Hindawi
Complexity
Volume 2018, Article ID 3469325, 16 pages
https://doi.org/10.1155/2018/3469325

Research Article

Enabling the Analysis of Emergent Behavior in Future Electrical Distribution Systems Using Agent-Based Modeling and Simulation

Sonja Kolen, Stefan Dähling, Timo Isermann, and Antonello Monti

Institute for Automation of Complex Power Systems, E.ON Energy Research Center, RWTH Aachen University, Mathieustraße 10, 52074 Aachen, Germany

Correspondence should be addressed to Sonja Kolen; skolen@eonerc.rwth-aachen.de

Received 2 June 2017; Revised 6 December 2017; Accepted 22 January 2018; Published 21 February 2018

Academic Editor: Liz Varga

In future electrical distribution systems, component heterogeneity and their cyber-physical interactions through electrical lines and communication lead to emergent system behavior. As the distribution systems represent the largest part of an energy system with respect to the number of nodes and components, large-scale studies of their emergent behavior are vital for the development of decentralized control strategies. This paper presents and evaluates DistAIX, a novel agent-based modeling and simulation tool to conduct such studies. The major novelty is a parallelization of the entire model—including the power system, communication system, control, and all interactions—using processes instead of threads. Thereby, a distribution of the simulation to multiple computing nodes with a distributed memory architecture becomes possible. This makes DistAIX scalable and allows the inclusion of as many processing units in the simulation as desired. The scalability of DistAIX is demonstrated by simulations of large-scale scenarios. Additionally, the capability of observing emergent behavior is demonstrated for an exemplary distribution grid with a large number of interacting components.

1. Introduction

The integration of new technologies enabling the energy transition towards an efficient, environment-friendly, and overall sustainable energy system increases the complexity of distribution systems immensely. New technologies encompass renewable energy sources, storage systems, information and communication infrastructures, and new control approaches. The rise in system complexity is a result of the diversity of these technologies with respect to their individual characteristics, time-wise properties, and level of controllability of their behavior. Considering the growing complexity of the system and the different levels of interaction of components, it can be expected that its overall nature cannot be determined based on the behavior of individual components [1]. The state of the system will no longer be given by the states of the components, but results from their nonlinear cyber-physical interactions. The specific evolution, which is a result of interactions in the system and does not coincide with any of the components' behaviors, is called *emergent behavior* [2].

Emergent behavior is in contradiction with linear system behavior because a linear superposition of subsystems' behaviors results again in a linear system with full predictability and easy calculation. In emergent systems, nonlinear interactions between system components preempt linearization of the system behavior [2], making classical small-signal modeling impossible. To analyze the system behavior of future distribution systems at a meaningful scale prior to their implementation in the real world, a scalable modeling and simulation approach is required which is able to create the emergent system behavior based on nonlinear interactions of the system's components. The tool DistAIX (for DISTributed Agent-based sImulation of compleX power systems) presented in this paper includes the following contributions addressing this requirement:

(i) Components: a system component is modeled as an agent with respect to the component's electrical, communicative, and control behavior.

(ii) Emergent behavior: system behavior results from agents' cyber-physical interactions. Edges between agents model the infrastructures of the system that are used for interactions of components, that is, electrical lines and communication links.

(iii) Scalability: the agent-based model is parallelized using processes with respect to the electrical topology of the system under study. This allows a distributed simulation of the model on multiple computing nodes.

Agent-based modeling is used for the definition of components as autonomous elements of a multiagent system. Other approaches such as metaheuristics have been previously used for analysis and optimization problems in complex energy systems [3]. This approach is not considered here because we do not target a search for the solution of an optimization problem, but the observability of a system behavior that emerges from components' interactions. Agent-based modeling has been proven to be a powerful method for modeling emergent behaviors of complex systems [4, 5]. Specialized simulation tools for agent-based modeling in general [6] and power system modeling and simulation in particular [7] have been proposed in the literature. For example, the tool GridLAB-D is capable of coupling the modeling of the domains power, thermal, control algorithms, and market in an agent-based modeling approach.

DistAIX is not the attempt of a GridLAB-D duplicate, but it aims specifically at improving scalability of agent-based modeling and simulation for distribution systems. GridLAB-D uses a power-flow module with an algebraic solver for the computation of currents and voltages of the system [7]. This limits its scalability to the one of the algebraic power-flow module. A parallelization using POSIX threads has been proposed for GridLAB-D [8]. Parallelization with threads instead of processes limits the number of usable computing nodes to one due to general memory locality restrictions of threads [9].

DistAIX parallelizes the computation of the agent-based model with processes and makes use of multiple distributed computing nodes, for example, as available in computing centers. The memory can be spread over multiple computing nodes as each process has its own memory address space. By using distributed memory and computing architectures, the simulation of large, for example, nation-sized, energy systems is possible [10]. We propose and combine methods that address the arising challenges for distribution system simulation, such as distributed power-flow computation and message handling.

For a combined modeling and simulation of power system, communication system, and control, several cosimulation platforms have been proposed in the literature [11–16] coupling different tools for distribution system, transmission system, communication, or market simulations. In addition to the stand-alone usage, DistAIX also extends the pool of modeling and simulation tools that can be cosimulated with others. A co-simulation of multiple large-scale distribution systems with DistAIX coupled with the simulation of a transmission system using a different tool is a possible use case. For example, the platform GridSpice [14] could be used as scaling interface if a respective interface is added to DistAIX. As we focus on the presentation and analysis of DistAIX, cosimulation with other tools is not considered here and left for further research.

Models of the components' behaviors are required to embed the possibility that emergent behavior, that is, nonlinearity, plays a critical role in the final solution. These models can either be derived from literature or be found experimentally. Especially for the control behavior, DistAIX can serve as a testing environment during development of a future control strategy. New component behavior models can be developed using DistAIX. Despite the individual behaviors of the components, the determination of their electrical as well as communication interactions is vital for an analysis of emergent behavior. Efforts have been made for the identification and formalization of the calculation of interactions on electrical level [17]. Combining electrical with communication interactions and cyber-physical interactions in one scalable modeling and simulation tool is one of the contributions of this work.

Unlike traditional energy system simulation tools, for which [18] provides an overview, DistAIX does not target a system level solution. A fully decentralized implementation of the iterative forward-backward sweep method [19] is used for the determination of voltages and currents at all nodes. For this method, only the local neighbors in the electrical grid need to be known to each node and component [19]. No system level solver for the power-flow calculation is required which eases the distribution of the model to multiple processes. Communication links are modeled based on individual properties such as latencies. Messages are exchanged between components along these links and are routed from one process to another where and when necessary. Thereby, all interactions of components can be calculated by distributed computing resources.

We believe that a simulative study of new control strategies for future distribution systems at large-scale is an essential part on the way towards realization. Distribution systems are the largest part of the electrical supply system with respect to the number of electrical nodes and components. In our opinion, a resilient analysis of emergent behavior of distribution systems requires a model that includes all components and their behaviors. For that reason, modeling and simulation for such systems need to be scalable. We present a simulation tool that finds a useful parallelization of an agent-based model for a given set of computing resources and distributes the computation to these resources. DistAIX is designed to keep the computation time for large-scale simulations of emergent system behavior manageable while at the same time enabling a highly flexible modeling of individual component behaviors in agents.

The paper is organized as follows: Section 2 introduces the challenges and related research in the field of scalable agent-based modeling and simulation of energy systems. In Section 3, DistAIX is presented in detail while Section 4 defines the evaluation methodology. Section 5 discusses the results and Section 6 concludes this work.

2. Scalable Agent-Based Modeling and Simulation of Energy Systems

Modeling and simulation of complex agent-based energy systems face the following challenges when the size of the system under study and/or the number of available computing nodes grows:

(1) Computation of power-flow

(2) Interagent data flow

(3) Result data acquisition.

A scalable modeling and simulation tool needs to address the aforementioned challenges. An outline on solutions found in the literature and how DistAIX makes use of them is discussed in the following subsections.

2.1. Computation of Power-Flow. For the execution of a distribution system model on distributed computation resources it is essential that methods working without a centralized point of knowledge on all system variables are chosen. This is especially important for the calculation of voltages and currents in the system under study. Hence, a decentralized power-flow calculation is needed, which can be formulated in the intended way. This topic has been extensively investigated in recent years, resulting in several decentralized power-flow formulations.

In [20, 21] the grid is divided into a set of subgrids. Two neighboring subgrids share common buses, the so-called boundary buses. For each subgrid, the power-flow is solved independently and the injected power from all neighboring subgrids is calculated subsequently considering the voltage mismatch of boundary buses. This process is repeated iteratively until the power-flow solutions of all subgrids converge. While the overall solution is found in a decentralized way, the power-flow computation within one subgrid is carried out centrally using algorithms such as Newton-Raphson.

Other approaches aim at solving the Gauss-Seidel algorithm in a decentralized way [22]. Each node calculates its voltage using the previously calculated voltage of all neighboring nodes. The new voltage value is passed to the next node which then also updates its own voltage. In this approach the calculation is performed on node level. However, the convergence of Gauss-Seidel can be considered comparatively slow. In [23] an accelerated Gauss-Seidel implementation using FPGAs is presented. Their approach is not compatible with the requirement for a decentralized power-flow calculation as mentioned above.

Another algorithm that can be used for decentralized power-flow calculation of radial power grids is the so-called forward-backward sweep method. It has been used for agent-based smart grid simulation [24, 25] and consists of three iteratively repeated steps [19]:

(1) Calculation of nodal currents: at each node, the current injections into all components connected to a node are calculated for a given nodal voltage. In the first iteration, all nodal voltages are initialized with the rated voltage. The nodal currents are obtained by summing up all component currents of a node.

(2) Backward sweep: starting at the last node of each feeder, the current flows through all branches are calculated by applying Kirchhoff's current law at the nodes.

(3) Forward sweep: starting at the slack node, all node voltages are updated based on the branch impedances and the previously calculated branch currents.

Note that there are also other formulations of the forward-backward sweep method which calculate power-flows instead of current flows in the backward sweep [26]. Forward-backward sweep provides considerable advantages for an agent-based simulation. The most important ones are as follows:

(i) The calculation is performed on node level and is therefore highly decentralized. The electrical system behavior emerges from the electrical interactions of agents.

(ii) The algorithm makes use of the radial structure of distribution grids. As a result, the processing of forward and backward sweep in each feeder can be computed in parallel.

(iii) The electrical behavior of grid components is calculated independently for each component in the first step. Hence, this step can be parallelized.

(iv) Sufficient convergence has been shown for this algorithm [27].

For DistAIX, we choose the forward-backward sweep technique presented in [19] as the given advantages ensure the required scalability feature and allow a system analysis based on the behavior and interactions of single components. Since the algorithm makes use of the structure of a distribution grid it is not able to deal with meshed grids. Extensions described in literature solve this problem for weakly meshed grids [19] such as electrical distribution systems.

2.2. Interagent Data Flow. The separation of a distribution system model into subparts for distributed computation demands an interagent data flow organization that handles the distributed memory architecture in a scalable way. Interagent data flow means the cyber-physical interactions of agents which can take place within a model subpart and between different model subparts. Two approaches addressing this challenge have been identified in existing agent-based modeling and simulation frameworks [28]:

(i) Message boards: Flame [29]

(ii) Agent copy and update: RepastHPC [30].

Both frameworks use the Message Passing Interface (MPI) to parallelize the simulation on multiple computing nodes. Flame uses a message board in each process for the realization of agent interactions. Agents use the message board in their process to exchange data with other agents and messages are broadcast via MPI either to all or to a set of other agents.

The agent copy and update approach of RepastHPC is illustrated in Figure 1 for a small example with two processes and two agents. The methodology is introduced in [30]. Agent A in process 1 requires data of agent B that belongs to process 2 and vice versa. The required data can be voltage and currents, for example. In RepastHPC, process 1 creates a copy of agent B in its own memory space. Process 2 does the same for agent A. If agent A in process 1 needs to access data of agent B, it reads from the copy in process 1 and not from the original agent. Agent-networks between original and copy agents within the same process (red arrows in Figure 1) are called SharedNetworks in RepastHPC and are maintained in their own data structure. Copies are only created for agent links that go across processes, that is, across subparts of the model. Whenever required, the copies can be updated with the state of the original agents by calling an MPI-based synchronization method of RepastHPC (blue arrows in Figure 1). In contrast to Flame, this method does not use broadcast but asynchronous message sending and receiving via MPI. In the example, the copy of agent B in process 1 is updated with the state of the original agent B and the copy of agent A is updated with the state of the original agent A. The state of the agent copy is never transferred back to the original agent. Thereby, frequent interprocess communication is avoided and the programmer has full control on the points in time when copy updates are necessary.

As has been shown in [28], RepastHPC's scalability outperforms the one of Flame because of this difference in the realization of agent interactions. Also, the memory consumption of RepastHPC stays constant for increasing number of processes used in the simulation while Flame's memory consumption increases [28]. Due to these benefits, RepastHPC is used for the implementation of DistAIX. Two agent network structures (SharedNetworks) are used in each process to organize physical and cyber interactions of agents. For more details on the implementation refer to Section 3.5.

2.3. Result Data Acquisition. Storing of simulation time dependent and independent data should put only minimal stress on the computation resources while at the same time the analysis of result data and postprocessing have to be flexible and convenient. The following two problems need to be solved in an efficient way by a data acquisition solution:

(i) Store huge amounts of data per simulation time step.

(ii) Read specific result data for postprocessing and analysis.

Connecting the simulation to a database system is one possible solution; for example, GridLAB-D offers an interface to a MySQL database [31]. Relational databases such as MySQL and PostgreSQL are useful for storing simulation time independent data, that is, the metadata of the agents and the model. However, they have problems in processing large amounts of data at once in either direction, writing to or reading from the database. Benchmarks have shown that NoSQL databases have a better performance when it comes to fast storing of large amounts of data [32, 33]. For these reasons, DistAIX is interfaced with both a PostgreSQL database for meta information and a NoSQL database cluster based on Cassandra for storing simulation time dependent data in a scalable database system.

With respect to the huge amounts of data that need to be acquired by the database system, serialization of data seems to be a good option. Protocol Buffers [34] have proven to be an efficient method for data serialization and deserialization in other fields of application [35, 36] and are therefore explored here for the decrease of time to write data to the Cassandra database.

3. Modeling and Simulation with DistAIX

DistAIX is presented in this section. Section 3.1 discusses how RepastHPC is used for the parallelization of a simulation and Section 3.2 discusses the distribution of agents to processes. In Section 3.3 we comment on the time step whereas details on the modeling of agent behaviors and all agent types are provided in Section 3.4. Section 3.5 focuses on agent interactions and discusses the computation of communication as well as electrical interactions. Finally, the setup of a simulation is addressed in Section 3.6.

3.1. Parallelization with RepastHPC. RepastHPC uses MPI to parallel a simulation on a high performance computing (HPC) system. A simulation is launched via `mpiexec` with options specifying the hosts to use and the number of processes to start. Each process executes one part of the agent-based distribution system model. In DistAIX, this is reflected by a C++ class `Model` which defines how each process is configured, for which agents it is responsible, and which connections exist among agents in this process and to agents in other processes. The `Model` class also defines the schedule of a process and data that need to be synchronized with other processes.

Figure 2 shows the schedule of a process. The first step is the initialization and encompasses the creation of all agents of the process and all electrical connections between agents in this process. Further, the electrical connections to agents in other processes are established by the creation of copies of these agents in the process. If these copies were not created, electrical connections between agents belonging to different processes would be missing in the simulated model. This would lead to wrong simulation results. The model is kept synchronous and consistent by updating copies with the state of the original agent according to the copy update method explained in Section 2.2. Agent copies are updated between each step of the process schedule (white boxes in Figure 2) and also during the forward-backward sweep step to reach convergence.

After the initialization, a process executes simulation steps for a given simulation step size (loop in Figure 2). The agent messages and individual control behaviors are processed in step 2. This part of the simulation considers

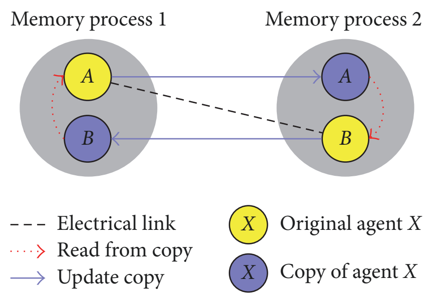
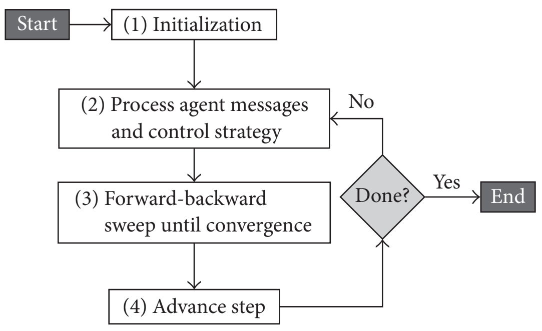


FIGURE 1: RepastHPC's agent copy and update synchronization.



Start
(1) Initialization
(2) Process agent messages and control strategy
(3) Forward-backward sweep until convergence
(4) Advance step
Done?
No
Yes
End

FIGURE 2: Schedule of each process as defined in `Model` C++ class.

the communication network infrastructure (how communication links are modeled is explained in Section 3.5) as well as the control strategy under study. At the beginning of step 2, agent messages have to be synchronized between processes. After that, a process executes the behaviors of all of its agents independently for each agent. Step 3 of a process is the calculation of current and voltage at each node using the iterative forward-backward sweep method until convergence.

The calculation of the forward-backward sweep algorithm is parallelized according to its three phases as explained in Section 2.1. Similar to the message processing, the calculation of nodal currents is done independently of other agents. The algorithm iterates backward and forward through the grid topology in sweeps. During a sweep, only one node per feeder can be active. After a sweep is completed, an agent copy update is required to synchronize newly computed values of currents and voltages at electrical agent connections spanning over two processes. The three phases of the forward-backward sweep algorithm are repeated until convergence is reached. Step 4 of a process is composed of preparing the next simulation time step, that is, saving results of this step and resetting internal state variables used during forward-backward sweep iterations. This final step is done independently for each agent.

3.2. Distribution of Agents to Processes. Agents need to be distributed to processes in such a way that processes wait for others the least amount of time. The topology of the electrical grid influences a good distribution of agents as only one agent per feeder can be active in the forward-backward sweeping at once. It can be expected that the number of processes beneficial for the parallelization depends on the electrical topology of the system and is therefore limited.

We adapt a node scheduling method [37] to work with contiguous sequences of successive nonbifurcating nodes of the electrical grid, called *workitems* in the following. Workitems are the parts of the grid in which only one node can be active in forward-backward sweeping at once. The electrical topology of the grid under study is analyzed at the start of a simulation. Workitems are determined by parsing the graph topology of the grid under study with a depth-first-search algorithm. The distribution of node agents to processes works in the following steps:

(1) Leaf node agents and their depth in the topology are determined.

(2) If the number of processes available is m, the m deepest leaf node agents are selected (or less if there are less than m leaf nodes in the topology).

(3) The m workitems in which the selected m leaf node agents are included are assigned to one of the m processes so that all m processes are responsible for the nodes contained in one workitem. If $n < m$ leaf node agents were selected in the previous step, the number of workitems and processes in this step is only n.

(4) From now on only the topology of the remaining nodes agents which are not yet assigned to a process is considered. Continue with step 1 if there are still nodes agents to be assigned to a process. Otherwise, all node agents are distributed to processes.

Transformer agents and slack agent are both treated as node agents regarding their distribution to processes. Component agents (load, electric vehicle (EV), photovoltaic (PV), combined heat and power (CHP), wind energy converter (WEC), battery, and compensator) are always created in the process where the node agent they are connected to is located. Note that this procedure does not guarantee the usage of all available processes. The topology of the electrical grid influences the number of processes in use. This is user-friendly as only the maximal available number of processes needs to be known, but not the optimal number with respect to the system under study. If more processes are available than useful, the dispensable processes are idle during the simulation and do not impede the calculations.

3.3. Simulation Time Step. The size of a simulation time step is fixed for one simulation. The main reason for this is that the occurrence of communication events cannot be predetermined due to emergence in the system. A variable time step would increase the chance to miss time steps in which communication occurs. Another aspect to bear in mind is the distributed simulation approach itself. As the time step has to be the same in all processes, additional synchronization would be necessary for the determination of the time step size and the spreading of this information to all processes. The overhead caused by the required synchronization could possibly outweigh the performance improvements of a flexible time step.

However, the number of calculations per time step on agent level can be reduced. For example, the recalculation of nodal currents in step 1 of the forward-backward sweep algorithm can be managed in an event-driven way so that a recalculation is done only when an event has occurred in the signal of the nodal voltage. This can be an attempt to reduce the computational effort for component agents in the future. However, for a first demonstration of DistAIX we have not implemented such a method yet.

3.4. Agent Behaviors. To draw conclusions on emergent behavior of the system, the agent models used in DistAIX are discussed in this section. Equations for the electrical agent models are provided to show their role in the forward-backward sweep iterations. Each agent represents a component of the distribution system and consists of an electrical model, behavior rules for the control, a state and knowledge, and communication capabilities. Figure 3 provides an overview about the interconnection of these parts within the agent model.

The electrical model of an agent and its communication capabilities influence the state of the agent and its knowledge about other components in the system. The control algorithm of an agent is defined by its behavior rules operating based on the current state and knowledge. Behavior rules result in both control signals influencing the electrical model (right side in Figure 3) and triggers for information exchange using the communication capabilities (left side in Figure 3). An agent can be connected to other agents in no more than two ways: via communication and/or electrical network. Detailed descriptions of the electrical steady state models are provided in the following subsections. In the equations below, variables indexed with "ctrl" indicate that this value is a control signal determined by the control behavior rules of an agent.

3.4.1. Slack Agent. In the forward sweep, the nodal voltage of the slack is calculated to be

$$v_{\text{node}} = \frac{V_n}{\sqrt{3}} \cos(\vartheta_{\text{slack}}) + j\frac{V_n}{\sqrt{3}} \sin(\vartheta_{\text{slack}}). \tag{1}$$

V_n is the nominal voltage and ϑ_{slack} is the defined voltage angle at the slack node, usually set to zero. In the backward sweep, the output current of the slack node flowing to the next node is

$$i_{\text{out}} = \sum_{n\in N} i_{\text{in},n} + i_{\text{leak},n}, \tag{2}$$

where N is the set of all next nodes, $i_{\text{in},n}$ is the incoming current of node n, and $i_{\text{leak},n}$ is the leakage current of the line connecting the slack node and node n.

3.4.2. Node Agent. In the forward sweep, the nodal voltage is obtained by

$$\begin{aligned} v_{\text{node}} &= v_p - i_{\text{out},p}(R_l + jX_l) \\ &\quad + v_p \frac{(G_l + jB_l)(R_l + jX_l)}{2}, \end{aligned} \tag{3}$$

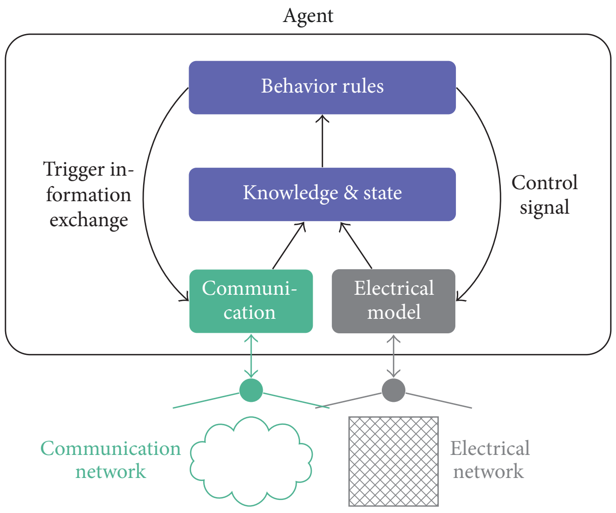


FIGURE 3: Model of a component agent and its connections to electrical and communication networks.

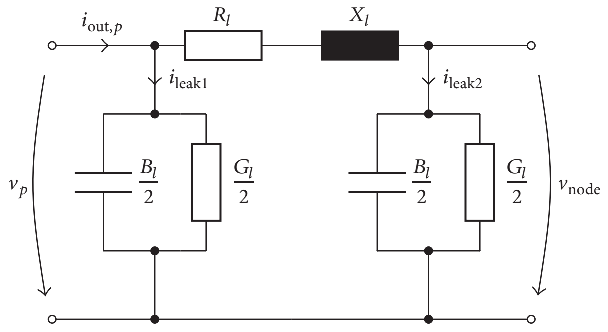


FIGURE 4: Single-phase reference circuit of a power line.

where v_p is the nodal voltage of the previous node, $i_{\text{out},p}$ is the output current of the previous node, R_l, X_l, G_l, B_l are parameters of the connecting PI-line (see Figure 4). In the backward sweep, the output current from the previous node is

$$i_{\text{out},p} = i_{\text{out}} + i_{\text{leak}} + \sum_{c\in C} i_c, \tag{4}$$

where i_{out} is the output current of the investigated node, i_{leak} is the leakage current of the line connecting this and the previous node, C is the set of all components connected to the node, and i_c is the component current.

3.4.3. Transformer Agent. The transformer model is composed of an RX-line and an ideal transformer as depicted in Figure 5. In the forward sweep, the secondary node voltage is obtained by

$$v_{\text{node}} = \frac{v_{\text{in}} - i_{\text{out},p}(R_t + jX_t)}{u}, \tag{5}$$

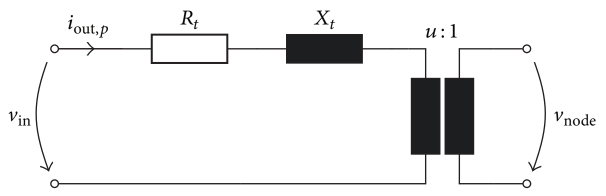


Figure 5: Single-phase reference circuit of a transformer.

with the input current from the previous node $i_{\text{out},p}$, the input voltage at the primary node

$$v_{\text{in}} = v_p - i_{\text{out},p}\left(R_l + jX_l\right) + v_p \frac{\left(G_l + jB_l\right)\left(R_l + jX_l\right)}{2}, \quad (6)$$

the transformer parameters R_t and X_t, and the turns ratio u. R_l, X_l, G_l, and B_l are parameters of the PI-line connecting the transformer with the previous node. In the backward sweep, the output current from the previous node is

$$i_{\text{out},p} = \frac{i_{\text{out}}}{u} + i_{\text{leak}}, \quad (7)$$

where i_{out} is the sum of all currents flowing to the next nodes and i_{leak} is the leakage current of the line between this and the previous node.

In DistAIX we use transformers with an On Load Tap Changer (OLTC) capability. Therefore, the turns ratio can be adjusted within a symmetrical range r around the nominal value u_{nom}. This is done in discrete steps N_{ctrl} where

$$-N \leq N_{\text{ctrl}} \leq N \quad (8)$$

has to be satisfied and N is the number of possible steps. The turns ratio is then calculated as

$$u = u_{\text{nom}} \cdot \left(1 + N_{\text{ctrl}} \frac{r}{N}\right). \quad (9)$$

3.4.4. Load Agent. The load model has an integrated profile providing the active and reactive power demand for each step. The component current is then calculated to

$$i_c = \frac{P + jQ}{3v_{\text{node}}}. \quad (10)$$

3.4.5. Electric Vehicle Agent. An EV is represented by a battery with a capacity C_{el} that is periodically connected to the grid. The connection status is provided by a profile. If the EV is not connected to the grid, it has a certain power consumption P_{con} which is also stored in a profile and leads to a discharge of the EV's battery. For times of disconnection the model output is

$$i_c = 0, \quad (11)$$

with state of charge (SOC) of the battery being

$$\text{SOC} = \text{SOC}_{\text{old}} - \frac{P_{\text{con}} \cdot t_{\text{step}}}{C_{\text{el}}}. \quad (12)$$

If the EV is connected, the battery can be charged with an adjustable active power. Moreover, the EV is able to provide a certain amount of reactive power as it is connected to the grid via a converter. With the control values for active and reactive power P_{ctrl} and Q_{ctrl} the component current is

$$i_c = \frac{P_{\text{ctrl}} + jQ_{\text{ctrl}}}{3v_{\text{node}}}, \quad (13)$$

and the SOC is

$$\text{SOC} = \text{SOC}_{\text{old}} + \frac{P_{\text{ctrl}} \cdot t_{\text{step}}}{C_{\text{el}}}. \quad (14)$$

The EV agent ensures that the component's hardware limitations such as converter capabilities as well as the battery capacity are not violated. Location variability of EV is not considered here as it does not influence the applicability of DistAIX but only the control strategy under study.

3.4.6. Photovoltaic Agent. The model of a PV system is based on a performance prediction model [38] calculating the amount of active power $P_{\max}$ that can be generated. It requires the solar irradiance and the temperature as input data. Similar to an EV, PV systems are connected via a converter and can therefore provide reactive power. Furthermore, the active power supply can be curtailed if necessary. This leads to the equation for the component current (see (13)) where the control value for reactive power has to be within the limits of the converter and the control value for active power has to satisfy

$$-P_{\max} \leq P_{\text{ctrl}} \leq 0. \quad (15)$$

Note that P_{ctrl} is negative since active power is produced and not consumed.

3.4.7. Wind Energy Converter Agent. The WEC model makes use of a polynomial relation between wind speed and output power [39]. Hence, the model uses a wind speed profile as input data. Power generation starts when the minimum speed v_s is reached. Up to the rated speed v_r, generation continues following a polynomial relation. Maximum generation P_r is present until the cut off speed v_c is reached. This relation is illustrated by the following equations:

$$\begin{aligned} P_{\max}(v) &= 0, \quad v < v_s, \\ P_{\max}(v) &= a \cdot v^3 + b \cdot v^2 + c \cdot v + d, \quad v_s \leq v \leq v_r, \\ P_{\max}(v) &= P_r, \quad v_r \leq v \leq v_c, \\ P_{\max}(v) &= 0, \quad v > v_c. \end{aligned} \quad (16)$$

Between speeds v_s and v_r, the relationship between wind speed and maximum output power is described by a third-order polynomial with coefficients a, b, c, and d which are found so that a continuous curve is obtained. Similar to the PV and EV, the WEC is connected to a converter, which leads to similar constraints and enables the control of active power P_{ctrl} and reactive power Q_{ctrl}, as shown in (13), while respecting the limits of the generation unit (see (15)).

3.4.8. Combined Heat and Power Agent. The CHP model consists of three main components: combustion engine, synchronous generator, and heat exchanger. In a CHP, mechanical power supplied by a combustion engine is converted to electrical power by a generator. The occurring heat can be used directly for building heating or stored for later use in a thermal storage. Thus, the total fuel input power P_c is composed of the thermal output power Q_{th}, the electrical output power P_{el}, and losses P_{loss}:

$$P_c = Q_{\mathrm{th}} + P_{\mathrm{el}} + P_{\mathrm{loss}}. \tag{17}$$

Thermal and electrical output power are determined considering a thermal and electrical efficiency η_{th} and η_{el}, respectively. With the control value for electrical active power P_{ctrl}, the required fuel input power is

$$P_{\mathrm{c}} = \frac{P_{\mathrm{ctrl}}}{\eta_{\mathrm{el}}}, \tag{18}$$

resulting in the thermal output power

$$Q_{\mathrm{th}} = \eta_{\mathrm{th}} P_c = P_{\mathrm{ctrl}} \frac{\eta_{\mathrm{th}}}{\eta_{\mathrm{el}}}. \tag{19}$$

Moreover, the CHP can produce reactive power, as the power factor of the synchronous machine can be affected by controlling its excitation within the limits of the machine.

To simulate the CHP, a profile is required providing the thermal power demand Q_{dem} for building heating at each time step. The agent considers Q_{dem} and the currently stored thermal energy E_{th} and controls P_{ctrl} and Q_{ctrl} of the CHP model which returns the actual i_c to the simulation analogously to (13). The SOC of the thermal storage with capacity C_{th} is updated accordingly to

$$\mathrm{SOC} = \mathrm{SOC}_{\mathrm{old}} + \frac{\left(Q_{\mathrm{th}} - Q_{\mathrm{dem}}\right) t_{\mathrm{step}}}{C_{\mathrm{th}}}, \tag{20}$$

for the next time step. One constraint of the operation of the CHP agent is the satisfaction of limits of the thermal storage.

3.4.9. Battery Agent. Similar to the EV agent, the battery agent controls a battery system. Its battery can be charged/discharged with an adjustable active power and is connected to the grid via a converter. This enables the control of P_{ctrl} and Q_{ctrl} (see (13)). With respect to the loads of nearby consumers, the battery agent releases power to or stores power from the grid when necessary. Meanwhile, it ensures that the battery's operation boundaries will not be exceeded in terms of (14).

3.4.10. Compensator Agent. The compensator can provide capacitive reactive power to the grid to neutralize inductive reactive power caused by most loads. Reduction of grid losses is the main reason for the utilization of a compensator. To counter losses, the compensator is operated through N stages of switchable shunt capacitors with a nominal susceptance of B_{nom}. The agent controls the step size N_{ctrl} of the shunt banks resulting in the grid connected susceptance

$$B = B_{\mathrm{nom}} \frac{N_{\mathrm{ctrl}}}{N}. \tag{21}$$

The output current i_c is calculated as follows:

$$i_c = jBv_{\mathrm{node}}. \tag{22}$$

3.5. Agent Interactions. As interactions play a major role in emergent systems, they have to be modeled accurately. RepastHPC's SharedNetwork structure is used to model the electrical network, where the agents are the nodes and components in the network and cable models represent the edges between them. A cable model includes an equation system and parameters of a cable (see PI-line in Figure 4). Each process stores the excerpt of the electrical network involving its own agents and boundary agents to other processes in a SharedNetwork object. This object is a part of the `Model` class introduced in Section 3.1. The electrical interactions are calculated based on the forward-backward sweep method as explained in Section 2.1 and the SharedNetwork objects in the processes.

A generic modeling of communication interactions and communication edges requires a different approach. Assuming that a control strategy may require flexible communication connections, communication between all agents in all processes has to be possible. As the system under study is emergent, it may be unknown in advance which communication links will be used, that is, which agents communicate with each other. If a SharedNetwork was used for the modeling of communication links similar to electrical edges, copies of all agents in all processes would have to be synchronized. Hence, the SharedNetwork is not a scalable method for explicit modeling of communication edges, especially when only a small subset of all available communication links is used.

For this reason, one message router is introduced in each process (Figure 6). A message router does not represent a hardware component of the distribution system but is a method to implement the simulation of communication between agents. In step 2 of a process schedule (see Figure 2), the task of the message router is the collection of messages to be sent from all agents in their process (orange in Figure 6). For each message to be sent, the message router checks the model of the communication link used by the message transfer, for example, the latency or packet error rate of the link. In case no specific communication link properties are available, the user can define default values for all links. The message router applies the communication link model to determine whether or not a message has to be transmitted in the current simulation time step. If the latency of a link requires delaying a message, it will stay in a pending queue of the message router and will be checked again in the next simulation time step. If a message has to be transmitted to the target agent, two cases have to be distinguished:

(1) If the target of a message is an agent in the process of the message router, it will route the message to the inbox of this agent.

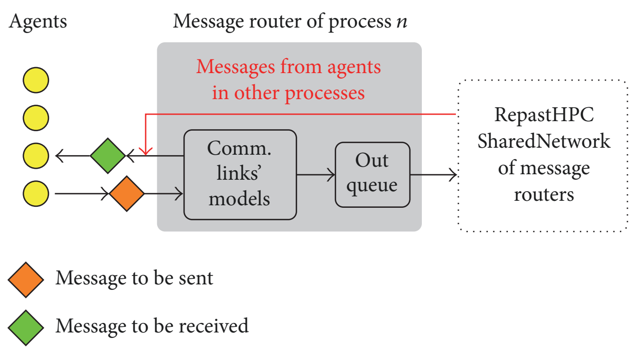

Agents
Message router of process *n*
Messages from agents in other processes
Comm. links' models
Out queue
RepastHPC SharedNetwork of message routers
Message to be sent
Message to be received

FIGURE 6: Modeling of communication infrastructure with message routers in process *n*.

(2) If the target of a message is an agent in a different process, the message is added to the out queue of the message router.

Each process has a RepastHPC SharedNetwork in which its own message router is connected to the message routers of all other processes. Once all messages to be transmitted are treated according to cases 1 or 2, the out queues of all message routers are synchronized between all processes via the SharedNetwork of message routers. Afterwards, each process has an up-to-date copy of the out queue of each message router. Each message router checks the others' out queues for messages that target an agent in its process and routes messages accordingly (red arrow in Figure 6). This way of communication interaction modeling allows an individual modeling of communication links and an efficient routing of agent messages in the distributed simulation.

3.6. Simulation Setup. Figure 7 shows the setup of a simulation with DistAIX. The simulation time step size and the number of time steps to simulate can be freely chosen. However, they should be selected in a meaningful way for the scenario and component models under study regarding the desired resolution of electrical dynamics and message exchange. Time series are used as input data for component agents that require profile information. If a time step smaller than the time step of the profile data is chosen, the profiles can be either linearly interpolated using the respective functionality of the GNU Scientific Library or the last value of the profile is held until the next value of the profile is reached by the simulation time. The user may choose between these options when configuring a simulation.

The simulation scenario is configured by lists of the nodes and components to be simulated and the electrical lines. Furthermore, the properties of communication links (e. g., package drop rates, latency) can be added as input for the simulation if these parameters are relevant for the scenario under study. It would also be possible to extend the interfaces of the simulator so that scenarios available in a Common Information Model (CIM) representation can be used as input for the simulation tool. A method for the automated transformation of CIM representations of distribution systems to C++ classes has already been presented

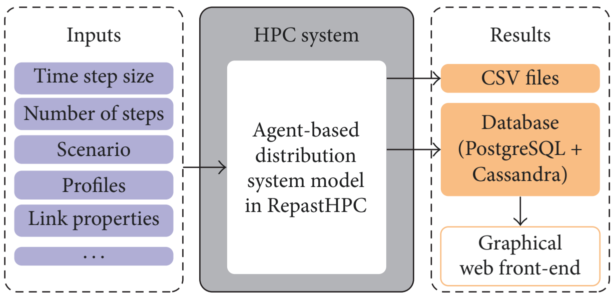


FIGURE 7: Simulation setup of DistAIX with inputs and results.

by our research group in [40]. For an initial demonstration of DistAIX, we choose a less complex method of scenario reading.

The user can choose between a simulation result output in a PostgreSQL and Cassandra database system, comma separated value (CSV) files, or both. Simulation results in CSV test simulations have proven to be useful during development of DistAIX due to their simplicity. Database output is integrated into the simulator because it allows a structured and efficient storage of large amounts of simulation data. It also enables a resource saving of the computing resources as database queries can be sent to the databases via a computer network and do not need to be handled by the executing computer system.

If the database is selected as result storage option, all simulation parameters, such as configuration of the simulation and constant agent properties, are stored as metadata in a PostgreSQL database. The result data of all agents for each simulation time step is stored in a Cassandra database system which currently consists of three nodes. The Cassandra cluster is horizontally scalable according to user demands. The three Cassandra nodes handle the storing of results into the database during simulation and split the workload automatically in a fair manner. For the database output, a graphical web front-end is available enabling an easy and efficient inspection and evaluation of simulation results. To further reduce the time it takes to send the result data of each simulation time step to the database, the data is serialized using Protocol Buffers and sent in a binary format. For the extraction of data from the database, the Protocol Buffers need to be applied again for deserialization.

4. Evaluation Methodology

For the evaluation of correctness and performance of DistAIX, we address three aspects: correctness of the distributed forward-backward sweep implementation, observability of emergent behavior, and the scalability of the simulator on an HPC system. The methodology used to address these three aspects is discussed in the following subsections.

4.1. Correctness of Distributed Power-Flow Calculation. The utilized method for the calculation of electrical interactions is the forward-backward sweep method. Convergence of this method has already been shown in [27]. DistAIX implements

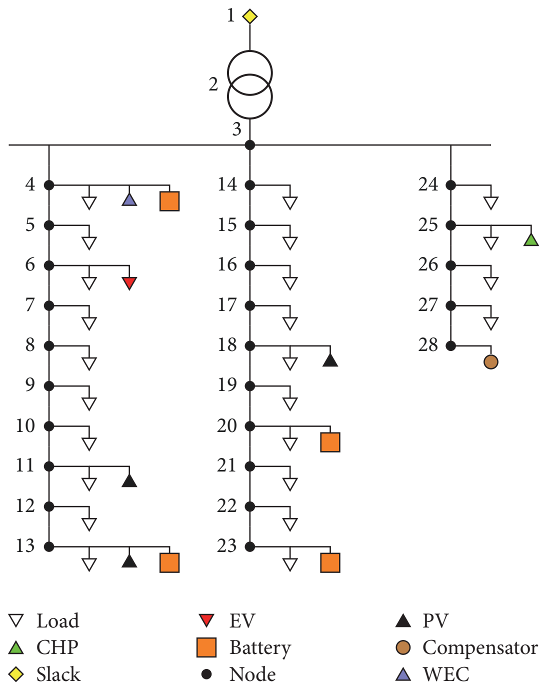


Figure 8: Low voltage distribution system for evaluation of correctness of distributed forward-backward sweep implementation.

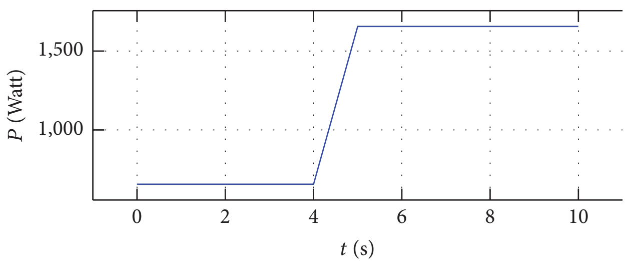


Figure 9: Profile of single family household load.

the algorithm in a distributed way. To demonstrate that our implementation with processes provides numerically correct results we select a small scenario and compare results to those obtained from Modelica modeling and simulation of the same scenario. Modelica is chosen for this demonstration, due to its flexible use and free availability as OpenModelica. As a metric to evaluate the correctness, we use the absolute differences Δ between node voltages and line currents obtained with DistAIX and Modelica as well as the number of occurrences of such differences. From the voltage and line current differences we derive conclusions on the correctness of our implementation.

To obtain comparable results, the electrical components are modeled according to Sections 3.4.1–3.4.10 in Modelica. Further, the same profile data are used as input for the electrical component models in both simulation environments. Control values for all components are found without communication interactions by simple rules as presented in the reference scenario in [41]. The distribution system scenario used as reference is shown in Figure 8. All agent types are used to ensure a general correctness of the presented approach. For the Modelica simulation, we use the DASSL solver with a tolerance of 10^{-6} and a time step of 60 s. The same time step is applied to our simulator. Convergence of simulation results is detected when all nodal voltages change less than a certain ϵ between two iterations. The ϵ is set to the value of the DASSL solver tolerance 10^{-6}. The simulation is executed for a complete day.

4.2. Observability of Emergent Behavior. Methods to detect and characterize emergent behavior in complex systems are discussed in the literature, for example, [42–44]. Such methods are always dependent on the application and system under study. In order to analyze emergent behavior characteristics of different control applications for distribution systems, DistAIX enables observability of emergent behavior in large-scale simulations. We demonstrate this functionality by selecting one control application called SwarmGrid [41]. SwarmGrid is expected to result in emergent behavior of the distribution system due to its bottom-up, decentralized control strategy. In the SwarmGrid control concept, the system level target is voltage stability of the system. Power producers and consumers (agents) negotiate the amount of power they supply or consume in a self-organized way. Their main objective is the usage of flexibilities for the local balancing of power production and consumption. Agents form so-called swarms, that is, groups of agents that need to interact with one another in order to achieve their goal.

The formulation of such a bottom-up heuristic control as one complete analytical model is infeasible. However, DistAIX allows the system behavior to emerge from the definition of the individual component behaviors. The agent-based concept and the need to simulate electrical and communicative interactions as well as component control together make the presented simulation tool suitable for an evaluation of SwarmGrid control. Furthermore, due to the vast number of components in the distribution grid that can be utilized in control approaches such as SwarmGrid, the system can be described as highly complex with various possible interactions among agents.

For a demonstration of observability of emergent behavior, we use the rural low voltage grid of 177 nodes as described in Section 4.3 and the electrical models given in Sections 3.4.1–3.4.10. In order to provoke emergent behavior, all input profiles have constant values except for the ones of single family household loads. These profiles feature a ramp up of 1 kW within 1 s as depicted in Figure 9. The ramps happen simultaneously, start at 4 s, and end at 5 s. As the total number of single family households in the grid is 140, the aggregated change in the total power exchange of the grid is 140 kW. The time step size of this assessment is set to 10 ms and 1000 time steps (10 s) are simulated. The communication link latency for all links is set to 100 ms to consider delays caused by message transmission. Thereby, the system behavior before, during, and after the household load ramp can be analyzed.

We determine (a) the aggregated active power behavior based on a reference simulation without self-organizing

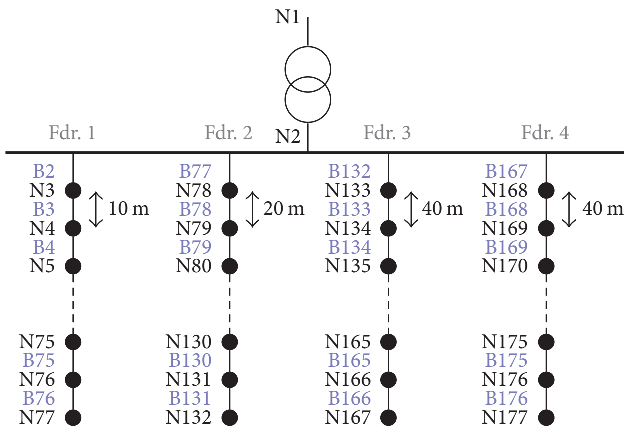


Figure 10: Low voltage grid with 177 nodes used in emergent behavior and scalability methodologies; figure taken from [41].

control and (b) the aggregated active power behavior of a simulation with SwarmGrid enabled. By comparing the results for (a) and (b), we can conclude on how emergent behavior appeared in the system and changed the overall behavior of specific component types and the system itself. As the emergent behavior results from the interactions of agents, the number of exchanged messages is evaluated as an additional indicator. Due to the individual and heterogeneous control behaviors of the system components (agents) and their situation-dependent interactions, it is virtually impossible to anticipate their behavior as a group or swarm. Even if their controls are designed to achieve a particular system level result, such a control cannot possibly have targets for all internal system variables. DistAIX enables the observation of even these internal variables and understanding their relation to the system level target.

4.3. Scalability. For the performance assessment of DistAIX with respect to scalability, a rural low voltage grid containing 177 nodes and 310 components (i.e., a total of 487 agents) is chosen. The low voltage grid is shown in Figure 10. It is identical to the one used by the authors in [41] and consists of four feeders with 75, 55, 35, and 10 nodes with different distances between single nodes. While two of the feeders are equipped with overhead lines the other two are built with cables. The installed power is listed in Table 1. The total installed producer power is 503.5 kVA and consumer power is 448.6 kVA. Moreover, it contains 275.2 kWh battery capacity and 60 kvar reactive power compensation. The low voltage grid is connected to a medium voltage feeder as many times as needed to upscale the system. This means the number of agents N_{agents} in the system is

$$N_{\text{agents}} = N_{\text{LV}} \cdot 487 - (N_{\text{LV}} - 1), \tag{23}$$

where N_{LV} is the amount of low voltage grids connected to the medium voltage feeder and $(N_{\text{LV}} - 1)$ is subtracted as there is only one slack bus.

This methodology is used to analyze the execution time of the simulation for different quantities of agents. In order

Table 1: Installed power in scenario of 177 nodes; table adapted from [41].

Component	Number	Installed power/capacity
Load	175	282.6 kVA
EV	35	166 kVA
PV (peak power)	30	220 kVA (165 kW)
WEC (peak power)	5	84 kVA (69 kW)
CHP	25	199.5 kVA
Storage	35	275.2 kWh
Compensator	4	60 kvar

to evaluate the impact of communicative and electrical interactions, two simulations are executed for each grid size:

(1) With agent communication: SwarmGrid [41] is enabled.

(2) Without agent communication: components behave in an uncoordinated manner based on currently valid guidelines for Germany (same as reference case in [41]).

The simulation execution time $T(p)$ for a number of p processes is measured and serves as scalability metric. The fastest execution time and the respective number of processes are used in the evaluation. The number of simulation time steps to execute is set to 1000 with a time step size of 1 s. All component profiles consist of two different values (one for $t = 0$ and one for $t = 1000$). The intermediate steps are linearly interpolated between these two values so that there is a power demand or generation change in each time step. Hence, the communicative interactions caused by SwarmGrid are representative for a case where the power system conditions change dynamically. Each simulation was carried out several times to ensure stable results for the execution time.

The available computation resources are four computing nodes. Each node includes 24 physical cores of type Intel Xeon E5-2658 v3 at 2.20 GHz. One computing node is used for simulations with 1–24 processes, two nodes for 25–48 processes, three nodes for 49–72 processes, and four nodes for 73–96 processes. Processes are always split equally among computing nodes. Each computing node has 126 GB RAM and all four nodes are connected via an Ethernet network for the benchmarks. They all run a CentOS 7.3 operating system using a 3.10.0 Linux kernel. The MPI implementation used on all computing nodes is the high performance MPI library ParaStation MPI [45]. Since the base grid for the scalability investigations has four feeders and our methodology duplicates this grid N_{LV} times to generate larger grids, $N_{\text{LV}} = 25$ is the largest scenario for the scalability study. This scenario has 100 feeders, that is, workitems, which can be optimally parallelized by the available computation resources. Additionally, the execution times of three scenarios containing considerably more feeders than available processors are investigated to give an impression of the examinable scenario sizes.

In practice, DistAIX can use separate computer networks (Ethernet and InfiniBand) for the interprocess communication and saving the result to the database. The database server

Table 2: Number and magnitude of absolute deviations Δ among node voltages and line currents (unit of Δ is V for voltages and A for currents).

Δ	Re $\{v_{\text{node}}\}$	Im$\{v_{\text{node}}\}$	Re$\{i_{\text{line}}\}$	Im$\{i_{\text{line}}\}$
$= 0$	37440	37431	35939	35987
$\in (0, 10^{-6}]$	0	6	50	2
$\in (10^{-6}, 10^{-5}]$	0	3	10	7
$\in (10^{-5}, 10^{-4}]$	0	0	1	4

is located on a resource different from the four computing nodes. Thereby, the simulation execution itself and the storage of result data are decoupled. The time required to initiate the savings to the database is regarded as a steady component of the simulation runtime and is therefore not considered here. Simulation results presented in the following section were obtained from a simulation setup without any result saving (except for runtime of the simulation) with no loss of generality.

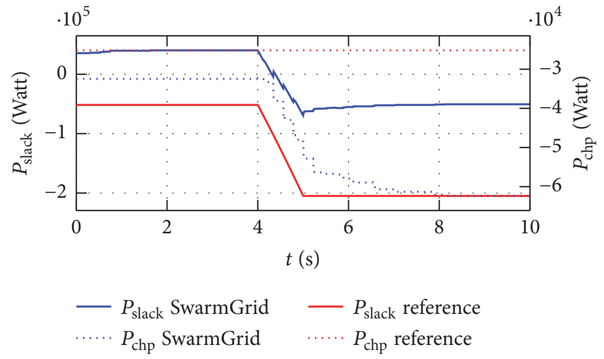


Figure 11: Active power behavior of slack bus and CHPs.

5. Evaluation

The methodologies defined in the previous section are used for an evaluation of DistAIX. The results are discussed in this section.

5.1. Correctness of Distributed Power-Flow Calculation. Table 2 provides the number of deviations Δ between node voltages and line currents of the two simulations in four intervals. The total number of current and voltage values differs because there are more busses than lines in the grid. As both simulations have an accuracy of 10^{-6}, smaller deviations are not discovered and are set to zero. Especially the difference between nodal voltages is small. While all Δ for node voltages are below or equal to 10^{-5} V for the line currents, some Δ are between 10^{-5} and 10^{-4} A. The reason for this is the application of the convergence criteria of the forward-backward sweep method to the node voltages. As the current is calculated by each component, this value can be considered the electrical interaction variable of agents. Hence, for a demonstration of correctness it is important to consider the line currents as well. For the imaginary part of the line currents only 4 values differ by more than 10^{-5} A. Experiments have been repeatedly carried out for different numbers of processes always yielding the same results. Therefore, the overall correctness of the implemented forward-backward sweep calculation is demonstrated and deviations to the reference simulation are negligible.

5.2. Observability of Emergent Behavior. Figure 11 shows the active power behavior of the whole grid at the slack bus for the reference simulation without a communicative control approach and for the SwarmGrid control approach. The different starting values of the two simulation results are caused by the way flexibility is used in the SwarmGrid approach. The starting conditions for the two simulations are identical. However, in SwarmGrid components negotiate immediately for a local balancing of production and consumption resulting in a smaller load at the slack bus compared to the reference case due to more efficient usage of storage. When the load starts to increase at $t = 4$ s the active power behavior at the slack bus in the reference simulation follows. The difference between the end and start value of active power at the slack bus is 153.2 kW, which corresponds to the change of load power (140 kW) plus losses in the grid due to higher currents. Therefore, the behavior of the whole grid in the reference simulation can be determined from the behavior of the single components, namely, in this case the single family household loads.

The difference between the end and start value in the SwarmGrid simulation is 90.1 kW which equals only 64.4% of the change of load power. Moreover, the shape of the active power at the slack bus indicates that the overall grid behavior does not follow the behavior of the loads. Instead, the change of load power is partly compensated. This is a dynamic process continuing after the single family household loads have reached their final value at 5 s. To find the cause for this behavior difference, the results for internal variables of single component types have to be investigated. Notice that the load change compensation is done by flexible components such as CHP. The total power production of all CHP is also depicted in Figure 11. In the reference case, the CHP determines the required power production according to the thermal demand, which is constant during the simulation time. In SwarmGrid control, the CHP utilizes thermal storage capacity to offer flexibility. As a result, the power production changes after the load power starts to increase.

Note that there is no centralized controller or optimization driving CHP towards this behavior. Instead, agents interact and determine their control values autonomously

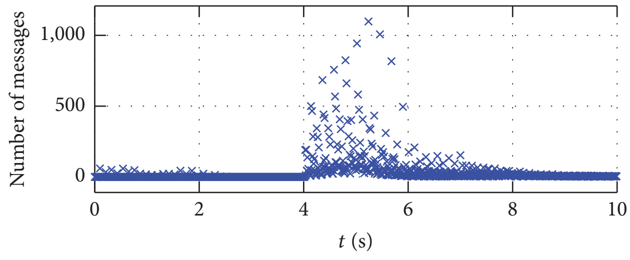


FIGURE 12: Number of messages sent between agents.

TABLE 3: Execution time results for different grid sizes.

SwarmGrid on/off	N_{LV}	p	$T(p)$ in sec
On	1	4	20.02
On	5	19	31.74
On	10	39	42.27
On	15	59	55.68
On	20	61	69.25
On	25	96	87.84
Off	1	3	10.66
Off	5	20	20.00
Off	10	39	27.58
Off	15	44	39.04
Off	20	61	51.27
Off	25	44	61.88

according to behavior rules. The overall system behavior, that is, the power behavior at the slack bus, emerges from the behavior of the single components and cannot be predetermined as in the reference case. Since the communication of agents plays a critical role in this process, Figure 12 shows the number of messages that are generated per time step. A large amount of interactions takes place during the change of load behavior and also afterwards. The number of messages decreases again when the system reaches a stable end value. The intensity of communication depends on the operating conditions and evolution of the physical system and can therefore not be anticipated precisely. Overall, these results demonstrate that emergent behaviors of distribution systems can be observed with DistAIX.

5.3. Scalability. The results of simulation benchmarks with respect to scalability are shown in Table 3. The minimal simulation execution time $T(p)$ scales almost linearly with the number of agents in the scenario for both cases with and without SwarmGrid control enabled. This result can also be observed in Figure 13 and is achieved by using more processes and spreading the simulation across multiple computing nodes; for example, for $N_{LV} = 10$ the simulation execution time is minimal for 39 processes. The simulation is spread across 2 computing nodes running 19 and 20 processes, respectively. The gradient of the $T(p)$ scaling factor is considerably lower than 1, indicating that the parallelization

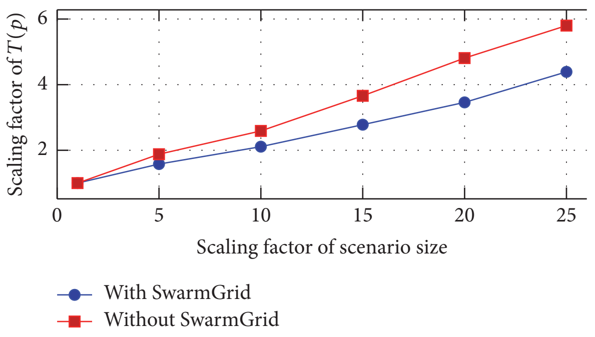


FIGURE 13: Scaling factors of scenario size in relation to scaling factors of minimal simulation execution time $T(p)$.

with processes is able to exploit computation resources efficiently for growing scenario sizes.

Linear scalability means that large electrical grids plus agent-based control can be simulated efficiently by an appropriate number of processes. It also indicates that the computation of communicative agent interactions adds an offset to the simulation execution time but does not change the scalability behavior in essence. Another aspect taken from Figure 13 is that the scaling for the case with communication is better than without. Keeping in mind that the benchmark scenarios were chosen in such a way as to require an extensive amount of agent communication, this leads to the conclusion that the message router concept is an effective method to parallelize the computational burden of agent message processing. However, the results presented here are only valid for the specific agent-based control strategy under study [41]. The experiments have to be repeated for different strategies resulting in potentially different communication link usages of agents to draw more general conclusions.

In most cases, the number of processes p of the minimal execution time $T(p)$ correlates well with the number of feeders in the electrical grid. This demonstrates the benefits of the proposed agent distribution methodology (see Section 3.2) for the simulation execution time and scalability. As the chosen benchmarking methodology uses grids with repetitive electrical topology with only a few workitems compared to the number of agents, it provides a worst case scenario for agent distribution. In more realistic grids of similar size containing more bifurcation nodes as considered here, the number of workitems will increase and the size (length) of each workitem will decrease. Hence, the execution time for a grid with a similar number of agents but more branched electrical topology can be expected to be equal to or smaller than the presented results.

Figure 14 shows the decrease of execution time with increasing number of processes for the $N_{LV} = 10$ scenario. The execution time decreases rapidly up to 10 processes and is close to the minimal value afterwards. DistAIX enables an efficient simulation even if less than the optimal number of processes for the given electrical topology is available. We observed this behavior for other scenario sizes as well. Future investigations on how to recommend a reasonable minimal number of processes to the user for a given electrical topology

Table 4: Execution time results for large grid sizes generated with 96 processes on 4 computational nodes.

SwarmGrid on/off	N_{LV}	Grid nodes	N_{agents}	$T(p)$ in sec
On	50	8,850	24,301	196.69
On	100	17,700	48,601	481.89
On	250	44,250	121,501	2019.86
Off	50	8,850	24,301	155.78
Off	100	17,700	48,601	401.05
Off	250	44,250	121,501	1824.61

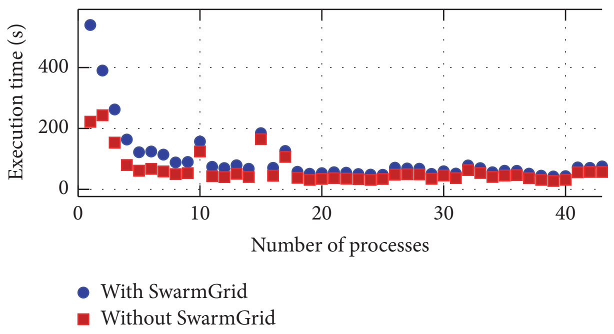


Figure 14: Execution time results of scenario with $N_{LVgrid} = 10$.

are necessary to exploit computation resources even more efficiently.

The results include only scenarios in which agents are assigned to all processes. Experiments with small testing scenarios such as the 27-node low voltage grid used for power-flow demonstrations showed that the influence of dispensable processes on the simulation execution time is negligible. Due to limitations of the hardware used for the benchmark simulations, only scenarios up to $N_{LV} = 25$ can be reasonably analyzed here with respect to scaling. The results for three additional scenarios are shown in Table 4. They demonstrate that even for grids exceeding the 40,000 nodes the presented approach provides a solution in a reasonable amount of time—even if not the optimal number of processes but only a maximum of 96 processes is available. Due to the memory limits of our computing nodes we do not provide results for larger scenarios here. It should be noted that the increase in execution time for the large scenarios is mainly caused by initialization procedures for the communication network setup which are specific for the control strategy used here and can be improved in the future.

6. Conclusion

This paper introduces DistAIX, an agent-based modeling and simulation approach for electrical distribution systems. It is designed to study emergent behavior of such systems at large scale. The simulation is parallelized by processes enabling the distribution of computations to multiple computing nodes with distributed memory. The distribution system components are modeled as agents in the required level of detail. No model simplifications or restrictions are needed for scalability. The properties of communication links between agents, such as latencies, can be modeled. DistAIX facilitates the design and analysis of agent-based bottom-up control concepts for distribution systems. Due to nonlinear cyber-physical interactions of the agents, such concepts may result in emergent system behavior. The emergent behavior can be observed in simulations with DistAIX. Process-based parallelization and linear scalability properties of DistAIX enable the study of models at nation scale if appropriate computation resources are used.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the German Federal Ministry of Education and Research in the project SwarmGrid (Grant no. 03EK3568A).

References

[1] C. E. Hmelo-Silver and R. Azevedo, "Understanding complex systems: Some core challenges," *Journal of the Learning Sciences*, vol. 15, no. 1, pp. 53–61, 2006.

[2] T. El-Mezyani, R. Wilson, M. Sattler, S. K. Srivastava, C. S. Edrington, and D. A. Cartes, "Quantification of complexity of power electronics based systems," *IET Electrical Systems in Transportation*, vol. 2, no. 4, pp. 211–222, 2012.

[3] S. Ikeda and R. Ooka, "Metaheuristic optimization methods for a comprehensive operating schedule of battery, thermal energy storage, and heat source in a building energy system," *Applied Energy*, vol. 151, pp. 192–205, 2015.

[4] W. K. V. Chan, Y.-J. Son, and C. M. Macal, "Agent-based simulation tutorial - Simulation of emergent behavior and differences between agent-based simulation and discrete-event simulation," in *Proceedings of the 2010 43rd Winter Simulation Conference, WSC'10*, pp. 135–150, USA, December 2010.

[5] P. Ringler, D. Keles, and W. Fichtner, "Agent-based modelling and simulation of smart electricity grids and markets - A literature review," *Renewable & Sustainable Energy Reviews*, vol. 57, pp. 205–215, 2016.

[6] A. Pokahr, L. Braubach, and W. Lamersdorf, "Jadex: A BDI Reasoning Engine," in *Multi-Agent Programming*, vol. 15 of *Multiagent Systems, Artificial Societies, and Simulated Organizations*, pp. 149–174, Springer US, Boston, Ma, 2005.

[7] D. P. Chassin, J. C. Fuller, and N. Djilali, "GridLAB-D: An agent-based simulation framework for smart grids," *Journal of Applied Mathematics*, vol. 2014, Article ID 492320, 2014.

[8] S. Jin and D. P. Chassin, "Thread group multithreading: Accelerating the computation of an agent-based power system modeling and simulation tool - GridLAB-D," in *Proceedings of the 47th Hawaii International Conference on System Sciences, HICSS 2014*, pp. 2536–2545, USA, January 2014.

[9] A. Tanenbaum and H. Bos, *Modern Operating Systems, , Pearson*, 4th edition, 2015.

[10] C. Kuschel and U. Rüde, "High-performance simulation of nation-sized smart grids," *International Journal of Parallel, Emergent and Distributed Systems*, vol. 32, no. 6, pp. 647–668, 2016.

[11] P. Oliveira, T. Pinto, H. Morais, and Z. Vale, "MASGriP a multi-agent smart grid simulation platform," in *Proceedings of the 2012 IEEE Power and Energy Society General Meeting, PES 2012*, USA, July 2012.

[12] J. C. Fuller, S. Ciraci, J. A. Daily, A. R. Fisher, and M. Hauer, "Communication simulations for power system applications," in *Proceedings of the 2013 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems, MSCPES 2013*, USA, May 2013.

[13] M. Stifter, E. Widl, F. Andren, A. Elsheikh, T. Strasser, and P. Palensky, "Co-simulation of components, controls and power systems based on open source software," in *Proceedings of the 2013 IEEE Power and Energy Society General Meeting, PES 2013*, Canada, July 2013.

[14] K. Anderson, J. Du, A. Narayan, and A. E. Gamal, "GridSpice: A distributed simulation platform for the smart grid," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2354–2363, 2014.

[15] R. Bottura and A. Borghetti, "Simulation of the volt/var control in distribution feeders by means of a networked multiagent system," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2340–2353, 2014.

[16] J. Vaubourg, Y. Presse, B. Camus et al., "Multi-agent Multi-Model Simulation of Smart Grids in the MS4SG Project," in *Advances in Practical Applications of Agents, Multi-Agent Systems, and Sustainability: The PAAMS Collection*, vol. 9086 of *Lecture Notes in Computer Science*, pp. 240–251, Springer International Publishing, Cham, 2015.

[17] M. Cvetkovic and M. Ilic, "Interaction variables for distributed numerical integration of nonlinear power system dynamics," in *Proceedings of the 53rd Annual Allerton Conference on Communication, Control, and Computing, Allerton 2015*, pp. 560–566, USA, October 2015.

[18] K. Mets, J. A. Ojea, and C. Develder, "Combining Power and Communication Network Simulation for Cost-Effective Smart Grid Analysis," *IEEE Communications Surveys Tutorials*, vol. 16, no. 3, pp. 1771–1796, 2014.

[19] D. Shirmohammadi, H. W. Hong, A. Semlyen, and G. X. Luo, "A compensation-based power flow method for weakly meshed distribution and transmission networks," *IEEE Transactions on Power Systems*, vol. 3, no. 2, pp. 753–762, 1988.

[20] Z. Haibo, Z. Boming, S. Hongbin, and A. Ran, "A new distributed power flow algorithm between multi-control-centers based on asynchronous iteration," in *Proceedings of the 2006 International Conference on Power System Technology*, pp. 1–7, Chongqing, China, October 2006.

[21] H. Sun and B. Zhang, "Distributed power flow calculation for whole networks including transmission and distribution," in *Proceedings of the Transmission and Distribution Exposition Conference: 2008 IEEE PES Powering Toward the Future, PIMS 2008*, USA, April 2008.

[22] J. M. Gonzalez de Durana, O. Barambones, E. Kremers, and L. Varga, "Agent based modeling of energy networks," *Energy Conversion and Management*, vol. 82, pp. 308–319, 2014.

[23] J.-H. Byun, A. Ravindran, A. Mukherjee, B. Joshi, and D. Chassin, "Accelerating the gauss-seidel power flow solver on a high performance reconfigurable computer," in *Proceedings of the IEEE Symposium on Field Programmable Custom Computing Machines, FCCM 2009*, pp. 227–230, USA, April 2009.

[24] C. P. Nguyen and A. J. Flueck, "A Novel Agent-Based Distributed Power Flow Solver for Smart Grids," *IEEE Transactions on Smart Grid*, vol. 6, no. 3, pp. 1261–1270, 2015.

[25] X. Zhang, A. J. Flueck, and C. P. Nguyen, "Agent-Based Distributed Volt/Var Control with Distributed Power Flow Solver in Smart Grid," *IEEE Transactions on Smart Grid*, vol. 7, no. 2, pp. 600–607, 2016.

[26] M. Wolter, H. Guercke, T. Isermann, and L. Hofmann, "Multi-agent based distributed power flow calculation," in *Proceedings of the IEEE PES General Meeting, PES 2010*, USA, July 2010.

[27] R. D. Zimmerman, *Comprehensive distribution power flow: modeling, formulation, solution algorithms and analysis [Ph.D. thesis]*, Cornell University, 1995, https://pdfs.semanticscholar.org/a1b6/eb1871701538ce2147523e8753a218ff0894.pdf.

[28] A. Rousset, B. Herrmann, C. Lang, and L. Philippe, "A survey on parallel and distributed multi-agent systems for high performance computing simulations," *Computer Science Review*, vol. 22, pp. 27–46, 2016.

[29] S. Coakley, M. Gheorghe, M. Holcombe, S. Chin, D. Worth, and C. Greenough, "Exploitation of high performance computing in the FLAME agent-based simulation framework," in *Proceedings of the 14th IEEE International Conference on High Performance Computing and Communications, HPCC-2012 - 9th IEEE International Conference on Embedded Software and Systems, ICESS-2012*, pp. 538–545, gbr, June 2012.

[30] N. Collier and M. North, "Parallel agent-based simulation with Repast for High Performance Computing," *Simulation*, vol. 89, no. 10, pp. 1215–1235, 2013.

[31] D. P. Chassin, K. Schneider, and C. Gerkensmeyer, "GridLAB-D: An open-source power systems modeling and simulation environment," in *Proceedings of the Transmission and Distribution Exposition Conference: 2008 IEEE PES Powering Toward the Future, PIMS 2008*, USA, April 2008.

[32] J. S. van der Veen, B. van der Waaij, and R. J. Meijer, "Sensor data storage performance: SQL or NoSQL, physical or virtual," in *Proceedings of the IEEE 5th International Conference on Cloud Computing (CLOUD '12)*, pp. 431–438, IEEE, June 2012.

[33] T. Rabl, M. Sadoghi, H.-A. Jacobsen, S. Gómez-Villamor, V. Muntés-Mulero, and S. Mankovskii, "Solving big data challenges for enterprise application performance management," pp. 1724–1735.

[34] K. Varda, "Protocol Buffers: Google's Data Interchange Format," 2008, https://opensource.googleblog.com/2008/07/protocol-buffers-googles-data.html.

[35] J. Feng and J. Li, "Google protocol buffers research and application in online game," in *Proceedings of the 2013 IEEE Conference Anthology*, pp. 1–4, China, January 2013.

[36] S. Popić, D. Pezer, B. Mrazovac, and N. Teslić, "Performance evaluation of using Protocol Buffers in the Internet of Things communication: Protobuf vs. JSON/BSON comparison with a

focus on transportation's IoT," in *Proceedings of the 1st IEEE International Conference on Smart Systems and Technologies, SST 2016*, pp. 261–265, Croatia, October 2016.

[37] T. C. Hu, "Parallel sequencing and assembly line problems," *Operations Research*, vol. 9, pp. 841–848, 1961.

[38] W. Zhou, H. Yang, and Z. Fang, "A novel model for photovoltaic array performance prediction," *Applied Energy*, vol. 84, no. 12, pp. 1187–1198, 2007.

[39] M.-R. Haghifam and S. Soltani, "Reliability models for wind farms in generation system planning," in *Proceedings of the 2010 IEEE 11th International Conference on Probabilistic Methods Applied to Power Systems, PMAPS 2010*, pp. 436–441, Singapore, June 2010.

[40] L. Razik, M. Mirz, D. Knibbe, S. Lankes, and A. Monti, "Automated deserializer generation from CIM ontologies: CIM++—an easy-to-use and automated adaptable open-source library for object deserialization in C++ from documents based on user-specified UML models following the Common Information Model (CIM) standards for the energy sector," *Computer Science - Research and Development*, pp. 1–11, 2017.

[41] S. Kolen, T. Isermann, S. Dähling, and A. Monti, "Swarm behavior for distribution grid control," in *Proceedings of the 2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, pp. 1–6, Torino, Italy, September 2017.

[42] W. K. Chan, "Interaction metric of emergent behaviors in agent-based simulation," in *Proceedings of the 2011 Winter Simulation Conference - (WSC 2011)*, pp. 357–368, 2011.

[43] C. Szabo and Y. M. Teo, "An integrated approach for the validation of emergence in component-based simulation models," in *Proceedings of the Winter Simulation Conference*, pp. 2412–2423.

[44] E. O'Toole, V. Nallur, and S. Clarke, "Towards Decentralised Detection of Emergence in Complex Adaptive Systems," in *Proceedings of the 2014 8th IEEE International Conference on Self-Adaptive and Self-Organizing Systems, SASO 2014*, pp. 60–69, UK, September 2014.

[45] C. Clauss, T. Moschny, and N. Eicker, "Dynamic Process Management with Allocation-internal Co-Scheduling towards Interactive Supercomputing," in *Proceedings of the 1st COSH Workshop on Co-Scheduling of HPC Applications, 2016*.