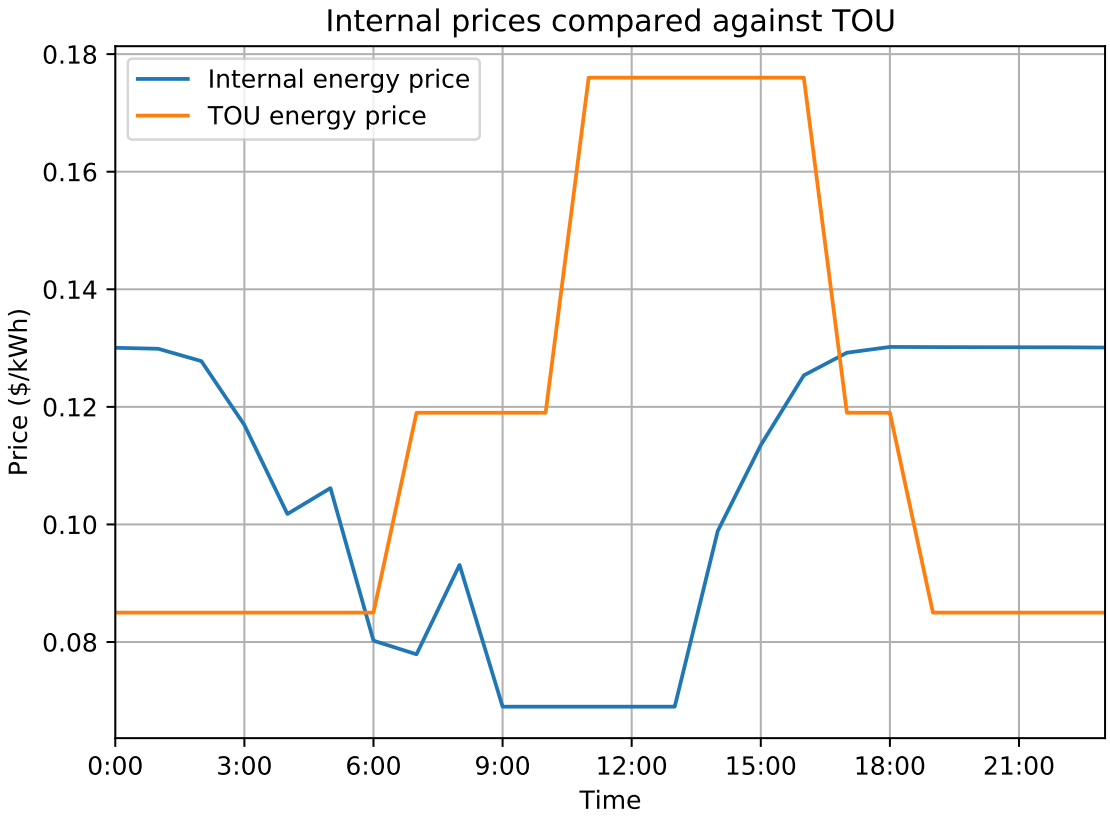

Figure 4.9: Local market pricing schedule compared against Ontario summer TOU prices.

The following analysis evaluates the the economic performance of all participants under this price curve. Since TOU is less relevant and will performe worse than net billing, the resulting electricity bills will only be compared to the net billing equivalent. The results of this comparison are shown in Fig. 4.10 and summarized in Table 4.2. In accordance with the operating principles of net billing, described in 2.1.4, the entire community is placed behind a primary meter, and energy exchanged directly between peers do not incur T&D fees.

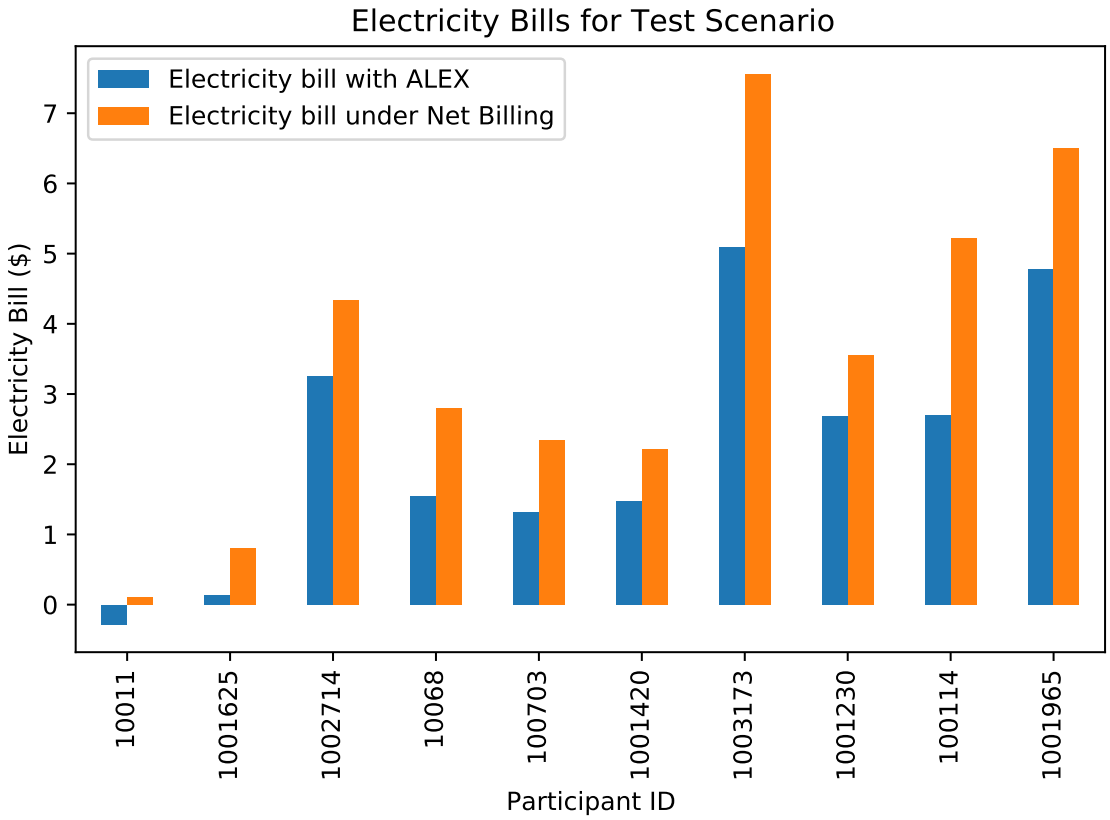

Figure 4.10: Electricity bill comparison between net billing and ALEX

The results in Table 4.2 show that the implementation of a local energy market can financially benefit the community as a whole, reducing the total community bill by 35.9%. The mean and median of individual bill reductions are 74.51% and 38.8%, respectively. This reduction in bills is due to the more efficient usage of local energy resources, which is more accurately reflected by the local market price. By putting the whole community behind-the-meter, financial benefits may even be gained by those who cannot afford the expense of acquiring and installing their own DERs. This is because they have direct access to residual generations of their neighbours, that may be priced lower in comparison to buying at retail prices from the grid. Similarly, there is an inherent financial incentive to sell excess generation to peers first, as the profits can be higher than selling to the grid at retail prices. In maximizing individual economic gains, local energy markets may further socialize the benefits

Table 4.2: ALEX vs. Net Billing (NB)

	Bill ($)			**Energy (kWh) bought from Grid**			**Energy (kWh) bought locally**		**Energy (kWh) sold to Grid**			**Energy (kWh) sold locally**	
Participant	**NB**	**ALEX**	**%**	**NB**	**ALEX**	**%**	**NB**	**ALEX**	**NB**	**ALEX**	**%**	**NB**	**ALEX**
10011	0.10	-0.28	383.2	6.83	2.35	65.59	-	4.48	12.88	0	100	-	12.88
1001625	0.80	0.13	83.2	10.73	2.28	78.75	-	8.45	10.97	0	100	-	10.97
1002714	4.34	3.26	24.8	32.45	13.10	59.63	-	19.35	5.31	0	100	-	5.31
10068	2.80	1.55	44.7	22.37	4.08	81.76	-	18.29	6.41	0	100	-	6.41
100703	2.35	1.32	44.0	22.47	5.94	73.56	-	16.53	13.17	0	100	-	13.17
1001420	2.21	1.47	33.6	17.63	11.96	32.16	-	5.67	4.98	0	100	-	4.98
1003173	7.56	5.09	32.7	61.73	22.00	64.36	-	39.73	20.03	0	100	-	20.03
1001230	3.55	2.69	24.2	24.89	8.03	67.74	-	16.86	0.80	0	100	-	0.80
100114	5.22	2.70	48.2	49.55	12.65	74.47	-	36.9	28.40	0	100	-	28.40
1001965	6.50	4.78	26.5	50.78	19.07	62.45	-	31.71	12.45	0	100	-	12.45
Total	**35.43**	**22.70**	**35.9**	**299.44**	**101.46**	**66.12**		**198**	**115.40**	**0**	**100**		**115.40**

of DERs. Although not directly comparable, recent work by Jogunola et al. [79] obtained average financial benefits of 35% by leveraging energy storage, and 55% when leveraging both PV and storage. While the proposed approach currently uses only PV, energy storage will be added in future studies. Similar or better performance than that reported by Jogunola et al. [79] should be expected.

The Effect of Number of Participants on Price Behaviour

After the article containing these results were published [1], discussions among peers brought up whether the price behaviour is affected by the number of participants. This is mainly due to the fact that the price behaviour of four agents was used to create the price model for a community of 10 participants. Since price theory is purely dictated by the ratio of supply and demand, which already includes the number of participants, the price behaviour should be similar regardless of the number of agents. However, for the sake of completeness, a set of simulations is performed for M3 using 10 agents following the same procedures as described in Section 4.6.1. The resulting distribution plots are shown in Figure 4.11.

As expected, the price distribution with 10 agents is highly similar to the four agent case previously shown in Figure 4.4. This means that price theory holds true, and the pricing model developed in equation 4.14 is a valid model regardless of the number of participants in the market.

4.6.3 The Effects of Battery Storage on Power Flow

A part of the financial gains seen in the previous experiment can be attributed to the more accurate tracking of energy flow, which include the source, destination, and amount of infrastructure used. However, these gains may be short-lived without a more flexible and robust way shape generation and loads to combat the negative effects of the non-dispatchable nature of renewable energy sources.

The currently technology landscape clearly favors battery energy storage, which

Figure 4.11: Price policies for bid, ask, and resulting settlement prices for 10 agents operating under M3.

includes electric vehicles with vehicle-to-grid (V2G) capabilities, as the provider of such micro-ancillary services. The most common use case of home battery storage today is to complement rooftop solar to improve self sufficiency, usually accomplished using a greedy policy. This often leads to batteries that are either undersized, or underutilized, since it is rarely possible for the battery capacity to perfectly match the generation and load profiles of any home. Although research exist to optimize battery sizes for residential systems [80], it is rare for vendors to offer more than one or two capacity options for purchase. Furthermore, studies have shown that the addition of a battery energy storage system is generally unprofitable over its lifetime under net billing or net metering [81, 82]. However, due to surrounding loads and generation made available through ALEX, BESS utilization and financial gains may both increase. More importantly, generation or load spikes may also be flattened as a side effect, resulting in a more stable power grid. Empirically testing this hypothesis is the focus of this experiment.

Experimental Design

This experiment requires power flow simulations to be performed in order to show the possible effects of load shaping. The simulation circuit used is the CIGRE North America low voltage residential benchmark circuit [77, 83], shown in Figure 4.12. Since a simulation model of this circuit is not available, one had to be created for OpenDSS, and is made available on GitHub for general use [84]. Baseline simulations are performed and evaluated against the published results to check for accuracy and validity. The simulation results show that node voltages are all within 2% of the values in the CIGRE report [77]. As node voltages are the primary metric used for evaluation, the circuit model is deemed sufficiently accurate for the purposes of this experiment.

The energy profile selection criteria differ from the previous experiment, as the original locations of the profiles and profile shapes can greatly affect power flow. For

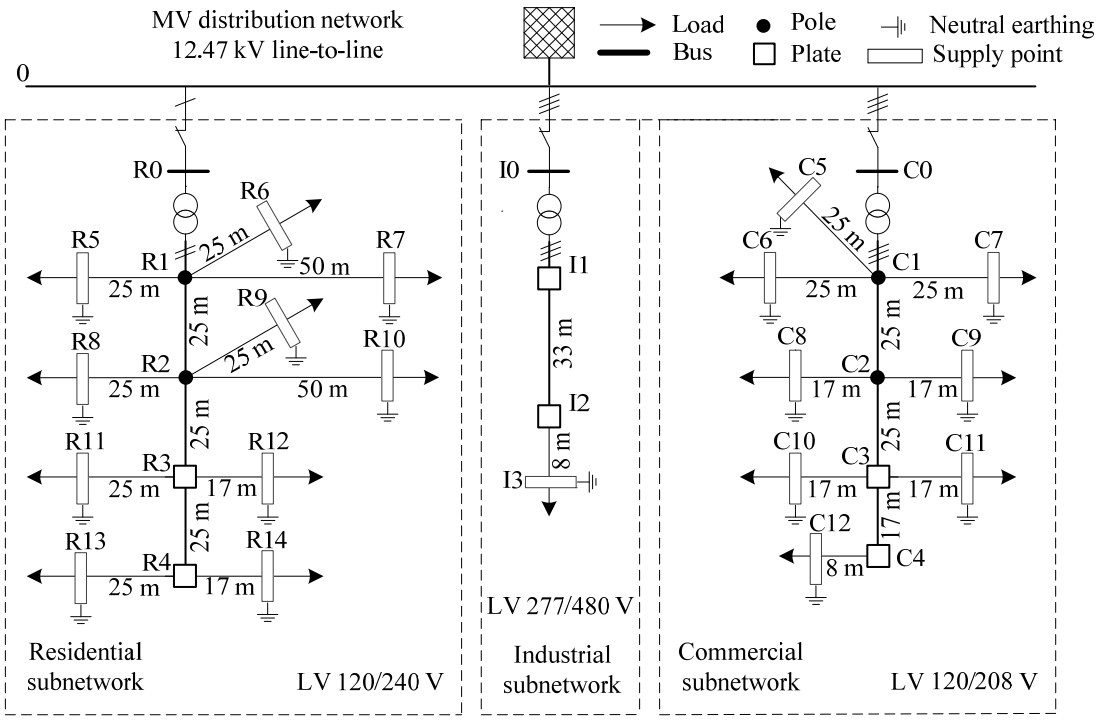

Figure 4.12: Topology of the North American low voltage distribution network benchmark. Originally from "Benchmark Systems for Network Integration of Renewable and Distributed Energy Resources", pg. 44 [77]

these reasons, the SunDance [71, 78] profiles are re-selected from the only the Denver time zone[5]. Monte-Carlo search is then performed to generate over 500 combinations of profile to node mappings, each followed by a one-day time-series power flow simulation. Finally, resulting voltage plots are both programmatically and visually inspected to choose the most suitable candidate to perform the experiment on. The resulting profile IDs and corresponding node locations are shown in Table 4.3. The energy profiles and node voltages for the final configuration is shown in Figure 4.13.

[5]The Denver time zone is an arbitrary choice. We hoped that the climate might be similar to central Canada, but it is not an important criteria

Table 4.3: Profile IDs and Corresponding Node Locations

Node	**Profile ID**
R5	1001230
R6	1001974
R7	100500
R8	10068
R9	10011
R10	10063
R11	100124
R12	100502
R13	1003842
R14	1002705

With the baseline (B1) established, two BESS-based load shaping strategies can be deployed for comparison. Since R10 is the only node experiencing voltage violations, it will be the target of load shaping strategies. The strategies are described as follows:

1. *B2:* Attach a BESS to R10. Use greedy control with no local market access.

2. *B3:* Attach a BESS to R10. Use rule-based controller with local market access.

The BESS used are modelled after a typical Tesla Powerwall 2, with a usable capacity of 13.5 kWh and a continuous charge/discharge power of 5 kW. B2 is designed to emulate existing control strategies that focus on self-sufficiency. The general strategy is to charge the battery with residual generation, and discharge the battery if there is residual load. This strategy relies upon having an oversized solar generation system and is often not feasible due to regulatory, roof area, or cost constraints.

B3 is designed as a simple way to demonstrate the effects of maximizing the utilization of the local market for one load shaping capable participant only. Ideally, a learning agent would be used. However, due to the complexities of the agent, development is slower than expected. As a result, a rule based agent that uses fixed prices

and a trading schedule is used instead. The agent used for B3 can be described by the pseudocode shown in Algorithm 1. The actual Python code can be found on the project repository on GitHub [85] as basic_trader.py.

Algorithm 1 Energy Trading and Management Algorithm for B3

```
for each market round do
    for last settled round do
        adjust BESS schedule based on successful settlements
    end for
    if residual load > 0 then
        if BESS is available then
            if next_settle is between 7AM to 3PM (inclusive) then
                Battery is allowed to charge
                Submit bid where,
                quantity = residual load + max charge capacity
                price = $0.14/kWh
            else
                Battery is allowed to discharge
                Submit ask where,
                quantity = max(0, max discharge capacity - residual load)
                price = $0.07/kWh
            end if
        else
            Submit bid where,
            quantity = residual load
            price = $0.14/kWh
        end if
    else
        if residual generation > 0 then
            if BESS is available then
                if next_settle is between 8AM to 4PM (inclusive) then
                    Battery is allowed to charge
                    Submit bid where,
                    quantity = max(0, max charge capacity - residual generation)
                    price = $0.14/kWh
                else
                    Battery is allowed to discharge
                    Submit ask where,
                    quantity = max discharge capacity + residual generation
                    price = $0.07/kWh
                end if
            else
                Submit ask where,
                quantity = residual generation
                price = $0.07/kWh
            end if
        end if
    end if
end for
```

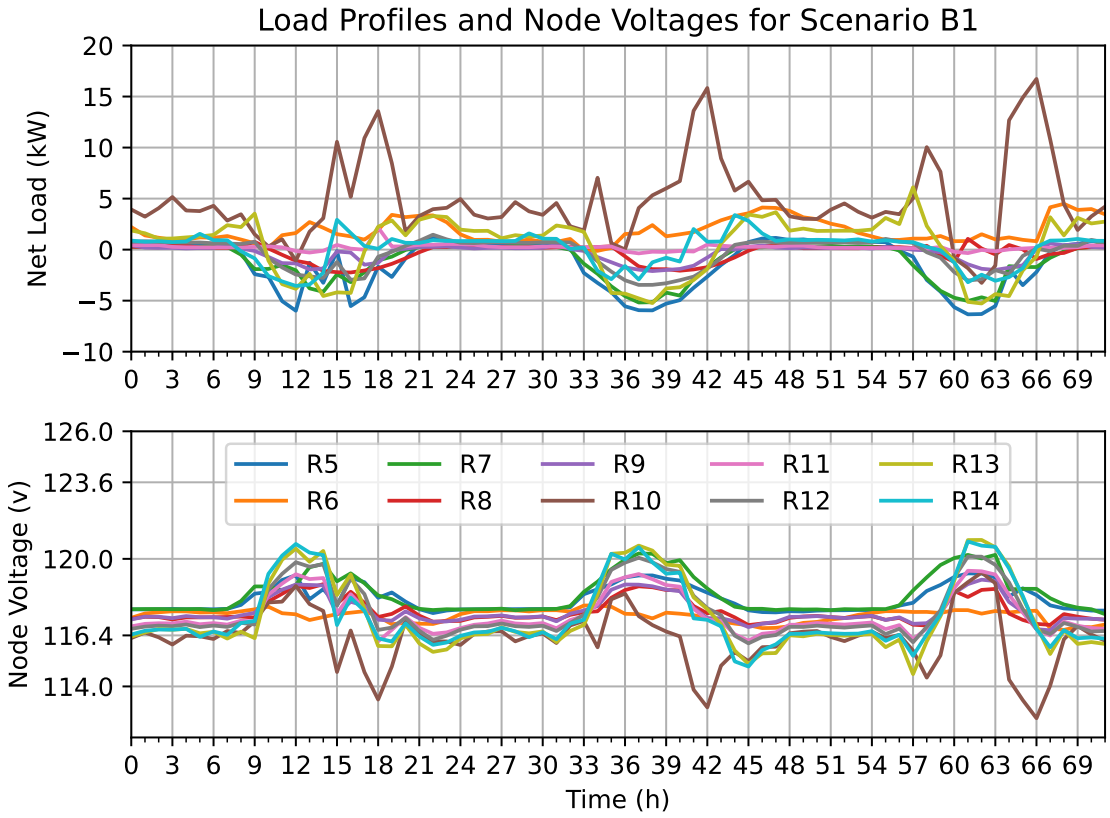

Figure 4.13: Energy profiles and node voltages for the baseline experiment (B1).

Figure 4.13 shows the energy profiles and node voltages for the baseline experiment, taking place across three summer days (June 1-3, 2015). Multiple days are used to show possible effects of leftover energy in storage from the previous day(s), if available. The top graph shows the hourly net loads, where positive means net consumption, and negative net generation. Major y-axes show the nominal operating voltage, the ± 3% voltage thresholds that some utility companies use, and the ± 5% standard instituted by IEEE. With the exception of R10, all other energy profiles show behaviours that are representative of typical residential prosumers. Consequently, R10's large loads caused four under-voltage violations over three days (hours 18, 41, 42, and 66). As such, R10 is the ideal target for studying load shaping strategies.

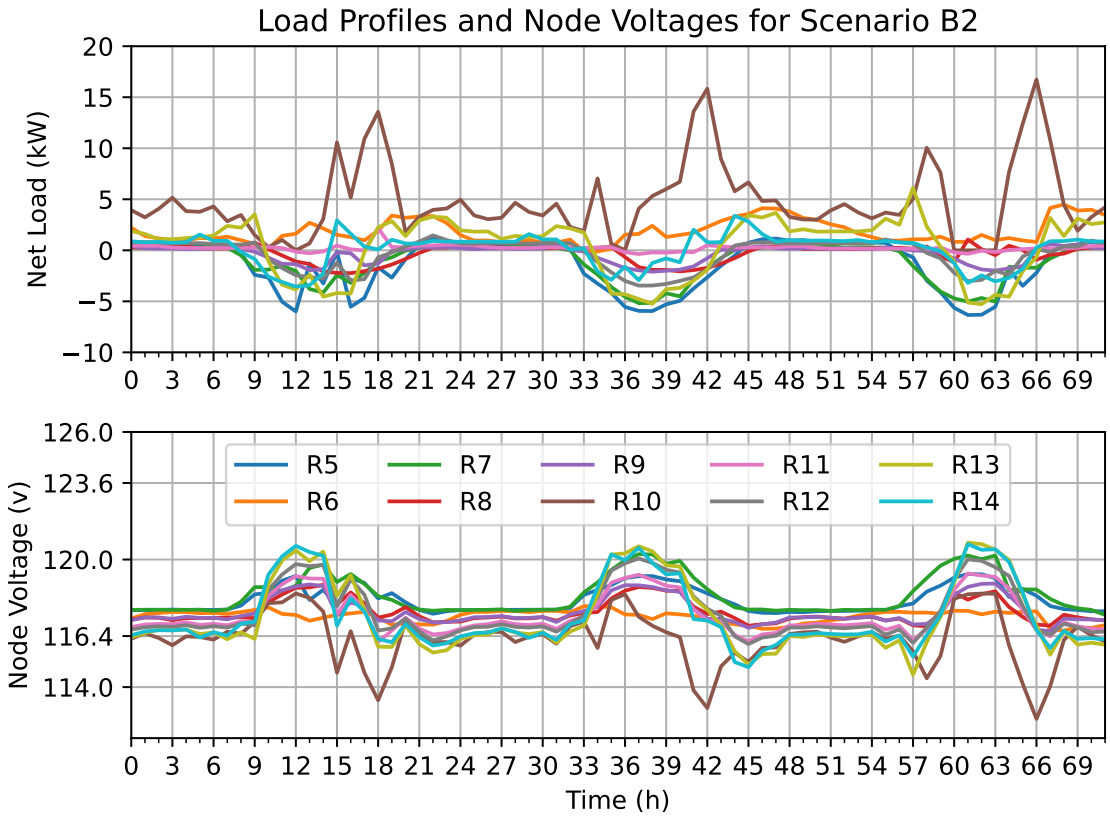

Figure 4.14: Energy profiles and node voltages for the experiment B2.

The typical application of BESS for a residential prosumer is to increase self sufficiency. The BESS usage strategy is generally greedy, i.e. charge if there is residual generation, and discharge if there is residual load. Charge/discharge times may be modified by a schedule if time-of-use or dynamic pricing is in play. Figure 4.14 shows the results when greedy battery management is used by R10. At first glance, the difference between B2 and B1 is minuscule. This is not unexpected, as R10 has short windows where small amounts of net generation is available (hours 12, 60 to 63, peaking at approximately 2.5kW). As a result, the loads at hours 13 and 64 are decreased by the BESS. However, these load decreases are not significant enough to eliminate the voltage violations. One can argue that the greedy policy is sub-optimal, and may have eliminated or reduced the under-voltage at hour 66 if the BESS was not discharged at hour 64. While this may be true in retrospect, predicting the load

spike at hour 66 with a high degree of precision can be very difficult for practical applications, and is one of the reasons why schedule-based transactive control have been generally unsuccessful [12].

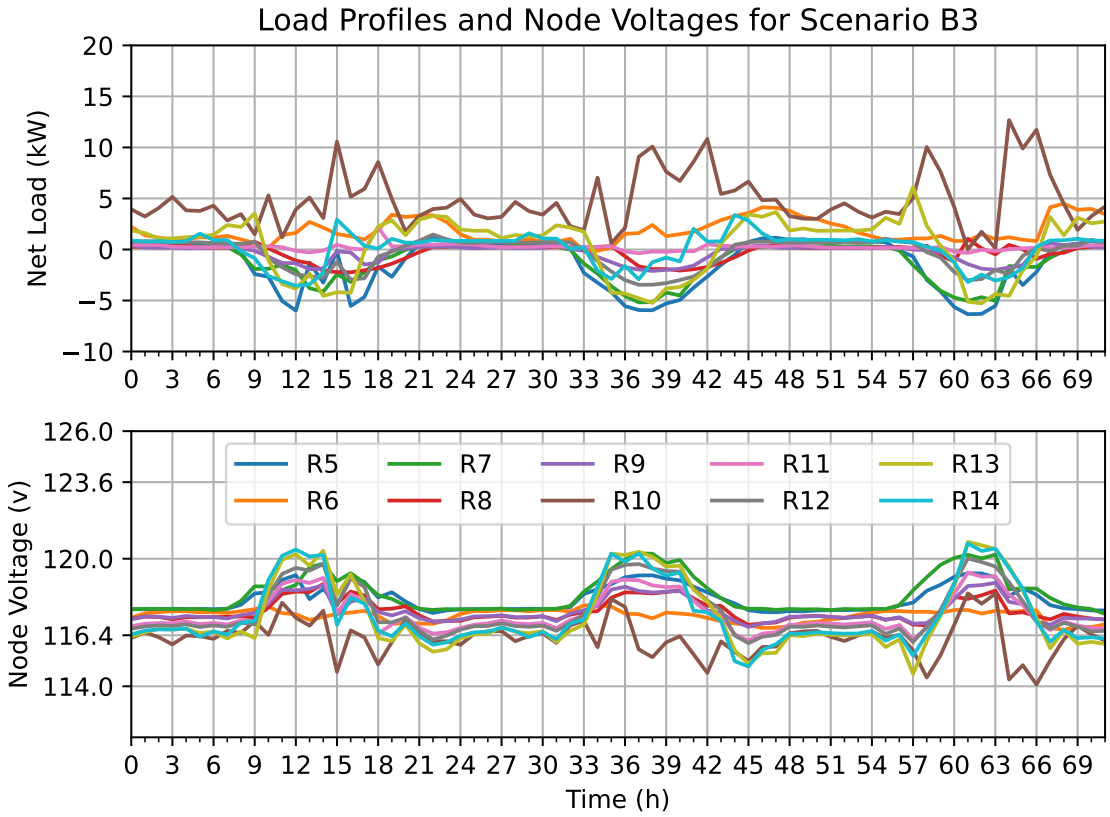

Figure 4.15: Energy profiles and node voltages for the experiment B3.

Unlike R10, its neighbours have much more residual generation available. The local energy market could allow R10's BESS to utilize this additional source of energy over longer periods of time. Figure 4.15 shows the resulting node voltages where local energy trading is enabled via ALEX. Compared to B1 and B2, the improvements are clear and drastic, with the most obvious being the elimination of all voltage violations. The clear cause of this is the reduction of R10's peak load, which is decreased by as much as 5 kW, the peak continuous discharge capacity of the BESS. This suggests that the BESS has enough energy charged to be utilized for longer periods of time, which can be seen in Figure 4.16, where the state of charges (SoC) of the BESS between

B2 and B3 are shown. As expected, as opposed to B2, which is unable to utilize the BESS's full capacity due to R10's lack of residual generation, B3 manages to almost always fully charge the BESS using energy bought via the local energy market. This is a much more effective use of local resources that would otherwise be simply sold to the grid. As shown previous experiments, better utilization of local resources leads to financial improvements for the community as a whole. Furthermore, the electric utility company can benefit as well. Typically, solving the under-voltage violation can involve transformer or line upgrades, which may cost tens of thousands of dollars in materials alone. A BESS installed in a single home is less expensive, and is shown here to be highly effective under the right conditions.

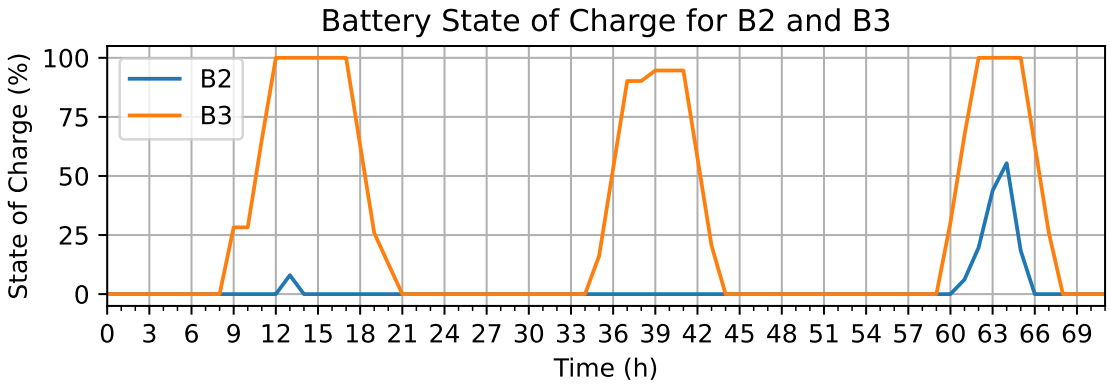

Figure 4.16: Comparing the state of charges of the BESS in B2 and B3.

This experiment uses a heavily solar penetrated community to test and compare BESS profile shaping strategies. The community consists of nine typical residential prosumers, and one heavily loaded prosumer (R10). A baseline time-series power flow simulation was performed, and shows that R10 suffers from multiple under-voltage violations throughout the three test days. Two profile shaping strategies (B2, B3) with a 13.5kWh BESS installed at R10 were tested and compared. B2 uses a greedy control strategy that maximizes self sufficiency, with no local market access. B3 uses a rule-based controller that maximizes local market interactions.

The results for B2 show that, due to R10's lack of residual generation, the BESS is

underutilized and is unable to make significant improvements to power flow. However, B3 is able to effectively utilize the full capacity of the BESS and reduce peaks loads by as much as 5kW. As a result, all of the under-voltage violations are eliminated. Finally, this experiment provides concrete evidence that targeted BESS installations can be used a more cost effective alternative to transformer, line, or other equipment upgrades.

Although the results achieved are impressive, they are still limited in scope. One of the biggest limitations is the rule based agent used for B3. Due to its current inability to learn prices, it is impossible to deploy multiple BESS enabled participants in ALEX and achieve deterministic, or probabilistic results with low variance without artificially introducing bias. This can be solved with more sophisticated reinforcement learning agents that can simultaneously learn energy trading and battery management. Work on the reinforcement learning agent is still being conducted at the time of writing, and will be published in a journal article in the future.

Chapter 5

Conclusion and Future Work

5.1 Summary

The ultimate objective of this thesis is to answer the following questions:

- What market mechanism and market properties will allow a set of independent reinforcement learning (RL) agents to trade energy in a way that reflects price theory?
- Can optimal, or near optimal power flow be achieved as a side effect of efficient market transactions?

Finding the answers to these questions started with a review of the current state of the art methods to simulate transactive energy. The findings in Chapter 2 show that the existing transactive energy simulators primarily focus on solving for optimal power flow, and relegate the resulting price signals as a proxy to control appliances, or influence human behaviours. There is a clear lack of an economy focused TE simulator that is required to complete to conduct this research. As such, such as simulator had to be created. Chapter 3 details the simulator design, and goes over some key topics, from the high level architecture, to implementations of some lower level functions.

The experimental part of this thesis is presented in Chapter 4. The experiments are presented in three parts, each build on top of the results of the previous experiment

to progressively answer the research questions. The first experiment (Section 4.6.1) focuses on investigating the necessary market properties for RL agents to behave in accordance to price theory. The second experiment (Section 4.6.2) shows the advantages of a market-based TE compared to contemporary DR pricing schemes in terms of responsiveness, relevancy, and bill cost. The last experiment (Section 4.6.3) shows the combined effectiveness of a local energy market and battery management in eliminating circuit violations.

5.2 Contributions

The original contributions described in this thesis can be summarized as follows:

- Developed T-REX, an economy focused transactive energy simulator that enabled the completion of subsequent tasks. The design of the simulator is detailed extensively in Chapter 3.

- Designed and implemented methods that allow agents to trade energy in a double auction market. This is introduced as the Autonomous Local Energy Exchange (ALEX) in the first part of Chapter 4.

- Examined the relationships between double auction market properties and reinforcement learning. Specifically, the experiments conducted in Section 4.6.1 investigated how the market properties affect RL policy development. The results show that the market must be truthful and weakly budget balanced in order for the agents to develop behaviours that reflect price theory. Truthfulness is necessary for agent behaviour policies to align with price theory, and weak budget balancing results in a stronger, denser reward signal that makes training complete in a reasonable amount of time.

- Inspected the developed RL policies, and devised methods to make comparisons to contemporary demand response pricing scheme. This work is presented

in Section 4.6.2. A generalized mathematical model that aggregates and converts individual agent policies to a global pricing scheme is first created, so that pricing schemes can be compared directly. Empirical testing shows that this math model is largely unaffected by the the total number of participants, and is instead only based on the global ratio of supply and demand. Then, the resulting price model is compared against net billing and time-of-use in terms of responsiveness, relevancy, and bill cost. The resulting price model is far more responsive and relevant compared to time-of-use, which is suggests that agent behaviours can be more accurate and efficient when used for load shaping. Furthermore, significant bill reductions are achieved when compared to net billing. The community as a whole experienced a bill reduction of 35.9%, and the mean and median of individual bill reductions are 74.51% and 38.8%, respectively.

- Studied the effects of integrating battery storage in the local energy market, with a focus on power flow. This is demonstrated in Section 4.6.3. The CIGRE North America low voltage residential benchmark circuit [77, 83] is used as the test circuit for power flow, and a set of load profiles from the SunDance data set was chosen using Monte-Carlo search to create voltage violations on the circuit as the starting condition. Two scenarios are tested with identical battery placement and specifications. The first is the baseline, where participants cannot trade energy with each other. The second enables energy trading, and the agents are designed to maximize market interactions to maximize battery utilization. The results show the elimination of all voltage violations when energy trading is allowed. Although this test scenario is one extreme and narrow example, it is clear that efficient market interactions can result in improvements in power flow.

In conclusion, the work presented in this thesis has adequately answered the research questions posed in Chapter 1. It also provides strong evidence for the practical

application of agent-based, economy focused transactive energy systems as a more robust and flexible alternative to transactive control.

5.3 Future Work

The two research objectives have been adequately answered, although opportunity still exists to further explore the load shaping capabilities. Because the experiment performed in this thesis is an controlled initial exploration, the setup and methods used had to be limited in scope. Future work will build upon the work presented, with the primary focus on more sophisticated RL agents that can learn both energy trading and battery management. This will, at minimum, allow multiple battery users on the local market. Holistic studies of the entire system will be enabled as well, and should allow simultaneous analysis of economic benefits, power flow improvements, and return on investment. Beyond further academic studies, T-REX and ALEX have garnered some interest in the industry, and a project is currently underway for potential field deployment. In terms of AI and ML related studies, some team members are developing methods to use T-REX with commonly used ML frameworks, such as gym [86]. This will allow machine learning researchers to use more familiar tools to research algorithms with T-REX.

Bibliography

[1] S. Zhang, D. May, M. Gül, and P. Musilek, "Reinforcement learning-driven local transactive energy market for distributed energy resources," *Energy and AI*, vol. 8, p. 100 150, 2022, ISSN: 2666-5468. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666546822000118.

[2] S. Zhang and P. Musilek, "The impact of battery storage on power flow and economy in an automated transactive energy market," *Energies*, vol. 16, no. 5, 2023. DOI: 10.3390/en16052251. [Online]. Available: https://www.mdpi.com/1996-1073/16/5/2251.

[3] S. M. Ismael, S. H. A. Aleem], A. Y. Abdelaziz, and A. F. Zobaa, "State-of-the-art of hosting capacity in modern power systems with distributed generation," *Renewable Energy*, vol. 130, pp. 1002 –1020, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0960148118307936.

[4] T. O. Olowu, A. Sundararajan, M. Moghaddami, and A. I. Sarwat, "Future challenges and mitigation methods for high photovoltaic penetration: A survey," *Energies*, vol. 11, no. 7, 2018.

[5] C Chapelsky, K Gerasimov, and P Musilek, "DER impacts to urban utilities study summary," 2019. [Online]. Available: https://www.epcor.com/products-services/power/Documents/micro-generation-research-solar-energy-electricity-grid-2019.pdf.

[6] D. Forfia, M. Knight, and R. Melton, "The view from the top of the mountain: Building a community of practice with the gridwise transactive energy framework," *IEEE Power and Energy Magazine*, vol. 14, no. 3, pp. 25–33, 2016.

[7] M. D. A.Fattahi Meyabadi, "A review of demand-side management: Reconsidering theoretical framework," *Renewable and Sustainable Energy Reviews*, vol. 80, pp. 367–379, 2017.

[8] S. Chen and C.-C. Liu, "From demand response to transactive energy: State of the art," *Journal of Modern Power Systems and Clean Energy*, vol. 5, no. 1, pp. 10–19, 2017.

[9] O. Abrishambaf, F. Lezama, P. Faria, and Z. Vale, "Towards transactive energy systems: An analysis on current trends," *Energy Strategy Reviews*, vol. 26, p. 100 418, 2019.

[10] J. Hu, G. Yang, C. Ziras, and K. Kok, "Aggregator operation in the balancing market through network-constrained transactive energy," *IEEE Transactions on Power Systems*, vol. 34, no. 5, pp. 4071–4080, 2019.

[11] M. S. Nazir and I. A. Hiskens, "A dynamical systems approach to modeling and analysis of transactive energy coordination," *IEEE Transactions on Power Systems*, vol. 34, no. 5, pp. 4060–4070, 2019.

[12] A. Soares, O. De Somer, D. Ectors, F. Aben, J. Goyvaerts, M. Broekmans, F. Spiessens, D. van Goch, and K. Vanthournout, "Distributed optimization algorithm for residential flexibility activation—results from a field test," *IEEE Transactions on Power Systems*, vol. 34, no. 5, pp. 4119–4127, 2019.

[13] S. H. Mengmeng Yu Renzhi Lu, "A real-time decision model for industrial load management in a smart grid," *Applied Energy*, vol. 183, Dec. 2016. DOI: 10.1016/j.apenergy.2016.09.021.

[14] X. Huang, S. H. Hong, and Y. Li, "Hour-ahead price based energy management scheme for industrial facilities," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 6, pp. 2886–2898, 2017. DOI: 10.1109/TII.2017.2711648.

[15] R. de Sá Ferreira, L. A. Barroso, P. R. Lino, M. M. Carvalho, and P. Valenzuela, "Time-of-use tariff design under uncertainty in price-elasticities of electricity demand: A stochastic optimization approach," *IEEE Transactions on Smart Grid*, vol. 4, no. 4, pp. 2285–2295, 2013. DOI: 10.1109/TSG.2013.2241087.

[16] L. Tesfatsion, "Agent-based computational economics: Growing economies from the bottom up.," *Artificial Life*, vol. 8, no. 1, pp. 55–82, 2002.

[17] L. Tesfatsion, "Modeling economic systems as locally-constructive sequential games," *Journal of Economic Methodology*, vol. 24, no. 4, pp. 384–409, 2017.

[18] E. F. Fama, "Efficient capital markets: A review of theory and empirical work," *The Journal of Finance*, vol. 25, no. 2, pp. 383–417, 1970, ISSN: 00221082, 15406261. [Online]. Available: http://www.jstor.org/stable/2325486 (visited on 11/29/2022).

[19] B. G. Malkiel, "The efficient market hypothesis and its critics," English, *Journal of Economic Perspectives*, vol. 17, no. 1, pp. 59–82, 2003, Cited By :841. [Online]. Available: www.scopus.com.

[20] D. Friedman and J. Rust, *The Double Auction Market: Institutions, Theories, and Evidence*. Reading, Mass., 1993.

[21] D. Friedman, "A simple testable model of double auction markets," *Journal of Economic Behavior & Organization*, 1991.

[22] M. Babaioff and N. Nisan, "Concurrent auctions across the supply chain," *CoRR*, vol. abs/1107.0028, 2011. arXiv: 1107.0028. [Online]. Available: http://arxiv.org/abs/1107.0028.

[23] R. B. Myerson and M. A. Satterthwaite, "Efficient mechanisms for bilateral trading," en, *Journal of Economic Theory*, vol. 29, no. 2, pp. 265–281, 1983. (visited on 02/24/2021).

[24] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[25] P. Werbos and P. John, "Beyond regression : New tools for prediction and analysis in the behavioral sciences," Jan. 1974.

[26] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

[27] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989, ISSN: 0893-6080. DOI: https://doi.org/10.1016/0893-6080(89)90020-8. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0893608089900208.

[28] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989. DOI: 10.1162/neco.1989.1.4.541.

[29] M. D. Zeiler and R. Fergus, *Visualizing and understanding convolutional networks*, 2013. DOI: 10.48550/ARXIV.1311.2901. [Online]. Available: https://arxiv.org/abs/1311.2901.

[30] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, 2015. DOI: 10.48550/ARXIV.1512.03385. [Online]. Available: https://arxiv.org/abs/1512.03385.

[31] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, M. Hasan, B. C. Van Essen, A. A. S. Awwal, and V. K. Asari, "A state-of-the-art survey on deep learning theory and architectures," *Electronics*, vol. 8, no. 3, 2019, ISSN: 2079-9292. DOI: 10.3390/electronics8030292. [Online]. Available: https://www.mdpi.com/2079-9292/8/3/292.

[32] D. Silver, A. Huang, C. Maddison, A. Guez, L. Sifre, G. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–489, Jan. 2016. DOI: 10.1038/nature16961.

[33] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. P. Agapiou, M. Jaderberg, A. S. Vezhnevets, R. Leblond, T. Pohlen, V. Dalibard, D. Budden, Y. Sulsky, J. Molloy, T. L. Paine, C. Gulcehre, Z. Wang, T. Pfaff, Y. Wu, R. Ring, D. Yogatama, D. Wünsch, K. McKinney, O. Smith, T. Schaul, T. Lillicrap, K. Kavukcuoglu, D. Hassabis, C. Apps, and D. Silver, "Grandmaster level in StarCraft II using multi-agent reinforcement learning," en, *Nature*, vol. 575, no. 7782, 2019, Number: 7782 Publisher: Nature Publishing Group. (visited on 02/24/2021).

[34] OpenAI, : C. Berner, G. Brockman, B. Chan, V. Cheung, P. Debiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, R. Józefowicz, S. Gray, C. Olsson, J. Pachocki, M. Petrov, H. P. d. O. Pinto, J. Raiman, T. Salimans, J. Schlatter, J. Schneider, S. Sidor, I. Sutskever, J. Tang, F. Wolski, and S. Zhang, *Dota 2 with large scale deep reinforcement learning*, 2019. arXiv: 1912.06680 [cs.LG].

[35] J. Vázquez-Canteli and Z. Nagy, "Reinforcement learning for demand response: A review of algorithms and modeling techniques," *Applied Energy*, vol. 235, pp. 1072–89, Feb. 2019. DOI: 10.1016/j.apenergy.2018.11.002.

[36] A. Barto and R. Sutton, *Reinforcment Learning, an Introduction*, 2nd ed. MIT Press, 2018.

[37] A. Slivkins, *Introduction to multi-armed bandits*, 2019. DOI: 10.48550/ARXIV.1904.07272. [Online]. Available: https://arxiv.org/abs/1904.07272.

[38] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, *Playing atari with deep reinforcement learning*, 2013. DOI: 10.48550/ARXIV.1312.5602. [Online]. Available: https://arxiv.org/abs/1312.5602.

[39] H. Hasselt, "Double q-learning," in *Advances in Neural Information Processing Systems*, J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, Eds., vol. 23, Curran Associates, Inc., 2010. [Online]. Available: https://proceedings.neurips.cc/paper/2010/file/091d584fced301b442654dd8c23b3fc9-Paper.pdf.

[40] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, *Prioritized experience replay*, 2015. DOI: 10.48550/ARXIV.1511.05952. [Online]. Available: https://arxiv.org/abs/1511.05952.

[41] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas, *Dueling network architectures for deep reinforcement learning*, 2015. DOI: 10.48550/ARXIV.1511.06581. [Online]. Available: https://arxiv.org/abs/1511.06581.

[42] M. G. Bellemare, W. Dabney, and R. Munos, *A distributional perspective on reinforcement learning*, 2017. DOI: 10.48550/ARXIV.1707.06887. [Online]. Available: https://arxiv.org/abs/1707.06887.

[43] M. Hessel, J. Modayil, H. van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, *Rainbow: Combining improvements in deep reinforcement learning*, 2017. DOI: 10.48550/ARXIV.1710.02298. [Online]. Available: https://arxiv.org/abs/1710.02298.

[44] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, *Mastering chess and shogi by self-play with a general reinforcement learning algorithm*, 2017. DOI: 10.48550/ARXIV.1712.01815. [Online]. Available: https://arxiv.org/abs/1712.01815.

[45] B.-G. Kim, Y. Zhang, M. van der Schaar, and J.-W. Lee, "Dynamic pricing and energy consumption scheduling with reinforcement learning," *IEEE Transactions on Smart Grid*, vol. 7, no. 5, pp. 2187–2198, 2016. DOI: 10.1109/TSG.2015.2495145.

[46] R. Lu, S. Hong, and X. Zhang, "A dynamic pricing demand response algorithm for smart grid: Reinforcement learning approach," *Applied Energy*, vol. 220, pp. 220–230, Jun. 2018. DOI: 10.1016/j.apenergy.2018.03.072.

[47] H. Zang and J. Kim, "Reinforcement learning based peer-to-peer energy trade management using community energy storage in local energy market," *Energies*, vol. 14, no. 14, 2021, ISSN: 1996-1073. DOI: 10.3390/en14144131. [Online]. Available: https://www.mdpi.com/1996-1073/14/14/4131.

[48] L. Xiao, X. Xiao, C. Dai, M. Pengy, L. Wang, and H. V. Poor, *Reinforcement learning-based energy trading for microgrids*, 2018. arXiv: 1801.06285 [cs.SY].

[49] E. Foruzan, L.-K. Soh, and S. Asgarpoor, "Reinforcement learning approach for optimal distributed energy management in a microgrid," *IEEE Transactions on Power Systems*, vol. 33, no. 5, pp. 5749–5758, 2018. DOI: 10.1109/TPWRS.2018.2823641.

[50] S. Zhou, Z. Hu, W. Gu, M. Jiang, and X.-P. Zhang, "Artificial intelligence based smart energy community management: A reinforcement learning approach," *CSEE Journal of Power and Energy Systems*, vol. 5, no. 1, pp. 1–10, 2019. DOI: 10.17775/CSEEJPES.2018.00840.

[51] T. Chen and W. Su, "Local energy trading behavior modeling with deep reinforcement learning," *IEEE Access*, vol. 6, pp. 62 806–62 814, 2018. DOI: 10.1109/ACCESS.2018.2876652.

[52] D. K. Gode and S. Sunder, "Allocative efficiency of markets with zero-intelligence traders: Market as a partial substitute for individual rationality," *Journal of Political Economy*, vol. 101, no. 1, pp. 119–137, 1993.

[53] T. Chen and W. Su, "Indirect customer-to-customer energy trading with reinforcement learning," *IEEE Transactions on Smart Grid*, vol. 10, no. 4, pp. 4338–4348, 2019. DOI: 10.1109/TSG.2018.2857449.

[54] J.-G. Kim and B. Lee, "Automatic p2p energy trading model based on reinforcement learning using long short-term delayed reward," *Energies*, vol. 13, no. 20, 2020, ISSN: 1996-1073. [Online]. Available: https://www.mdpi.com/1996-1073/13/20/5359.

[55] S. Bose, E. Kremers, E. M. Mengelkamp, J. Eberbach, and C. Weinhardt, "Reinforcement learning in local energy markets," *Energy Informatics*, vol. 4, no. 1, p. 7, May 2021, ISSN: 2520-8942. DOI: 10.1186/s42162-021-00141-z. [Online]. Available: https://doi.org/10.1186/s42162-021-00141-z.

[56] E. Mengelkamp, J. Gärttner, and C. Weinhardt, "Intelligent agent strategies for residential customers in local electricity markets," in *Proceedings of the Ninth International Conference on Future Energy Systems*, ser. e-Energy '18, New York, NY, USA: Association for Computing Machinery, 2018, 97–107, ISBN: 9781450357678.

[57] I. Erev and A. E. Roth, "Predicting how people play games: Reinforcement learning in experimental games with unique, mixed strategy equilibria," *The American Economic Review*, vol. 88, no. 4, pp. 848–881, 1998, ISSN: 00028282. [Online]. Available: http://www.jstor.org/stable/117009.

[58] E. Mengelkamp, P. Staudt, J. Garttner, and C. Weinhardt, "Trading on local energy markets: A comparison of market designs and bidding strategies," in *2017 14th International Conference on the European Energy Market (EEM)*, 2017, pp. 1–6.

[59] J. Nicolaisen, V. Petrov, and L. Tesfatsion, "Market power and efficiency in a computational electricity market with discriminatory double-auction pricing," *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 5, pp. 504–523, 2001.

[60] D. Holmberg, M. Burns, S. Bushby, A. Gopstein, T. McDermott, Y. Tang, Q. Huang, A. Pratt, M. Ruth, F. Ding, Y. Bichpuriya, N. Rajagopal, M. Ilic, R. Jaddivada, and H. Neema, "NIST transactive energy modeling and simulation challenge phase II final report," National Institute of Standards and Technology, Gaithersburg, MD, Tech. Rep. NIST SP 1900-603, 2019.

[61] S. Ciraci, J. A. Daily, J. C. Fuller, A. R. Fisher, L. D. Marinovici, and K. Agarwal, "Fncs: A framework for power system and communication networks co-simulation," 2014.

[62] Q. Huang, T. E. McDermott, Y. Tang, A. Makhmalbaf, D. J. Hammerstrom, A. R. Fisher, L. D. Marinovici, and T. Hardy, "Simulation-Based Valuation of Transactive Energy Systems," *IEEE Transactions on Power Systems*, vol. 34, no. 5, 2019.

[63] S. Mittal, M. Ruth, A. Pratt, M. Lunacek, D. Krishnamurthy, and W. Jones, "A System-of-Systems Approach for Integrated Energy Systems Modeling and Simulation: Preprint," en, 2015.

[64] M. D. Ilić, R. Jaddivada, and X. Miao, "Scalable Electric Power System Simulator," in *2018 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, 2018.

[65] H. Neema, J. Sztipanovits, M. Burns, and E. Griffor, "C2WT-TE: A model-based open platform for integrated simulations of transactive smart grids," in *2016 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES)*, 2016. DOI: 10.1109/MSCPES.2016.7480218.

[66] H. Neema, J. Sztipanovits, D. J. Hess, and D. Lee, "TE-SAT: Transactive Energy Simulation and Analysis Toolsuite," in *2020 IEEE Workshop on Design Automation for CPS and IoT (DESTION)*, 2020.

[67] L. Tesfatsion, "Chapter 13 - electric power markets in transition: Agent-based modeling tools for transactive energy support," in, ser. Handbook of Computational Economics, C. Hommes and B. LeBaron, Eds., vol. 4, Elsevier, 2018, pp. 715 –766.

[68] Steven, Dc-May, and P. Atrazhev, *Trex-ai/trex-core: V3.7.5*, version v3.7.5, Oct. 2022. DOI: 10.5281/zenodo.7240411. [Online]. Available: https://doi.org/10.5281/zenodo.7240411.

[69] D. Arrachequesne, *Socket.io*, 2021. [Online]. Available: https://github.com/socketio/socket.io (visited on 02/24/2021).

[70] S. Zhang, *TREX Analysis Tools*, https://github.com/sd-zhang/TREX-Analysis-Tools, 2020.

[71] D. I. Dong Chen, "Sundance: Black-box behind-the-meter solar disaggregation," in *e-Energy '17: Proceedings of the Eighth International Conference on Future Energy Systems*, May 2017, pp. 45–55.

[72] EPRI, *Epri distribution system simulator*, 2021. [Online]. Available: https://sourceforge.net/projects/electricdss/ (visited on 02/24/2021).

[73] *Opendssdirect.py, version 00.* [Online]. Available: https://www.osti.gov//servlets/purl/1374532.

[74] A. Latif, M. Ikechi, USDOE, S. E. Industries, and I. Hawaiian Electric Company, *Pydss*. DOI: 10.11578/dc.20190514.1. [Online]. Available: https://www.osti.gov//servlets/purl/1512458.

[75] J. C. Fuller, S. E. McHann, and W. Sunderman, "Using open source modeling tools to enhance engineering analysis," in *2014 IEEE Rural Electric Power Conference (REPC)*, 2014, pp. C4–1–C4–5. DOI: 10.1109/REPCon.2014.6842209.

[76] J. Hu and M. P. Wellman, "Nash q-learning for general-sum stochastic games," *Journal of machine learning research*, vol. 4, no. Nov, pp. 1039–1069, 2003.

[77] K. Strunz, E. Abbasi, R. Fletcher, N. Hatziargyriou, R. Iravani, and G. Joos, *TF C6.04.02 : TB 575 – Benchmark Systems for Network Integration of Renewable and Distributed Energy Resources*. Apr. 2014, ISBN: 9782858732708.

[78] S. Barker, A. Mishra, D. Irwin, E. Cecchet, P. Shenoy, and J. Albrecht, "Smart*: An open data set and tools for enabling research in sustainable homes," *Proc. SustKDD.*, Jan. 2012.

[79] O. Jogunola, Y. Tsado, B. Adebisi, and R. Nawaz, "Trading strategy in a local energy market, a deep reinforcement learning approach," in *2021 IEEE Electrical Power and Energy Conference (EPEC)*, 2021, pp. 347–352. DOI: 10.1109/EPEC52095.2021.9621459.

[80] H. C. Hesse, R. Martins, P. Musilek, M. Naumann, C. N. Truong, and A. Jossen, "Economic optimization of component sizing for residential battery storage systems," *Energies*, vol. 10, no. 7, 2017, ISSN: 1996-1073. DOI: 10.3390/en10070835. [Online]. Available: https://www.mdpi.com/1996-1073/10/7/835.

[81] S. Zhang, R. Martins, M. Gul, and P. Musilek, "Economy of residential photovoltaic generation and battery energy storage in alberta, canada," in *2017 IEEE Electrical Power and Energy Conference (EPEC)*, 2017, pp. 1–5. DOI: 10.1109/EPEC.2017.8286177.

[82] C. CRISTEA, M. CRISTEA, I. BIROU, and R.-A. TÎRNOVAN, "Techno-economic evaluation of a grid-connected residential rooftop photovoltaic system with battery energy storage system: A romanian case study," in *2020 International Conference on Development and Application Systems (DAS)*, 2020, pp. 44–48. DOI: 10.1109/DAS49615.2020.9108954.

[83] K. Strunz, R. H. Fletcher, R. Campbell, and F. Gao, "Developing benchmark models for low-voltage distribution feeders," in *2009 IEEE Power and Energy Society General Meeting*, 2009, pp. 1–3. DOI: 10.1109/PES.2009.5260227.

[84] S. Zhang, *CIGRE North American Low Voltage Residential Circuit OpenDSS Model for T-REX*, https://github.com/sd-zhang/TREX-Analysis-Tools/tree/powerflow, 2022.

[85] S. Zhang, *TREX-Core-sd-zhang-dev*, https://github.com/sd-zhang/TREX-Core/tree/dev, 2020.

[86] P. Atrazhev and P. Musilek, "Investigating effects of centralized learning decentralized execution on team coordination in the level based foraging environment as a sequential social dilemma," in *International Conference on Practical Applications of Agents and Multi-Agent Systems*, Springer, 2022, pp. 15–23.

Appendix A: Configuration File

Listing A.1: example of configs.json used in T-REX

```
{
  "version": "3.7.0",
  "study": {
    "name": "descriptive_name",
    "description": "long_description",
    "start_datetime": "YYYY-MM-DD hh:mm:ss",
    "start_datetime_sequence": "sequential",
    "timezone": "America/Vancouver",
    "days": 1,
    "generations": 10,
    "profiles_db_location": "postgresql://un:pws@localhost/profiles",
    "output_db_location": "postgresql://un:pw@localhost",
    "sim_root": "/home/trex/sim/"
  },
  "server": {
    "host": "localhost",
    "port": "5100"
  },
  "training": {
    "hyperparameters": {
      "alpha_critic": {"start": 0.1, "stop": 0.2, "num": 2},
      "alpha_actor": 0.1
    },
    "curriculum": {
      "0": {
        "learning": false
      },
      "1": {
        "learning": true
      },
      "50": {
        "anneal": {"exploration_factor": [0.95, "multiply", 0.1]}
      },
      "100": {
        "anneal": {"learning_rate": [0.98, "multiply", 1e-7]}
      }
    }
  },
  "market": {
    "id": "",
    "type": "MicroTE3B",
    "close_steps": 2,
    "grid": {
      "price": 0.069,
      "fee_ratio": 1.1,
      "tou": {
        "5,6,7,8,9,10": {
          "7,8,9,10,17,18": 0.144,
          "11,12,13,14,15,16": 0.208
        },
        "1,2,3,4,11,12": {
```

```
          "11,12,13,14,15,16": 0.144,
          "7,8,9,10,17,18": 0.208
        }
      }
    }
  },
  "participants": {
    "R5": {
      "type": "Residential",
      "trader": {
        "track_metrics": true,
        "type": "qlearn_bandit",
        "bid_price": 0.07,
        "ask_price": 0.14,
        "learning": true,
        "reward_function": "net_profit",
        "use_synthetic_profile": "test_profile_1kw_constant_gl",
        "trader_specific_parameters": param(s)
      },
      "load": {
        "scale": 1
      },
      "generation": {
        "scale": 10
      },
      "storage": {
        "type": "Bess",
        "capacity": 10000,
        "power": 5000,
        "efficiency": 0.95,
        "monthly_sdr": 0.05
      }
    }
  }
}
```