Electrify Europe, 19 - 21 June 2018, Vienna

Economic Optimization and Control of Distributed Energy Resources – Deep Reinforcement Learning as an Option for Automated Real-Time Operation

Thomas Kalitzky

Managing Director

Qantic, Berlin, Germany

6th of April 2018

1. Introduction

The efficient operation of distributed energy resources (DER) can pose some severe challenges especially if DER ought to handle signals from a market or a grid operator:

a) To successfully integrate DER into larger energy systems (e.g. smart grids, microgrids) and markets (e.g. intraday and balancing market or peer-to-peer trading) it is crucial to enable the distributed units to quickly react to real-time signals. This increases the requirements for the speed of computation of the algorithms used to determine the units' optimal dispatch.
b) More and more measurement data that contains valuable information is available from distributed units. Algorithms should be empowered to appropriately make use of it.
c) Compared to centralized systems the optimal dispatch has to be determined simultaneously for a large number of smaller energy systems, each with its own technical particularities. This further increases the computational burden imposed by the necessary optimization algorithms.

In the recent past Deep Reinforcement Learning (DRL) has emerged from the field of artificial intelligence and since then solves complex automation tasks across a continuously broadening range of industries. This contribution will show how DRL can help to face the described challenges of efficiently operating DER in real-time environments. We present an algorithm that can dispatch numerous complex energy systems while frequently receiving near real-time information e.g. on market prices, weather forecasts and measurement data. We aim to prove that DRL is well fitted to extract the relevant information from large data streams and to deal with the stochastics of prices, demand and renewable production as well as time-varying quality of data and forecasts. The algorithm is compared to conventional optimization methods.

This paper is structured by first giving an overview about the general challenges of optimally dispatching an energy system and the advantages and limitations of conventional methods. We then explain the basic functioning of DRL and finally illustrate its benefits by presenting two case-studies. We try to give the reader a good intuition about DRL and its potential benefits without diving too deep into any mathematical details as we want this topic to be accessible to a broad audience of energy experts from different sub-domains.

The content is structured by following chapters:

1. Introduction
2. Background – Optimization Methods
3. Background – Deep Reinforcement Learning
4. Case Study A – Operating a Hybrid Microgrid
5. Other Applications
6. Case Study B – Planning a Hybrid Microgrid
7. Summary
8. References and Recommendations for Further Reading

2. Background – Optimization Methods

Before stepping into the details of Deep Reinforcement Learning and its applications in system operation, in this section we want to clarify some general definitions and specify what it is meant when we talk about optimally controlling a system.

What to control and optimize?

The general terms “optimization” and “optimization problems” refer to a broad class of mathematical problems and corresponding analytical, numerical or heuristic methods to solve these problems. Diverse approaches from the domains of mathematics, economics, operations research, statistics and engineering claim to solve optimization problems [2][7]. In the domain of engineering optimal control theory is often utilized to find a control law for a given system such that a certain optimality criterion is achieved. It was decided to use the term Economic Optimization in the title of this contribution to emphasize that the optimality criterion we want to apply is fully based on economic measures such as costs or profits, i.e. we are seeking to minimize costs or maximize profits by optimally controlling a system.

When we refer to the task of controlling a system we mean the process of choosing and applying a certain action among a subset of permitted actions that can be performed on the system given the current state of the system. These actions (e.g. starting a power plant, charging a battery, turning off an interruptible load) affect the system’s mode of operation and lastly its economic performance. In other words we are looking for the optimal dispatch of all controllable energy resources in a given energy system which maximizes profits or minimizes

the total costs. In many applications this simply means we have to find the combination of energy sources that can satisfy a given demand at least costs.

Why optimize?

If we stick with this very simple definition of our task, shouldn't it then be a trivial problem to find the cheapest energy sources needed to meet a certain electricity demand? The answer is of course: It depends. Assuming that all our energy sources have well known decision-relevant costs of operation, we can simply order all sources by costs at every point in time and pick the cheapest to satisfy the demand at a given time (merit-order-approach). This will automatically give us the optimal solution to control the system and is quite trivial to implement. If we look a little bit closer, we will find that this idealized approach can be quite complicated to implement for any real-world system. This is because determining the correct decision-relevant costs is often far more challenging than we might assume at first glance.

Energy systems are often marked by intertemporal effects. This means that the actions we are taking to control our system at a certain point in time will affect the possibilities and costs to take actions on our system in the future.

Let us assume we have to decide whether to start a plant. The plant might have relatively low costs of electricity generation compared to other electricity sources but relatively high costs to perform a start when being cold. Simply considering the total costs of electricity generation and starting the plant might let it seem unprofitable to turn it on. But perhaps this might have been the optimal action as we could use the plant also in the following time-steps to produce electricity at low costs once it has been started.

An even more obvious example of intertemporal effects is given by storage technologies. Discharging a battery will often be a cheap source to serve a given demand as it can avoid using energy from fossil sources or buying at higher costs from a market. But discharging a battery obviously requires that the battery must have been charged somehow in the past. If our control-strategy only considers the current costs, why should we ever charge a battery if this will probably by associated with some costs and does not yield any immediate profit? The simple merit-order-approach obviously does not give us a good solution here. The deeper we dive into the subject and the more detailed we are investigating the characteristics of the technologies of our energy system the more intertemporal effects we will find. In complex energy system we are likely to find a diversity of those.

How to optimize?

The more pronounced the intertemporal effects are in our considered system the more money we will lose by operating it with a simple merit-order-approach. The first thing we can do to avoid this is to adjust the way we order and choose our energy sources by costs (merit-order). We can use some simple heuristics which tell us for example when to charge or discharge a battery, or when to start or stop a plant by following some rule of thumb.

In our context heuristics are any approaches to problem solving that employ a practical method not guaranteed to be optimal or perfect, but sufficient for the immediate goals [6]. In other words, a heuristic will almost be surely missing the optimum when dispatching a complex energy system, but there is nothing wrong with using one as long as we can live with that limitation. However, in the following case studies we will show that the performance of heuristics can differ significantly from the full economic potential which could be opened up by advanced optimization techniques. Perhaps some users of heuristics might be less satisfied with their approach if they would see the performance gap they have caused by oversimplifying the control of their system.

The two commonly used concepts to implement advanced optimization techniques for control of energy systems we want to consider in this paper are Dynamic Programming (DP) and Mixed Integer Linear Programming (MILP) [2][4][7]. It is far beyond the scope of this contribution to explain these algorithms in detail and there is already a variety of literature describing the application of these methods on energy topics. Hence, we will simply refer to these methods as a benchmark and compare their advantages and disadvantages to our proposed algorithm without diving into any mathematical details.

As hinted in the examples of intertemporal effects, it is somehow necessary to look into the future to optimally control an energy system. But how can we accomplish that in a real-world situation, where we usually do not have perfect foresight on all relevant system metrics (e.g. wind and solar production, grid load)? One way is to simply conduct forecasts on these relevant metrics and then to optimize using these forecasts. This type of approach could also be classified as a simple form of model predictive control. A big drawback of this approach is that, in its simple form, it does not take uncertainty into account. This approach is only guaranteed to perform optimally if no deviations between forecasted and actual values occur. In this sense the presented approach is a kind of deterministic optimization.

We could try to make some refinements to incorporate forecast errors but to fully take uncertainty into account we will have to move on to another approach called stochastic optimization. These types of algorithms can deal with forecasts as well, but will also take an expectation of the error of forecast into account.

Using a stochastic model may lead to a completely different system dispatch: Let us for example consider a grid-disconnected microgrid. The electricity demand in this microgrid has to be met by electricity sources within the microgrid. Given a certain load, wind and solar forecast and a given state of charge of a battery, it might not be necessary to start a plant in the microgrid as the load can be served cheaper from the other sources. But if the load exceeds the forecasted values above a certain level it cannot be served without using the plant. Let us furthermore assume that the plants needs some time to start up. If the load immediately climbs up above a certain level, the grid will break down if we have followed the simple predictive approach, potentially leading to very high costs, because there is no time left to start the plant. This is because the model has no possibility to consider how risky its decisions are in the presence of forecast errors. It also has no idea of the precision of the forecasts that it receives. A stochastic model might have started the plant just out of security and will therefore outperform the simple predictive (deterministic) approach at least when evaluated over a longer time horizon.

What are the challenging tasks?

As we have now seen why using stochastic optimization might be the best approach to control energy systems, and have understood the mechanisms that allow it to potentially outperform other methods, let's try it out.

The optimization techniques DP and MILP mentioned above are both generally suited for stochastic optimization. This is usually done by implementing some sort of scenarios or tree-structures that indicate which range of potential states the system is expected to take in the future. Compared to a simple optimization based on a single forecast this can dramatically increase the calculation time of the model, especially when using a sufficiently large number of scenarios which also cover the less probable but crucial system states (consider the microgrid black-out example above).

These are really bad news as our computational resources will be already pretty much occupied to fight another phenomenon which is often called "The Curses of Dimensionality"

[7]: Even a simple storage resource can theoretically have (at least from a mathematical point of view) an infinitive number of possible levels of charge as this is a continuous variable (from completely full to completely empty everything in-between is possible). To model the storage using a simple DP-approach we need to break it down into a number of discrete potential states (e.g. 100%, 75%, 50%, 25% and 0% full). The smaller the number of levels we consider the less accurate will be our result. On the other hand every additional level of storage we consider in our model will significantly increase our calculation time. Another problem, especially MILP will suffer from, is non-linear behavior of system components. We can duck and dive by modeling the non-linear behavior through piece-wise linearization and adding new state variables but sooner or later this can also blow-up our model and our calculation time will increase dramatically [2][7].

If we have to model an energy system which exceeds a certain level of complexity we are often forced to make one or more of the following simplifications to keep it computationally tractable, when using conventional stochastic optimization:

- Reduce number of considered scenarios (poor covering of uncertainty)
- Only consider a part of all possible system states
- Simplify physical behavior of system components

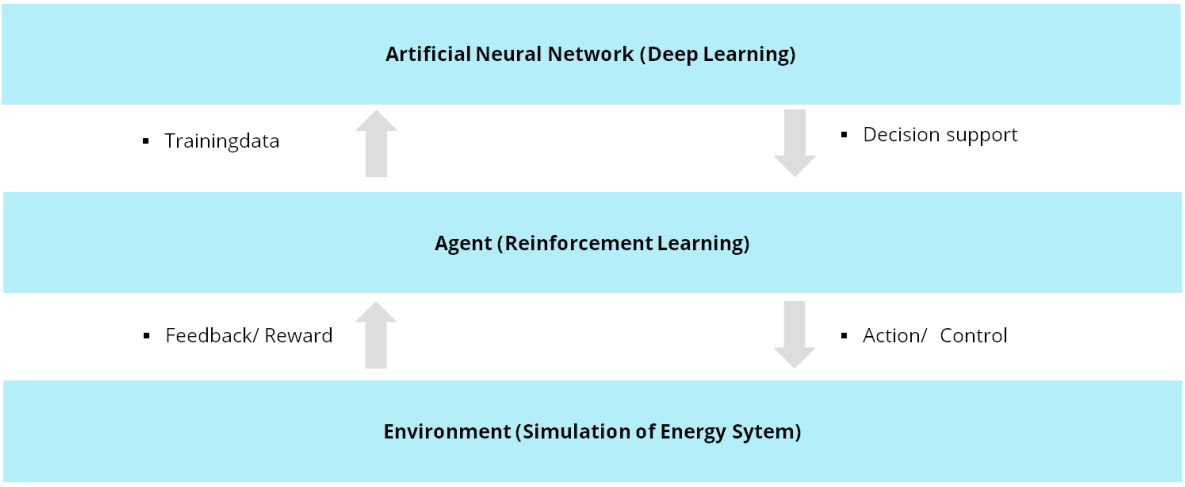
All of these simplifications must result in missing the theoretical optimum. In the worst case these effects might be so bad that the model is even outperformed by an approach which is only heuristic in nature but based on a precise representation of the real physical system. So, is all hope lost or is there any possibility to break the curses of dimensionality and to implement stochastic optimization without losing money by oversimplifying the representation of the physical system?

3. Background – Deep Reinforcement Learning

Deep Reinforcement Learning (DRL) is based on scientific results in Machine Learning (ML) and particularly Artificial Neural Networks (ANN) [1]. It is a novel technique in the field of Artificial Intelligence (AI) with has most prominently been applied by the company DeepMind, acquired by Google in 2014. DeepMind has shown the ability of DRL to learn computer games on human-level. In October 2015 a DRL-algorithm was the first computer program to defeat a professional human player in the game of Go which is one of the most challenging of classic board games [3]. In March 2016, a refined version of that algorithm

defeated Lee Sedol (the strongest go-player of the last decade with 18 world titles) by 4 games to 1. DRL is a very promising approach to optimally controlling complex systems across a broad range of industries.

DRL mixes two different approaches which will briefly be outlined in this section. *Reinforcement Learning (RL)* is a technique of Machine Learning and tries to find optimal policies to control a given system [2]. Similar approaches have also been developed under the name of Approximate Dynamic Programming which emerged to overcome the "Curses of Dimensionality" in large-scale economic optimization problems [7]. Independently from this field of research new techniques have been developed to make it possible to successfully train large ANN with several layers of neurons which resemble the structure of the human brain. This domain is often referred to as *Deep Learning (DL)* [5]. Both approaches are combined in DRL to make it possible to train an agent to control very complex systems using some sort of artificial intelligence. In our application DRL consists of a detailed simulation environment of the energy system and an agent which is allowed to perform actions on the simulation environment. The simulation environment determines how the system reacts under the chosen action and also calculates the costs which are corresponding with taking the action. The direct process of interacting with the environment (and learning about the system) is performed by a RL-framework. During the RL-process data is collected to train a deep ANN, which can give decision support for choosing a good action to control the system. In the following sections a little more insight is given how RL and DL work.



Reinforcement Learning

Reinforcement learning (RL) is inspired by the concept of reinforcement which is investigated in behavioral psychology. A reinforcement learning agent interacts with its environment (e.g.

a real system or in our case a simulation of an energy system) in discrete time steps. At each time, the agent receives an observation of the system state (e.g. storage level, generation, load) which typically includes a reward. In our applications the reward will often be the total costs/revenues of all components of the energy systems. It then chooses an action (e.g. start a plant) from the set of available actions, which is sent to the environment in a next step. The environment moves to a new state and the procedure starts again. The goal of a reinforcement learning agent is to collect as much reward as possible (or collect least costs as possible). The agent trains itself by (randomly) exploring actions on the environment while also exploiting the knowledge it has received in the learning process so far [2][7].

Many refinements have to be made to guide the learning process and actually make RL work, which are far beyond the scope of this contribution. But to understand the link between RL and DRL it is helpful to briefly touch a concept called Q-Learning which is a sub-part of RL.

Q-learning works by learning an action-value function Q (*s, a*), which ultimately gives the expected utility of a given action *a* while in a given state *s* [7]. When such an action-value function is learned, the optimal policy (i.e. optimal control) can be constructed by selecting the action with the highest value in a given state. Q (*s, a*) can be implemented as a simple lookup-table if the action-state-space is discrete and not to vast. It can also be a mathematical function which parameters are learned using some sort of regression technique. In the case of DRL, Q (*s, a*) is represented by a deep ANN which is trained during the learning process. This means the decision which action to pick given a certain state of the system is guided by some sort of AI, more specifically Deep Learning.

Deep Learning

Q (*s, a*) is now an ANN which tells us which value a certain action *a* has when the system is in a given state *s.* After the learning process is finished the system can easily be controlled by just asking the ANN which values the possible actions on the system have and picking the most valuable.

DL is based on ANN, a theory that has been around for a while but has recently received some boosts in performance and applicability due to some new research and progress in computer hardware [5].

As there are a lot of good formal explanations available on how DL works concerning its mathematical structure, a more intuitive insight will be given by explaining some neuro-scientific phenomena which reflect the idea of DL pretty well.

Everyone knows the situation where we coincidentally meet a person we have not seen for a long time, e.g. an old schoolmate. Although the haircut might have changed radically and the skin is significantly more wrinkled as the last time we saw that person, chances are good that we will recognize him or her immediately.

What seems to be a very simple task for us becomes quite a big challenge if we wanted to train a machine to perform that same action. So how do humans cope with that very challenging task of pattern recognition? One important key is that we obviously do not learn to recognize visual objects separately one by one, but rather learn very deep hierarchical structures that help us to distinguish different objects from each other.

Neuroscience has shown that the part of our brain which is responsible for processing visual information (the visual cortex) is organized in multiple layers of neurons which are somehow stacked. It can be shown that the first layers are only responsible to recognize very simple abstract geometrical forms such as vertical or horizontal lines. Deeper layers summarize this abstract information to other abstract but more complicated geometrical forms (maybe circles or squares). Hence, it is plausible to assume that before recognizing the face of our old schoolmate, we might have first recognized something that is round, then decided that this looks like a face, further decided that this face is from a human and not from a dog and so on. We obviously apply a whole structure of abstract hierarchical concepts. At the highest level we are recognizing something in the face of our old schoolmate that is very unique to him or her. Whatever that is, probably we are not even conscious about it, we have recognized it. And we even looked out for something that is not only unique but also persistent over time, so the changed haircut can barely fool us.

Perhaps there are also situations where we will not be able to recognize our old schoolmate. But another interesting point about learning in abstract hierarchical structures is that even if we fail to recognize our old schoolmate, we will probably never classify him or her in a totally wrong way e.g. as our neighbor’s old dog. If we would have learned all visual objects just one by one without using any hierarchical structures this could more easily happen.

Lastly, it is impressive how quickly we are able to decide whether we have seen our old schoolmate or not. We know that newborn children may take some time to develop the full set of visual skills, so there is obviously some training part going on in our brain to learn all the difficult hierarchical structures that we need. And probably we have no chance to recognize a person we have only once seen many years ago in a meaningless situation. But we had the chance of training to recognize our schoolmate throughout each of our school days. After all training is accomplished we are obviously very fast visual processors and pattern recognizers.

Without having been given many explicit definitions the reader should by now already have a good intuition how DL works. Generally speaking, it tries to mimic all of the learning behavior discussed above. DL consists of an ANN which is organized in many stacked layers enabling the network to learn very deep hierarchical structures of abstract concepts [5]. The ANN is trained to perform a certain task, e.g. to classify visual information, on a large dataset. After that training is accomplished the ANN is able to classify any new visual information it is given to.

Putting it all together

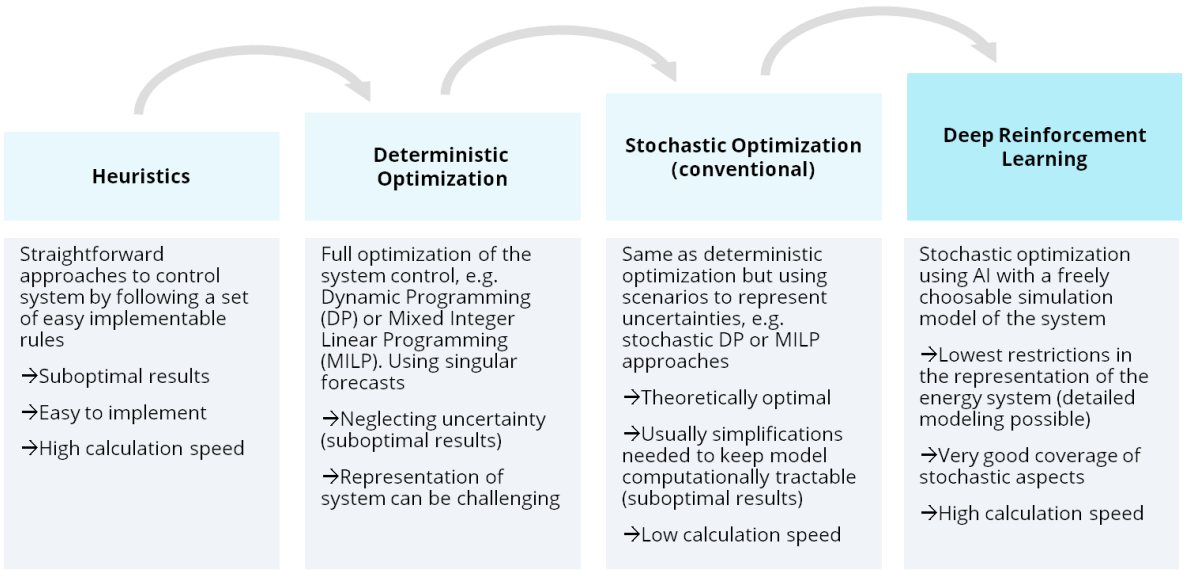
As mentioned in the prior sections we use DL to train our action-value function Q (s, a) from the RL framework. In other words we use DL to tell us how good it is to perform a certain action on the energy system in a given situation (state s). As DL is very powerful, we may put any additional information that we consider as relevant for our problem into our system state variable s (e.g. weather forecasts, measurement data, etc.). Fortunately, we do not have to care too much about performance issues as we find that DRL has miraculously helped us to overcome the curses of dimensionality (at the moment we just stay grateful about that without asking for any mathematical explanation). DL will decide on its own how relevant any of the additional information is and for which special patterns in the data it has to look. In a way DL may seem closer to a rule-of-thumb-like heuristic than to conventional optimization techniques. One may imagine that DRL finds out its own, possibly very complex and hierarchical, heuristics to convert the input data to the desired output.

Same as we do in the example of recognizing our schoolmate, DRL will look out for some persistent patterns in our input data. Consequently, it will not so easily be fooled by time-varying forecast or measurement errors as it will certainly find its way around to handle this (same as we won't get fooled by the changed haircut of our old schoolmate). In the same sense DRL is also very well suited to deal with any stochastic patterns. Due to its hierarchical

learning approach it can, to a certain degree, avoid giving us bad solutions if it fails for any reasons to give us the optimal solution (remember us rarely recognizing our old schoolmates as our neighbors' old dogs, even if we fail to recognize them correctly as our old schoolmates).

It is another very important aspect that a trained DRL-model will be order of magnitude faster in controlling a system than conventional optimization techniques. This makes DRL applicable for challenging tasks of real-time-control which otherwise could only be handled by simple heuristic approaches.

Compared to conventional methods DRL does not need any mathematical representation of the system it has to control as long it can directly interact with the system. Although we choose in this paper to let DRL learn by interacting with a detailed simulation model of an energy system it must be emphasized that we are relatively free to choose how detailed we want this simulation to be. Not only that computational aspects are far less severe as with conventional approaches (we have broken the curses of dimensionality) we are for example not restricted to stick to linear behavior of components and not even to a deterministic behavior at all.



What are the benefits?

A critical reader might have already noticed that so far we have successfully sidestepped any qualified statement on how well DRL performs economically compared to other approaches. This is because the answer is somehow tricky and it is good to accompany the theoretical solution with some practical considerations.

We know that conventional optimization techniques (DP or MILP) are guaranteed to find the optimal solution in cases where the structure of the energy system allows them to do so, i.e. they can be correctly implemented without any simplifications [2][7]. Heuristics are likely to give us a suboptimal solution but are under best circumstances as good as conventional optimization techniques [6]. A DRL-algorithm normally stops its training after reaching a certain level of convergence [1][2][7]. We can quite easily check that it is only stopped after being significantly better than the best available heuristic by simply implementing that heuristic in parallel. But we do not know how well it performs compared to a conventional technique without having implemented also a conventional optimization in parallel. Having said that, we know in advance that also our DRL-algorithm can at best only find the optimal solution, what is exactly what the conventional optimization is guaranteed to find.

From a practical point of view it consequently does not make very much sense to implement a DRL-model when we are also able to build a conventional model at the same level of detail without reaching computational limits. We already know that under best circumstances the DRL model can be as good as the conventional model, so why should we use one additionally. It is consequently only sensible to implement DRL where conventional models cannot be implemented at the same level of detail. But in these cases we simply have to accept that we do not know how well DRL would have performed compared to a similar conventional optimization model.

Apart from that, we might want to compare the performance of all three approaches (heuristic, conventional optimization, DRL) to just get a feeling whether DRL is a good optimization technique at all. Hence, we decided to compare the approaches on some demanding optimization tasks which have no obvious solutions. We implemented all three approaches at the same level of detail and applied it to tasks such as dispatching an idealized storage facility to generate arbitrage in a highly volatile market. We found that DRL can easily find (almost) the optimal solution and heuristic approaches perform clearly worse. The differences between conventional optimization and DRL were neglectible. Generally, DRL seems to be a good optimization technique.

In the following case study of operating an energy system using DRL we compare our model performance only to a reasonable heuristic. This is because we found no way to implement our case study with conventional models at the same or even similar level of detail.

Although the case studies might not seem too complex at a first glance we want to give a few examples for complex system behavior that can easily be implemented in DRL but might be challenging when using conventional stochastic methods. Fully implementing these effects can make conventional methods prohibitively slow, while neglecting these effects will lead to suboptimal results.

- non-linear fuel consumptions
- maintenance and lifetime (e.g. time-between-overhaul of genset) dependent on all past modes of operation (operation times and load levels)
- reserve power restrictions
- restrictions from exhaust emission standards
- recovery-time after critical loads (e.g. for gas-engines in low-load)
- effects of starts on lifetime and maintenance
- divers effects of ambient temperature and operation temperature of components
- load-dependant converter efficiencies
- varying maximum input and output power as well as efficiencies of batteries
- battery degradation and replacement based on
    - throughput
    - depth of discharge
    - state of charge
    - cell temperature

And there is even far more complexity we can bring into a DRL-model without getting any severe computational problems. Even if we find ways to incorporate the mentioned system behavior into a conventional model or at least succeed in simplifying some aspects without getting too far away from an optimal control, there are two aspects that can still make conventional models unusable: Complicated topologies and real-time applicability.

Complicated topologies can for example be found in industrial CHP-applications where steam demands of different pressures have to be satisfied and multiple steam sources are available, potentially also with several options for demand side management. But complicated topologies can also arise from sheer scaling, e.g. in microgrids or virtual power plants containing many similar but not identical distributed energy resources with diverse site-specific restrictions.

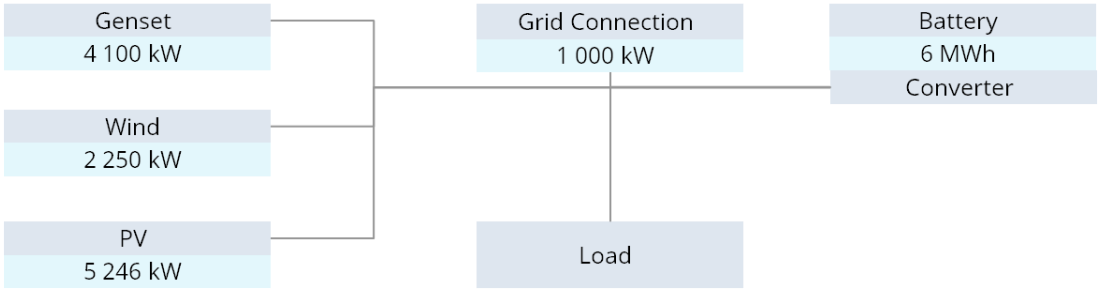
Real-time-applicability may also prohibit the use of conventional models. Imagine that an operator of an energy system can continuously buy or sell electricity on an intraday wholesale market. For example prices on the continuous European intraday market EPEX change with a high frequency and large part of market participants already use automated trading algorithm to keep track of the market. In this application it is crucial for optimal system control to consider market prices. Our conventional optimization might theoretically be perfect in doing so, but the calculation time it takes to react to changes in market prices will probably burn money. In a volatile market such as EPEX intraday there might be no (or even a negative) benefit from using an optimization model which only gives a strongly delayed signal. It would actually only tells us which optimal decision we should have taken in the past. We can be almost sure to miss market opportunities.

But continuous price signals can also come from a peer-to-peer-trading system or from a grid operator. And even in the absence of any market or grid operator (e.g. in an islanded microgrid) it might be necessary for the system to quickly react to fluctuations in demand and generation as it is crucial for the system to balance all deviations immediately and DRL can help to do this in a more cost-efficient way.

4. Case Study A – Operating a Hybrid Microgrid

Introduction

In this case study we investigate a microgrid where the demand of several thousand households and small businesses has to be met primarily by local distributed energy sources. In total 2.2 MW of wind generation and 5.2 MW of solar generation is available within that microgrid. To balance the production from renewable energy sources (RES) the operator of the microgrid decided to build a 6 MWh storage capacity. There is also the ability to purchase electricity from a public grid at a fixed tariff. The capacity of the grid connection is limited as our microgrid has grown significantly in the past and it showed to be unprofitable to enforce the grid given that there are enough local sources of electricity to meet the demand. A 4.1 MW genset delivers energy in some peak-hours and is used as a redundancy for the unreliable grid connection at this site.



Results

We modeled the microgrid using the DRL approach and evaluated its performance over a typical year. We compared the result to a widely used heuristic also called load following. The heuristic consists of serving the demand with the cheapest source of electricity (merit-order approach) with an additional rule to dispatch the battery. The battery is filled every time when the local RES-production exceeds the demand. The battery is discharged when its costs are lower than those of using the genset or buying from the market. Start costs of the genset are neglected in both approaches for a fair competition of the models as the described load following heuristic would have no chance to cope with this sort of condition. Because our heuristic is not able to deal with it, we also abstain from using any load or RES forecasts although DRL is well suited to process large amounts of additional data, which would surely improve the results of our DRL-model. In this sense, our results only mark a lower boundary of the benefits that can be achieved by DRL and must be interpreted as conservative estimate.

Although the heuristic is quite reasonable and seems to be widely used – e.g. Homer Pro, a popular software claiming to be the global standard for optimizing microgrids, uses this exact strategy as a default – there is a large potential to reduce costs by implementing DRL. The total costs of buying energy from the market, operating the genset and costs for maintenance and replacement of battery and genset (i.e. total costs without RES) are reduced by 16.9%. Costs for RES sources stay the same in both approaches as they cannot really be influenced by the optimization.

16.9 % cost reductions from optimized usage of genset, grid and battery compared to a heuristic

As it is crucial for a reliable microgrid operation that the demand has to be balanced immediately and that the restriction from the grid connection is met continuously, we decided to optimize our microgrid in 20 millisecond time steps (at a 50Hz frequency). In every time

step we calculate the results from the heuristic and the full stochastic optimization of our DRL model. Even if we would somehow be able to model the system with a conventional stochastic optimization technique it would be probably impossible to conduct a full stochastic optimization run within 20 ms even on high performance hardware. Using DRL, this task can easily be performed on a consumer notebook with a simple Intel Core i3-3110M unit, and it does not even bring the machine to its limits. Of course the DRL needs to be trained in advance on another independent set of test data (so that it has no chance to simply memorize our evaluation dataset). This can easily be performed in far less than one hour on the same machine to receive the presented accuracy of results.

5. Other Applications

The results of the presented case study can also be carried over to other applications. We want to give some examples where similar levels of costs reduction can be expected.

Flexible Grid Tariffs

Even if there would be no physical constraint in our grid connection it might be that there are some monetary incentives or penalties from the grid operator to reduce the maximum withdrawal from the grid. These incentives or penalties may even be time-dependant or flexibly taken by the grid operator on short notice. In any case DRL will be a good solution to optimally use these incentives or avoid penalties as it has already proven to deal with a hard physical restriction of grid-use.

Virtual Power Plants and DER

Furthermore, our system may not consist of centralized units for genset and battery but of a virtual power plant and virtual storage made up by several smaller DER. In this case we would have to deal with a far more complex system. But this would not impress us, as we have broken the curses of dimensionality and could simply adjust our simulation environment. Even in the worst (very unrealistic) case, the calculation time can be at most directly proportional to the calculation time of the simulation model (our model is still far away from that). But even in this overly pessimistic case, we would at maximum need one second to optimize a system which is 50 times more complex (needs a simulation of the environment that calculates 50 times longer) and here we are not even considering a faster hardware or parallel computing.

Peer-to-Peer-Trading and Blockchain

Perhaps we can also (partially) replace our conventional market access by a peer-to-peer-trading system in which DER from in- and outside the microgrid can participate (e.g. using blockchain technology). A topic that is often disregarded when talking about blockchain and peer-to-peer is that the challenge of optimizing the system stays the same or might even get harder because of the higher level of disaggregation – blockchain alone as a technology does not comprise any suitable optimization techniques. If price-signals are no longer (exclusively) received from a conventional market or by incentives from a grid operator, the peer-to-peer-system has to balance itself out automatically or generate the appropriate price signals to incentivize its users to do so. In any case the dispatch of the units should remain optimal, independently from the chosen market design. DRL could be implemented at a central instance of the system (producing appropriate price signals) or decentralized at the level of DER (processing price-signals and potentially placing bids and offers in a peer-to-peer-system). Due to its high performance DRL seems to be a good partner for any (potentially blockchain-based) peer-to-peer system.

6. Case Study B – Planning a Hybrid Microgrid

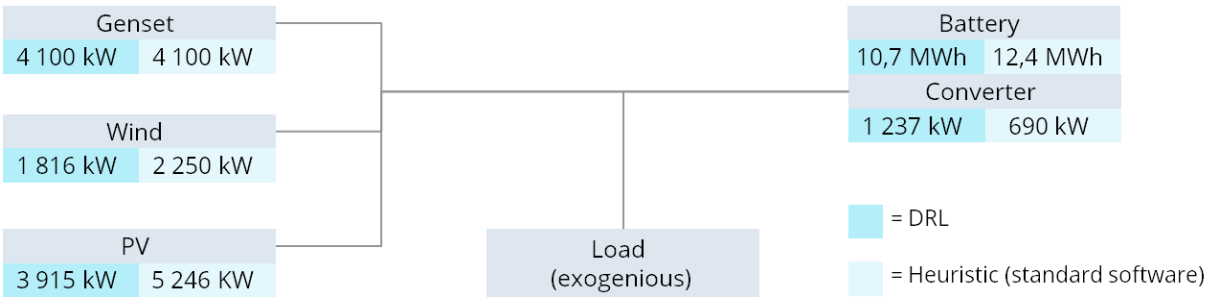
Introduction

As we have shown that operating a microgrid with DRL will yield significant benefits, how can we even get more out of DRL? Perhaps some of the system components of our microgrid are not dimensioned appropriately and when operating with DRL we might find that some components seem to be over- or undersized. Perhaps we could save money in advance, in addition to the reduced costs from operating the system with DRL, by reducing failed investments.

Consequently, we decided to let DRL help us finding the optimal dimensioning of all components in our microgrid. We use the same base-case as for our prior example. But this time we also want to investigate how we could operate the microgrid even without using any external grid and energy market. By this, we want to make clear that the benefits of DRL are not limited to finding any sort of market arbitrage and also fully arise in the absence of a proper market. Another slight difference is how we model system components, especially our battery and converter. We do that, because we want to compare our results directly to a standard software for microgrid optimization which is using a load-follow-heuristic.

To assure that we make a fair comparison, we firstly implemented a load-follow-heuristic by ourselves and adjusted the behavior of all our components until we were able to mimic the results of the software with a deviation of less than 0.5 % in all important metrics of the components. Overall we were able to reproduce the total costs calculated by the software with a deviation of only 0.02 %. Hence, we can be pretty sure that we are modeling components in a very similar way as the software and that all benefits we will see from using DRL in the next step will almost exclusively come from dispatching and planning the system in a more cost-efficient manner.

Results



We can see significant differences in the optimal dimensioning when using DRL vs. the heuristic (the external software) with deviations of over 75% in some components. This leads to a reduction in initial investments of more than 20% compared to the heuristic. Furthermore, the carbon emissions produced by operating the system with DRL are 30 % lower. This is surprising as DRL is also installing less wind, solar and storage capacity. Obviously DRL can make a better use of the resources and avoid curtailment of RES which results in a higher use of renewable energy.

20 % lower investment when using DRL in system-planning
30% lower CO_2-Emissions

It has to be stated clearly that there is nothing wrong with the compared software. It solely assumes that the microgrid will be operated using a load-following-heuristic. If this is really the case, then it gives a pretty good recommendation of the system design. It can be said that in any case the algorithms for planning and operating the system should be aligned. But we also successfully demonstrated that it is a good idea to operate and additionally also plan systems using DRL.

7. Summary

We have shown that stochastic optimization techniques are generally well suited to control and operate energy systems. Other approaches such as deterministic models or heuristics cannot open up the full economic potential and will lead to higher costs of operation. However, in many applications the computation time of conventional stochastic optimization techniques can be prohibitively high. Other approaches have to be found in cases where the system complexity has exceeded a certain level or when real-time-applicability is necessary. It has been shown that Deep Reinforcement Learning is well suited for real-time-applications as it is order of magnitude faster than conventional optimization techniques. It is also capable of modeling very complex system behavior. Deep Reinforcement Learning is very suited to perform stochastic optimization in domains where system control and operation could otherwise only be accomplished by heuristic approaches. We have presented case studies which prove that the use of Deep Reinforcement Learning can lead to significant reductions in costs and emissions as compared to widely used heuristic approaches.

8. References and Recommendations for Further Reading

[1] A Beginner's Guide to Deep Reinforcement Learning visited on 06. April 2018 at https://deeplearning4j.org/deepreinforcementlearning

[2] Busoniu, Lucian; Babuska, Robert; De Schutter, Bart; Ernst, Damiem. Reinforcement Learning and Dynamic Programming using Function Approximators, CRC Press, 2010

[3] Hassabis, Demis. AlphaGo: using machine learning to master the ancient game of Go visited on 06. April 2018 at https://www.blog.google/topics/machine-learning/alphago-machine-learning-game-go/

[4] Hugo Morais, Pedro Faria, Zita A. Vale, H. M. Khodr, Péter Kádár. Optimal scheduling of a renewable micro-grid in an isolated load area using mixed-integer linear programming, Renewable Energy, Vol: 35, Issue: 1, Page: 151-156, 2010

[5] Li Deng, Dong Yu. Deep Learning: Methods and Applications, Foundations and Trends in Signal Processing Volume 7 Issues 3-4. 2014

[6] Pearl, Judea. Heuristics: Intelligent Search Strategies for Computer Problem Solving, New York, Addison-Wesley, 1983

[7] Powell, Warren. Approximate Dynamic Programming – Solving the Curses of Dimensionality, Second Edition, John Wiley & Sons, 2011