Dengler *et al. Energy Informatics* 2022, **5**(Suppl 4):51
https://doi.org/10.1186/s42162-022-00243-2

Energy Informatics

METHODOLOGY Open Access

Simulation of a Cellular Energy System including hierarchies and neighborhoods

Gabriel Dengler*, Peter Bazan and Reinhard German

From Energy Informatics.Academy Conference 2022 (EI.A 2022)
Vejle, Denmark. 24-25 August 2022

*Correspondence:
gabriel.dengler@fau.de

Computer Science 7, Friedrich-Alexander-Universität Erlangen-Nürnberg, Martensstraße 3, 91058 Erlangen, Germany

Abstract

The massive use of small energy resources and storage units causes a transition from a traditionally centralized to a decentralized energy system. To structure and coordinate the emerging changes in the energy system, the concept of Energy Cells (ECs) was developed. Several ECs can be combined to form a hierarchically superordinate EC. These hierarchically superordinate ECs can in turn be combined and thus form a complex hierarchy. An EC encapsulates coherent parts of an energy system and can communicate as well as exchange energy with other ECs at a different or the same level. It follows the idea of local balance of energy provision and demand. A network of ECs forms a Cellular Energy System (CES). In this paper, we develop a concept for modeling and simulating a CES. We accomplish this by beginning with atomic components like consumers, producers, and storage units and aggregating them with Hierarchical Controllers (HCs). Such a hierarchically structured energy system is part of various proposals. However, we are able to add neighborhood relations by introducing Local Controllers (LCs). This is more realistic and also opens many degrees of freedom for control strategies in such a system. Following the recursive structure of the CES itself, we define recursive functions for visiting the CES architecture and realizing various control strategies. We evaluate our approach in a series of partially randomized scenarios, showing notable differences in the performance of the CES regarding different control strategies in a larger example. We also provide a theoretical analysis of the computational complexity of the suggested approach.

Keywords: Cellular approach, Energy cell, Local neighborhoods, Electric power system, Simulation

Introduction

Energy systems of the past could be characterized by the presence of a few central energy *producers* running mainly by the consumption of fossil resources and many *consumers*. The energy production of these producer types is scaleable and is very stable, as the amount of electricity feed-in is—at least in most cases—independent from environmental influences and can be adjusted to the current consumers' needs, but has a significant environmental impact. Renewable energies are much more

environmentally friendly but are not suitable to replace conventional producers without further adjustments, as the actual energy feed-in depends on factors outside our control, such as the season and the time of the day. Apart from that, many renewable energy producers like photovoltaic (PV) systems are often decentralized (e.g., integrated into a household), which makes it even more difficult to control them centrally. With the help of *storage* units, we can compensate for over- and underproduction of energy without the need for conventional energy forms. This has especially been made possible due to the advancements in storage technologies in recent years (Ma et al. 2021). However, since these are often placed distributed like renewable energy producers, we have to coordinate them in a new way.

In this context, the concept of the Cellular Energy System (CES) has arisen and has been researched by many organizations, especially in Germany by the Verband der Elektrotechnik (VDE) (2019). This system should help to create a scientific and technological view on the reliability of supply regarding electricity and other forms of energy. The basis is the Energy Cell (EC) which encapsulates geographically or structurally coherent parts of an energy system. An EC could be of any scale, e.g., a house, a neighborhood, or even an entire city. It has the target to fulfill all its demands by itself and it can provide or receive energy to/from other ECs. Furthermore, an EC can contain other ECs or be connected with neighbor ECs. This allows the coordination of an energy system with an increasing amount of distributed elements.

In Fig. 1 we see an illustration of a small EC system by definition of the VDE: this example consists of multiple ECs, which are either connected locally or hierarchically with each other and can transmit energy as well as data. For EC 1 it is additionally stated what a typical EC can look like: in this EC there are a producer, a consumer, and a storage component. All of these three components themselves could also be seen as ECs. They are connected to a controller which can exchange energy and information with EC 2. EC 2 contains three other ECs (or differently speaking, is hierarchically connected to three other ECs) and is connected with three neighbors on the same level.

In this paper, we propose a methodology for modeling CESs that have both a hierarchical structure and neighborhood relationships based on energy flows. We begin

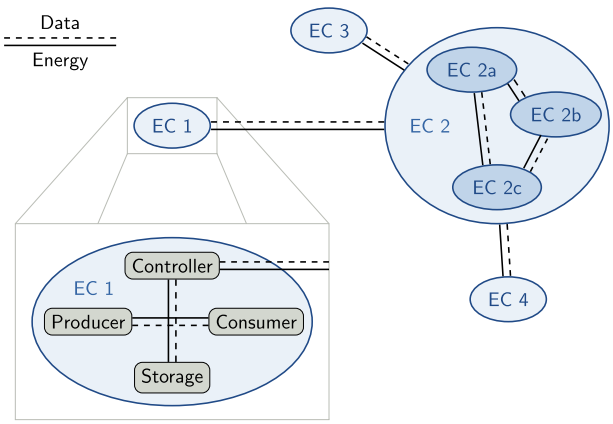

Fig. 1 A small CES based on the definition of VDE (2019)

with reviewing related work. Then we present how we model the previously presented concept of the VDE and what algorithms are used, following an analysis of the computational complexity. Finally, our approach is evaluated with examples of different sizes.

Related work

The German project *C/sells* is a research and implementation project for Smart Grids (SGs) and one of the main contributors in the field of the CES (Website 2022). Their goal is to realize the energy transition in a "cellular, participatory and diverse" manner (Haller et al. 2020). C/sells discusses multiple approaches to solutions, which may also be in competition with each other. This is intended to serve the various customer requirements in a variety of ways in order to find the best implementation in each individual case. Regardless of the exact interpretation of C/sells concepts in corresponding projects, C/sells is based on the idea of an EC similar to the definition of the VDE: an EC summarizes multiple coherent parts of an energy system. Connected ECs can exchange data and any form of energy. According to C/sells, a CES is either autonomous, which means that it can organize itself independently, or autarkical, referring to a system that does not need to interact with other components. They focus on the first type of CESs and want to integrate a CES into the existing power grid. In this context, C/sells introduces the *FlexPlattform* which is a central marketplace where the components can sell energy to or buy energy from. It can be used as a further way of exchanging energy between the individual participants in the system. Especially the influence of this FlexPlattform has been analyzed by different researchers. For example, Klempp et al. (2020) discuss the effects of strategical bidding and the resulting problems as well as possible solutions in this market, while Bauknecht et al. (2021) did some investigations of legal regulations.

Outside the cellular approach, there are various projects to realize decentralization in energy systems. One of them is Local Energy Management (LEM), where the basic idea is similar to the CES: neighborhoods try to meet their own demands directly but still have the possibility to make requests to the outside. Related work mainly covers the realization of some thinkable scenarios: Eid et al. (2016) examined issues with market integration of local energy systems with European regulations. Additionally, Feng et al. (2020) analyzed *local energy communities* as a possibility to implement LEM by utilizing a coalitional game model in their research.

Another popular method of coordinating energy in neighborhoods is by utilizing *energy hubs* which can receive, provide, as well as store different types of energy (e.g. electricity, heat, or water) and, in particular, convert between those types if physically possible and implemented into this hub. This concept shows many similarities with the idea of ECs and is used in research to address the problem of managing energy in neighborhoods: Walker et al. (2017) reviewed the current state of the art back in 2017 in managing energy via energy hubs in energy positive neighborhoods and considered energy hubs as a widely researched and reasonable concept. Concrete modeling and testing of energy hubs at a neighborhood scale were done by Kristina et al. investigating optimal configurations and building layouts for a village in the mountains of Switzerland containing 29 buildings, showing an overall decrease in energy demand peaks and a

reduction of the overall consumption (Orehounig et al. 2015). A variation of this concept has been done by Perea et al. (2021) by extending the hub concept by introducing an urban cellular architecture that facilitates sector and spatial coupling, which comes even closer to the EC concept.

In the context of decentralized energy management there is an increasing trend in investigating the capability of introducing *Peer-to-Peer (P2P) energy trading* in energy systems. It refers to the direct energy exchange between two components in a system. These components are typically consumers or prosumers (which denotes participants that produce and consume energy), e.g. households. They can directly communicate with other participants in the system. In doing so, they usually make use of a market: the participants in the system can sell energy to or buy energy from other components. Those trading procedures can be combined with various technologies. One of these is blockchain as it allows to securely record all tradings without the need of one central instance (Cali and Fifield 2019). Each component tries to minimize the resulting costs but ensures that it still satisfies its consumption. One implementation from Monroe et al. implements an agent-based model following this idea (Monroe et al. 2020): each household acts like an individual agent with its own decision-making. Every agent can utilize any type of control strategy including machine learning techniques or forecasting algorithms. This approach was tested in a case study with 18 households in Western Australia utilizing a blockchain-enabled trading platform. A different approach is taken by Tushar et al. (2019) by first summarizing different types of P2P energy trading topologies, e.g., trading within a microgrid or between multiple microgrids. After that, they apply game-theoretical methods to answer the question of how to encourage prosumers to participate in P2P energy trading, so that it becomes sustainable. Additional research was done by Shrestha et al. (2019), reviewing a broad range of methods for P2P energy trading models, including game theory, distributed algorithms, or a multi-market driven approach. Their paper discusses the challenges and opportunities of using P2P energy trading in the energy system of Nepal. To sum up, the most related work regarding P2P energy trading takes a multifaceted view on the topic, also beyond technical aspects. However, our proposed CES simulation combines hierarchical and local decision-making by utilizing different controller types, which is often missing in P2P energy trading models.

Modeling a Cellular Energy System

In the "Introduction", we took a look at the definition of a CES according to the VDE. However, since it is mainly described as a concept, it is not very well suited to be used directly in a simulation framework. Therefore, we precise the concept of a CES and give a more formal definition of an EC. In our approach, an EC is either

- an *atomic* component, such as a *consumer* (taking predetermined energy from the system), a *producer* (adding predetermined energy to the system), or a *storage* unit (possibility to store energy),
- a Hierarchical Controller (HC) with one or more subordinate ECs (this models energy management with a central coordinator) or

- a Local Controller (LC) with one subordinate EC. An LC is furthermore connected and can interact with an arbitrary amount of neighbor ECs (this represents decentralized management).

As the resulting structure is tree-like, every non-root component has one overlying component as a parent, while each controller aggregates one or more subordinate components. A visualization of an example CES is presented in Fig. 2. If we consider an LC or HC with the associated child elements as a unit, we obtain an EC similar to the definition of the VDE. The limitation for an LC to one child is by purpose since hierarchical controlling should be done by the HC. This enables us to separate the control from the neighbors and the children into two components, which increases the modularity of the system. Furthermore, it is also possible to omit the LC completely and use only an HC for certain ECs.

Energy flows

Before we start with the actual system, we need to introduce the concept of an *energy flow*. An energy flow describes how much energy flows of a certain type (e.g., electricity, gas, or heat) from one part of the system to another. More precisely, each EC A in the system can transmit energy with a power $P(t)$ (usual unit kW) at a time t to any other connected EC B. For simplicity, we say that the flows can be conducted through controllers. We safely assume that for all t the value of $P(t) \geq 0$. To model an energy flow in the opposite direction (B to A) we simply introduce an additional energy flow. Therefore, the complete transmitted energy (usual unit kWh) from the start of the simulation ($t = 0$) until a time t_end between those participants can be calculated as $E(t\_\text{end}) = \int_0^{t_\text{end}} P(t)dt$.

This approach is a vast simplification in comparison to other simulation models (such as the *complex alternating current calculation*). But more precise models are not necessary in our case and would not suit the main goals of this simulation: our framework

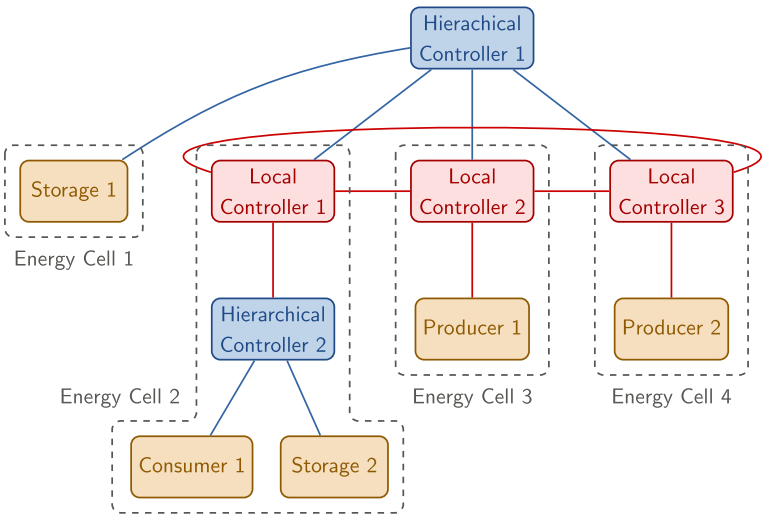

Fig. 2 Visualization of a hierarchical CES and correlation to the definition of the VDE

is mainly designed to develop control strategies within the system, for example deciding how to resolve an energy excess when there are multiple possibilities. Energy flows help to understand the impact of each component and identify parts that might turn out to be weak spots regarding the overall system's performance. Furthermore, a simulation using energy flows is less computationally expensive compared to more precise calculation models and can be scaled more easily to large systems. We can use the simulation's results to find out possible weak spots in our system which we can investigate with more accurate models.

In the scope of energy flows, we introduce a naming pattern to describe and distinguish multiple flows. This naming pattern is only valid *locally*, which means that the flows are named from the point of view of one component. Every flow is classified as follows:

$$[type]\_[source]\_[drain]\_[meaning] \tag{1}$$

The *type* describes the energy carrier of the flow, e.g., electricity, gas, water, or (in our case) a generic form '*x*'. *Source* and *drain* describe the direction of the flow, as mentioned, from the point of view of the current component. Possible values are *ext* referring to the overlying component, *int* to the component itself, and *ec* to any subordinate EC. Since every energy flow must not be negative by definition, we usually use pairs of energy flows between two components to express flows in both directions. The possible *meanings* depend on the application, and do not necessarily describe an actual existing flow: it can also be used to exchange data and is then denoted as *information flow*. What they are in our case is described in the following section.

Interfaces

Each component in the CES has a number of different flows to its overlying component. From their perspective, these flows can be described as *x_ext_int_** for flows from the overlying component and *x_int_ext_** in the opposite direction ('*' stands for a certain meaning, see below). HCs are furthermore connected to an arbitrary number of subordinate components and share flows with them, while LCs have one subordinate component and an arbitrary number of neighbors (see Fig. 3).

Based on this information we can define the following meanings of energy flows:

- *f*: it stands for *flow* and resembles the actual energy flow. It describes the amount of energy that flows currently in or out of an EC. Similar to Kirchhoff's current law, the sums of the incoming and the outgoing *f* flows of a controller have to be the same.
- *r*: this information flow stands for *request*. It is set by the overlying component or a neighbor EC and gives information about what the optimal adjustments are for *f* to answer. More precisely, a *r* flow describes the desired total energy flow outgoing from or incoming to an EC, so in the optimal case, it holds, e.g., that $x\_int\_ext\_f = x\_int\_ext\_r$ and $x\_ext\_int\_f = x\_ext\_int\_r$.
- *z* and *g*: that information flows stand for *zero* and *grant*. The *z* flow describes the flow *f* an EC emits after the initial call of *calc_z*() (see below). In *g* it is stated what an EC allows additionally after some (maybe multiple) calls of *calc_f*().

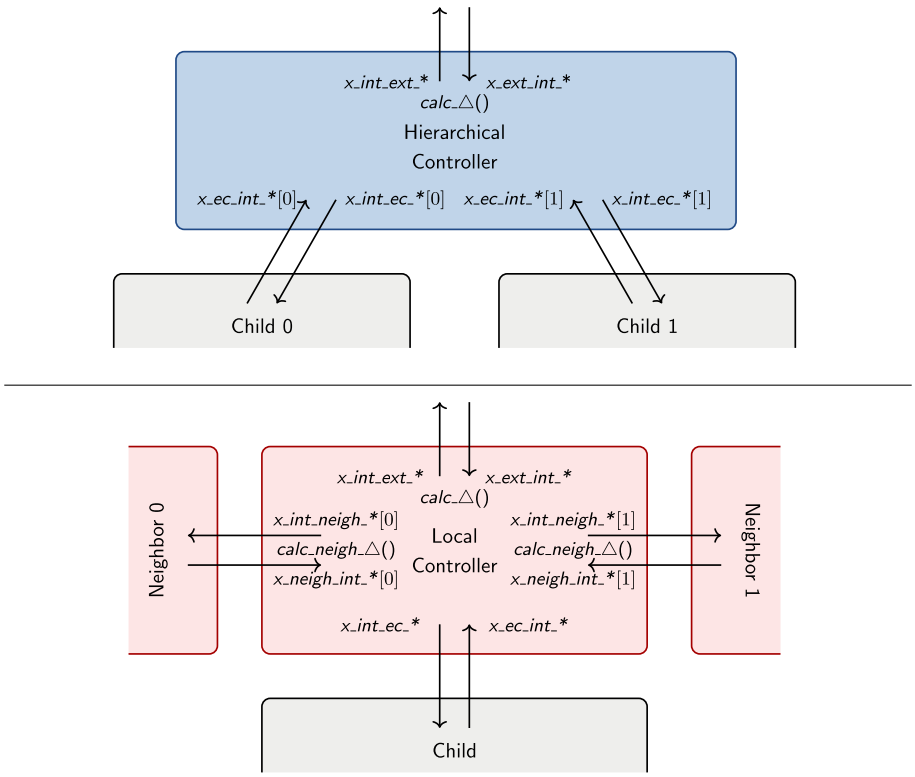

Fig. 3 Interfaces of an HC (above) and an LC (below), $* \in \{f, r, z, g\}$, $\Delta \in \{z, f\}$

Besides the flow meanings stated above, it is also possible to exchange further information to the parent EC via information flows: one example is a *r_min* as well as a *r_max* flow meaning which contains an estimation of what is the minimum (unequal to 0) and maximum amount of energy an EC can offer to its parent. Formally it should hold for all flows unequal to 0:

$$x\_int\_ext\_r\_min \leq x\_int\_ext\_f \leq x\_int\_ext\_r\_max \tag{2a}$$

$$\text{and} \qquad x\_ext\_int\_r\_min \leq x\_ext\_int\_f \leq x\_ext\_int\_r\_\text{max} \tag{2b}$$

Those limitations lie in the modeling of storage components: for each battery there exist restrictions for the outgoing and incoming flows. A controller can use this information to compute requests optimally based on the current strategy. Another parameter that could be exchanged is the current State of Charge (SoC) of the attached storage units.

These flows can now be used for the definition of some functions every component offers to overlying elements (or neighbors in the case of the LC). The goal of an EC is to fulfill its own needs but is furthermore able to exchange energy with connected ECs. These two properties are modeled with two functions:

- *calc_z*(): this function calculates an EC's *initial* flow and is invoked once per EC and calculation step. It determines which energy surplus an EC produces or which demand an EC has if the EC tries to fulfill its requirements on its own. When the

EC is not able to balance its own demand and production, it returns a flow that the overlying EC has to handle.

- *calc_f*(): this function receives a request as information from the *request* flows and adjusts its flows to determine its *grant* for this request. It is called multiple times per system calculation.

With the information gained from the subordinate cells, a control algorithm can now request the current state of the subordinate ECs and adjust the energy distribution based on this information.

Require: *(Initially called once per time step)*
1: **procedure** *calc_z*()
2: **for** every child EC *child_ec*$[i]$ **do**
3: *child_ec*$[i]$.*calc_z*()
4: Calculate *balance* according to Equation 3
5: Set unresolved *balance* to *x_int_ext_f* and *x_ext_int_f*
6: Fix $x\_int\_ext\_r = x\_ext\_int\_r = 0$
7: *calc_f*()

Algorithm 1 HC routine when requesting the initial flows

Managing energy

Previously we described the proposed modeling of a CES. However, we did not specify how to implement the described functions and how to calculate possible flow constellations. Therefore we design a calculation procedure that utilizes the recursive structure of the CES. Another goal for the creation of the general principle is that we are able to realize different control strategies without major changes.

Discrete time step simulation

The flows determined via the *calc_f*() and *calc_z*() functions calculate the flows for a given time step. However, we are interested in the system's behavior over time. The main idea in the CES simulation is to recalculate the flows in fixed time steps. The time between any recalculations should be chosen such that the overall computation time of the simulation is not too expensive, but the calculations are accurate enough. The recalculations are managed by a sequencer that initializes the flow balance calculation from the top EC iteratively. At first, it advances one time step in the simulation by updating the SoC of the batteries and statistics. After that, the flows will be reset and then recalculated on the new status of the system by invoking *calc_z*() at the top EC.

Hierarchical Controller routine

For an HC, we have to implement two functions *calc_z*() and *calc_f*(). Both maintain a variable *balance* defined as:

$$balance = \sum(incoming\,flows) - \sum(outgoing\,flows) \tag{3}$$

At the end of a calculation step, the *balance* has to be 0 to guarantee that there is no energy excess or demand.

Since *calc_z*() (see Algorithm 1) is the first function called per component during a time step, it is responsible for preparing its children by calling *calc_z*(). The resulting imbalance is redirected to the caller to ensure that $balance = 0$. However, the current flows could be optimized (e.g., the energy should be managed as locally as possible), whereas *calc_z*() calls *calc_f*() requesting no incoming or outgoing flow. If this can be achieved, the EC can work fully self-sufficient.

Require: Desired requests *x_int_ext_r* and *x_ext_int_r*
1: **procedure** *calc_f*()
2: ▷ Consider the flows *x_int_ext_r* and *x_ext_int_r*
3: ▷ Prepare how to send requests to children based on the strategy
4: **for** every child EC $child\_ec[i]$ **do**
5: Send requests *x_int_ec_r* and *x_ec_int_r* to $child\_ec[i]$
6: ▷ Order and magnitude of request calls are determined by the strategy
7: $child\_ec[i].calc\_f()$
8: Adjust *balance* according to Equation 3
9: Set unresolved *balance* to *x_int_ext_f* and *x_ext_int_f*

Algorithm 2 HC routine answering requests

The function *calc_f*() (see Algorithm 2) starts with a possible constellation of incoming and outgoing flows fulfilling that $balance = 0$. However, they are probably not optimal as the energy may be managed more locally, so the function tries to improve them. It considers the initial request and tries to fulfill it by requesting the children. The order and magnitude are determined by the strategy. The effects on the *balance* are then redirected to the caller.

Figure 4 illustrates the idea behind the two functions when applied to a small system consisting of an HC and a consumer, a producer, and a storage unit attached. We are interested in how the system behaves when it tries to be as self-sufficient as possible. Therefore we start with the calculation of *calc_z*() at the top HC. At first, the controller requests the original flow from each component (figure above). Then, the consumer and producer return their production or demand. The imbalance can be temporarily redirected to the caller of the HC. After that, the controller asks its children to adjust their flow to a new value. Only the battery in this example can store the surplus (figure below).

Local Controller routine

Similar to the HC, we can realize the *calc_z*() and *calc_neigh_z*() functions as shown in Algorithm 3 and Algorithm 4. For both, we have to prepare the only children, and for *calc_z*() we have to additionally take care of the neighbors. The imbalance in *calc_neigh_z*(), however, is redirected to the overlying EC and not to the neighbor, as flows between neighbors are only determined via *calc_neigh_f*(). In any case, we have to ensure that an LC is only prepared once per time step, as the initialization can both be initiated by a parent or neighbor.

For *calc_f*() (see Algorithm 5) it is our goal to handle the request at first only by the usage of the subordinate EC, as we want to handle the energy as local as possible. Only then the neighbors are considered, in a similar manner as for the HC. The *calc_neigh_f*() function (see Algorithm 6) first checks, if *calc_f*() has already been called from the top. If not, the flows have not been sent to the overlying EC. Then, if applicable, we can redirect

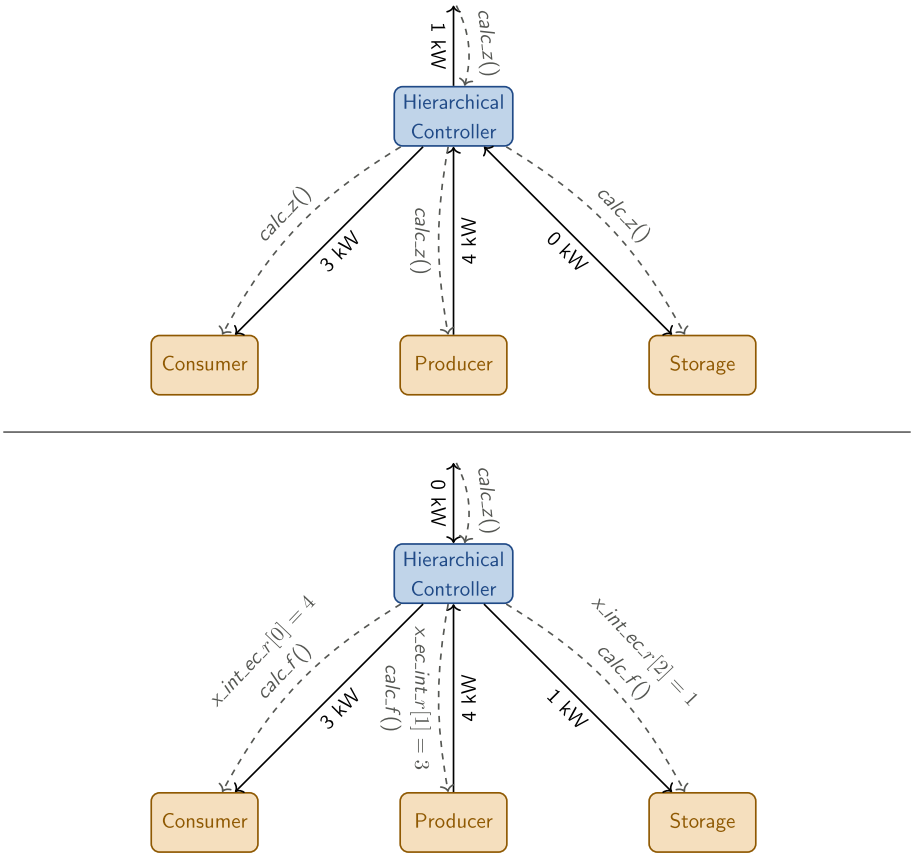

Fig. 4 Example describing the idea behind the functions *calc_z*() and *calc_f*()

parts of these flows to the neighbor. Afterwards, the controller tries to resolve the initial request with the child EC.

Minimum/Maximum request estimation

As mentioned in the definition of the Interfaces, we can determine flow estimators with the meaning *r_min* as well as *r_max* describing a minimum and maximum energy flow an EC can offer. Note that while these values are perhaps not exact, they are still very helpful for estimating the possible capacity of an EC which can be used for control strategies.

For consumers and producers, the corresponding flows are fixed, which means that sending special requests would not change the flows. Therefore, the calculation of the *r_min* and *r_max* flows can be omitted here. For our storage unit models, we set minimum and maximum flows in the incoming as well as outgoing directions. However, also more complex battery models can be implemented which consider external factors such as the working temperature.

The capacity of controllers is dependent on their subordinate ECs. For the *r_min* flow, we only have to be aware of the subordinate EC with the lowest corresponding *r_min* information: we can redirect all energy to or from this EC. When we calculate the maximum possible flow, we consider the left buffer (e.g. difference between the *r_max*

estimator and the current flow) for each subordinate EC. The sum of those buffers indicates how much power we can redirect additionally into the current EC. Therefore this sum can be added to the existing incoming or outgoing flow to obtain the maximum request. For LCs we only consider the subordinate EC but not the neighbors, as we do not see neighbors as part of the current EC.

Strategies

Until now we did not specify how exactly the requests are sent to the subordinate ECs (in case of the HC) or the neighbors (in case of the LC). Depending on the chosen strategy the behavior of the system can differ vastly. In this paper, we present three simple strategies with different goals:

Require: *(Initially called once per time step)*
1: **procedure** *calc_z*()
2: **if not** origin from *child_ec* calculated **then**
3: *child_ec.calc_z*()
4: **for** every neighbor EC *neigh_ec*$[i]$ **do**
5: **if not** origin from *neigh_ec*$[i]$ calculated **then**
6: *neigh_ec*$[i]$.*calc_neigh_z*()
7: Calculate *balance* according to Equation 3
8: Set unresolved *balance* to *x_int_ext_f* and *x_ext_int_f*
9: Fix $x\_int\_ext\_r = x\_ext\_int\_r = 0$
10: *calc_f*()

Algorithm 3 LC routine when requesting the initial flows from top

Require: *(Used to be prepared by neighbors)*
1: **procedure** *calc_neigh_z*()
2: **if not** origin from *child_ec* calculated **then**
3: *child_ec.calc_z*()
4: Calculate *balance* according to Equation 3
5: ▷ **No** *calc_neigh_f*(), as related *z* flow is known to be 0
6: Set unresolved *balance* to *x_int_ext_f* and *x_ext_int_f*

Algorithm 4 LC routine when requesting the initial flows from a neighbor

Greedy One simple strategy to determine the requests is the *Greedy* strategy: after calculating the initial flows from and to the attached ECs, the controller tries to compensate for the complete current energy surplus or shortage using only one *calc_f*() call by sending a corresponding request to a child or neighbor. Ideally, the current EC can directly fulfill the current request. Otherwise, the caller has to ask another EC or let the initial request remain unsatisfied.

Equal request As introduced earlier, we calculate flows of the type *r_max*. Those flows give an estimation of how capable an EC is to answer a request. For example, a high value for this estimation indicates that this EC consists of many components and can therefore also provide a large amount of energy. We use this information to distribute the request equally to all neighbors but weighted according to the *r_max* values. There-

fore, we ask each candidate once, evaluate its answer, rearrange the new target request and go on with the next EC. We further divided this strategy into two variants:

- In *variant 1*, we split the difference between the initial request and the actual flow. For every child or neighbor, we use as weighting, which buffer is between the current flow and the corresponding *r_max* flow. The achieved result for the next EC is then added to the existing flows to obtain the request flows.
- In *variant 2*, we redistribute the total request. Since we reorganize the complete request, we do not consider the buffer, but instead, the complete possible requests given by the *r_max* flows. The achieved result for the next EC is then considered as the new request.

Equal SoC In the context of this strategy, we look at some observations from Abgottspon et al. (2018). They analyze how to control a Battery Energy Storage System (BESS) and observe many parameters. One of them is the *heterogeneity*, describing how different the storage components regarding parameters like size or charging/discharging speed are. One result of the paper is that a homogeneous BESS can usually perform better than a heterogeneous one. The goal of our approach is to keep the SoC of the batteries on a similar level. Therefore we use variant 2 from the *Equal Request* strategy. Variant 1 cannot be easily adapted, as it requires knowing what part of the maximum possible flow is already used. We use the entire free storage of all attached child batteries for charging and saved energy for discharging as weighting. Due to the modular structure of the framework, further strategies are possible, also ones that consider and/or control not only the current controller but also a complete subtree. Apart from it, possible thinkable is to establish a market in which the components participate, e.g. by setting a price a controller has to pay for demanding a certain request.

Require: Desired requests *x_int_ext_r* and *x_ext_int_r*
1: **procedure** *calc_f*()
2: ▷ Consider the flows *x_int_ext_r* and *x_ext_int_r*
3: Send requests *x_int_ec_r* and *x_ec_int_r* to *child_ec*
4: ▷ Choose requests such that ideally the initial request is fulfilled
5: *child_ec.calc_f*()
6: Adjust *balance* according to Equation 3
7: Set unresolved *balance* to *x_int_ext_f* and *x_ext_int_f*
8: **for** every neighbor EC *neigh_ec*$[i]$ **do**
9: Send requests *x_int_ec_r* and *x_ec_int_r* to *neigh_ec*$[i]$
10: ▷ Order and magnitude of request calls are determined by the strategy
11: *neigh_ec*$[i]$.*calc_neigh_f*()
12: Adjust *balance* according to Equation 3
13: Set unresolved *balance* to *x_int_ext_f* and *x_ext_int_f*

Algorithm 5 LC routine answering requests from the top

Require: Desired requests *x_int_neigh_r*$[i]$ and *x_neigh_int_r*$[i]$

procedure *calc_neigh_f*()
  Determine which index i corresponds to the caller
  ▷ Consider the flows *x_int_neigh_r*$[i]$ and *x_neigh_int_r*$[i]$
  if not *calc_z*() called for this controller **then**
    Try to redirect *x_int_ext_f* and *x_ext_int_f*
    to *x_int_neigh_f*$[i]$ and *x_neigh_int_f*$[i]$
  Send requests *x_int_ec_r* and *x_ec_int_r* to *child_ec*
  ▷ Choose requests so that that ideally the initial request is fulfilled
  child_ec.calc_f()
  Calculate *balance* according to Equation 3
  Set unresolved *balance* to *x_int_neigh_f*$[i]$ and *x_neigh_int_f*$[i]$
  ▷ **No** further requests to neighbors
  ▷ (But adjustment of algorithm would be possible if wanted)

Algorithm 6 LC routine answering requests from a neighbor

Computational complexity

Since the general relatively complex procedure consists of various cascade-like function calls, it is hard to determine the simulation's complexity by intuition. However, an estimation of the computational effort can be really helpful to better assess the simulation.

We consider a CES as a tree as presented in Fig. 2 and define h as the height of the tree ($h = 0$ refers to a tree with one node). Regarding the tree's height, we consider each pair of an LC and HC as a node. Each HC in the tree has d children and each LC has k neighbors on the same height. The lowest layer consists of producers, consumers, or storage units which are treated equally here.

At first, we analyze the number of function calls for a strict hierarchical system. Let $f(h)$ be the number of calculation steps to determine the system's flows for a tree of height h with d defined as above. We consider each function call as one calculation step. The runtime is also dependent on the overhead occurring at each function body, which is for example affected by the number of children at this node, but for general analysis, we consider this estimation as sufficient. Furthermore, we specify $f(0) = 1$, suggesting that the flow calculation complexity for consumers, producers, and storage components is constant. Every calculation step of *calc_z*() and *calc_f*() leads to at maximum two function calls (again *calc_z*() and *calc_f*()) for each child. Therefore we can conduct for $h > 0$:

$$f(h) = 2 \cdot d \cdot f(h-1) + 1 \tag{4}$$

The 1 indicates that every function call has a constant overhead. For further simplification, we erase the 1 as summand. In this case, this simplification does not lead to falsification in terms of complexity. A detailed explanation is omitted here, but, in short, we would only observe a small multiplicative offset. We follow:

$$f(h) = 2 \cdot d \cdot f(h-1) \tag{5}$$

As one can see, this equation can be simplified to $f(h) = (2d)^h$. We now extend the given approach to a system with LCs. Again, the tree has as root one HC, following the LCs in each layer with an HC attached to each of them. $f_{\text{hierarchical}}(h)$ is the number of

calculation steps remaining when starting with an HC at height h. $f_{\text{neigh}}(h)$ describes the number of calculation steps remaining when starting with an LC which has been requested from the top, while $f_{\text{neigh\_side}}(h)$ refers to the number of calculation steps when requested from a neighbor. Then we can derive

$$f_{\text{hierarchical}}(h) = 2 \cdot d \cdot f_{\text{neigh}}(h-1) + 1, \tag{6a}$$

$$f_{\text{neigh}}(h) = 2 \cdot f_{\text{hierarchical}}(h) + 2 \cdot k \cdot f_{\text{neigh\_side}}(h) + 1 \tag{6b}$$

$$\text{and} \qquad f_{\text{neigh\_side}}(h) = 2 \cdot f_{\text{hierarchical}}(h) + 1 \tag{6c}$$

with $f_{\text{hierarchical}}(0) = 1$. The difference between the runtime for a request from an LC when requesting from the top or a neighbor lies in the fact that in the second case an LC only asks the subordinate HC to answer the given request. This is a strong overestimation of the actual computational effort since this calculation assumes that during every function call *calc_z*() and *calc_f*() have to be completely recalculated. However, since *calc_z*() on each component has to be called only once per time step, the cascade may be often interrupted earlier.

Deleting the summand 1 and doing some simplifications leads to:

$$\begin{aligned} f_{\text{hierarchical}}(h) &= 2 \cdot d \cdot f_{\text{neigh}}(h-1) \\ &= 2 \cdot d \cdot [2 \cdot f_{\text{hierarchical}}(h-1) + 2 \cdot k \cdot f_{\text{neigh\_side}}(h-1)] \\ &= 2 \cdot d \cdot [2 \cdot f_{\text{hierarchical}}(h-1) + 2 \cdot k \cdot (2 \cdot f_{\text{hierarchical}}(h-1))] \\ &= (4d + 8dk) \cdot f_{\text{hierarchical}}(h-1) = (4d + 8dk)^h \end{aligned} \tag{7}$$

If we consider a complete tree with n nodes, the height of the tree will be logarithmic to n, formally $h \approx c \cdot \log_d n$ with c being a constant. For the hierarchical system we can derive

$$f(h) = (2d)^h \approx (2d)^{c \cdot \log_d n} = \left[(2d)^{\log_d n}\right]^c = \left[n^{\log_d (2d)}\right]^c = n^{c \cdot (1 + \log_d (2))}. \tag{8}$$

Therefore, we can expect a polynomial runtime with the number of children per node having a minor influence on the exponent. E.g., when choosing $d = 2$ and $c = 1$, the resulting runtime lies in $\mathcal{O}(n^2)$.

Applying these calculations to the second formula we derive

$$\begin{aligned} f_{\text{hierarchical}}(h) &= (4d + 8dk)^h \approx (4d + 8dk)^{c \cdot \log_d n} \\ &= \left[n^{\log_d (d \cdot (4 + 8k))}\right]^c = n^{c \cdot (1 + \log_d (4 + 8k))}. \end{aligned} \tag{9}$$

For simplification, we set this time $d = k = 2$ and $c = 1$ to follow:

$$n^{1 + \log_2 (4 + 8 \cdot 2)} = n^{1 + \log_2 20} \approx n^{5.32} \tag{10}$$

This time we obtain a larger exponent, but the result is still a polynomial. A higher number of neighbors per LC tends towards higher complexity.

Regardless of the details in the runtime analysis, we can conclude that the system's runtime in normal use cases is polynomial and not exponential in the number of

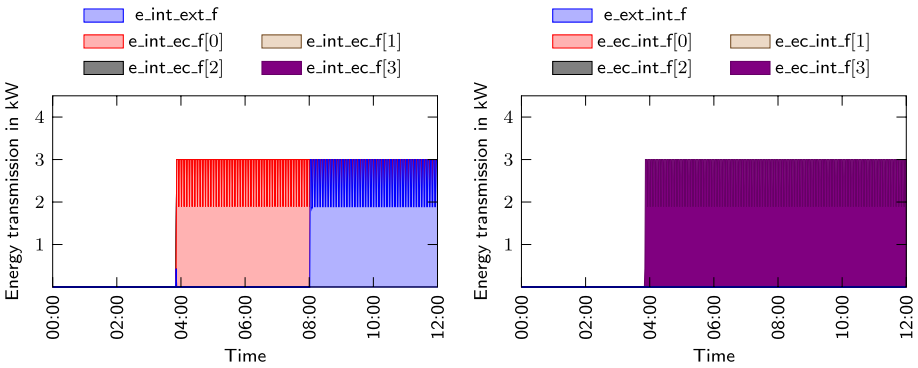

Fig. 5 Outgoing (left) and incoming (right) energy flows for HC 1 in the example from Fig. 2 when using the *Greedy* strategy, ec[0] ≙ Storage 1, ec[1] ≙ LC 1, ec[2] ≙ LC 2, ec[3] ≙ LC 3

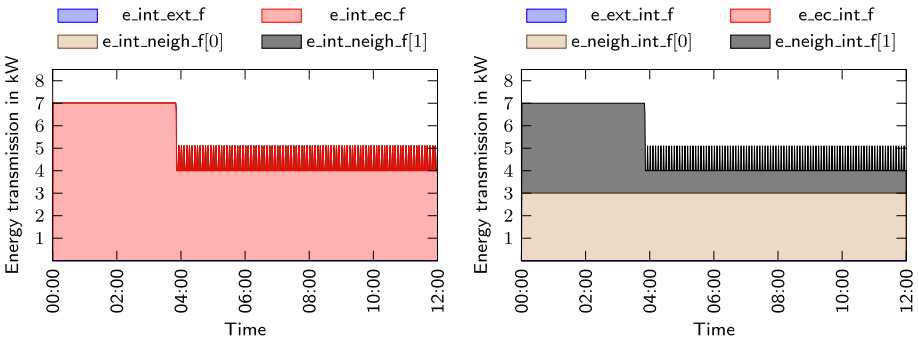

Fig. 6 Outgoing (left) and incoming (right) energy flows for LC 1 in the example from Fig. 2 when using the *Greedy* strategy, neigh[0] ≙ LC 2, neigh[1] ≙ LC 3

elements of the tree. This means that the simulation is capable of handling larger systems and is not limited to very small examples, although the algorithm design might imply it differently.

Evaluation

Testing example

For demonstration, we select the example CES presented in Fig. 2. Furthermore, we define the following properties: Both controller types follow the *Greedy* strategy. The HC 1 first asks Storage 1 and then the LCs 1 to 3, while the handling of the LC always begins with the LC with a smaller digit. Consumer 1 needs a constant power of 4 kW, while both producers provide a power of 3 kW and 4 kW to the system. All storage components have a capacity of 10 kWh, are initially fully discharged, support charging and discharging between 1 and 3 kW, leak some energy when charging/discharging, and constantly lose energy due to self-discharging. The simulated time frame is 12 h long and a time delta between two time steps amounts to 1 min, yielding a total of $12 \cdot 60 = 720$ time steps with 721 points in time.

Figures 5 and 6 demonstrate the imported and exported energy of HC 1 and LC 1. As one can see, the system's behavior can be divided into three phases:

1. HC 1 begins with requesting the initial flows of its children. During this, LC 1 receives from its child the information that it has a power shortage of 4 kW. Therefore, LC 1 asks LC 2 to compensate for this, but it can only provide 3 kW. The remaining 1 kW power shortage is delivered by LC 3. However, LC 3 still has to deal with an energy surplus which will be redirected to LC 1.
2. After about 4 h, Storage 2 is fully charged. Then the energy surplus from LC 3 has to be exported to HC 1, which can conduct this energy to Storage 1 by sending a corresponding request call.
3. After that, the overproduction in the system cannot be compensated anymore by the storage components. The excess must be exported by HC 1.

The flickering effects beginning at about 4 h are caused by the constant energy loss of the storage components. When the battery has been discharged to a level where it is again able to receive a power of at least 1 kW, this battery will be charged again, resulting in the observed anomalies.

In short, our proposed algorithms show the wanted behavior: The energy surplus is managed as locally as possible because Storage 2 is loaded first and solely over neighborhood connections. Only afterwards HC 1 has to be utilized.

Comparison of measurements

To better assess the capability of the presented procedures for larger systems, we have to aggregate information about the calculated flows. Therefore we defined the following measurements to evaluate the overall performance of our chosen strategies:

(a) Power that is flowing over an HC, formally

$$x\_ext\_int\_f + \sum_{i=0}^{n-1} x\_ec\_int\_f[i] \tag{11a}$$

$$\text{or} \qquad x\_int\_ext\_f + \sum_{i=0}^{n-1} x\_int\_ec\_f[i]. \tag{11b}$$

This measurement informs us about how much power can be managed locally instead of being handled by an HC. A low value, especially for higher placed HCs in the tree, indicates that much energy can be managed locally without the need for higher placed controllers. If the LC can share energy flows over its neighborhood connections, the amount of energy flowing over an HC should on average be lowered.

(b) Percentage of outgoing and incoming flows at each LC being distributed hierarchically ($x\_ext\_int\_f$ and $x\_int\_ext\_f$) or locally ($x\_neigh\_int\_f$ and $x\_int\_neigh\_f$) in each direction to serve the needs of the subordinate EC, formally for n neighbors:

$$\frac{\sum_{i=0}^{n-1} x\_neigh\_int\_f[i]}{x\_ext\_int\_f + \sum_{i=0}^{n-1} x\_neigh\_int\_f[i]} \tag{12a}$$

$$\text{and} \qquad \frac{\sum_{i=0}^{n-1} x\_int\_neigh\_f[i]}{x\_int\_ext\_f + \sum_{i=0}^{n-1} x\_int\_neigh\_f[i]} \tag{12b}$$

This measurement is only applicable to systems with LCs. In the best-case scenario, these values are 1, which means that all flows through that controller are managed locally, in contrast, a value of 0 indicates that the neighborhood connections could not be utilized. However, these values usually do not reach 1 as imported or exported energy for this LC must, in any case, be conducted over the corresponding HC and cannot be managed locally.

(c) Unresolved energy flows ($x\_ext\_int\_f$ and $x\_int\_ext\_f$) at the top HC. This describes if all needs of the CES could be fulfilled and gives us information on whether the presented strategies improve or even worse the system's stability. If the CES does not demand any energy export or import, both flows are 0. In general, we want to achieve the smallest possible value.

These measurements are depicted for each calculation step and then averaged for all time steps and controllers. There is no weighting in regards to the importance of a controller (e.g., the root controller is more important than one near the leaves), but for a general estimation, we consider this as accurate enough.

Large-scale example

We evaluate the efficiency of our algorithms with the help of a randomly created larger CES. We analyze the difference between a strict hierarchical system and a mixed one including LC as well as the different presented strategies. The CES is constructed by setting up a tree to define the hierarchical structure and afterward adding neighborhood connections if LCs are included. We use a tree with a height of 6, with ≈ 3 attached controllers per HC and ≈ 3 neighbors per LC. Furthermore, at each HC there is on average 1 producer, consumer, or storage component for inner nodes and 2 for leaf nodes. For the particular example, we, therefore, obtain 75 HCs, 73 LCs, 47 consumers, 28 producers, and 32 storage units. The time-varying impact of the producers and consumers is modeled with a sinus-shaped function with two peaks for consumers (forenoon and afternoon, maximum consumption 3 kW, representing a typical household) and one peak for a producer (midday, maximum production 6 kW, representing a PV system). The average capacity of a battery is ≈ 20 kWh, in total they can hold $659.58 \approx 20 \cdot 32$ kWh. For each battery, we model again some losses during operation, and each of them supports charging and discharging between 1 and 5 kW. Initially, they have a random SoC. As before, one time step amounts to one minute, resulting in overall $24 \cdot 60 = 1440$ time steps.

By having a look at Figs. 7 and 8, we see that the system is designed so that it has to import energy in the forenoon due to the low SoC, but must export energy during midday. Especially due to the limitation of the maximum charging/discharging power of the storage units we observe an export/import at the top HC even if the SoC limits of the batteries have not been reached. Here, the distributions of the requests can have a significant impact, as they, e.g., determine how the batteries will be charged and therefore how fast and efficient a swarm of batteries can be discharged.

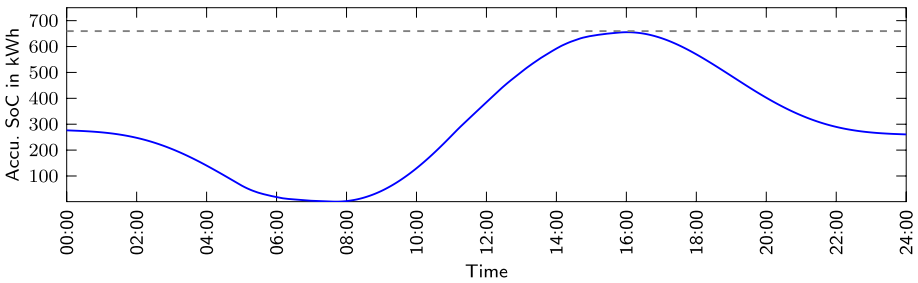

Fig. 7 Accumulative SoC and maximum capacity of all storage components at the larger example without LCs when using the *Greedy* strategy

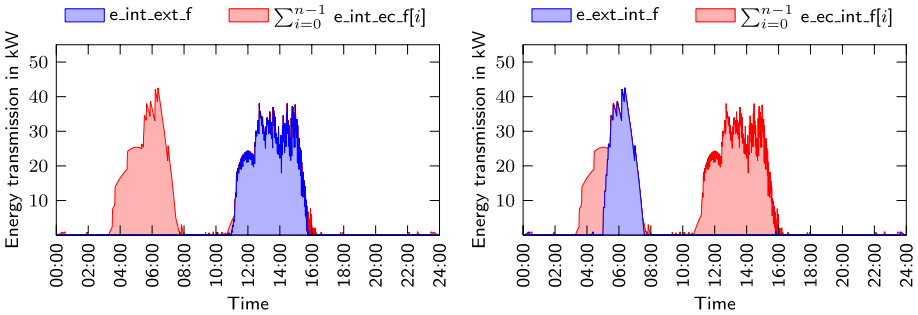

Fig. 8 Outgoing (left) and incoming (right) energy flows for the topmost HC at the larger example without LCs when using the *Greedy* strategy

Table 1 Evaluation of the presented metrics (in short: (a) Power over HC in kW, (b) Percentage resolved over LC, (c) Unresolved power at top HC in kW) to the first large-scale example when using either a strict hierarchical CES or one with LCs by applying different control strategies

	Strategy	(a)	(b)		(c)	
			Import	Export	Import	Export
Only HCs	Greedy	5.273	/	/	2.894	4.858
	Equal Request v1	5.204	/	/	2.902	4.885
	Equal Request v2	5.231	/	/	3.596	6.267
	Equal SoC	5.268	/	/	3.241	6.435
HCs and LCs	Greedy	3.477	0.576	0.584	3.095	5.098
	Equal Request v1	3.430	0.580	0.545	3.005	4.955
	Equal Request v2	5.457	0.582	0.624	3.181	5.556
	Equal SoC	6.038	0.608	0.617	3.135	5.398

In Table 1 we see the previously defined performance measures for strictly hierarchical systems in comparison to some using LCs. In general, the *Greedy* strategy and the *Equal Request 1* strategy show the overall best results regarding the needed imported and exported energy. Furthermore, for these strategies, the overall amount of energy conducted by the HCs could be lowered significantly with the use of LCs. However, the *Equal Request 2* and the *Equal SoC* variant show different results: the system has to import/export more energy at the top HC, and surprisingly, the average energy flows of all HCs have become larger, whereas the import/export could be lowered. One explanation for this lies in the functionality of these strategies: at first, the requests are always

set up without the information on existing flows. Secondly, since every attached EC in the controller routine during one recursion step is only visited once, the full potential of an EC is maybe not be called to compensate for the answer of another EC. This problem could be partially improved by asking an EC multiple times and would be part of further research.

Conclusion and future work

In this paper, we presented a methodology for modeling CES that have both a hierarchical structure and neighborhood relationships based on energy flows. Therefore we structured the CES into various components, including consumers, producers, and storage units as well as HCs and LCs to represent aggregations of ECs. Each ECs can control its energy locally by an exchangeable controller. As part of the controller implementations, we presented three different simple strategies for how a controller can handle the energy. Furthermore, we analyzed the complexity of the implemented functions and showed that the overall needed computing power is reasonable. The operating principle and the effectiveness of the proposed methods were evaluated in examples of different sizes.

The presented CES simulation could be extended and modified in various ways: Due to the modular architecture of our work, new components and strategies are thinkable. This includes, among others, complex battery models imitating real-world storage systems or advanced controllers based on linear programming, that minimizes a given cost function and has control over a complete subtree. Since the concept of energy flows is not limited to electrical energy systems, heat storage systems and electricity/heat coupled systems can be integrated and the interplay between the multiple forms of energy can therefore be simulated. We can also introduce parallelism into our calculations to both enhance performance and model conflicting situations during our decision-making. Apart from that, also different EC simulation approaches could be created, that take, e.g., a more sophisticated view of the communication structure between the controllers.

Abbreviations

EC	Energy Cell
HC	Hierarchical Controller
LC	Local Controller
SG	Smart Grid
PV	Photovoltaic
P2P	Peer-to-Peer
LEM	Local Energy Management
SoC	State of Charge
BESS	Battery Energy Storage System
CES	Cellular Energy System

List of mathematical acronyms

calc	Calculate
int	Internal
ext	External
ec	(child) Energy Cell
neigh	Neighbor
f	Flow
r	Request
z	Zero
g	Grant
x	Generic flow type
e	Electricity

Acknowledgements
Not applicable.

About this supplement
This article has been published as part of Energy Informatics Volume 5 Supplement 4, 2022: Proceedings of the Energy Informatics.Academy Conference 2022 (EI.A 2022). The full contents of the supplement are available online at https://energyinformatics.springeropen.com/articles/supplements/volume-5-supplement-4.

Author contributions
GD designed, programmed, and tested the neighborhoods and Local Controllers (LCs). He was a major contributor in writing the manuscript. PB designed, programmed, and tested the hierarchies and Hierarchical Controllers (HCs). He was a contributor in editing the manuscript and supervising the project. RG suggested the flow-based simulation of Energy Cells (ECs). He was a contributor in writing and editing the manuscript and was responsible for supervision and project administration. All authors read and approved the final manuscript.

Funding
This paper is funded by the authors' affiliations (Computer Science 7, Friedrich-Alexander-Universität Erlangen-Nürnberg).

Availability of data and materials
Not applicable.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Published: 21 December 2022

References

Abgottspon H, Schumann R, Epiney L, Werlen K (2018) Scaling: managing a large number of distributed battery energy storage systems. Energy Inform 1(1):20

Bauknecht D, Bekk A, Klobasa M, Kühnbach M, Stute J, Pelka S (2021) Evaluation rechtlicher und regulatorischer Rahmenbedingungen in C/sells. Fraunhofer ISI, Karlsruhe

Cali U, Fifield A (2019) Towards the decentralized revolution in energy systems using blockchain technology. Int J Smart Grid Clean Energy 05(8):245–256

Eid C, Bollinger LA, Koirala B, Scholten D, Facchinetti E, Lilliestam J et al (2016) Market integration of local energy systems: is local energy management compatible with European regulation for retail competition? Energy 114:913–922

Feng C, Wen F, You S, Li Z, Shahnia F, Shahidehpour M (2020) Coalitional game-based transactive energy management in local energy communities. IEEE Trans Power Syst 35(3):1729–1740

Haller B, Langniß, O, Reuter A, Spengler N. 1,5 °Csellsius - Energiewende zellulär, partizipativ, vielfältig umgesetzt [1.5 ° Csellsius—energy transition realized cellularly, participatively, diversely]. Silberburgstr. 112, 70176 Stuttgart: C/sells Selbstverlag; 2020

Klempp N , Heilmann E , Pelka S, Wetzel H, Zeiselmair A, Wohlschlager D, Köppl S. Strategisches Gebotsverhalten im Kontext der C/sells FlexPlattform [Strategic bidding behavior in the context of the C/sells FlexPlattform]. Forschungsstelle für Energiewirtschaft e. V. and Forschungsgesellschaft für Energiewirtschaft mbH; 2020

Ma J, Li Y, Grundish NS, Goodenough JB, Chen Y, Guo L, Wan L (2021) The 2021 battery technology roadmap. J Phys D Appl Phys 54(18):183001

Monroe J, Hansen P, Sorell M, Berglund E (2020) Agent-based model of a blockchain enabled peer-to-peer energy market: application for a neighborhood trial in Perth, Australia. Smart Cities 09(3):1072–1099

Orehounig K, Evins R, Dorer V (2015) Integration of decentralized energy systems in neighbourhoods using the energy hub approach. Appl Energy 154:277–289

Perera ATD, Javanroodi K, Wang Y, Hong T (2021) Urban cells: extending the energy hub concept to facilitate sector and spatial coupling. Appl Energy 06:3

Shrestha A, Bishwokarma R, Chapagain A, Sandesh S, Aryal S, Mali B, Korba P (2019) Peer-to-peer energy trading in micro/mini-grids for local energy communities: a review and case study of Nepal. IEEE Access 7:131911–131928

Tushar W, Saha TK, Yuen C, Morstyn T, McCulloch MD, Poor HV et al (2019) A motivational game-theoretic approach for peer-to-peer energy trading in the smart grid. Appl Energy 243:10–20

VDE Verband der Elektrotechnik Elektronik Informationstechnik e V . Zellulares Energiesystem - Ein Beitrag zur Konkretisierung des zellularen Ansatzes mit Handlungsempfehlung [The Cellular Energy System—a contribution to the concretization of the cellular approach with a recommendation for action]. Frankfurt am Main; 2019

Walker S, Labeodan T, Maassen W, Zeiler W (2017) A review study of the current research on energy hub for energy positive neighborhoods. Energy Procedia. 122:727–732. In: CISBAT 2017 international conference future buildings & districts—energy efficiency from nano to urban scale
Website of *C/sells*. https://www.csells.net/. Accessed 29 June 2022

Publisher's Note