Annals of Operations Research
https://doi.org/10.1007/s10479-018-3122-6

S.I.: STOCHASTIC MODELING AND OPTIMIZATION, IN MEMORY OF ANDRÁS PRÉKOPA

Improving the computational efficiency of stochastic programs using automated algorithm configuration: an application to decentralized energy systems

Hannes Schwarz[1] · Lars Kotthoff[2] · Holger Hoos[3,4] · Wolf Fichtner[1] · Valentin Bertsch[5,6,7,8]

Abstract
The optimization of decentralized energy systems is an important practical problem that can be modeled using stochastic programs and solved via their large-scale, deterministic-equivalent formulations. Unfortunately, using this approach, even when leveraging a high degree of parallelism on large high-performance computing systems, finding close-to-optimal solutions still requires substantial computational effort. In this work, we present a procedure to reduce this computational effort substantially, using a state-of-the-art automated algorithm configuration method. We apply this procedure to a well-known example of a residential quarter with photovoltaic systems and storage units, modeled as a two-stage stochastic mixed-integer linear program. We demonstrate that the computing time and costs can be substantially reduced by up to 50% by use of our procedure. Our methodology can be applied to other, similarly-modeled energy systems.

Keywords OR in energy · Large-scale optimization · Stochastic programming · Uncertainty modeling · Automated algorithm configuration · Sequential model-based algorithm configuration

✉ Hannes Schwarz
hannes.schwarz@kit.edu

1 Chair of Energy Economics, Institute for Industrial Production (IIP), Karlsruhe Institute of Technology (KIT), Hertzstraße 16, 76187 Karlsruhe, Germany

2 Department of Computer Science, University of Wyoming, Laramie, WY, USA

3 Leiden Institute of Advanced Computer Science (LIACS), Universiteit Leiden, Leiden, The Netherlands

4 Department of Computer Science, University of British Columbia (UBC), Vancouver, BC, Canada

5 Economic and Social Research Institute (ESRI), Dublin, Ireland

6 Department of Economics, Trinity College Dublin, Dublin, Ireland

7 Department of Energy Systems Analysis, German Aerospace Center (DLR), Stuttgart, Germany

8 University of Stuttgart, Stuttgart, Germany

Published online: 25 January 2019

1 Introduction

With the expansion of renewable energy sources (RES) around the world, decentralized energy systems play an increasingly important role (Altmann et al. 2010; Owens 2014; Velik and Nicolay 2016; Yazdanie et al. 2016; Kobayakawa and Kandpal 2016; Schwarz et al. 2018b). Planning and implementing these systems in an optimized fashion is therefore becoming more prominent. Since electricity generation from RES is fluctuating and uncertain, in particular in the case of wind power and photovoltaic (PV) systems, a high temporal resolution is required. For real energy systems, this results in large-scale optimization problems which are difficult to solve in practice. Furthermore, the decentralization of the energy system introduces non-negligible uncertainties on the supply side. To meet these challenges, different analytic and computational techniques from Operations Research (OR) can be leveraged (Andriosopoulos et al. 2016). Stochastic Programming is an OR technique that enables an adequate consideration of various uncertainties (see, e.g., Dantzig 1955; Prékopa et al. 1980; Wallace and Fleten 2003; Beraldi et al. 2008; Kuznia et al. 2013). In order to solve a stochastic program computationally, the problem is described by its deterministic-equivalent formulation, where a set of scenarios represent the uncertain conditions. This typically results in programs that are much larger than the original and very expensive to optimize (Fragnière et al. 2000). Such optimization techniques are commonly used to tackle real-world problems (see, e.g., Ben-Ayed et al. 1992; Zokaee et al. 2017; Khalilpourazari and Arshadi Khamseh 2017).

Recently, techniques from Artificial Intelligence (AI) have gained traction in optimizing the process for solving such difficult problems (see, e.g., Hutter et al. 2010). They can be applied to systems that expose parameters affecting their performance and intelligently change those parameters to reduce the time it takes to solve the problem. In many cases, these optimized parameter settings generalize, i.e., it is sufficient to run this automated configuration process on a subset of the problems that are to be solved, and simply apply the result to similar problems to be solved subsequently.

In this work, we consider a specific real-world decentralized energy system as a case study for automated configuration: the optimization of a residential quarter with a PV system, heat pumps and heat storage units (Schwarz et al. 2018a). Since the optimal implementation of an energy system depends predominantly on the investment at the first stage and on their operation at the second stage, the problem is formulated as a two-stage stochastic program with recourse. To keep the program with more than 100 million variables computationally feasible, the problem is decomposed by fixing the first-stage variables of the program and optimizing them iteratively with a derivative-free optimization (DFO) approach. The sub-problems at the second stage are solved in parallel on a high-performance computing (HPC) system using the commercial MILP solver CPLEX. At each step of the DFO process, thousands of sub-problems are solved by CPLEX. The default parameter configuration of the solver is unlikely to provide the best performance for all of these problems (see, e.g., Hutter et al. 2010). We therefore automatically determine sub-problem-specific parameter configurations using the state-of-the-art algorithm configuration tool SMAC (Sequential Model-based Algorithm Configuration, Hutter et al. 2011). There are similar approaches in the literature but they are less general and comprehensive. Avdoulas et al. (2018) use genetic algorithms to optimize the parameters of their model, but require heavy customization of the tuning procedure. Similarly, Talbi (2016) uses meta-heuristics, but their procedure is also not generic. Pintér (2017) proposes an approach to find optimized general parameter settings, but does not apply this approach beyond artificial benchmark problems. To the best of our knowledge, our paper is the first to demonstrate automatic configuration of a MIP solver to be effective on a real-world problem of the magnitude considered here. The main contribution of this work is the description of a general procedure to improve the solution of energy system optimization problems (modeled as stochastic programs) by using automated algorithm configuration, including a discussion of the rationale behind the achieved performance improvements.

The remainder of this article is structured as follows. Section 2 provides background on automated algorithm configuration. The real-world stochastic optimization problem considered in our work is presented in Sect. 3. The automated performance optimization approach we employ here is described in Sect. 4, and the results achieved by it are presented in Sect. 5, followed by further discussion in Sect. 6. We conclude with a summary of our findings and point out several avenues for future work in Sect. 7.

2 Automated algorithm configuration

Most modern software systems, such as the CPLEX solver we use here, expose a multitude of parameters to the user. The default values of these parameters usually do not provide optimal performance (see, e.g., Atamtürk and Savelsbergh 2005). Configuring them on a case-by-case basis is imperative to be able to solve problems quickly and efficiently (see, e.g., Hutter et al. 2010). Unfortunately, the space of possible parameter settings is vast, and there is little theory to guide the setting of such parameters for a given problem. Therefore, in most applications of CPLEX and similar highly parametric solvers, default parameter settings are used or custom parameter settings are determined in an ad-hoc manner with tedious manual experimentation.

Automatic algorithm configuration techniques from AI provide an effective way of solving this problem (see, e.g., Hutter et al. 2007, 2011). Instead of manually experimenting with different parameter settings, the user only specifies the target algorithm (i.e., the algorithm whose performance is to be optimized), the parameter space (defined through the names of the parameters and their permitted and default values), a set of representative problem instances, and a performance metric. Then the automated procedure does the rest: it intelligently and efficiently selects and evaluates promising candidate parameter settings, with the goal of optimizing the given performance metric. The general approach is depicted in Fig. 1.

Automatic algorithm configuration approaches treat the target algorithm as a black box, in that they do not require knowledge of its inner workings; instead, they evaluate its performance by observing it empirically and use the information gathered in this way to find better-performing parameter settings. This is a great practical advantage of this approach—users

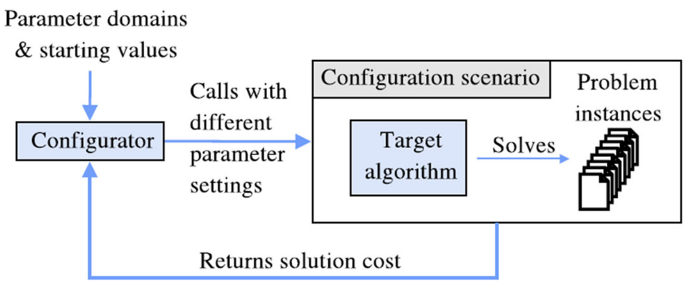


Fig. 1 Automated algorithm configuration (Hutter et al. 2010)

do not need to be CPLEX experts, as is the case for most practitioners who use CPLEX to solve problems in their specific domain of expertise. Automated configuration can be applied by users with no background in the used AI techniques. Hence, this is a powerful, domain-independent approach that provides the ability to improve empirical performance for all potential users, regardless of their background.

A simple approach to algorithm configuration is to evaluate a very large number of different configurations, either chosen systematically to cover the given configuration space, or sampled randomly. For target algorithms with a very large configuration space, such as CPLEX in our application, this is usually infeasible because of resource constraints. Instead, state-of-the-art model-based algorithm configuration methods approximate the parameter-performance response surface with a so-called surrogate model (see, e.g., Hutter et al. 2011). The surrogate model is cheap to evaluate and provides predictions of how the target algorithm will perform for parameter settings that have not been evaluated. An acquisition function uses these predictions, along with the uncertainty associated with them, to propose the next configuration on which to evaluate the target algorithm. This configuration is where the model predicts the highest potential for performance improvement, taking into account both the performance of past configurations and the uncertainty associated with unexplored configurations. In this way, the procedure naturally tends to explore more diverse configurations at the beginning, when uncertainty is high, and focuses on the most promising areas of the configuration space as more and more data is gathered and uncertainty decreases. The surrogate models used by such sequential model-based optimization (SMBO) procedures are induced using machine learning methods, based on the performance of the target algorithm evaluated on a relatively small number of configurations. SMBO iterates between fitting surrogate models and using them to make choices about which configurations to evaluate. Procedures that are based on the results of SMBO or the surrogate models can also be used to quantify the importance of each parameter and parameter interactions (Hutter et al. 2011). Altogether, this provides a principled way to intelligently and efficiently explore large configuration spaces.

After each evaluation, the new information on the actual performance of the target algorithm with the proposed configuration is incorporated into the surrogate model, and the acquisition function predicts the next configuration to be evaluated; this informs the sequential behavior. The process stops when a user-specified configuration budget is exhausted. The approach provides a solution at any time: the incumbent configuration can be retrieved at each step of the process.

A large body of existing research has been devoted to this problem; a comprehensive survey is beyond the scope of this article. Further information can be found, for example, in Jones et al. (1998), Hutter et al. (2009), Ansótegui et al. (2009), or Mascia et al. (2014).

SMAC is a state-of-the-art automatic algorithm configurator based on the SMBO approach (see Hutter et al. 2011 for details). We decided to use it in this work because it is known to perform well on a broad range of algorithm configuration tasks, is readily available, and relatively easy to use. Alternative SMBO-based configuration procedures include the Tree-based Parzen Estimator (TPE) (Bergstra et al. 2011) and Spearmint (Snoek et al. 2012). However, it has not been shown that either of those would reach or surpass the performance of SMAC when configuring CPLEX.

3 Problem description

We demonstrate the automated algorithm configuration methodology described above and, specifically, SMAC on the real-world case study of a decentralized energy system. This system can be modeled as a two-stage stochastic MILP, as explained in the following Sects. 3.1 and 3.2. As it is not feasible to solve the program on a single machine, the problem is decomposed into sub-problems and solved in parallel on the HPC system described in Sect. 3.3. The application of SMAC and the differences between the optimized configuration and the default configuration of the MILP solver are analyzed with respect to the most important parameters in Sects. 4 and 5.

3.1 Residential quarter as a decentralized energy system

Energy systems are considered decentralized when a portion of the energy required to satisfy demand is produced on-site, within the boundaries of, or located nearby and directly connected to, a building, community or development (Wolfe 2008). The residential quarter in our case study pools multi-family and row houses with 70 residential units into a living and energy community on 7708 m^2 for up to 180 residents (see Schwarz et al. 2018a). The quarter has photovoltaic (PV) generators and can handle flexible load through heat pumps and thermal storage units. The planning task is to determine the optimal capacities of the storage units and their operation under weather-related uncertainties of the electrical and thermal demand as well as energy supply. Figure 2 depicts the energy setup of the quarter.

On the supply side, there is a PV system of 240 kW$_p$ installed, providing power between 0 and up to approximately 200 kW$_{el}$. If the PV supply is insufficient, electricity can be purchased from an external energy supplier at a given tariff. On the demand side, 70 households

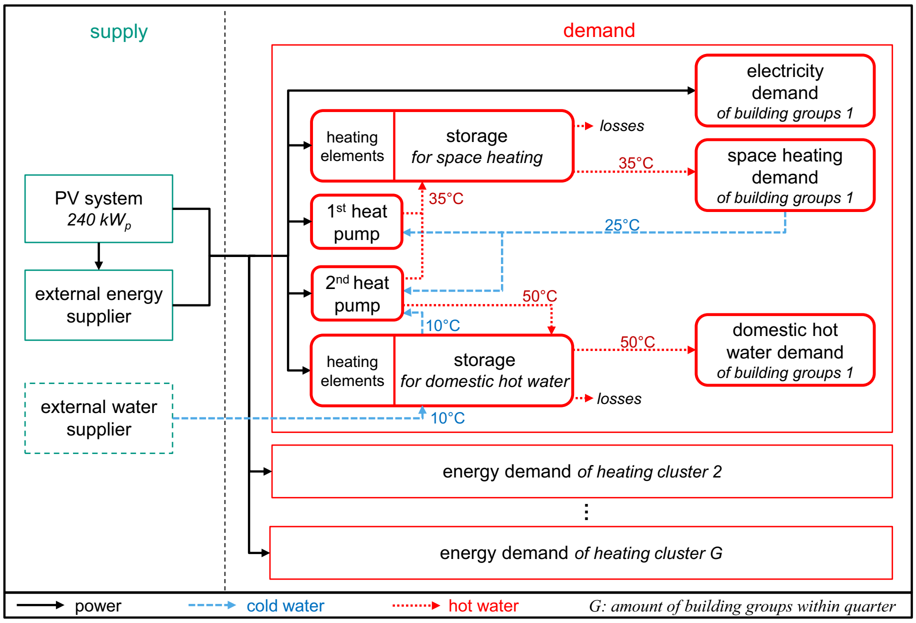

supply
demand
PV system
240 kW$_p$
external energy supplier
external water supplier
electricity demand of building groups 1
heating elements
storage for space heating
losses
35°C
space heating demand of building groups 1
1st heat pump
35°C
25°C
2nd heat pump
50°C
10°C
domestic hot water demand of building groups 1
50°C
heating elements
storage for domestic hot water
losses
10°C
energy demand of heating cluster 2
energy demand of heating cluster G
power
cold water
hot water
G: amount of building groups within quarter

Fig. 2 Energy setup of the residential quarter (Schwarz et al. 2018a)

are clustered into $G = 4$ building groups, each with a fluctuating, uncertain demand of electricity, domestic hot water, and space heating. To cover the heat demand, each building group is equipped with heat storage units in combination with two air–water heat pumps that can provide heat at half or full load up to 120 kW_{th}, depending on ambient air temperature. Additional heating elements in the storage units ensure that the thermal demand can be covered during peak times and provide a disinfection function. The heating system is separated into two cycles: the closed cycle for space heating runs at lower temperatures than the one for domestic hot water, resulting in a higher coefficient of performance (COP) of the first heat pump and lower heat losses of the storage. The target temperature is assumed to be 35 °C and can drop by approximately 10 Kelvin below this target. For the domestic hot water requirements, fresh water is obtained from an external water supplier and heated by a second heat pump in an open loop from about 10 to 50 °C.[1]

3.2 Modeling of decentralized energy systems as two-stage stochastic MILP

In order to determine the optimal storage size that leads to minimal energy system costs under uncertain conditions, the residential quarter is modeled as a two-stage stochastic program (for a compact introduction to stochastic programming, see Prékopa 1995; Shapiro et al. 2009). The objective function of the program is defined as:

$$costs = \min_{c_{g,i}, e^{grid}_{\omega,t}, e^{fi}_{\omega,t}} ANF \cdot \sum_{g=1}^{G=4} \sum_{i=1}^{k_1} cost_i \cdot c_{g,i} \\ + \frac{1}{N} \cdot \sum_{\omega=1}^{N=100} \sum_{t=1}^{T=35040} \left(p^{grid} \cdot e^{grid}_{\omega,t} - p^{fi} \cdot e^{fi}_{\omega,t} \right). \tag{1}$$

At the first stage, the capital cost of each investment for building group , such as the storage for space heating and for domestic hot water, is converted into an equivalent series of uniform amounts per period. The lifetime of the investment and an alternative investment opportunity at a certain interest rate of the fixed capital is taken into account by the annuity factor ANF. In this case study, a technical lifetime of 20 years is assumed, with an interest rate of 10%. At the second stage, the energy costs of each scenario $\omega = \{1, \ldots, \Omega\}$ are computed at each time step t by the energy obtained from the external grid $e^{grid}_{\omega,t}$ at price p^{grid}, minus the energy fed into the grid $e^{fi}_{\omega,t}$ at feed-in tariff p^{fi}. The period $t = \{1, \ldots, 35{,}040\}$ comprises 1 year, with a temporal resolution of 15 min time steps. In total, the optimization is carried out for 100 scenarios generated by a Markov process.[2] An essential constraint of the system is that the electrical and thermal demand and supply are balanced at any time. The thermal supply in the system is limited by heat pumps plus heating elements and heat storage units. The heat pumps can only run stepwise at idle, half or full load, while the heating elements can modulate their heat output on a continuous scale. The storage levels connect the states of time step t to step $t + 1$ and result in a complex stochastic MILP. The entire program is listed in "Appendix A". For further information about the program and the scenario generation, see Schwarz et al. (2018a).

[1] Note that the higher temperature difference results in a larger energy content at the same volume compared to storage units for space heating.

[2] Markov processes have proven suitable to generate PV generation and energy demand of the decentralized energy system that depend essentially on fluctuating and uncertain meteorological parameters (see Schwarz et al. 2018a, b for details).

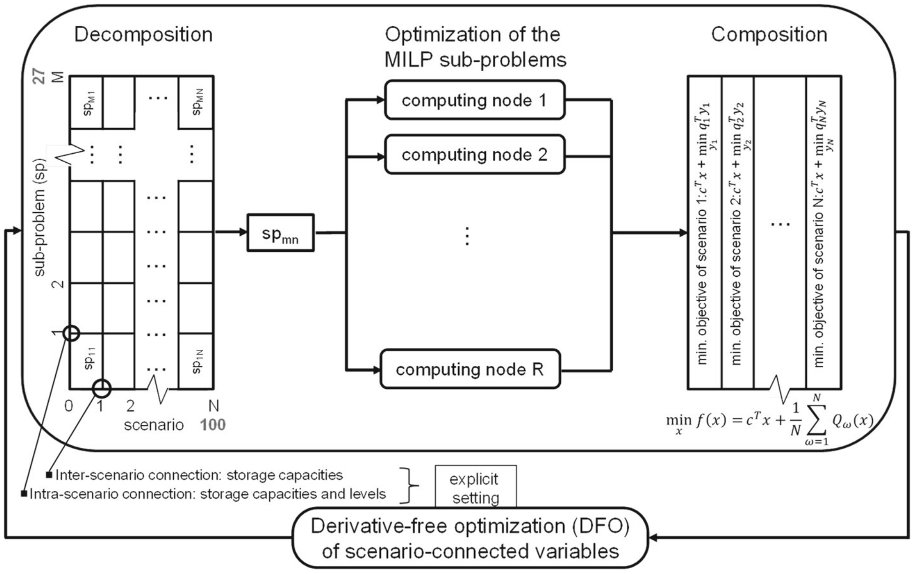

Decomposition
Optimization of the MILP sub-problems
Composition
sub-problem (sp)
27 M
sp_{M1}
sp_{MN}
sp_{11}
sp_{1N}
0 1 2 N
scenario 100
sp_{mn}
computing node 1
computing node 2
computing node R
min. objective of scenario 1: $c^T x + \min_{y_1} q_1^T y_1$
min. objective of scenario 2: $c^T x + \min_{y_2} q_2^T y_2$
min. objective of scenario N: $c^T x + \min_{y_N} q_N^T y_N$
$\min_x f(x) = c^T x + \frac{1}{N}\sum_{\omega=1}^{N} Q_\omega(x)$
Inter-scenario connection: storage capacities
Intra-scenario connection: storage capacities and levels
explicit setting
Derivative-free optimization (DFO) of scenario-connected variables

Fig. 3 Optimization approach of the two-stage stochastic MILP (Schwarz et al. 2018a)

3.3 Initial optimization approach with default configuration

The sub-problems of the two-stage stochastic MILP are solved in parallel by explicitly setting first-stage variables. The sub-problems are solved by CPLEX for given first-stage variables that are optimized iteratively by an outer DFO. The entire optimization procedure is depicted in Fig. 3.

The first-stage variables, i.e., the storage capacities as inter-scenario connections, are optimized using the steepest-ascent hill-climbing DFO algorithm (Taborda and Zdravkovic 2012). In order to minimize the costs, the storage capacities for space heating and domestic hot water of each building group are altered sequentially by a positive and negative step size s_i. The costs are minimized for each altered storage capacity, and the step with minimal costs is accepted. When there is no improvement in terms of the objective function, the step size is halved. This process is repeated until the relative change of the objective function value is smaller than 0.1%.

The DFO in the first stage allows the problem to be decomposed into $N = 100$ sub-problems (one for each scenario). By fixing the storage capacities that connect the time steps of the period $t = \{1, \ldots, T\}$ within a scenario, each scenario ω can be decomposed over time t: the 1-year period of a scenario is decomposed into periods of 2 weeks, resulting in $M = 27$ sub-problems per scenario.[3] Hence, $4 \cdot N \cdot M = 10{,}800$ sub-problems for each building group need to be solved for a storage capacity given by the outer DFO; the factor 4 stems from the number of positive and negative steps for the storage of space heating and domestic hot water. The sub-problems are solved using CPLEX (version 12.6.2) with a MILP gap of 0.6% and a cutoff-time of 1 800 s, so that the hill-climbing approach efficiently progresses to the

[3] The scenarios are further decomposed, because one scenario cannot be solved within 48 h for a MILP gap of 0.6% on a single computer. The intra-scenario connecting storage levels are not optimized by the DFO, but set to reasonable levels resulting in a negligible difference to the optimum of less than 1%.

optimum.[4] The sub-problems are solved in parallel on a HPC cluster. Even when using the HPC system, the initial optimization process requires more than a week of wall-clock time for just one building group.

4 Methodology

The application of SMAC to the integrated MILP solver is described in Sect. 4.1. Our approach to analyzing the differences between the optimized configuration (obtained from SMAC) and the default configuration of the MILP solver is outlined in Sect. 4.2.

4.1 MILP solver configuration using SMAC

To improve performance, we partition the sub-problems, select problem instances for each partition, and apply SMAC to each of these groups of instances to find optimized parameter settings. Partitioning the sub-problems is necessary, because they are structurally different and we expect different configurations to be most suitable for each. By applying SMAC to each partition individually, we allow for partition-specific configurations that result in a higher overall performance. Equally, an appropriate, representative selection of training instances is important to enable finding a well-performing solver configuration for the entire partition.

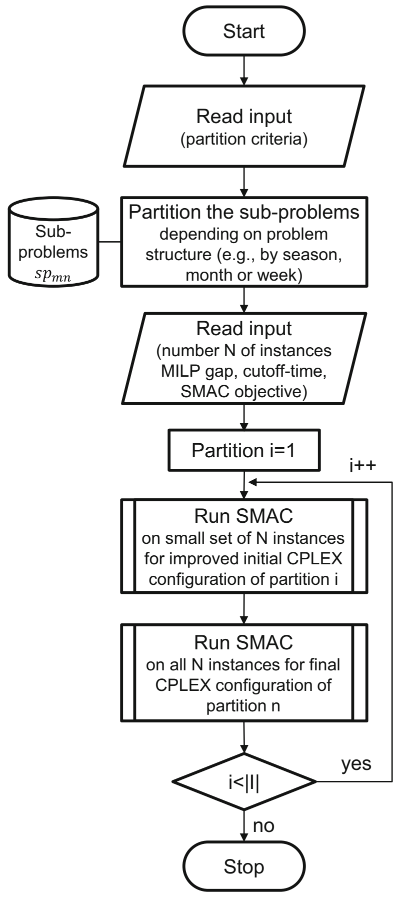
As the sub-problem with the longest runtime provides a lower bound of the running time of each hill-climbing iteration, the most difficult instances are selected for each group. Improvements on these instances will achieve the highest performance improvements for the overall method. The first choice are sub-problems that are not solved by the default CPLEX parameter configuration within the given cutoff-time of 1800 s, the second choice are sub-problems that are solved successfully, sorted by running time in descending order. In practice, we found it beneficial to apply an iterative approach, where we run SMAC on a small number of instances first and use the optimized configuration from this step as a starting point for running SMAC on the full set of instances. The complete procedure is illustrated by the flow chart in Fig. 4. Details specific to the optimization of the residential quarter are shown in *italics* on the right side of Fig. 4.

The partition-specific optimized configuration is used for optimization of the energy system of the residential quarter. The training phase of SMAC and the MILP solving process must be executed on the same computer system (the SMAC training and the CPLEX solving process are executed on a Linux-based HPC cluster using 512 CPU cores with two threads at 2.6 GHz and 16 GB RAM per core). If the resulting performance gain is not sufficient, the procedure is repeated with a finer partitioning. Usually, finer partitions lead to better improvements but make the configuration phase more expensive due to a higher number of partition-specific parameter configurations that need to be found.

4.2 Ablation analysis

To analyze optimized configurations, ablation analyses (Fawcett and Hoos 2016) can be used. Ablation analysis assesses the effect of each parameter that differs between two configurations

[4] The MILP gap is set to 0.6%, because we observed no improvement of the sub-problem solution quality in practice after several days of additional computing time; this is almost always achieved within 1800 s.


Start
Read input
(partition criteria)
Sub-problems
sp_{mn}
Partition the sub-problems
depending on problem structure (e.g., by season, month or week)
Read input
(number N of instances MILP gap, cutoff-time, SMAC objective)
Partition i=1
i++
Run SMAC
on small set of N instances for improved initial CPLEX configuration of partition i
Run SMAC
on all N instances for final CPLEX configuration of partition n
i<|I|
yes
no
Stop

Execution details of the procedure for the case study

The sub-problems are partitioned into 27 parts, as each scenario is decomposed into periods of 2 weeks, where partition 1 represents week 1 and 2 of period t={1,…,T}, partition 2 represents week 3 and 4, and so on. Partition 27 presents the last part of the one-year period which remains as only 1 day.

Overall, 9 instances per partition are selected, where 8 of 9 difficult instances cannot be solved by the default CPLEX parameter configuration with a given MILP gap of 0.6% and a cutoff-time of 1 800s. One instance is chosen randomly to avoid finding configurations that are too specific to difficult instances and do not perform well on computationally easy instances.

SMAC is initially applied to 3 difficult instances for each partition (total configuration time of 24h per partition on 1 separate CPU core). The mean wall-clock time on these instances is minimized. Unsuccessful runs that could not be solved within the cutoff-time of 1 800s are counted as having taken 18 000s (i.e., 10 times the cutoff).

SMAC is run on the 9 difficult instances from Step 2 per partition on 1 separate CPU core (total configuration time of 72h per partition on 1 separate CPU core).

Fig. 4 Automated algorithm configuration procedure of CPLEX using SMAC. On the right side, specific details of the procedure for the residential quarter are shown in italics. (The final optimized configuration of most partitions is already found after a total configuration time of 24 h, only 5 transition partitions require the full time of 72 h on 1 separate CPU core.)

on the target algorithm performance in order to identify the most impactful parameter changes. Given a default and an optimized configuration, this is done by changing the value of one parameter at a time to determine what part of the performance difference between the two configurations it accounts for. This process moves incrementally from one configuration to the other and thus takes into consideration interactions between parameters; it proceeds in a greedy fashion, determining in each iteration the largest possible performance improvement and the parameter responsible for it. Ablation analysis determines which parameters are most important to achieve improved performance and which have little or no effect in a particular configuration scenario.

5 Results

We present computational results for our automatic configuration approach in Sect. 5.1, presenting the average reduction in running time of the problem described in Sect. 3. We

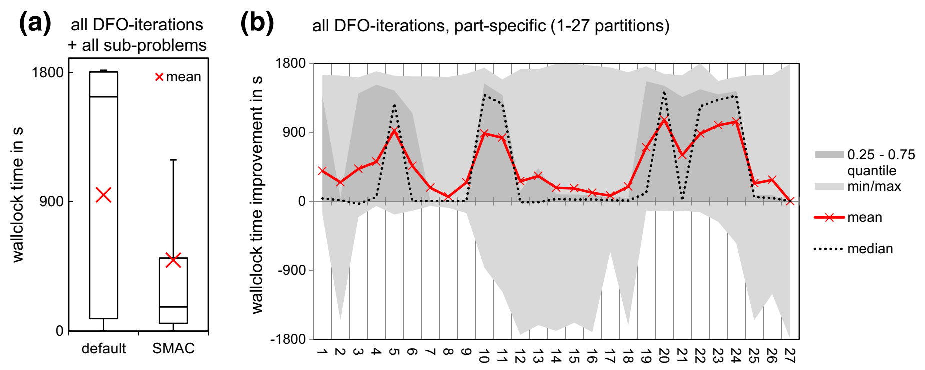


Fig. 5 Wall-clock time for the sub-problems (MILP gap = 0.6%, cutoff-time = 1800 s) for **a** the default configuration and **b** the optimized partition-specific configurations obtained from SMAC

use ablation analysis to determine the three most important parameters in reducing running times as described in Sect. 5.2. The temporal and monetary gain of using automated algorithm configuration, depending on computing parallelization, is revealed in Sect. 5.3.

5.1 Reduction in running time

We use 27 different partition-specific CPLEX configurations determined by SMAC for the inner optimization of the previously described MILP sub-problems. In total, six outer hill-climbing iterations are needed to find optimal storage capacities for one building group. The mean wall-clock time of all 54,000 computations is reduced substantially, from 947 s to 493 s, by using the optimized configurations.[5] The Tukey boxplot shown in Fig. 5a illustrates this difference.[6] Even more drastically than the mean, the median running time drops from 1 631 s to 168 s, while the upper quartile is reduced from 1800 to 509 s. The deviation between the mean and the median of the default case is due to the skewness of the wall-clock time distribution: the sub-problems tend to be solved either quickly or towards the end of the given cutoff-time of 1 800 s. Approximately one in six sub-problems is not solved, but achieves a MILP gap of 1% or less, which still provides a reasonably accurate basis for the hill-climbing approach. For the optimized configurations, the deviation between the mean and the median is mainly caused by outliers corresponding to unsuccessful runs. In comparison to the default, 30% more sub-problems are solved.

Figure 5b provides more detailed results for the 27 partitions and shows that the biggest improvements are achieved for the transition seasons (partitions 5–11 and 20–23). These sub-problems are more complex than those for the winter (partitions 1–4 and 24–27) and summer season (partition 12–19), for which little or no reduction of running time could be achieved. In summer, the space heating demand is zero and, accordingly, the heat pump is turned off. The integer variables of this heat pump can be set to zero, which simplifies the optimization task. In winter, the PV supply minus the electricity usage of the households

[5] There are four storage capacities that do not require optimization. Therefore, only 54,000 sub-problems are solved, instead of 64 800 (= 4 storage capacities per iteration times 27 parts per scenario × 100 scenarios times 6 iterations).

[6] The ends of the whiskers represent the lowest wall-clock time within the 1.5 interquartile range (IQR) of the lower quartile, and the highest wall-clock time within the 1.5 IQR of the upper quartile. Outliers are not shown.

is low or even zero. Consequently, the potential for profitable load shifting is lower than in transition seasons, which also simplifies the optimization task. The wall-clock times for solving the winter and summer sub-problems are much lower than those of the transition sub-problems for the default configuration. Thus, the potential wall-clock time reduction is higher for more complex sub-problems in the transition seasons: there is a wall-clock time improvement of 1 403 s within the 0.75 quantile. The negative lower quartile shows that the optimized configuration, which is based on nine of 100 instances per partition, can lead to worse performance on some sub-problems. More training instances or a finer partitioning could remedy this issue, but would increase the resource requirements for the automatic algorithm configuration process.

5.2 Importance of parameters in reducing running times

The path of improvements (ablation path) shows the reduction of the mean wall-clock time for the nine selected instances per partition. It is important to note that this path is not the one followed by SMAC, but the optimal improvement path determined by ablation analysis post hoc. For each partition, 20–50 CPLEX parameters differ between the default and the final optimized configuration obtained from SMAC. The mean wall-clock time for solving the selected instances across all partitions with the default configuration (P-0_Default) is 1 556 s and is reduced to 481 s by the optimized configuration (P-All_SMAC). Note that these values differ from the wall-clock time improvement in Sect. 5.1, as not all computations of the sub-problems, but only the nine selected instances per partition are considered. Figure 6 shows the path of improvements for all 27 partitions: beginning from P-0_Default, over the first three most effective parameter adjustments P-1, P-2, and P-3 and ending with P-All_SMAC. For each partition, the improvement paths are shown uniformly for the seasons of transition (dotted red line), winter (solid black line) and summer (dashed orange line).

Figure 6 shows that similar problems, i.e. instances of the same season, tend to show similar ablation paths. However, within the winter, summer, and transition partitions, there is still a notable difference of improvements, indicating that a finer segmentation into 27 partitions may achieve further improvements. In particular, the wall-clock time of partitions

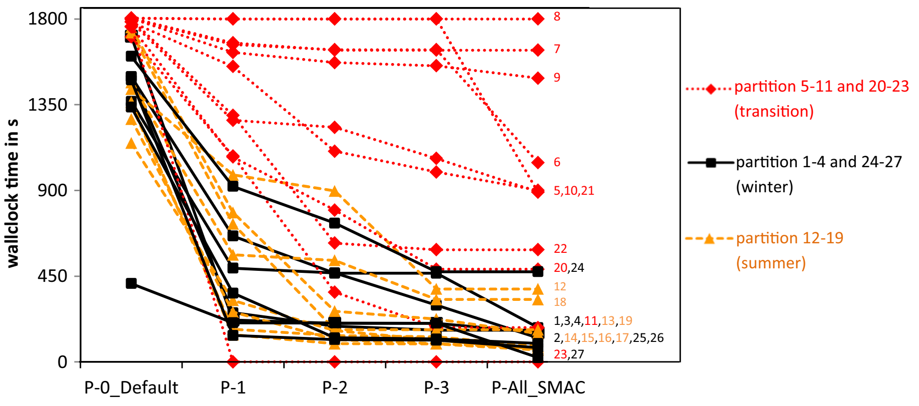


Fig. 6 Improvement paths for all 27 partitions (mean wall-clock time for the 9 selected instances) for the optimized configuration for the three parameters with the highest effect and the final configuration obtained from SMAC. (The default configuration (P-0_Default) has a lower mean wall-clock time of 477 s on partition 27 because of the lower time horizon of 1 day compared to 2 weeks for the other partitions.)

Table 1 The three parameters with the highest impact on wall-clock time leading to a total relative reduction of 40% (descriptions of the parameters are obtained from the CPLEX User's manual IBM 2016)

Parameter name	Description	Avg. reduction (total rel. reduction)
1. MIP strategy rinsheur	Sets the frequency to apply the relaxation induced neighborhood search (RINS) heuristic	618 s (27%)
2. MIP strategy nodeselect	Rules the selection of the next node to process when backtracking	384 s (9%)
3. MIP limits aggforcut	Limits the number of constraints that can be aggregated for generating flow cover and mixed integer rounding (MIR) cuts	293 s (4%)

in the transition seasons is reduced to a few seconds in some cases and remains close to the time of the default configuration. There is no wall-clock time reduction for partition 8. For these nine difficult instances, SMAC was not able to find a CPLEX configuration that leads to a solution within the cutoff-time of 1 800 s in multiple runs and very high configuration budgets of three days in total on a processor with 1 TB RAM. Eventually, we solved the sub-problems in partition 8 using the configuration optimized for the similar partition 7 and obtained an improvement in this way (see Fig. 5b). Table 1 lists the three parameters that have the largest effect on performance on average, a brief description, and their mean wall-clock time reduction and their total relative wall-clock time reduction over all partitions in the same order as in Fig. 6. Table 3 in Appendix B presents the corresponding results for all 27 partitions in parentheses.

The RINS heuristic parameter changed from the default of 0 to values around 80 in the optimized configurations. This means that instead of letting CPLEX automatically decide when to run the heuristic, setting it to be run at fixed intervals of around 80 nodes of the MILP problem was better.[7] Thus, running the heuristic per se does not result in performance improvements, but increases the overhead, and that CPLEX was unable to detect this and adjust the frequency accordingly. SMAC, on the other hand, successfully recognized and exploited this situation.

In the cases where the "nodeselect" parameter was changed from its default of 1, it was set to 0 or 2 in the optimized configurations. This means that rather than best-bound search, either depth-first search (0), or best-estimate search (2) was performed. This may indicate that the objective values derived from the LP-relaxation of the problem are not informative for selecting the best next node and that simply choosing the most recently created node (depth-first search) is more effective. For our particular set of problems, many nodes may have the same objective values, leading to an essentially random selection that turns out to be bad in many cases.

The "aggforcut" parameter changed from its default of 3 to larger values (between 7 and 10) in the optimized configurations, indicating that a more aggressive aggregation of constraints is beneficial.

[7] Note that the nodes of the MILP problem are different from the computing nodes of the HPC cluster.

5.3 Advantages of the applied methodology

The user cares most about the performance improvement achieved for the overall wall-clock time as a function of the number of CPU cores utilized. We kept track of the computing time required for each step of our approach, including solving all MILP sub-problems. Based on these times, we determined overall wall-clock time. The computing cost is based on Amazon EC2 (https://aws.amazon.com/ec2/pricing): US-$ 0.239 per full hour and node with two CPU cores and 16 GB RAM that is required for the CPLEX optimization of the sub-problems. Figure 7 compares the default and SMAC-optimized configurations with respect to the total wall-clock time and cost as a function of the number of CPU cores. In addition, the SMAC configuration effort is illustrated. Note that the overhead of finding the optimized configuration is only incurred once, but the optimized configuration can be used to solve MILP instances from different scenarios more efficiently. In the case of parameter changes and further analyses, the deployed optimized configuration would quickly amortize the configuration cost and achieve increasingly larger relative improvements over time.

For one CPU core, the total wall-clock time is reduced from 592 days (51,149,983 s) for the default configuration to 308 days (26,629,672 s) for the optimized configuration. As a result, the total reduction of the computing cost is about 50% of the initial optimization with the default configuration (about 30% when the overhead of running SMAC to find optimized configurations is taken into account). Up to about 100 CPU cores, this ratio of wall-clock time to computing cost can be maintained, because there is a sufficient number of sub-problems that can be solved simultaneously at any given time. Beyond 100 CPU cores, the cost increases, because some CPU cores are idle while other CPU cores are still computing difficult sub-problems that are required for the DFO approach to process. At about 6000 CPU cores, this effect negates the individual wall-clock time reduction of the sub-problems achieved by the optimized CPLEX configurations. Consequently, there is no further time and cost reduction in this case.

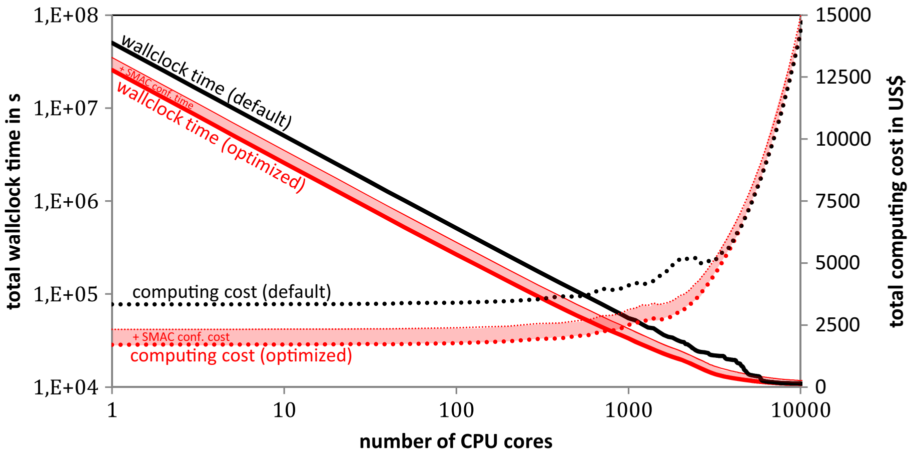


Fig. 7 Total wall-clock time (solid lines, left log-scaled vertical axis) and total computation cost (dotted lines, right vertical axis) versus utilized CPU cores (log-scaled horizontal axis) using the default CPLEX configuration (black) and the optimized CPLEX configuration obtained from SMAC plus the configuration effort (red). (We employed SMAC on up to 27 CPU cores and assume the same scaling effect on total wall-clock time and cost, when SMAC is employed in parallel mode on more CPU cores.)

For this paper, we used up to 512 CPU cores of one HPC cluster. This corresponds to a wall-clock time of about 28 h and computing cost of US-$ 3716, which are reduced by approximately 45% with the optimized configurations.[8] Even when the SMAC run time is taken into account, about 30% of the time and cost are saved.

6 Discussion

The results presented in Sect. 5 demonstrate that careful partitioning and selection of problem instances is crucial to achieving good results with automated algorithm configuration methods for a real-world problem of the magnitude considered here. The fewer partitions are used, the smaller the potential performance improvement becomes. Too few instances can lead to configurations that are too specific and do not perform well on other instances of the partition. On the other hand, too many partitions and instances increase the cost of running the automated configuration method and thus diminish the subsequent performance improvements relative to the overall cost. Partitioning based on winter, summer, and transition seasons enabled an overall performance improvement of about 33%. A finer partitioning into 27 partitions based on 2-week intervals achieved a performance improvement of 50%, while maintaining acceptable resource requirements for the configuration procedure.

CPLEX comes with an internal tuning tool. It tries up to 30 different pre-determined configurations and chooses the best of these. While this process takes much less time, the achieved performance improvements are much smaller. For example, a randomly chosen instance that is not solved within 1 800 s by the default configuration could be solved within 420 s with the best configuration found by the CPLEX tuning tool, but the same instance is solved four times faster with the best configuration found by SMAC. This is consistent with results indicating that state-of-the-art automatic algorithm configuration achieves substantially better results than the CPLEX tuning tool (Hutter et al. 2010).

Our ablation analysis shows that only a few parameter adjustments are necessary to achieve major improvements: changing just three parameters (MIP strategy rinsheur, MIP strategy nodeselect, and MIP limits aggforcut) already achieves 40% of the total improvement on average. While these findings, as well as the actual configurations we determined, are specific to our case study, our overall methodology is applicable more broadly, and we expect that similar performance improvements and ablation results can be obtained on challenging, large-scale problems similar to the one considered here. In particular, SMAC can be applied to any similar problem.

Further performance improvements can be achieved by changing the MILP gap. In this specific optimization problem, the outer hill-climbing DFO requires a MILP gap of about 0.6% for the sub-problems to work efficiently, but only when the outer DFO is close to the optimum. One approach to reducing the computational requirements would be to start with a wider gap and reduce it dynamically as the outer DFO converges on optimal values. A similar approach could be used for the cutoff-time.

A different way to reduce the required computational resources would be to optimize the amount of memory required as well as the running time. Some sub-problems require up to 16 GB RAM with the current best configuration we have found. A reduced memory requirement would allow more runs on a single node, or the use of cheaper computing nodes with less RAM.

[8] In practice, due to time restrictions per job of the HPC queuing system, the computation takes about a week using CPLEX in its default configuration, and less than half a week when using the optimized configurations.

7 Conclusions and future work

Our energy system structure is changing from centralized to decentralized energy systems, which are subject to manifold sources of uncertainty such as the electrical and thermal demand and the energy supply. Stochastic Programming helps to avoid insufficient investment decisions but typically results in extremely large-scale optimization problems. Here, we have modeled a real-world residential quarter as a two-stage stochastic mixed-integer linear program, which was subsequently solved on a high-performance computing (HPC) system. With the default configuration of CPLEX, the problem requires about 28 h of computation time on 512 CPU cores to be solved. By applying the automatic algorithm configuration tool SMAC, we were able to determine a set of performance-optimized configurations and achieve performance improvements of up to 50% overall, and up to 30% when taking into account the effort of finding the optimized configurations. This enables not only a faster solution of the given problem, but also facilitates additional analyses, and ultimately makes it possible to tackle more complex energy systems.

Further computing time and cost reductions could be achieved by adapting the MILP gap and/or cutoff-time parameters. Moreover, the exact CPLEX optimization could be substituted by a heuristic method (e.g., using machine learning for quick approximation of solutions to sub-problems) subject to the condition of a sufficient solution quality for the outer DFO approach. Another promising direction would be to adapt the DFO method to be more efficient.

It might seem tempting to apply algorithm configuration to the full problem, without decomposition. Tackling this challenge would require substantially more powerful MILP solvers than are currently available. Therefore, using a combination of state-of-the-art algorithm configurators, MILP solvers and decomposition techniques will likely remain for some time as the best approach to solving complex energy system optimization problems, such as the one considered here.

Acknowledgements The authors would like to gratefully acknowledge the funding provided by the Helmholtz Association of German Research Centers via the Research Programme “Storage and Cross-Linked Infrastructures”. In addition, the authors also acknowledge support by the state of Baden-Württemberg through bwHPC and the German Research Foundation (DFG) through Grant No. INST 35/1134-1 FUGG, as well as through an NSERC Discovery Grant. We also acknowledge the use of Compute Canada/Calcul Canada computing resources. Valentin Bertsch acknowledges funding from the ESRI’s Energy Policy Research Centre. All omissions and errors are our own.

Appendix A

The entire two-stage stochastic MILP of the residential quarter is shown in the following (nomenclature is listed in Table 2):Objective function:

$$costs = \min_{c_{g,i}, e^{grid}_{\omega,t}, e^{fi}_{\omega,t}} ANF \cdot \sum_{g=1}^{4} \sum_{i=1}^{k_1} cost_i \cdot c_{g,i} + \frac{1}{N} \cdot \sum_{\omega=1}^{N} \sum_{t=1}^{T} \Big(p^{grid} \cdot e^{grid}_{\omega,t} - p^{fi} \cdot e^{fi}_{\omega,t} \Big), \tag{A.1}$$

- the installed PV capacity of the quarter: $\sum_{g=1}^{4} c_{g,i=PV} = 240$,

Table 2 Nomenclature of the residential quarter modeled as a two-stage stochastic program

Parameters	
ANF	Annuity factor
$cost_i$	Variable capacity costs of component i plus a fixed amount
$COP_{\omega,u,t}$	COP of the heat pump in scenario ω of building group g for use u at time t
$d^{hp,max}_{\omega,t}$	Maximal heating power of the heat pump at time t
$d^{he,max}$	Maximal heating power of the heating element
$d^{ee}_{\omega,t}$	Electricity demand for electrical usage in scenario ω of building group g at time t
$d_{\omega,g,u,t}$	Thermal demand in scenario ω of building group g for use u at time t
$e^{pv,kwp}_{\omega,t}$	Supplied electrical energy per kilowatt-peak of the PV system in scenario ω at time t
$e^{pv}_{\omega,t}$	Supplied electrical energy from the PV system in scenario ω at time t
f	Compensation factor for not-covered heat demand
l_u	Loss factor of heat storage for use u
m	Possible power modes of the heat pump
r_u	Ramp-up loss factor of heat pump for use u
p^{grid}	Price of electricity from grid
p^{fi}	Price of feed-in compensation
η	Efficiency of the heating element
Variables	
$\boldsymbol{c_{g,i}}$	Capacity of building group g of component i
$\boldsymbol{c_{g,i=PV}}$	Installed PV capacity of building group g
$\boldsymbol{c_{g,i=HP_{SH}}}$	Number of heat pumps of building group g for SH
$\boldsymbol{c_{g,i=HP_{DHW}}}$	Number of heat pumps of building group g for DHW
$\boldsymbol{c_{g,i=HE_{SH}}}$	Number of heating elements of building group g for SH storage
$\boldsymbol{c_{g,i=HE_{DHW}}}$	Number of heating elements of building group g for DHW storage
$\boldsymbol{c_{g,i=S_{SH}}}$	Maximal capacity of heat storage of building group g for SH
$\boldsymbol{c_{g,i=S_{DHW}}}$	Maximal capacity of heat storage of building group g for DHW
$\boldsymbol{d^{hp}_{\omega,g,u,t}}$	Used electricity of heat pump in scenario ω of building group g for use u at time t
$\boldsymbol{d^{he}_{\omega,g,u,t}}$	Used electricity of heating element in scenario ω of building group g for use u at time t
$\boldsymbol{e^{grid}_{\omega,t}}$	Used electricity from the grid in scenario ω at time t
$\boldsymbol{e^{fi}_{\omega,t}}$	Fed-in energy of the PV system in scenario ω at time t

Table 2 continued

$\boldsymbol{pos_{\omega,g,u,t}}$	Pos. variable for positive shift of heat pump in scenario ω of building group g for use u at time t
$\boldsymbol{neg_{\omega,g,u,t}}$	Pos. variable for negative shift of heat pump in scenario ω of building group g for use u at time t
$\boldsymbol{q_{\omega,g,u,t}}$	Not covered heat demand in scenario ω of building group g for use u at time t
$\boldsymbol{s_{\omega,g,u,t}}$	Stored heat in scenario ω of building group g for use u at time t
$\boldsymbol{s_{g,u}^{min}}$	Minimal heat storage level of building group g for use u
$z_{\omega,g,u,t}$	Integer/continuous heating power level in scenario ω of building group g for use u at time t
Indices	
g	Building group $1,\ldots G$ of the quarter with $G = 4$
i	Component $i \in \{PV, HP_{SH}, HP_{DHW}, HE_{SH}, HE_{DHW}, S_{SH}, S_{DHW}\}$ of the energy system with $\lvert i\rvert = k_1 = 7$
u	Use $u \in \{SH, DHW\}$ for space heating or domestic hot water with $\lvert u\rvert = 2$
t	Time index $1,\ldots,T$ indicating the time step of the year
ω	Scenario index $1,\ldots,N$

- the number of heat pumps for SH within a building group: $c_{g,i=HP_{SH}} = 1$,
- the number of heat pumps for DHW within a building group: $c_{g,i=HP_{DHW}} = 1$,
- the number of heating elements for the SH storage: $c_{g,i=HE_{SH}} = 4$,
- the number of heating elements for the DHW storage: $c_{g,i=HE_{DHW}} = 4$.

Additionally, electrical supply and demand have to be balanced:

$$e_{\omega,t}^{pv} + e_{\omega,t}^{grid} = d_{\omega,t}^{ee} + \sum_{g=1}^{4}\sum_{u=1}^{2}\left(d_{\omega,g,u,t}^{hp} + d_{\omega,g,u,t}^{he}\right) + e_{\omega,t}^{fi} \quad \forall\omega\forall t, \tag{A.2}$$

with supplied PV energy $e_{\omega,t}^{pv} = \sum_{g=1}^{4} e_{\omega,t}^{pv,kwp} \cdot c_{g,i=PV}$ and balanced thermal supply and demand:

$$\begin{aligned} &COP_{\omega,u,t} \cdot d_{\omega,g,u,t}^{hp} + \eta \cdot d_{\omega,g,u,t}^{he} + (1 - l_u) \cdot s_{\omega,g,u,t} + q_{\omega,g,u,t} \\ &\quad = d_{\omega,g,u,t}^{th} + L_{\omega,g,u,t} + s_{\omega,g,u,t+1} + pos_{\omega,g,u,t} \cdot r_u \quad \forall\omega, \forall g, \forall u, \forall t, \end{aligned} \tag{A.3}$$

with the storage heat losses $l_{u=SH} = 0.003$ and $l_{u=DHW} = 0.006$ and ramp-up losses $r_u = 0.05$.

The storage possibility is restricted by:

$$s_{g,u}^{min} \le s_{\omega,g,u,t} \le c_{g,i=S_u} \quad \forall\omega, \forall g, \forall u, \forall t, \tag{A.4}$$

where $s_{g,u}^{min} = 0$. Load changes are taken into account by:

$$z_{\omega,g,u,t+1} - z_{\omega,g,u,t} = pos_{\omega,g,u,t} - neg_{\omega,g,u,t} \quad \forall\omega, \forall g, \forall u, \forall t. \tag{A.5}$$

The heating element supply for each building group is given by:

$$\eta \cdot d_{\omega,g,u,t}^{he} \leq c_{g,i=HE_u} \cdot d^{he,max} \quad \forall\omega, \forall g, \forall u, \forall t, \tag{A.6}$$

and the heat pump supply by:

$$COP_{\omega,u,t} \cdot d_{\omega,g,u,t}^{hp} = \frac{1}{m} \cdot d_{\omega,t}^{hp,max} \cdot z_{\omega,g,u,t} \quad \forall\omega, \forall g, \forall u, \forall t, \tag{A.7}$$

$$z_{\omega,g,u=DHW,t} \leq m \cdot c_{g,i=HP_{DHW}} \quad \forall\omega\forall g\forall t, \tag{A.8}$$

$$\sum_{u=1}^{2} z_{\omega,g,u,t} \leq m \cdot \sum_{u=1}^{2} c_{g,i=HP_u} \quad \forall\omega, \forall g, \forall t. \tag{A.9}$$

If heat pumps run only at idle, half or full load, then $m = 2$ with $z_{\omega,g,u=SH,t} \in \{0, 1, 2, 3, 4\}$ and $z_{\omega,g,u=DHW,t} \in \{0, 1, 2\}$, otherwise $z_{\omega,g,u=SH,t}, z_{\omega,g,u=DHW,t} \in R_{+}$. The following constraints equal the element of the first and last time step t:

$$s_{\omega,g,u,t=T} = s_{\omega,g,u,t=1} \quad \forall\,\omega, \forall g, \forall u, \tag{A.10}$$

$$z_{\omega,g,u,t=T} = z_{\omega,g,u,t=1} \quad \forall\,\omega, \forall g, \forall u. \tag{A.11}$$

All presented variables need to be positive:

$$c_{g,i}, e_{\omega,t}^{grid}, e_{\omega,t}^{fi}, q_{\omega,g,u,t}, d_{\omega,g,u,t}^{hp}, d_{\omega,g,u,t}^{he}, s_{\omega,g,u,t}, L_{\omega,g,u,t}, pos_{\omega,g,u,t}, \neg_{\omega,g,u,t}, z_{\omega,g,u,t}$$
$$\geq 0\,\forall\omega, \forall g, \forall i, \forall u, \forall t. \tag{A.12}$$

Appendix B

Table 3 lists the three parameters that have the largest effect on performance and their mean wall-clock time reduction, itemized in detail for all 27 partitions.

Table 3 Results of the ablation analysis listing the three parameters that have the largest effect on performance altered from the default to the partition-specific optimized configuration of CPLEX. The parameters are determined by SMAC on nine instances for the given 27 partitions. The values in the brackets show the mean wall-clock time of the nine instances to achieve a MILP gap of at most 0.6%, whereby unsuccessful runs that could not be solved within the cutoff-time of 1800 s are counted as having taken 18,000 s

	P-0_default ⟶	P-1 ⟶	P-2 ⟶	P-3 ⟶	P-ALL_SMAC
Partition 1	(1498 s)	Mip cuts flowcovers (661 s)	Read scale (466 s)	Mip strategy bbinterval (466 s)	(180 s)
Partition 2	(1367 s)	Preprocessing reduce (492 s)	Preprocessing repeatpresolve (467 s)	Preprocessing fill (298 s)	(125 s)
Partition 3	(1498 s)	Mip strategy rinsheur (258 s)	Mip limits submipnodelim (188 s)	Mip strategy heuristicfreq (167 s)	(167 s)
Partition 4	(1709 s)	Mip strategy rinsheur (221 s)	Feasopt mode (203 s)	Barrier limits growth (203 s)	(154 s)
Partition 5	(1784 s)	Mip strategy rinsheur (1551 s)	Mip limits aggforcut (1106 s)	Mi p cuts covers (996 s)	(901 s)
Partition 6	(1800 s)	Mip strategy rinsheur (1665 s)	Preprocessing linear (1639 s)	No improvment (1639 s)	(1045 s)
Partition 7	(1800 s)	Mip strategy rinsheur (1675 s)	Mip limits submipnodelim (1637 s)	No improvment (1637 s)	(1637 s)
Partition 8	(1800 s)	No improvements for P-1, P-2, P-3 and P-ALL			
Partition 9	(1800 s)	Simplex pgradient (1625 s)	Barrier crossover (1571 s)	Mp limits aggforcut (1555 s)	(1489 s)
Partition 10	(1785 s)	Mip strategy rinsheur (1268 s)	Mip limits gomorycand (1231 s)	Preprocessing repeatpresolve (1068 s)	(893 s)
Partition 11	(1706 s)	Mip strategy rinsheur (1081 s)	Mip limits aggforcut (366 s)	Mp strategy heuristicfreq (180 s)	(180 s)
Partition 12	(1432 s)	Mip strategy nodeselect (982 s)	Mip strategy rinsheur (898 s)	Mp limits aggforcut (383 s)	(383 s)
Partition 13	(1729 s)	Mip strategy rinsheur (725 s)	Mip limits gomorycand (169 s)	Preprocessing linear (169 s)	(169 s)

Table 3 continued

	P-0_default ⟶	P-1 ⟶	P-2 ⟶	P-3 ⟶	P-ALL_SMAC
Partition 14	(1348 s)	Preprocessing reduce (171 s)	Mip strategy nodeselect (138 s)	Mip strategy variableselect (131 s)	(82 s)
Partition 15	(1149 s)	Mip strategy nodeselect (324 s)	Mip strategy rinsheur (164 s)	Mip limits submipnodelim (115 s)	(66 s)
Partition 16	(1363 s)	Mip strategy heuristicfreq (143 s)	Mip limits aggforcut (95 s)	Mip cuts gubcovers (95 s)	(63 s)
Partition 17	(1274 s)	Mip strategy nodeselect (262 s)	Mip limits aggforcut (117 s)	Simplex limits perturbation (96 s)	(74 s)
Partition 18	(1466 s)	Mip strategy rinsheur (562 s)	Mip strategy startalgorithm (533 s)	Network pricing (327 s)	(327 s)
Partition 19	(1755 s)	Mip strategy rinsheur (784 s)	Preprocessing repeatpresolve (266 s)	Mip strategy nodeselect (225 s)	(149 s)
Partition 20	(1759 s)	Mip strategy rinsheur (1075 s)	Mip strategy nodeselect (797 s)	Preprocessing reduce (486 s)	(486 s)
Partition 21	(1800 s)	No improvements for P-1, P-2 and P-3			(891 s)
Partition 22	(1796 s)	Mip strategy rinsheur (1294 s)	Mip strategy nodeselect (624 s)	Preprocessing fill (589 s)	(589 s)
Partition 23	(1762 s)	Mip strategy rinsheur (1043 s)	Mip strategy nodeselect (773 s)	Mp limits aggforcut (472 s)	(472 s)
Partition 24	(1604 s)	Mip strategy rinsheur (922 s)	Preprocessing reduce (729 s)	Mip strategy nodeselect (473 s)	(473 s)
Partition 25	(1480 s)	Emphasis numerical (139 s)	Mip cuts gomory (117 s)	Perturbation constant (114 s)	(75 s)
Partition 26	(1338 s)	Mip strategy rinsheur (361 s)	Mip limits submipnodelim (126 s)	Mip strategy nodeselect (121 s)	(97 s)
Partition 27	(411 s)	Mip strategy heuristicfreq (205 s)	Emphasis mip (204 s)	Read scale (203 s)	(22 s)

References

Altmann, M., Brenninkmeijer, A., Lanoix, J.-C., Ellison, D., Crisan, A., & Hugyecz, A., et al. (2010). Decentralized energy systems. Technical report European Parliament's Committee (ITRE). http://www.europarl.europa.eu/document/activities/cont/201106/20110629ATT22897/20110629ATT22897EN.pdf. Accessed 29 Sept 2016.

Andriosopoulos, K., Zopounidis, C., & Doumpos, M. (2016). Editorial to the special issue "OR in energy modeling and management". *Computers & Operations Research, 66,* 225–227. https://doi.org/10.1016/j.cor.2015.11.005.

Ansótegui, C., Sellmann, M., & Tierney, K. (2009). A gender-based genetic algorithm for the automatic configuration of algorithms. In D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, et al. (Eds.), *Principles and practice of constraint programming—CP 2009* (Vol. 5732, pp. 142–157)., Lecture Notes in Computer Science Berlin: Springer.

Atamtürk, A., & Savelsbergh, M. W. P. (2005). Integer-programming software systems. *Annals of Operations Research, 140,* 67–124. https://doi.org/10.1007/s10479-005-3968-2.

Avdoulas, C., Bekiros, S., & Boubaker, S. (2018). Evolutionary-based return forecasting with nonlinear STAR models: Evidence from the Eurozone peripheral stock markets. *Annals of Operations Research, 262,* 307–333. https://doi.org/10.1007/s10479-015-2078-z.

Ben-Ayed, O., Blair, C. E., Boyce, D. E., & LeBlanc, L. J. (1992). Construction of a real-world bilevel linear programming model of the highway network design problem. *Annals of Operations Research, 34,* 219–254. https://doi.org/10.1007/bf02098181.

Beraldi, P., Conforti, D., & Violi, A. (2008). A two-stage stochastic programming model for electric energy producers. *Computers & Operations Research, 35,* 3360–3370. https://doi.org/10.1016/j.cor.2007.03.008.

Bergstra, J. S., Bardenet, R., Bengio, Y., & Kégl, B. (2011). Algorithms for hyper-parameter optimization. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems* (Vol. 24, pp. 2546–2554). Red Hook: Curran Associates Inc.

Dantzig, G. B. (1955). Linear programming under uncertainty. *Management Science, 1,* 197–206.

Fawcett, C., & Hoos, H. H. (2016). Analysing differences between algorithm configurations through ablation. *Journal of Heuristics, 22,* 431–458. https://doi.org/10.1007/s10732-014-9275-9.

Fragnière, E., Gondzio, J., & Vial, J.-P. (2000). Building and solving large-scale stochastic programs on an affordable distributed computing system. *Annals of Operations Research, 99,* 167–187. https://doi.org/10.1023/a:1019245101545.

Hutter, F., Babic, D., Hoos, H. H., & Hu, A. J. (2007). Boosting verification by automatic tuning of decision procedures. In *Austin, TX, USA* (pp. 27–34). https://doi.org/10.1109/famcad.2007.9.

Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2010). Automated configuration of mixed integer programming solvers. In D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, et al. (Eds.), *Integration of AI and OR techniques in constraint programming for combinatorial optimization problems* (Vol. 6140, pp. 186–202)., Lecture Notes in Computer Science Berlin: Springer.

Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. In D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, et al. (Eds.), *Learning and intelligent optimization* (Vol. 6683, pp. 507–523)., Lecture Notes in Computer Science Berlin: Springer.

Hutter, F., Hoos, H. H., Leyton-Brown, K., & Stützle, T. (2009). ParamILS: An automatic algorithm configuration framework. *Journal of Artificial Intelligence Research, 36,* 267–306. https://doi.org/10.1613/jair.2808.

IBM. (2016). ILOG CPLEX optimization studio: CPLEX user's manual, version 12 release 6. http://www.ibm.com/support/knowledgecenter/en/SSSA5P_12.6.1/ilog.odms.studio.help/pdf/usrcplex.pdf. Accessed 3 June 2016.

Jones, D. R., Schonlau, M., & Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global Optimization, 13,* 455–492. https://doi.org/10.1023/a:1008306431147.

Khalilpourazari, S., & Arshadi Khamseh, A. (2017). Bi-objective emergency blood supply chain network design in earthquake considering earthquake magnitude: A comprehensive study with real world application. *Annals of Operations Research, 52,* 168. https://doi.org/10.1007/s10479-017-2588-y.

Kobayakawa, T., & Kandpal, T. C. (2016). Optimal resource integration in a decentralized renewable energy system: Assessment of the existing system and simulation for its expansion. *Energy for Sustainable Development, 34,* 20–29. https://doi.org/10.1016/j.esd.2016.06.006.

Kuznia, L., Zeng, B., Centeno, G., & Miao, Z. (2013). Stochastic optimization for power system configuration with renewable energy in remote areas. *Annals of Operations Research, 210,* 411–432. https://doi.org/10.1007/s10479-012-1110-9.

Mascia, F., López-Ibáñez, M., Dubois-Lacoste, J., & Stützle, T. (2014). Grammar-based generation of stochastic local search heuristics through automatic algorithm configuration tools. *Computers & Operations Research, 51,* 190–199. https://doi.org/10.1016/j.cor.2014.05.020.

Owens, B. (2014). The rise of distributed power. General electric (ecomagination). https://www.ge.com/sites/default/files/2014%2002%20Rise%20of%20Distributed%20Power.pdf. Accessed 30 Sept 2016.

Pintér, J. D. (2017). How difficult is nonlinear optimization? A practical solver tuning approach, with illustrative results. *Annals of Operations Research, 31,* 635. https://doi.org/10.1007/s10479-017-2518-z.

Prékopa, A. (1995). *Stochastic programming*. Dordrecht: Springer.

Prékopa, A., Ganzer, S., Deák, I., & Patyi, K. (1980). The STABIL stochastic programming model and its experimental application to the electrical energy Sector of the Hungarian economy. In M. A. H. Dempster (Ed.), *stochastic programming*. London: Academic Press.

Schwarz, H., Bertsch, V., & Fichtner, W. (2018a). Two-stage stochastic, large-scale optimization of a decentralized energy system: A case study focusing on solar PV, heat pumps and storage in a residential quarter. *OR Spectrum, 40,* 265–310. https://doi.org/10.1007/s00291-017-0500-4.

Schwarz, H., Schermeyer, H., Bertsch, V., & Fichtner, W. (2018b). Self-consumption through power-to-heat and storage for enhanced PV integration in decentralised energy systems. *Solar Energy, 163,* 150–161. https://doi.org/10.1016/j.solener.2018.01.076.

Shapiro, A., Dentcheva, D., & Ruszczynski, A. P. (2009). *Lectures on stochastic programming: modeling and theory (MPS-SIAM series on optimization)* (Vol. 9). Philadelphia, PA: SIAM.

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical Bayesian optimization of machine learning algorithms. In F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems* (Vol. 25, pp. 2951–2959). Red Hook: Curran Associates Inc.

Taborda, D., & Zdravkovic, L. (2012). Application of a hill-climbing technique to the formulation of a new cyclic nonlinear elastic constitutive model. *Computers and Geotechnics, 43,* 80–91. https://doi.org/10.1016/j.compgeo.2012.02.001.

Talbi, E.-G. (2016). Combining metaheuristics with mathematical programming, constraint programming and machine learning. *Annals of Operations Research, 240,* 171–215. https://doi.org/10.1007/s10479-015-2034-y.

Velik, R., & Nicolay, P. (2016). Energy management in storage-augmented, grid-connected prosumer buildings and neighborhoods using a modified simulated annealing optimization. *Computers & Operations Research, 66,* 248–257. https://doi.org/10.1016/j.cor.2015.03.002.

Wallace, S. W., & Fleten, S.-E. (2003). Stochastic programming models in energy. In *Stochastic programming* (Vol. 10, pp. 637–677, Handbooks in Operations research and management science). Amsterdam: Elsevier.

Wolfe, P. (2008). The implications of an increasingly decentralised energy system. *Energy Policy, 36,* 4509–4513. https://doi.org/10.1016/j.enpol.2008.09.021.

Yazdanie, M., Densing, M., & Wokaun, A. (2016). The role of decentralized generation and storage technologies in future energy systems planning for a rural agglomeration in Switzerland. *Energy Policy, 96,* 432–445. https://doi.org/10.1016/j.enpol.2016.06.010.

Zokaee, S., Jabbarzadeh, A., Fahimnia, B., & Sadjadi, S. J. (2017). Robust supply chain network design: An optimization model with real world application. *Annals of Operations Research, 257,* 15–44. https://doi.org/10.1007/s10479-014-1756-6.