Shared powergrid in residential neighborhoods

Effective cost reduction using genetic algorithms

Christian Staufenbiel

April 25, 2019

Contents

1 Introduction

Solar power systems were installed on residential houses to a large extend. Due to high revenues, when selling power to the grid, this setup was feasible for a long time. However the rewards (so called feed-in tariffs) were reduced over the last years and storage systems (batteries) began to become more and more feasible, also for residential households[1]. However I think that setting up all these storage systems separately is not the most efficient way to do so. A more flexible solution, that allows sharing energy with neighbors possibly gives an even higher outcome and more efficient usage of produced energy. Therefore I want to explore, how to simulate the general behavior of such a neighborhood, using a data set of real-life measurements.
Another aspect is the distribution of solar panels within the neighborhood. If a shared storage system is introduced, other possibilities of setting up solar systems can emerge. For example, a house with roof that has lots of space for solar panels, should install an even bigger system than what would be feasible for the single household. It can produce more energy that would then be shared with neighbors, that might not have that much space available.
I developed a model, that is able to mimic the behavior of the described model. This implementation can be found in an online repository[2] and I try to keep the documentation as simple as possible. I used *python* to write a general model and used *Jupyter Notebooks* to perform the optimization task. As a backend I used the ERDA[3] (Electronic Research Data Archive) system of the SCIENCE faculty. With this backend I was able to run the algorithm in parallel and speed up the whole process of optimizing.
My main focus, while implementing the model was to keep it general, so one can easily edit certain parameters (such as costs and losses) or pass a different data set into the simulation routine. I want to emphasize that I have no affiliation to power grid constructions and the model may not be realistic. I don't have the expertise to make it more realistic yet, but I left a lot of general parameters open for changes, which can increase the feasibility of the model.
Exploring the data set and the model behavior took a long time in this process and therefore I did not come to all the aspects of the model I wanted to study. Mostly I will focus on optimizing the storage structure within the neighborhood, taken the already installed solar systems as fixed.

2 A simple grid model

For every optimization an implementation of a model is needed. The model represents the current state of the whole system and also provides sufficient functions in order to edit its current state. It also provides the functionality to determine the current fitness of the model.

[1]I speak here about my home country (Germany). Further information here: `https://en.wikipedia.org/wiki/Feed-in_tariff`
[2]`https://github.com/christian512/powergrid_neighbors`
[3]`www.erda.dk`

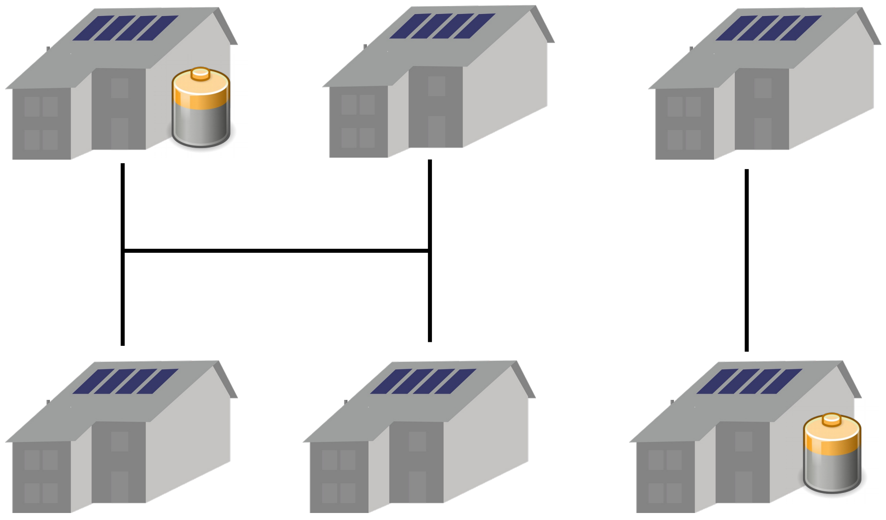


Figure 1: Small neighborhood

2.1 Representation

As shown in figure 1, a neighborhood consists of a fixed number of houses. Every house has a specific consumption schedule for every day, since there live different people, with different daytime schedules. They receive their energy from the *normal* energy grid, which costs them a certain amount of money per kilo-Watt-hour [kWh].
In order to produce their own energy, a photovoltaic (pv) system is attached on the roofs of the house.
Thus, for every daytime there is a specific consumption and production amount. We will focus here on time steps of 30 minutes. Hence we have the amount of energy consumed and produced within 30 minutes in units of kWh.
We expect that there might be times for each household, where their pv system provides more energy than they consume at that time. Since the reward you get for selling one kWh to the system is lower than buying one kWh, it might be feasible to build up a storage system. One can also think of a shared storage, that can be used by several households and therefore might reduce the amount imported energy from the normal power grid even more.
To describe the whole configuration of the neighborhood we need three lists.

- pv system installed on each house
- connection to storage for each house
- maximum capacity of each storage

2.2 Number of storage connections

Each house can be connected to a different storage (battery). We constrain the positions of the storages, to be at a specific house. A house, identified by the number x, is connected to a storage at position $y = f(x)$. The function f gives the connected storage for every house. Assuming equally spaced houses, the length of all connections can be calculated by:

$$s = \sum_i |i - f(i)| \tag{1}$$

If each house is connected to its own storage system, i.e. $f(i) = i \; \forall i$, the total length results in $s = 0$. The other extreme is, that each house is connected to one storage, e.g. storage number 0. That implies $f(i) = 0 \; \forall i$ and s resolves to $s = n(n+1)/2$, where n is the number of houses.
This quantity is used later on, to determine the connectivity between storages and houses.

2.3 Genetic operations

The genetic algorithm we use for our implementation is described later, but for all genetic algorithms we need operations as mutations and crossovers, which we will describe in this section.
As stated in the section before, the neighborhood is completely defined by three lists. In a usual genetic algorithm, we use a single gene representation `gr` for each individual. In the simplest form we can write this representation as a list of bits (a bit string). A *crossover* between two individuals will then create two children. The genes of the children are combined genes of the two parents, as seen in fig. 2.

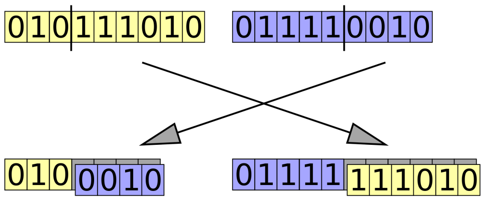


Figure 2: Genetic crossover

In the case for our grid neighborhood, we have three lists, that we can interpret as gene representations. Since we not only have two types of pv modules or storage systems, we have more than just zeros and ones in our gene representations. Additionally, there are three lists describing the individual. If we were to append all three list to one, only the last parts are crossed. Hence the properties described by the end of the joined representation (e.g. capacities) would have a higher crossover probability than the starting properties . Thus a crossover in our model can happen in each of the three lists individually.
The other genetic operation, we use later on, is *mutation*. In the picture of bit string

representations this is a change of the bit on a random position. For our model, we again separate this operation between the three lists. Therefore we can define a probability for a mutation in each of the lists separately in the genetic algorithm.

3 Data set

For simulating the behavior of the neighborhoods energy consumption, we use a data set of the Australian electricity distribution company *Ausgrid*. Their data set consists of data from 300 houses, randomly chosen within their supply grid. For each house the consumption was measured in a half-hourly rate. Additionally each of the houses has a photovoltaic system installed on its roof, whose energy generation was measured at the same frequency. The data set covers a whole year of measurements. Further information can be found on the Ausgrid website[4].
We will now plot some basic properties of the data set and discuss some choices for later simulation purposes.

3.1 Average day

Visualising a usual day throughout the year is the most interesting part of the analysis, since it shows us, if there is an energy overproduction at any time of the day. Only if this overproduction exists, it is feasible to use storages for later consumption. Otherwise the energy can be consumed directly.

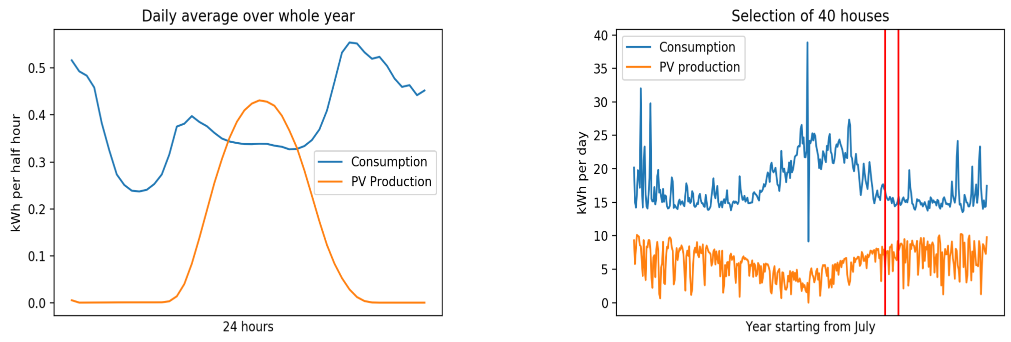


Figure 3: Average day and year

The consumption pattern looks as expected. A peak during the morning hours, when the residents are waking up and leaving for work. The absence of residents leaves the consumption rather constant during the daytime. During the evening residents start using most of their electrical devices to prepare dinner and for general entertainment.
The power generation peaks during daytime, where the consumption is low. Hence there

[4]`https://www.ausgrid.com.au/Industry/Innovation-and-research/Data-to-share/Solar-home-electricity-data`

is an overproduction, that could be used later during the day. This detail makes storage systems (batteries) feasible. But which specific setup of storages gives the optimal energy usage and reduces the costs for electricity, will be studied in the following.

3.2 Average year

To inspect the general quality of the data, we plot the daily averages of energy consumption and generation over all houses for the whole year.

In fig. 3 we observe the offset between pv power generation and consumption. The majority of houses seems to have pv systems, that do not cover their consumption at any time. However they still might benefit, from their installation. There are also some houses, that have approximately the same production as consumption (on a daily average) during the summer months (houses not shown here explicitly).
During the winter months the consumption rises, most probably, due to heating usage. As expected, the pv power generation decreases in this time period. The large peak between December and January can not be explained by any reasonable fact and might be a problem with measurement devices. Peaks in the consumption during the summer could be a result of cooling systems that some houses have installed.
For later simulations, taking the whole data set with this resolution of data points results in a high computational cost that needs to be spend when calculating objective functions. In order to avoid this problem, we will choose a smaller time period from that year, which represents the whole year relatively good. Explicitly, we chose a two week time frame (marked in red in fig. 3) during spring. If we can find an optimal solution for this time frame, this will most probably also perform well during summer (of course not optimal). For the winter period this might not be the best choice, but it should give an intermediate result. If one has more computation time available, the whole optimization can be done using the whole year data set, by just one change in the code. Hence we leave the option to extend the model/optimization to any complexity.

4 Costs and losses

Every optimization includes an objective function that gives every configuration a specific fitness. In our case that would be the cost for setting up the grid structure and the continuous expenses for electricity. The considered costs in our model are:

- Cost per kWp of the pv panel[5] (not used for optimization in this report)
- Cost per kWh of the storage
- Cost per kWh bought from the grid
- Reward per kWh sold to the grid

[5]This is the kilo-Watt-peak measure, which defines the efficiency of the solar panel by standardized conditions.

Another thing that needs to be considered, but is not implemented yet, is the price per cable to the storage. Currently we only include the loss that is due to the transmission of power through cables to the storage. Therefore we assume that each storage has its position at one specific house. Hence if a house has a overproduction it can put the energy into its storage without losses. As soon as it delivers or fetches energy from a storage not on its premises, losses are introduced.

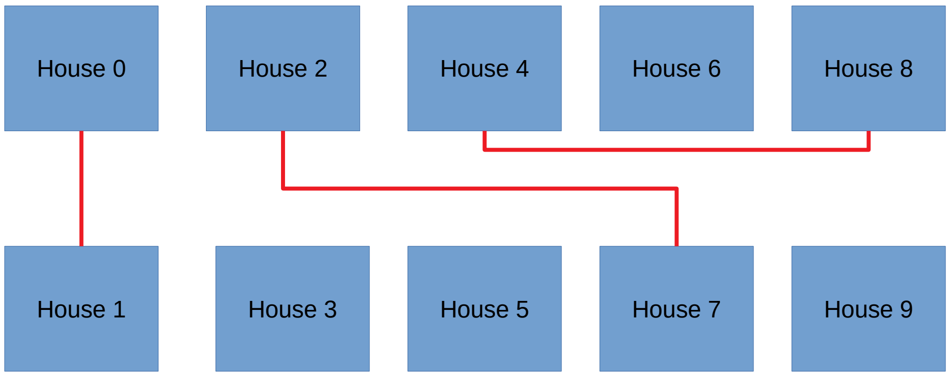


Figure 4: Connections in the neighborhood

Fig 4 illustrates our choice for the spatial distribution of houses and thereby storages (attached to each house). We define a loss A per length unit. If we connect house no. 0 with house no. 1 there will be a loss of A, since they are direct neighbors. A connection between house no. 4 and 8 has a loss of $2A$ since there is one house in between. Thus the connection between house no. 2 and 7 has a loss of $3A$.

4.1 Numerical values

We will just give a small overview of the estimations of costs we did for this simulation. Note that all prices are in Euros.

1 kWp of solar panel	1400 EUR
1 kWh of storage	500 EUR
1 kWh from grid	0.25 EUR
1 kWh to grid	0.10 EUR
loss per length unit	5%

The loss per length unit is chosen to be that high, since we do not include the prices of cables yet. If you think of houses that are approx. 30 meters apart, this would be a rather high loss. The other values are taken from various online resources. Again, we want emphasize here, that there exists a simple function within our model, that can change these values to more realistic ones, if needed.

4.2 Amortization time frame

Since we can not look on the whole life of a resident and the expenses during that time, we need to define a time period, after which the investment in the storage system should amortize. We set this time frame to be 10 years here. Again this is our own choice and can be changed easily within the implementation. If we now look at the two week period, the storage system in that time costs:

$$\text{costs for two weeks} = \frac{\text{investment cost} \cdot 14\text{days}}{10\text{years} \cdot 365\text{days}}$$

The same holds for the costs of cables and pv systems.

5 Genetic algorithm

As we already discussed during the lecture, genetic algorithms do not have strict boundaries in their implementation. Therefore we describe briefly the implementation we used here. Consider the two properties of the system: *storage connections* and *storage capac-*

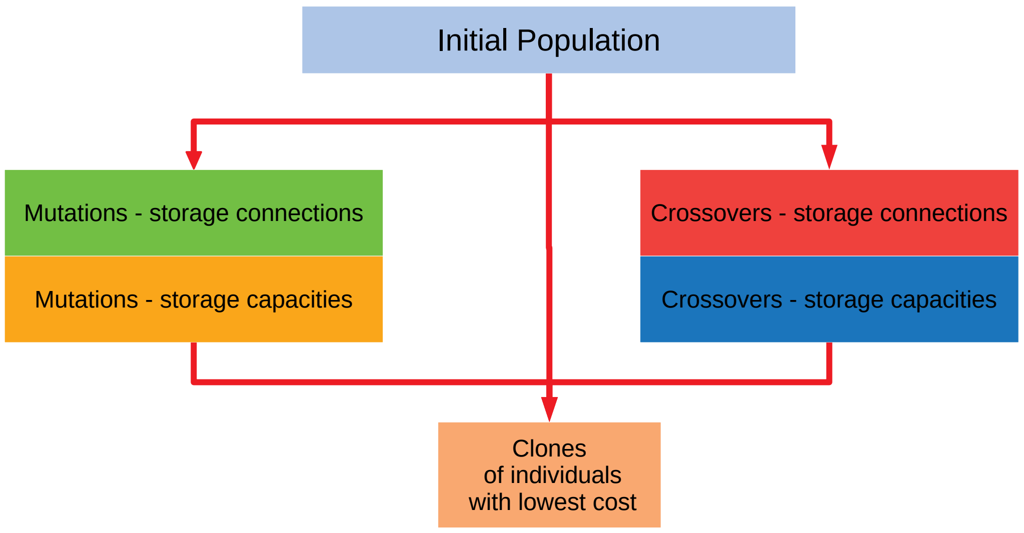

Initial Population
Mutations - storage connections
Mutations - storage capacities
Crossovers - storage connections
Crossovers - storage capacities
Clones of individuals with lowest cost

Figure 5: Genetic algorithm

ities.
For each property we define two probabilities: one for mutations and one for crossovers. Taking any of these probabilities p we then get the number of affected individuals, by

multiplying it with the size of the initial population (and rounding it to an integer value). Next step is to choose the individuals randomly from the initial population, create a copy of them and apply the corresponding genetic operation. This modified individual is then appended to the population.
After all mutations and crossovers happened, a certain percentage of the best individuals (with regards to the overall costs) is cloned. The population size is now somewhat higher and to move on to the next generation we drop the worst individuals, until we reach the size of the initial population.

6 Optimizing the storage system

The initial idea was to optimize the size of the pv system on each roof top and the storage system shared by all houses in order to have minimum expenses within the whole grid. However it turned out to be a really complex task to do so, especially if we consider many houses. Therefore we start off with taking a given number of houses and their attached pv system (just the ones from the direct measurement) and then optimize the infrastructure of the storage connections and their capacities. We neglect here, that the data was not only taken of houses that are close to each other, i.e. share the same weather data, and thus it is not realistic choose these as we did. But it will still show us if the genetic algorithm is able to find an optimal infrastructure.

6.1 Dimensions of the phase space

We will calculate the size of the phase space here. Therefore we need to consider the following properties:

- Number of houses n (40)
- Number of storages k (40)
- Number of possible capacities for each storage m (20)

Each house can connect to one storage, and each storage can have a capacity from the possible capacities. Hence the dimension of the phase space is given by:

$$D = k^n \cdot m^k = 40^{40} \cdot 20^{40}$$

6.2 Boundaries and expectations

The results of our optimization must be compared to specific grid setups, such that we know, if the result is better than other configurations. The simplest storage infrastructure would be no infrastructure at all. Another setup, which we will refer to as *individual-storages*-configuration, is that each house has its own battery. Hence there are no losses, as we set them to zero if energy is stored on-site. The other boundary

case would be only one storage for all houses (*one-storage*-configuration). This setup implies maximum sharing between houses, but also includes the highest losses possible. These two boundary cases are plotted for two different sizes of neighborhoods (20 and 40 houses) in fig. 6.
We expect that for small neighborhoods (approx. $1 - 10$ houses) the *one-storage*-configuration reduces the cost best. For a larger number of houses, we then expect that smaller sharing communities are created within the whole neighborhood. Closer neighbors would still share their energy, while neighbors with higher distance to each other, are unlikely to share a storage (due to high losses).

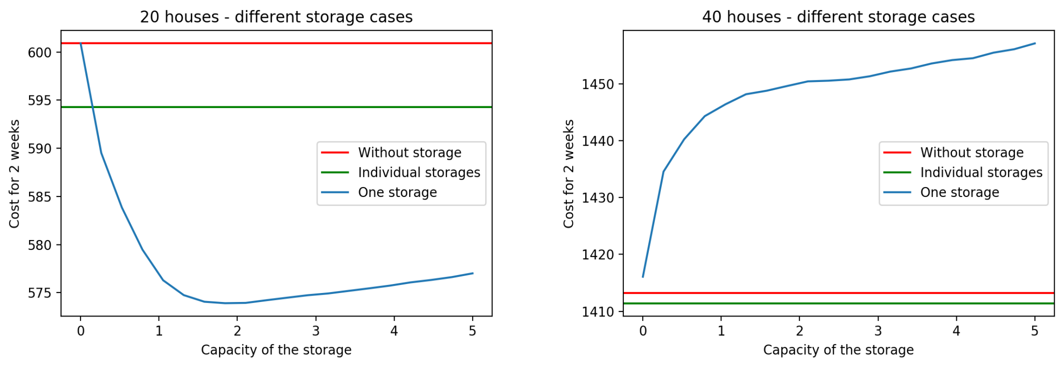


Figure 6: Boundary cases of storage infrastructure for different sizes of neighborhoods

We will focus on optimizing a neighborhood consisting of 40 houses in the following sections.

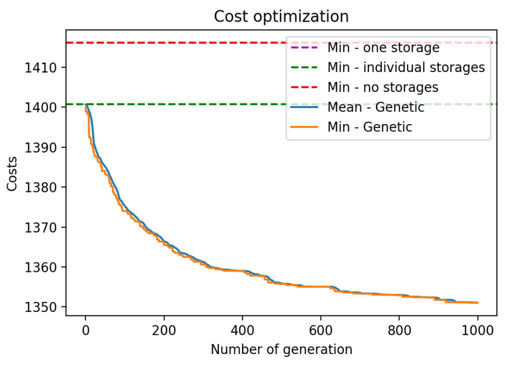
6.3 Initial population

We have seen before that at least one of the two boundary cases is better than the one without any storage. Normally, the initial population would consist of random individuals that, which then increase their fitness by all the genetic operations. In our case, we can use the knowledge of the two boundary cases to model our initial population in such a way, that it includes the optimal boundary cases already and optimize from there one. However we still need a variety of genes in order to pick the best properties. Thus we use a some individuals of the optimal boundary case as a germ cells for the whole population. These individuals will survive most of the starting generations, until we find a better solution than that. Contrarily we can also end up in a point where the population consists only of boundary case individuals. Then the algorithm starts its search, from the current optimal solution.

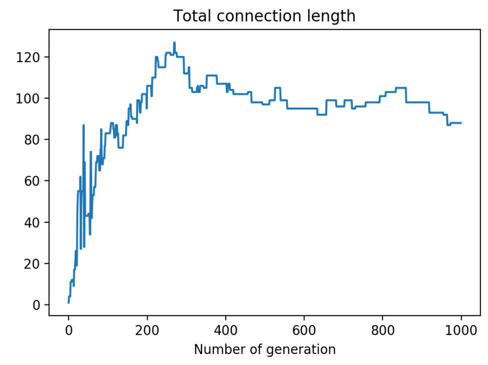
6.4 Connection optimization

For a first benchmark, we let the population evolve only through changes of house - storage connections. This reduces the size of the available phase space, since we are

not varying the storage capacities. We aim on finding a better structure of connecting the houses to storages, in order to verify our expectation of smaller clusters around one storage.
In fig. 7 we can see that the this strategy already improves the result from the *individual-storage* configuration.



(a) Costs of the population and boundary cases



(b) Total length of connections for the best storage infrastructure

Figure 7: GA for 50 individuals with 25% chance for connection mutation/crossover

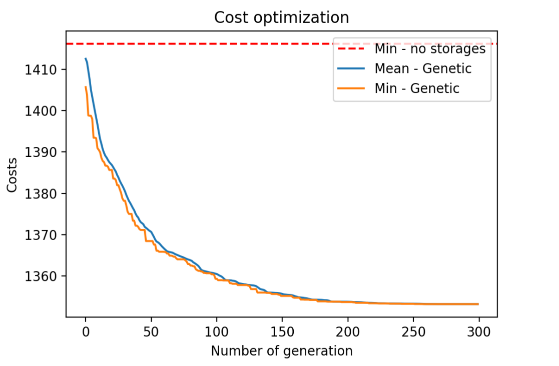
To inspect the setup of the grid infrastructure we can use the *number of storage connections* s that we introduced before. As we start of with the *individual storage* configuration, this implies $s_{initial} = 0$. The optimal solution found with this strategy corresponds to $s \approx 90$. The maximum s_{max} for this optimization with $n = 40$ houses is $s_{max} = 820$. This intermediate value of s indicates smaller clusters, that have formed within the grid. Due to the space, we can not visualize the connections in a plot. But an example list of each house and its connected storage is given below.

house no.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
storage pos.	9	1	1	9	9	1	9	9	9	9	9	15	9	9	22	9	15	19	25	19	19	25	27	22	22	25	22	27	22	22	33	32	32	32	33	32	32	33	32	33

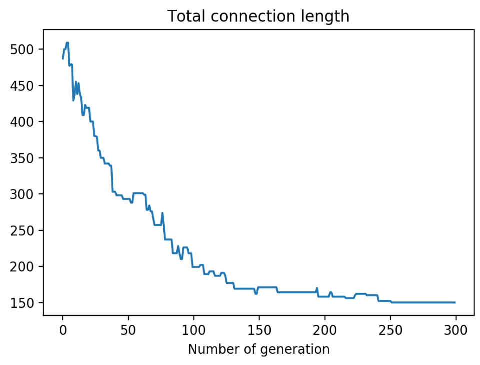
6.5 Artificial cluster individuals

After we've seen that our expected cluster solution seems to be perform very good, we construct artificial individuals of these, as a new population. This will reduce our phase space, as the individuals before all included the same number of storages as houses. Here we will cut down the number of storages for the 40 houses neighborhood to 5 storages. While optimizing the connections, we will now also optimize the size of the storages.

As we can see in fig. 8 a relatively constant value is reached in just $\frac{1}{3}$ of the generations than before. The optimal solution of this approach is slightly higher ($\approx 1353EUR$) than the one from before ($\approx 1351EUR$). Even though we also optimized the storage size in this example, we are not able to improve the results. This might indicate that some houses can interact better with other houses nearby. Hence giving each house an



(a) Costs of the population and boundary cases



(b) Total length of connections for the best storage infrastructure

Figure 8: GA for 50 individuals with 5 storages and 25% chance for each mutation/crossover

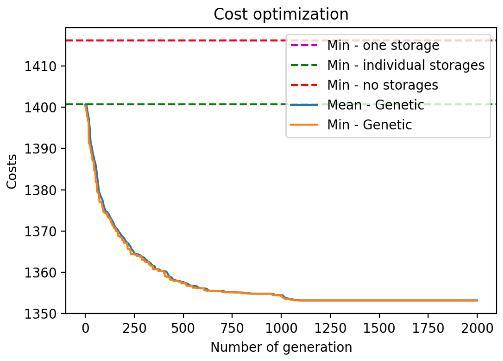
individual storage and then discarding some storages seems to be the better solution. Comparing the total length of connections, the result of this artificial storage clusters also ends at a higher value. This shows the higher losses in the artificial storage system. The freedom given by the individual storage approach, seems to be needed in order to minimize the total cost.
Even though we were not able to improve our results, reproducing them with a smaller computational cost was beneficial for analysing parameter choices for crossovers,mutations and cloning.

6.6 Capacity optimization

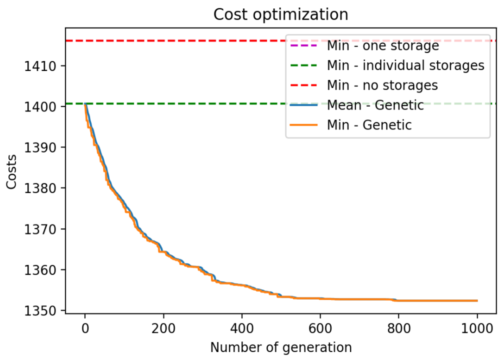
Adding the variation of the capacities extends the phase space to a even higher dimensions and therefore a search for the optimal solution will take even longer.
To avoid unnecessary mutations in the capacities, we only allow changes to storages that are actually connected to by a house within the neighborhood. As initial population we choose the *individual-storage* configuration. In order to explore the phase space around the best individuals a bit more , we set the probability of cloning the best individuals to 10%.
We'll follow two strategies within this part of the optimization.

- First optimize the connections, then the capacities
- Optimize connections and capacities at the same time

For the first schedule we choose 1000 generation for connection optimization and capacity optimization each. Optimizing both properties at the same time takes more computational power. Therefore we only use 1000 generations for the second optimization schedule.



(a) First connections, then capacities



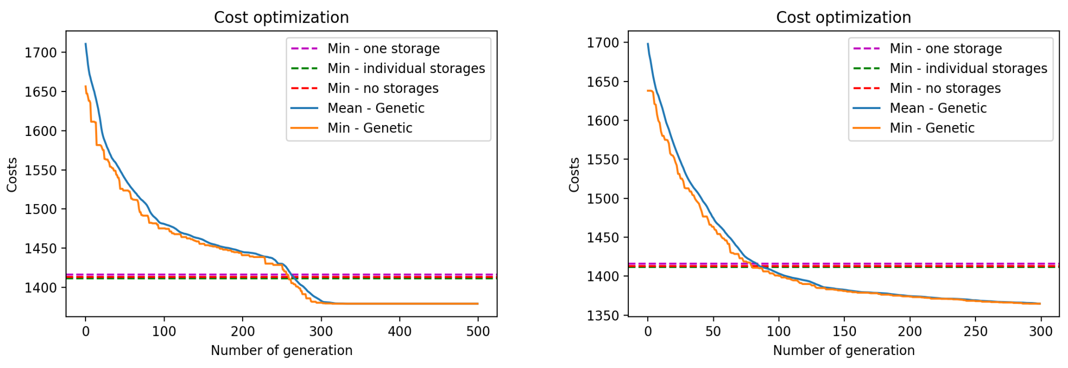
(b) Connections and capacities at once

Figure 9: Optimization for the two different schedules, as described above.

The results, shown in fig. 10, reveal the same what we've seen before: Changing the capacities of the used storages gives little to no change in the objective (cost) function. We can assume that our losses dominate the optimization and therefore the price per kWh of a storage battery, affects the overall cost only to a small extend.

7 Shorter amortization time

To increase the influence of the capacity, one has to increase the influence of the installation cost of the storages. This can be done by shortening the amortization time frame. If we cut this time frame to 5 years an installation for a single household is not feasible anymore (for most houses), i.e. optimal capacity is zero. But we could still try to find a configuration of storages, that optimizes the costs below the no-storage costs. Therefore we start with a setup, where every house has a storage of $5kWh$. The costs for this setup are indeed higher than without any storage. But with optimizing the connections (and thereby neglecting some of the storages) and their capacities, we can find a configuration that performs better, see fig. **??**. In the figure we can also observe that changing the capacities of the storage now have a larger influence on the result. We find that varying connections and capacities at the same time is more effective than varying both separately.



(a) First connections, then capacities

(b) Connections and capacities at once

Figure 10: Optimization with 5 year amortization time frame for the two different schedules

8 Conclusion

We implemented a model, that is able to describe our simplified neighborhood setup and can handle external data sets for simulating real life consumption and pv production schedules. Additionally we kept the model general, by leaving different parameters open for later changes (e.g. more realistic parameters). Given the model configuration we performed an optimization using a genetic algorithm. Using this setup we were able to improve our results from two boundary cases, which might be considered in real world problems. Hence our algorithm gave us a better solution in terms of saving money for energy expenses. We were also able to minimize energy expenses, where both of the boundary cases were not feasible. However we don't know if the given solution is optimal, since the phase space is large and we have no possibility to calculate the optimal result analytically.

8.1 Outlook

The most important step would be setting cost parameters to more realistic values and also introduce of costs for the house - storage connections. Concerning the data set, we could start using the whole data set (1 year) instead using the two week time frame. For the genetic algorithm itself, more time can be spend on finding better parameters to retrieve better results (especially higher population size). All of these steps require more computation time, but can be realized by changing just a few lines of codes in our implementation.
Other optimization strategies might include changing the pv panel size on each house. However therefore we need to investigate the maximal pv panel size for each house, that we can not determine from the current data set. Further investigations might also include different amortization time frames or even try to find the best amortization time frame to choose.