Data Management in the MIRABEL Smart Grid System

Matthias Boehm* [1], Lars Dannecker[2], Andreas Doms[2], Erik Dovgan[3], Bogdan Filipič[3], Ulrike Fischer[1], Wolfgang Lehner[1], Torben Bach Pedersen[4], Yoann Pitarch[4], Laurynas Šikšnys[4], Tea Tušar[3]

[1] Dresden University of Technology, Database Technology Group, Germany
[2] SAP Research, SAP AG, Germany
[3] Jožef Stefan Institute, Department of Intelligent Systems, Slovenia
[4] Aalborg University, Center for Data-intensive Systems, Denmark

ABSTRACT

Nowadays, Renewable Energy Sources (RES) are attracting more and more interest. Thus, many countries aim to increase the share of green energy and have to face with several challenges (e.g., balancing, storage, pricing). In this paper, we address the balancing challenge and present the MIRABEL project which aims to prototype an Energy Data Management System (EDMS) which takes benefit of flexibilities to efficiently balance energy demand and supply. The EDMS consists of millions of heterogeneous nodes that each incorporates advanced components (e.g., aggregation, forecasting, scheduling, negotiation). We describe each of these components and their interaction. Preliminary experimental results confirm the feasibility of our EDMS.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

1. INTRODUCTION

Energy demand is increasing rapidly worldwide. Fossil fuels are a problematic way of producing energy due to greenhouse gas emissions and potential exhaustion of oil supplies, while nuclear energy is risky. Instead, renewable energy sources (RES) such as wind, waves and solar power is seen as the promising sustainable alternative.

Thus, many countries aim to increase the share of energy from RES such as wind turbines and solar panels. However, the integration of renewable energy is challenging as production from RES highly depends on weather conditions and thus cannot be planned.

In order to facilitate the more efficient utilization of an intermittent RES supply, the EU's 7^{th} Framework project *MIRABEL (Micro-Request-Based Aggregation, Forecasting and Scheduling of Energy Demand, Supply and Distribution)* designs and prototypes an Energy Data Management System (EDMS). Particularly, this project copes with the problem of RES by balancing supply and demand with the help of flexibilities. Indeed, flexible demand (e.g., the usage of a washing machine or charging an electric vehicle) can often be shifted to the time when production from RES is available. Conversely, non-flexible demand (e.g., the usage of lights, TV, or a cooking stove) must be satisfied at the time when it is demanded. Figure 1 visualizes situations before (top graph) and after (bottom graph) the MIRABEL system balances demand and RES supply. Here, the solid gray and shaded areas depict non-flexible and flexible demand, aggregated from hundreds of consumers. The EDMS can utilize production from RES (dashed line) more efficiently by shifting flexible demand in time (bottom graph). Interestingly, the EDMS also contributes to reduce peak demand by planing energy flows in a physical grid based on the automatic scheduling of energy demands from millions of consumers.

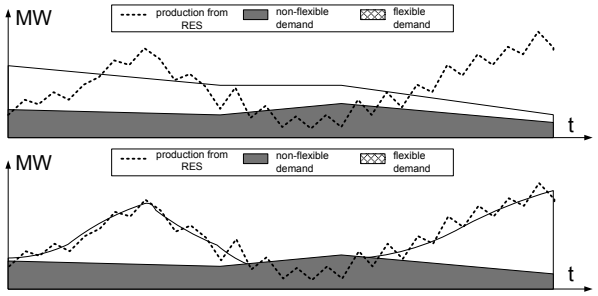


Figure 1: Balancing the consumption and the RES production

From an architectural point of view, the EDMS consists of millions of homogeneous nodes that are organized hierarchically to reflect the harmonized model of the European electricity market [4] (Figure 2). Highest level nodes are used by electricity network operators (TSOs – transmission system operators), middle level nodes – by traders (balance-responsible parties – BRPs), and lowest level nodes – by prosumers (entities that are consumers and/or producers). We anticipate that the EDMS will eventually span over the entire Europe, thus encompassing millions of nodes.

When trying to balance energy supply and demand ef-

*The author is currently visiting IBM Almaden Research Center, San Jose, CA, USA.

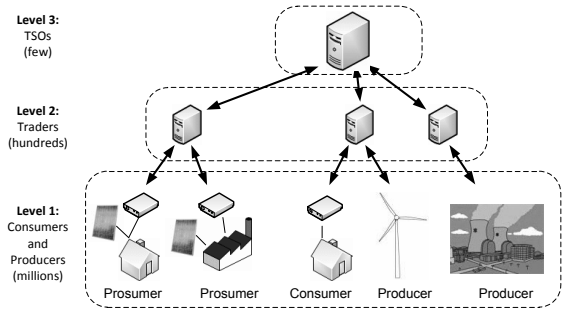


Figure 2: Architecture of the EDMS

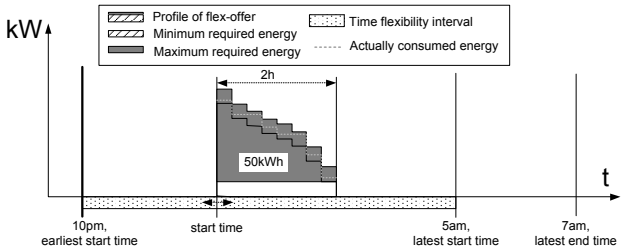


Figure 3: Flex-offer for charging the car's battery

fectively, a number of general data management challenges arise. These include managing very large-scale wide-area distributed systems, providing high availability and fault tolerance, supporting near-realtime data synchronization and integration, and enabling advanced analytics. Some of these challenges are inherently addressed in our system. Indeed, even in critical scenarios (e.g., nodes unreachable, failed execution deadlines) the overall system would gracefully behave as in the traditional setting because pending flexibilities simply timeout and customers fall back to the open contract. Furthermore, specific research challenges are found within data aggregation, forecasting, scheduling, and negotiation. In this paper, initial results about the EDMS architecture are presented through the following contributions. First, we provide an efficient technique to aggregate flex-offers that preserves as much flexibilities as possible. Second, an accurate and efficient forecasting technique is proposed since it is a fundamental precondition to the overall quality of the system. Third, a scheduling model for balancing energy demand and supply is provided. Fourth, a negotiation module is defined to find an agreement between the prosumers and its BRP about the price for flex-offers. Finally, we describe how these components interact and provide preliminary experimental results to validate the feasibility of the EDMS.

Many other projects work on balancing energy demand and supply, including MEREGIO (www.meregio.de), FENIX (www.fenix-project.org), EU DEEP (www.eu-deep.com), ADDRESS (www.addressfp7.org), EDISON (www.edison-net.dk), MORE MICROGRIDS (www.microgrids.eu), and EcoGrid EU (www.eu-ecogrid.net). Overall, these projects tend to be focused on quite specific demand types such as heat pumps or electric vehicles, or on specific ways of controlling devices. The MIRABEL approach to flexibility is able to generalize and combine these more specific approaches. Indeed, a major strength of the MIRABEL approach is that it is able to accommodate all forms of both flexible demand, e.g., heat pumps, dishwashers, washing machines, freezers, and supply, e.g., from private solar panels, in a completely general way. Additionally, the flexible demand and supply of not just large, but also small and medium-size, industrial prosumers can easily be handled. The solutions developed in MIRABEL thus have an impact far beyond the project itself.

The remainder of the paper is structured as follows. Section 2 introduces the MIRABEL use scenario. Section 3 gives an overview of a node architecture. Section 4 introduces the problem of aggregating flex-offers and outlines our solution. Section 5 describes the challenges and solutions for forecasting energy time series. Section 6 discusses the MIRABEL scheduling approach, while Section 7 outlines the negotiation approach. Section 8 discusses the interaction and interdependencies between the individual components, and Section 9 presents initial experimental results. Finally, Section 10 concludes the paper.

2. MIRABEL USE SCENARIO

The following sequence illustrates a typical flow of events within the MIRABEL system.

Step 1. A consumer arrives home at 10pm and wants to recharge the electric car's battery at lowest possible price by the next morning. Once plugged in, the system recognizes the vehicle and chooses a default energy consumption profile, which, among other information, defines consumption completion time at 7am.

Step 2. The prosumer's node (level 1) generates an energy planning object, called *flex-offer* (Figure 3). The solid gray and shaded area represents the *profile* and the dotted area at the bottom shows the time flexibility interval - *earliest start time* is 10pm and *latest start time* is 5am to finish consumption by 7am.

Step 3. Based on weather forecasts, the trader's node (level 2) schedules the flex-offer to start energy consumption at 3am and sends back a message to the prosumer's node.

Step 4. The consumer's node of EDMS starts supplying energy to the electric vehicle at 3am. Actually consumed energy is visualized with the gray dashed line in Figure 3. The car's battery is fully charged at 5am.

The trader's node collects such flex-offers from millions of consumers. Then, it aggregates these (micro) flex-offers into larger ones, called *macro flex-offers*. These - define larger energy quantities in their energy profiles due to the aggregation. Later, the aggregated flex-offers are scheduled so that they match supply from RES thus minimizing imbalances in the part of electricity network, for which the trader has balance responsibility. Such scheduled flex-offers are disaggregated into *micro scheduled flex-offers* and finally reported back to the consumers. A consumer is given a discount for energy if she provides flexibilities using flex-offers. Producers can also issue flex-offers. These are treaded equivalently to flex-offers for consumption.

The aggregated flex-offers are used to buy and sell energy at the energy market (via the market operator or directly with other traders). This means that the process is essentially repeated at a higher level: the aggregated flex-offers are sent to a TSO's node (level 3) for further aggregation, scheduling, and disaggregation. When the TSO's node for-

wards back scheduled flex-offers to the trader, they are disaggregated and reported back to respective prosumers in the same way as locally managed flex-offers.

3. NODE ARCHITECTURE OVERVIEW

In the global MIRABEL architecture, nodes represent the atomic entity and has to be designed with great care to guarantee good overall performances. In the rest of the paper, we thus focus on the description of Local Energy Data Management System (LEDMS) running on each of the MIRABEL homogeneous nodes. We start by providing an overview of the LEDMS architecture. Details on key components are given in the next sections.

The Control component is the central component orchestrating all other components and processes within the node. The Communication component is responsible for exchanging messages (flex-offers, supply and demand measurements, forecasts, etc.) between the current and other LEDMSs nodes. The Forecasting component is responsible for forecasting the expected demand and supply within the electricity grid based on historical measurements. In the LEDMS, these forecasts primarily serve as input to the flex-offers scheduling, handled by the Scheduling component. The primary task of the Scheduling component is to balance supply and demand in the relevant part of the electricity grid, based on flexibilities provided by flex-offers. It is infeasible for the Scheduling component to schedule the millions of micro flex-offers individually. Instead, similar micro flex-offers are aggregated into larger macro flex-offers that can then be scheduled within reasonable time. The aggregation is done by the Aggregation component. The Negotiation component handles the pricing of individual scheduled flex-offers, and the contracting between two users. Physical users can interact with LEDMS, set parameters, and analyze the data through the User Interface component. Finally, all historical and current time demand/supply, forecasting model parameters, flex-offers, price and contracts are stored and managed by the Data Management component. Thus, it plays a major role since it interacts with most of LEDMS component. To meet the above mentioned data management requirements, data are persistently stored using a multidimensional schema [6] that can be seen as a combination of star and snowflake schemas. This single, unified schema is flexible enough to support actors at all levels, some of which only use subparts of the schema, *e.g.*, prosumers nodes do not make use of market area data.

4. AGGREGATION

As noted in Section 2, *Aggregation* inputs a large set ($> 10^6$ per day) of flex-offers (e.g., from prosumers) and outputs a substantially smaller set of aggregated flex-offers. *Disaggregation* transforms these into a set of scheduled (micro) flex-offers. This process must always satisfy the:

Disaggregation requirement. It should be always possible to correctly disaggregate scheduled flex-offers. Aggregated scheduled flex-offers can always be converted into (non-aggregated) scheduled flex-offers while respecting the initial flex-offer constraints. If not, the macro-level schedule does not correspond to the micro-level one, and it is impossible to balance energy supply and demand at the micro-level.

Additionally, the process should try to meet a trade-off between the following three conflicting requirements:

Compression requirement. The number of aggregated flex-offers should be as small as possible to reduce scheduling time.

Flexibility requirement. The loss of flexibility in the aggregation should be as small as possible. When aggregating two or more flex-offers into a single one, either energy flexibility (the ability to scale energy up or down at a given time) or time flexibility (the ability to shift energy use/production in time) is often lost because of the many possible flexibility combinations, only one of which can be chosen.

Efficiency requirement. Aggregation, scheduling, and disaggregation must complete within a given (short) time.

We now describe how aggregation and disaggregation were designed to satisfy these requirements.

Aggregation Component Overview

First, it is assumed that a set of flex-offers is always aggregated into a single aggregated flex-offer. All internal constraints of an aggregated flex-offer are conservatively produced so that (1) all profiles of the underlying flex-offers can always be shifted in the time flexibility range of the aggregated flex-offer; (2) energy values in the aggregated flex-offer profile are computed by summing the values from the underlying flex-offers profiles. Such approach satisfies the disaggregation requirement because the schedule that is produced using the aggregated flex-offers can always be disaggregated into an equivalent schedule with the non-aggregated flex-offers.

Second, in order to provide control over the compression factor and flexibility losses, a set of user-defined aggregation thresholds (e.g., *duration tolerance*, *start after tolerance*) is introduced. They allow determining similar flex-offers that can potentially be aggregated. Specifically, two flex-offers are allowed to be aggregated together only if their attribute values (e.g., *duration*, *start after time*) deviate by no more than user-specified thresholds. As shown in the experiment section, combinations of the aggregation parameter values allow controlling the flexibility loss and the preferred compression factor. Moreover, the aggregation parameters might not be sufficient when aggregating a large number of identical flex-offers. In such a case, all identical flex-offer will be aggregated into a single aggregated flex-offer thus losing the flexibility to schedule them individually. To prevent this, a so called *bin-packer* is designed. It allows to specify lower and upper bounds on one of the following aggregated flex-offer properties: (1) the number of flex-offers included into a single aggregate, (2) the amount of energy (or time flexibility) an aggregated flex-offer has to offer, etc. It should be noticed that this *bin-packer* is an optional feature and can be turned off.

Third, an incremental aggregation is supported. Therefore, aggregated flex-offers can be incrementally updated to avoid a *from-scratch* re-computation (an aggregation from scratch is also supported). Thus, a more efficient flex-offer aggregation can be achieved.

Research Results

Based on these design decisions, the aggregation component was implemented. It accepts a set of *flex-offer updates*, i.e., information about accepted or expiring flex-offers (those with approaching assignment before time), and produces a set of *aggregated flex-offer updates*, i.e., information about created, deleted, and changed aggregated flex-offers.

The aggregation component consists of the following 3 sub-components: (1) *group-builder*, (2) *bin-packer*, and (3) *n-to-1 aggregator*. These sub-components are chained so that provided flex-offer updates traverse them sequentially. First, flex-offer updates are accumulated within the *group-builder* until their further processing is invoked. Then, when invoked (e.g., by the control component. See Section 2), the *group-builder* internally maintains similar flex-offer groups and produces *group-updates*, i.e., information about created, deleted, and changed groups. Then, when the *bin-packer* receives those group-updates, it utilizes them to maintain so called sub-groups, i.e, bounds-satisfying groups of similar flex-offers, and to produce *sub-group updates*, i.e., information about created, deleted, and changed sub-groups. Finally, the produced sub-group updates are issued to the *n-to-1 aggregator*. This sub-component utilizes sub-group updates (or group-updates if the *bin-packer* is disabled) to maintain a set of aggregated flex-offers and to produce respective aggregated flex-offer updates. The disaggregation of flex-offers is also performed by the *n-to-1 aggregator*. Since our approach meets the *disaggregation requirement*, the disaggregation technique is quite straightforward and is hence not further detailed here.

Research Directions

The current solution satisfies the defined requirements but some challenges remain to be addressed. They are challenges related to the current aggregation component design and the aggregation of flex-offers in general.

First, it is a challenge to integrate the *bin-packer* with a *group-builder*, i.e., the component that partitions flex-offers into disjoint groups based on their similarity. In the current design, these two elements are independent and, therefore, the *group-builder* is unaware about the goals of the *bin-packer*. A better partitioning of flex-offer can be achieved if all flex-offers are partitioned in one iteration so that they suit better for the bin-packing. Performing the partitioning incrementally is a part of the challenge.

Second, it is a challenge to develop a flex-offer aggregation technique that simultaneously supports additional types of flexibility, e.g., price, energy interval duration, or power flexibilities. Third, a challenge is to find a more advanced flex-offer representation which, for the same compression ratio, can preserve flexibility of multiple prosumers with lower flexibility loss. Building aggregation techniques for such representation is a part of the challenge. Finally, it is a challenge to generalize flex-offer aggregation approaches into a multi-criteria grouping operator and a user defined aggregation operator for a relational database management system.

5. FORECASTING

Accurate and efficient forecasts of energy consumption and production are a fundamental precondition for dynamic and fine-grained scheduling. Based on forecasts, schedules for RES supply and demand are initially computed and afterwards incrementally maintained if forecast values change over time. Specific characteristics of energy time series like multi-seasonality (daily, weekly, annual) or dependency on external information like weather or calendar events motivate the employment of forecast models tailor-made for the energy domain. In addition, different forecast horizons (short-term, mid-term, long-term) as well as the forecasting of flex-offers have to be provided. Finally, forecasting faces the challenge of a large scale hierarchical system with high update rates of new measurements and evolving time series, which require continuous model evaluation and adaptation.

Overview Forecasting Approach

Our system architecture consists of two main components: (1) the transparent forecast model creation and usage and (2) the transparent forecast model update and maintenance. The model creation component automatically creates forecast models, either beforehand or when the respective forecasts are demanded, where we apply the *Engle, Granger, Ramanathan, and Vahid-Arraghi (EGRV) Model* [11] and the *Triple Seasonality Holt Winters (HWT) Model* [12]. The EGRV-Model is a multi-equation energy demand forecast model that uses an individual model for each intra-day period (e.g., one model for each hour). In addition, weather information, calendar events (e.g., holidays) and context knowledge of energy types (e.g., constraints on the produced energy) are included. If the EGRV model does not provide accurate results, we fall back to the alternative (more robust) HWT-Model, which is a energy specific adaptation of the general purpose Holt-Winters exponential smoothing forecast model. Besides forecasting traditional energy demand and supply, we provide the possibility of forecasting flex-offers. Flex-offers can be viewed as multi-variate time series that consists of a vector of observations (e.g., min power, max power) per time slice. To forecast flex-offers, we decompose this multi-variate time series into a set of univariate time series and apply our already defined forecast model types to the individual time series.

Model creation involves computationally expensive parameter estimation, where we reuse existing well-established local (e.g., Downhill-Simplex [8]) and global (e.g., Simulated Annealing [1]) parameter estimators. The scheduling component can explicitly request forecast values or may register forecast queries as continuous queries in order to obtain notifications whenever the forecast values change significantly. A continuous stream of new measurements require a continuous maintenance of forecast models. For each new time series value, we update our forecast models that consists of a simple update of smoothing constants or the shift of lagged input values. This implies low additional costs. Due to changing time series characteristics, the accuracy of the forecast models might be reduced over time, which poses the necessity of adapting the model parameters. To evaluate the need for a model adaptation, we offer different model evaluation strategies (e.g., time- or threshold-based). Furthermore, the model adaption exploits the context knowledge of previous model estimations in order to speed up this time-consuming process of parameter re-estimation.

Research Results

Furthermore, we briefly summarize selected research results that enhance and further specify the default MIRABEL forecasting approach introduced above. Forecasting always needs to cope with the trade off between forecast accuracy and runtime of parameter estimation. We offer different optimizations that address this challenge in terms of model creation on physical (parallelized model creation) and logical level (hierarchical forecasting), model usage (publish-subscribe forecast queries) and model maintenance (context-aware model adaption).

Parallelized Model Creation Energy-domain-specific

multi-equation forecast models (e.g., EGRV-Model) comprise a large number of parameters, for what reason the estimation of such models is time consuming. As multi-equation models consist of several independent individual models, we can reduce the time needed for estimating such models by partitioning and parallelization. Therefore, we horizontally partition the time series according to the multi-equation access pattern and parallelize the model estimation process according to the resulting independent data partitions.

Hierarchical Forecasting Based on the hierarchical organization of the energy market, the macroscopic system architecture is inherently distributed, where at each system node, one or several forecast models might be created and used according to the scope of the particular role. Beside the use of individual forecast models, forecast models can be used to aggregate or disaggregate forecast values without the need for individual models at each system node. Therefore, we provide an advisor component that computes for a given hierarchical structure a configuration of forecast models according to specified accuracy and runtime constraints [5].

Publish-Subscribe Forecast Queries The scheduling component does not always need or even not want to have the most up-to-date forecast values as every new forecast value triggers the computationally expensive maintenance of schedules. Only if forecast values change significantly, notifications are required. Therefore, in addition to requesting forecast values, we offer the interaction scheme of so-called publish-subscribe forecast queries. Hereby, our goal is to minimize the overall costs of the subscriber.

Context-Aware Model Adaptation The development of energy time series strongly depends on background processes and influences that together form the context of a time series. Observing these context information offers the possibility of storing previous models in conjunction to their corresponding context information within a repository to reuse them whenever a similar context reoccurs. This kind of case-based reasoning approach [2] achieves a higher forecast accuracy in less time, especially for complex models.

Research Directions

Our initial research results can be further extended and improved in terms of model creation, usage and maintenance. The creation time of models might not only be reduced by inter-model parallelizing, but also by intra-model parallelizing, i.e., parallel parameter estimation of one model. Our hierarchical forecasting approach can be further extended to continuously adapt the model configuration to changing time series characteristics and to globally optimize model parameters over several system nodes. The usage of models through publish-subscribe forecast queries might be further improved by including context information to specify the notification length. Finally, model maintenance should not only include the context for adaption but also for evaluation, e.g., to determine a dynamic error threshold.

6. SCHEDULING

Scheduling Component Overview

Each time there is a significant change in the forecasts or in the pool of aggregated flex-offers, the scheduling component is invoked. The scheduling component tries to find the best schedule for the given aggregated flex-offers by taking into account the forecast energy production and consumption and the possibility of selling energy to (and buying energy from) the market (other BRPs).

More specifically, scheduling consists of fixing start times and energy flexibilities of all given flex-offers and setting the amount of energy that will be sold to (and bought from) the market, while optimizing the total cost of the resulting schedule. The schedule cost is calculated as the sum of (1) costs of remaining mismatches, (2) costs of all given aggregated flex-offers and (3) costs of energy sold to (and bought from) the market. The lower the cost, the better the schedule. Only schedules that respect all flex-offer constraints are considered.

The MIRABEL scheduling problem differs from the scheduling problems treated in the literature either in the context of production systems [10] or energy sector [14]. Unlike the usually scheduled activities, flex-offers are structured. In addition to the start time, flex-offer and market energy amounts need to be determined, which substantially increases the problem complexity in terms of the number of candidate solutions. Furthermore, the objective function is not related to a time measure, but is rather a composed cost function. Finally, flex-offer constraints contribute to the problem specificity. These characteristics and the expected large number of flex-offers to be processed make the MIRABEL scheduling problem non-standard and highly complex.

Research Results

The biggest challenge of scheduling is to efficiently find a good approximation of the optimal schedule. When addressing this challenge, the following issues were faced.

Scheduling problem formulation. While scheduling in MIRABEL is an optimization problem with the objective of balancing energy supply and demand, the exact formulation of the schedule evaluation function deals with the schedules from the point of view of expenses for the BRP. Such a formulation allows us to weight the remaining mismatches according to their costs (mismatches at peak periods cost the BRP more than at other periods) and differentiate among flex-offers and among market energy with different prices.

Investigation of schedule optimality. When solving any optimization problem it is advantageous to know its optimal solution (so an assessment of the distance to the optimum can be computed). Because energy amounts can take on an infinite number of values and flex-offer energy constraints construct dependences among different intervals of a single flex-offer profile, an infinite number of possible solutions may exist and thus the optimal solution of this problem is generally not known. Only if a few flex-offers need to be scheduled or if there are no flex-offer energy constraints, it is possible to find the true optimum. In a preliminary experiment with 10 flex-offers without energy constraints it took almost three hours to explore all (almost 850 million) sensible solutions and find the optimal schedule.

Implementation of the scheduling algorithms. As known scheduling heuristics cannot be applied to this problem, we used two stochastic metaheuristic algorithms to solve it: randomized greedy search and an evolutionary algorithm. The *randomized greedy search* constructs the schedule gradually—at each step a randomly chosen flex-offer is scheduled in the best possible position. This is repeated until all flex-offers have been scheduled. While it is pos-

sible to schedule a single flex-offer in an optimal way, a sequence of such optimal placements does not produce an overall optimal schedule. Therefore, we also developed an *evolutionary algorithm* [3] that starts with a population of randomly created solutions and uses evolutionary principles of selection, crossover and mutation to find progressively better solutions. Results of an initial experiment with these two methods are presented in Section 9.

Research Directions

As shown by the experiment in Section 9, the number of flex-offers to be scheduled importantly influences the efficiency of the applied scheduling algorithms. However, the complexity of the search space heavily depends also on the start time flexibilities of the included flex-offers. As this influence was not researched in detail yet, it shall be explored in the future. Moreover, we will consider implementing and testing additional scheduling algorithms as well as hybridizing the existing ones to improve their efficiency.

7. NEGOTIATION

Each flex-offer potentially increases the profit of the BRP. He can avoid costs on the reserve energy market or trade capacities. Negotiation in MIRABEL finds an agreement between the prosumer and its BRP about the price for flex-offers. Depending on the business strategy of the BRP various price setting schemes are possible:

Monetize Flexibility

Potential cost savings and trading opportunities result from different flexibilities offered by the prosumer:

Assignment flexibility is the time left for re-scheduling a flex-offer. The BRP needs a minimum of time to process a flex-offer. Any Assignment flexibility that exceeds the time until the next trading period of the day-ahead market is marginalized by the option for the BRP to trade on the day-ahead market.

Scheduling flexibility is the time range within a flex-offer can be scheduled. If the *earliest start time* and *latest start time*, parameter, see figure 3, of a flex-offer are equal there is no Scheduling flexibility for the BRP. Such a flex-offer may still provide a benefit for the BRP if it offers Energy flexibility.

Energy flexibility is the amount of energy which is dispatchable by the BRP. The Energy flexibility per time period must be above zero and the grid capacity. Any other parameters have no additional value for the BRP.

Each of the described flexibility parameters can be normalized to flexibility potentials by applying a function, e.g. the sigmoid function, that maps the flexibility parameter to value between 0 and 1. The total value of each flex-offer is the weighted sum of its flexibility potentials and can be computed before execution time.

Share Realized Profit

An alternative price setting scheme takes the context of an executed flex-offer into account. As a consequence the value of a flex-offer can only be computed after the execution time. Here the BRP calculates the realized profit that this flex-offer has generated and shares it with the Prosumer.

The advantage over a price setting before execution time is that incentives for the Prosumer are based on the realized value for the BRP. Any price setting after execution time can not be used as an acceptance criteria, in contrast to the price setting described before.

Flex-Offer Acceptance

Before taking a flex-offer into account the BRP has to decide whether it is potentially profitable. The BRP must be able to reject a flex-offer that generate loss or can not be processed in time. It is important to note that the rejection of a flex-offer does not imply that the Prosumer is not allowed to produce or consume the energy based on his tariff. The BRP just waives the option to control the load in the energy grid.

Research Directions

Future research will evaluate price setting strategies for the BRP. Due to the complexity of the planning and the large number of flex-offers it is necessary to develop better heuristics to estimate the value of individual flex-offers before execution time.

8. COMPONENT INTERPLAY

Interaction of Aggregation and Scheduling

The first major interaction between the components is between aggregation and scheduling. Here, two major concerns, time and flexibility loss, must be balanced. The total time spent on these two tasks is very important to fit within the available time window. As explain earlier, scheduling is computationally heavy. Thus, aggregation is first used to reduce the number of flex-offers substantially. The aggregation parameters must thus be set to achieve a sufficiently high compression ratio. As seen in the next section, more aggressive aggregation (higher compression ratio) will take somewhat more time, but this is (much) more than offset by the savings in scheduling time. However, more aggressive aggregation will often lead to a considerably higher loss of flexibility in the aggregate result, in comparison to the flexibility in the original flex-offers, unless the flex-offers are very similar. This leads to an interesting two-dimensional optimization problem: how do we choose the best aggregation result size (number of aggregated flex-offers), and the corresponding aggregation parameters, to preserve as much as possible of the flexibility, while still keeping the overall run time within the limits?

Interaction of Forecasting and Scheduling

The second major interaction can be found between forecasting and scheduling. First, the time spent on parameter estimation as well as maintenance parameters influence forecast accuracy and thus scheduling results. As shown in the next section, the more time we spent on parameter estimation the higher the resulting forecast accuracy (until convergence). In addition, model maintenance parameters (e.g., when to trigger parameter re-estimation) influence maintenance time and forecast accuracy as well. Second, the forecast horizon, i.e., the number of values provided to scheduling, has a major impact on scheduling time and costs. A higher forecast horizon allows scheduling to plan for a longer time horizon (and maybe achieve lower costs) but might also require rescheduling. As shown in the next section, the higher the forecast horizon the lower the fore-

cast accuracy. However, with each new measurement the forecast models can be maintained and accuracy can be improved. Thus, if new forecast values significantly differ from previous ones, we need to re-execute the computationally expensive scheduling component. In contrast, a low forecast horizon achieves a higher accuracy but requires constantly restarting the scheduling component to plan appropriately. This implies higher scheduling time as well. We address this trade off with our concept of publish-subscribe forecast queries explained in Section 5.

Global Distribution of Time

Following these considerations, the distribution of time between the three components strongly influences the imbalance costs and has to be set accordingly. Higher aggregation time might allow higher compression ratios, higher forecasting time might allow higher forecast accuracy and higher scheduling time might allow lower total costs. The detailed time assignment is challenging and depends on many factors like the desired flexibility loss and forecast accuracy as well as the convergence characteristics of scheduling and forecasting algorithms. However, in a parallel setting, we do not need to exactly assign the available time to all components, but we can exploit asynchronous approaches. For example, forecast models might already start maintenance even if production and consumption measurements are not up-to-date yet, accepting a slightly lower accuracy. Finally, the global time consumption obviously impacts on the reactivity of the system. Indeed, the smaller the time to perform aggregation, scheduling and disaggregation, the more last-minute generated flex-offers could be considered in time. This point is of crucial importance since we aim at designing a highly reactive EDMS. Experiment results reported in the next section show that our objective is realistic.

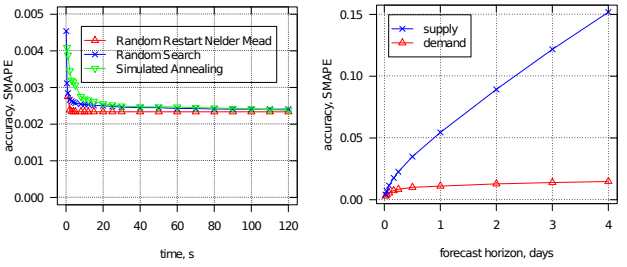
9. EXPERIMENTAL RESULTS

The previous section was dedicated to describe the component interactions and ended up by discussing timing aspects. We now present some experimental results to support this discussion. All the experiments were run on a standard PC.

Aggregation Experiments. An experiment was performed to evaluate the aggregation component in term of the compression, efficiency, and flexibility loss. We used a flex-offer dataset with around 800000 artificially generated flex-offers. Only flex-offer inserts and no deletes were used in the experiment. The *bin-packer* was disabled. Two aggregation parameters and four different their value combinations were used in the experiment. A combination $P0$ ensures that *Start After Time* and *Time Flexibility* values are equal for all flex-offers being aggregated together. A combination $P1$ allows the small variation of *Time Flexibility* attribute values, but requires identical *Start After Time* values. On contrary, a combination $P2$ allows the small variation of *Start After Time* values, but requires identical *Time Flexibility* values. And finally, a combination $P3$ allows the small variations of both attribute values.

Results of the experiment are depicted in Figure 5(a-d). As it is seen in the figures, different aggregation parameter values lead to different compression ratios, aggregation times, and time flexibility losses. The combination $P0$ offers no time flexibility losses, leads to an efficient aggregation, but does not yield a good compression ratio (it is still above 4). The $P1$ leads to a better compression ratio, efficient aggregation, and increased time flexibility loss, which occurs due to the allowed variations of *Time Flexibility* values. The $P2$ offers a very good compression ratio, low time flexibility loss, but results in slower aggregation. This can be explained by the need to traverse flex-offer energy profiles with increased number of intervals every time a new flex-offer has to be aggregated. Finally, $P3$ results into an increased flexibility loss and a worse aggregation performance, but offers a good compression ratio. Moreover, from Figure 5(d), it can be seen that the disaggregation is approx. 3 times faster than aggregation regardless of the flex-offer count and aggregation parameter settings.

Forecasting Experiments. In our first experiment, we compared the error development of three important global search algorithms that are used in our forecasting component for an initial parameter estimation. The experiment was conducted using the Holt-Winters Triple Seasonal Exponential Smoothing (HWT), a forecast model tailor-made for the energy domain [13]. We performed our tests on the publicly available UK energy demand dataset from UK NationalGrid [7]. As it can be seen in Figure 4(a) all algorithms converge to a result having similar accuracy, with Random Restart Nelder Mead having a slight advantage. Overall Random Restart Nelder Mead also slightly beats both other algorithms namely Simulated Annealing and Random Search in the error development over time. For this reason, we employ Random Restart Nelder Mead as our main global search algorithm, when estimating forecast model parameters from scratch.

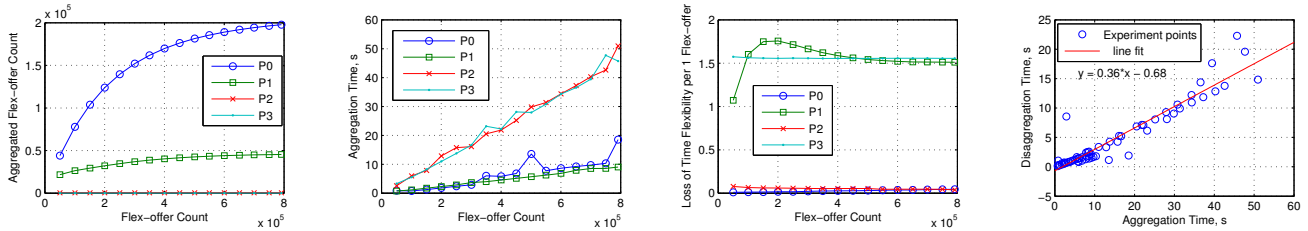


(a) Accuracy vs. Efficiency (b) Accuracy vs. Forecast Horizon

Figure 4: Results of Forecasting Experiments.

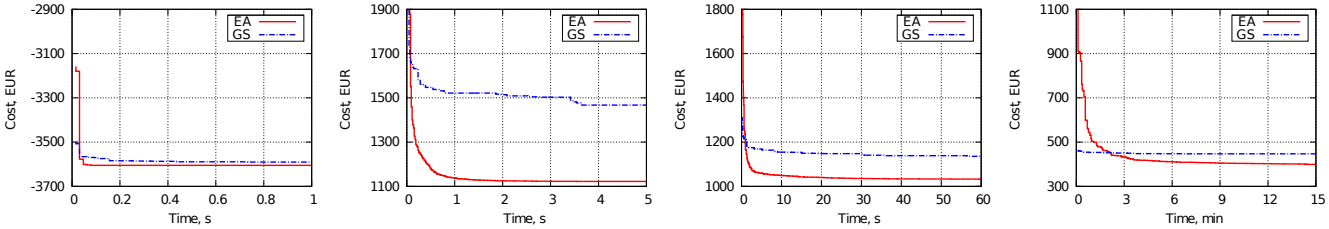
In a second experiment, we measured the forecast accuracy according to different forecast horizons. Again, we used the demand data set described above as well as the HWT forecast model. In addition, we used a supply data set, which contains wind energy data and is publicly available [9]. Naturally with increasing forecast horizon the forecast error increases (Figure 4(b)). For both data sets, we achieve a very high accuracy with forecast horizons covering only a few hours. As supply data is in general harder to forecast and contains less seasonal effects, the supply data set shows a much higher decrease in accuracy with increasing horizon. Note that we did not include any external information (e.g., wind speed) in this experiment. To conclude, the forecast horizon has a high impact on the forecast accuracy and needs to be set accordingly to achieve robust and efficient scheduling results.

Scheduling Experiments. An experiment was performed to test how scheduling deals with various numbers of aggregated flex-offers. Both scheduling algorithms were



(a) Compression performance (b) Aggregation time (c) Time flexibility losses (d) Relationship between aggregation disaggregation times

Figure 5: Results of the aggregation experiments



(a) 10 flex-offers (b) 100 flex-offers (c) 1000 flex-offers (d) 10 000 flex-offers

Figure 6: Results of the scheduling experiment with an evolutionary algorithm (EA) and randomized greedy search (GS).

run five times on four different intra-day scheduling scenarios with 10, 100, 1000 and 10000 aggregated flex-offers. The averaged results are presented in Figure 6. We can see that a large number of flex-offers considerably slows down the convergence of the algorithms. While the problem with 1000 flex-offers can still be solved efficiently, to deal with larger problems, a proper degree of flex-offer aggregation needs to be performed.

10. CONCLUSION AND FUTURE WORK

We have described the MIRABEL system that facilitates the more efficient utilization of RES supply by taking benefit from flexibilities. In particular, our attention was focused on the LEDMS components: (1) the aggregation component can group similar flex-offers and guarantees minor flexibility loss; (2) the forecasting component provides forecasts for traditional energy demand and supply and flex-offers and offers some nice optimizations; (3) the scheduling component uses either a randomized greedy algorithm or an evolutionary one to balance energy supply and demand; (4) a negotiation component finds an agreement between the prosumer and its BRP about the price for flex-offers. Finally, the component interactions are discussed and supported by some satisfactory initial experimental results.

Additionally to component-specific future directions, interesting data management future directions that we are considering for future work are (1) seamless integration of past, current and forecast data, (2) design of highly scalable, tailor-made data management and query processing techniques and (3) capture of uncertainty levels in the result of queries.

11. REFERENCES

[1] D. Bertsimas and J. Tsitsiklis. Simulated annealing. *Statistical Science*, 8:10–15, 1993.

[2] L. Dannecker, R. Schulze, M. Boehm, W. Lehner, and G. Hackenbroich. Context-aware parameter estimation for forecast models in the energy domain. In *SSDBM*, 2011.

[3] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Springer, Berlin, 2003.

[4] ENTSO-E. The Harmonized Electricity Market Role Model, Version 2009-01, 2009. `https://www.entsoe.eu/fileadmin/user_upload/edi/library/role/role-model-v2009-01.pdf`.

[5] U. Fischer, M. Böhm, and W. Lehner. Offline design tuning for hierarchies of forecast models. In *BTW*, 2011.

[6] R. Kimball and M. Ross. The data warehouse toolkit: the complete guide to dimensional modeling. *John Wiley & Sons Inc*, 2002.

[7] Nationalgrid UK. *Metered half-hourly electricity demands*, 2010. `http://www.nationalgrid.com/uk/Electricity/Data/Demand+Data/`.

[8] J. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7:308–313, 1965.

[9] NREL. *Wind Integration Datasets*, 2010. `http://www.nrel.gov/wind/integrationdatasets/`.

[10] M. L. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Springer, New York, 2008.

[11] R. Ramanathan, R. Engle, C. W. Granger, F. Vahid-Araghi, and C. Brace. Short-run forecasts of electricity loads and peaks. *International Journal of Forecasting*, 13:161–174, 1997.

[12] J. W. Taylor. Triple seasonal methods for short-term electricity demand forecasting. *European Journal of Operational Research*, 204:139–152, 2009.

[13] J. W. Taylor and P. E. McSharry. Short-term load forecasting methods: An evaluation based on european data. *IEEE Transactions on Power Systems*, 22:2213–2219, 2007.

[14] H. Y. Yamin. Review on methods of generation scheduling in electric power systems. *Electric Power Systems Research*, 69(2–3):227–248, 2004.