Deliverable 5.2
Report on assessment of the tool performance and test application results

Project: **PlaMES**
Full Title: **Integrated Planning of Multi-Energy Systems**

Funding:	European Commission, Horizon 2020
Project N°:	863922

Delivery Date:	30/11/2021
Deliverable type:	Report
Dissemination level:	Public

Editors: Klemens Schumann, Philipp Hälsig, Henrik Schwaeppe, Luis Böttcher, Lukas Hein

Version 1.0
Issue Date 30th November 2021

This deliverable may be subject to final acceptance by the European Commission. The content of this deliverable reflects only the authors' view. The European Climate, Infrastructure and Environment Executive Agency (CINEA) is not responsible for any use that may be made of the information it contains.

History of Changes

Version	Date	Change	Page(s)
1.0	30/11/2021	First Version released	

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 863922.

Contents

This project has received funding from the European Union's Horizon 2020
research and innovation programme under grant agreement No 863922.

List of Figures

This project has received funding from the European Union's Horizon 2020
research and innovation programme under grant agreement No 863922.

Chapter 1

Introduction

PlaMES intends to help stakeholders in the energy system to take decisions about how the energy system should be designed in 2050 to meet the carbon reduction goals. Therefore, we have developed the six models and tools as sketched in Deliverable 2.2[1] and shown in Figure 1.1. The models can be used in different combinations. As exemplary use cases, we have defined the Central Use Case and Decentral Use Case. In the Central Use Case, the generation and network expansion for the European energy system can be planned. In the Decentral Use Case, distribution systems can be planned considering different coordination mechanisms for decentral energy systems.

Both use cases in PlaMES can lead to large optimization models. On the one hand, modeling the central European energy system leads to very large data sets and thus to large models. On the other hand, decentral energy systems are rather small compared to the central energy system but need to be modeled with higher granularity. This also leads to large models.

To determine the simulation requirements for the two use cases of PlaMES, the hardware requirements need to be tested. Therefore, in the following chapters the models are tested and evaluated based on their run time, their required RAM and the required number of cores. Each model is tested individually to take into account its specific requirements. In section 2, the tools for the Central Use Case, Decentral Energy System Aggregation (DESA), Central Energy System (CES) and Transmission Expansion Planning (TEP) are tested. In section 3, the tests for the tools for the Decentral Use Case, Decentral Energy System Disaggregation (DESD), Decentral Energy System Operation (DESOP) and Distribution Network Expansion Planning (DNEP) are described.

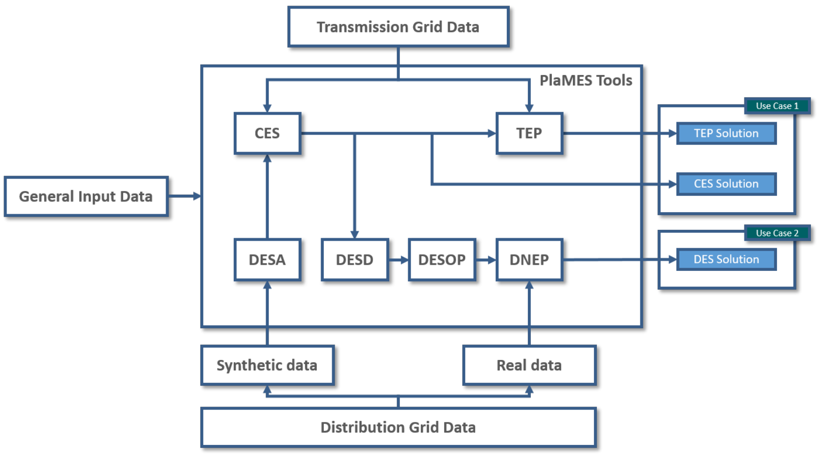


Figure 1.1: Set of tools used in PlaMES.

[1]Mathematical formulation of the model (Deliverable 2.2): `https://cordis.europa.eu/project/id/863922/results`

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 863922.

Chapter 2

Central Use Case Benchmarks

The demonstrated 'Central Use Case' relies strongly on the overall performance of the tools. Compared to the Decentral Use Case, a large quantity of data has to be processed within each model. Furthermore, two of the three models implement mathematical optimization problems to model assumptions about the future energy system. In theory, linear programs classify as P, mixed-integer problems as NP-hard. In practice, however, it says little about the actual run time of the tools. Statements about run time can best be made with actual calculations. Therefore, we introduce test scenarios to run benchmarks. These test scenarios are only introduced to the degree which is required for understanding how the performance is impacted. In addition, input parameters, problem sizes and hardware are varied. The main object of investigation is not the performance of the test cases, but in particular the determination of hardware and information technology limits as well as an outlook on possible problem classes for future projects or model calculations. These performance tests are carried out for Decentral Energy System Aggregation (DESA, section 2.1), Central Energy System Planning (CES, section 2.2) and Transmission Expansion Planning (TEP, section 2.3).

An important part of the PlaMES project is the development of custom solvers, used to solve the CES and TEP models. Unexpectedly, the performance of commercial solvers is still very decent and at their stage of development, the custom solvers do not outperform commercial solvers consistently. Further understanding is required and will be provided in later documents. We use Gurobi, a commercial optimization solver to benchmark the models. Benchmarks were also run with CPLEX, but Gurobi outperformed CPLEX in all comparable requests.

2.1 Decentral Energy System Aggregation

The DESA model is not used to optimize but to aggregate information from decentral energy systems to the central energy system instance. DESA consists of multiple processing steps which are handled independently. Therefore, the data processing on the respective regions in Germany and Turkey are not benchmarked. To derive the distribution system expansion costs, the DNEP model is used to determine the necessary expansion on synthetic networks modeled to representative network areas. The DNEP model and the solution methodology are benchmarked in 3.3.

2.2 Central Energy System Planning

The CES model can be classified as a single-stage generation & transmission expansion planning (G&TEP), implemented as a purely linear program in MATLAB. As outlined in Deliverable 2.2[1], the model minimizes the total annual system cost with respect to (hourly) energy demand and total Greenhouse gas emissions. A first application of the model has been shown in [6]. Substantial additions to the model have been sparse storage implementations (as discussed in [1]) and power flow formulations for the transmission grid (described as "Kirchhoff" in [4]).

[1]Mathematical formulation of the model (Deliverable 2.2): `https://cordis.europa.eu/project/id/863922/results`

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 863922.

2.2.1 Solver setup

As the benchmark scenarios result in linear programs with several 10s of millions of variables and constraints, the problems are forced to be solved with the barrier algorithm (sometimes also described as interior point method) with no cross-over activated[2]. Deactivating cross-over means that the solutions are non-basic and cannot be considered optimal in mathematical terms. However, the solutions are *very* close to the optimal, almost outweighed by numerical issues (see section 2.2.4).

After several tests the barrier tolerance has been set to $1e-5$. In practice barrier tolerance did not make a significant difference in finding good solutions, but impacts the required solver iterations. Due to the indicated numerical issues, substantially better tolerance than $1e-5$ was not possible. Seldom did solutions reach a tolerance of $1e-6$. To exclude numerical issues and enable stable comparisons, barrier tolerance is set to $1e-5$ and will not be varied in any considered benchmark.

2.2.2 Benchmark setup

In total, two basic test cases are run on three different computing setups. In addition, several sensitivities are performed, either by changing configurations of the hardware setup or by altering scenario details, e.g. by changing the amount of time periods. Albeit model build-up can take up to 10 minutes - depending on the problem size - it is neglected and the raw solving duration of the models is being compared.

Test cases

We use two instances for performing calculations. They differ in various parameters, described in Table 2.1. The smaller test case *IEEE-118* is based on and enhanced version of the well-established IEEE-118 node grid. The larger instance *UC1-575* represents the *Central Use Case* or *Use Case 1* of the PlaMES project.

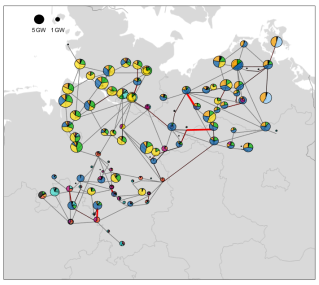
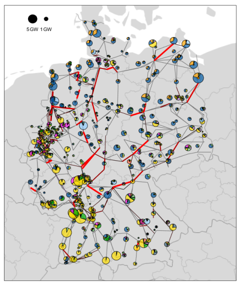
	IEEE-118	UC1-575
		
# of electrical nodes	118	575
# of electrical branches	186	802
# of heat nodes	296	755
# of storages	106	465
# of expansion variables	990	3122
# of time steps	up to 8760	
max # of variables	2.11e7	6.87e7
max # of constraints	2.84e7	9.35e7

Table 2.1: Various test case properties of the CES instances

[2] Making efficient use of the sparse formulations.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 863922.

Test machines

Three types of machines are used. Apart from a local machine, various machines are available at the RWTH High Performance Cluster[3]. For better comparison and smaller instances we use CLAIX2018 machines, as CLAIX2018 offers various nodes of the same machine type. Because CLAIX2018 is limited to 192 GB RAM, larger instances are run on the IAEW cluster. The IAEW computing cluster is also located at the RWTH computing cluster. However, various computing nodes are available. Performance varies depending on the assigned machine. The largest units offer up to 1 TB RAM, which is sufficient for all test performed and can be considered to be the available limit of computing units in practical terms. Although numerous cores are available per cluster, unless stated otherwise, calculations on the High Performance Cluster (HPC) are performed with 16 cores.

	local	High Performance Cluster	
	LAPTOP	CLAIX2018	IAEW
cpu	Intel Core i7-7700HQ 2.80GHz (4 cores)	2x Intel Skylake 24 cores	up to 2x56 threads (various)
ram	32 GB	192 GB	up to 1 TB
os	Windows 10	Linux	Linux

Table 2.2: Computing units used to run CES benchmarks

2.2.3 Performance results and comparison

For an outlook that is as complete as possible, a wide variety of instances are compared with each other. The time to find an optimal solution depends on the use case and the machine it is being performed on. However, the performance under the best-known conditions are given in Table 2.3.

Instance	**IEEE-118**	**UC1-575**
LAPTOP (4 cores)	**3:42h** (13331 sec)	Does not compute
CLAIX2018 (16 cores)	**2:10h** (7548 sec)	Does not compute
IAEW (16 cores)	**1:26h** (5163 sec)	**11:25h** (41092 sec)

Table 2.3: CES best-case wall-clock run time for finding an optimal solution at 8760 time-steps

Required memory

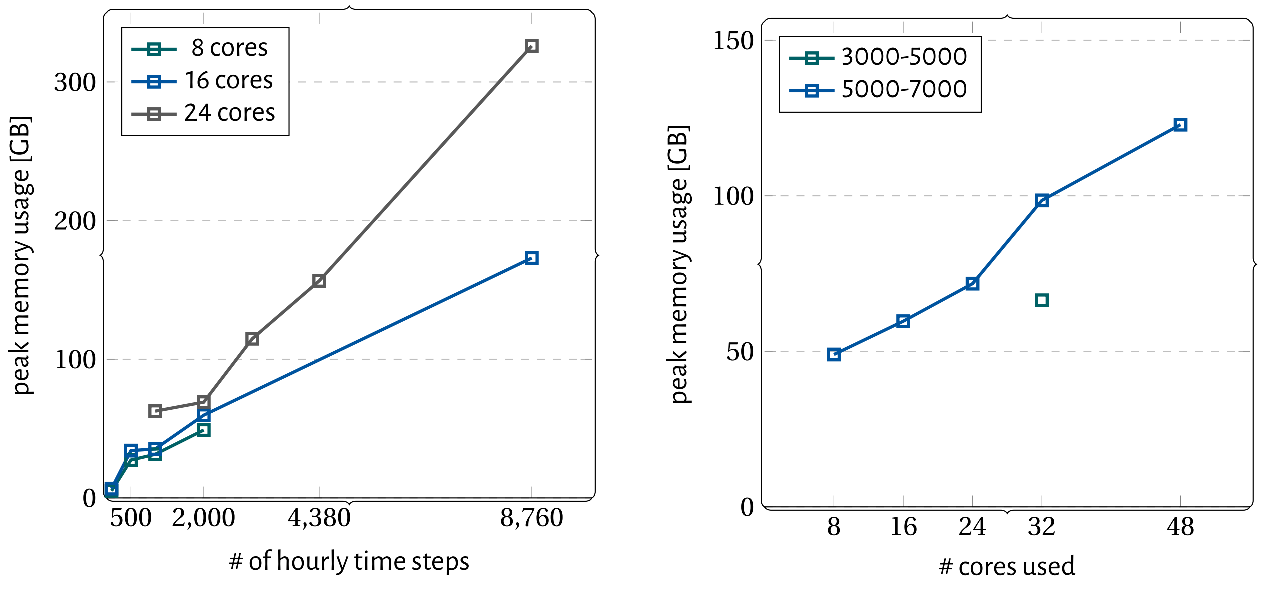
The memory requirement of *UC1-575* very much depends on the number of time steps and number of cores used to solve the problem, as shown in Figure 2.1a. While the dependence on time steps can be expected, further investigations regarding the memory-per-core requirements have been undertaken comparing the same 2000 time steps (and another one) as shown in Figure 2.1b. It appears that memory requirements are not only impacted by the two factors, but also by certain properties that relate to the scenario, i.e. how much the solutions space is confined/constrained. Memory requirements can be determined upfront. Nonetheless, when scaling the properties of timesteps/threads/scenarios we do not experience polynomial or even exponential growth, but rather linear growth. Even though the actual memory requirements cannot be determined, linear growth means that the models can be scaled. Worldwide, large computing nodes are typically specced with 512GB-1TB of RAM, meaning, memory-wise the models can still be scaled by 2 to five times without reaching the limits of available information technology. However, theoretically, much larger instances are still possible.

Performance of UC1-575 under various inputs

Run time of the models can be measured in two ways: wall-clock and cpu time. Although cost-wise cpu time may be an interesting benchmark, wall-clock time measures after what time a result can be expected. To our surprise, wall-clock run time

[3]`https://www.itc.rwth-aachen.de/cms/IT-Center/Forschung-Projekte/High-Performance-Computing/~eucm/Infrastruktur/?lidx=1` (more information)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 863922.



(a) Memory requirements on CLAIX2018 & IAEW

(b) Memory usage at 2000 time steps on CLAIX2018

Figure 2.1: CES memory tests with *UC1-575* on the High Performance Cluster

varies strongly under various conditions, as shown in Figure 2.2. We have not only included the same instance and varied the number of cores, but over the course of the project also did some sensitivity analysis that do not change the properties given in 2.1. However, changing the cost of storage can lead to significant performance improvements - probably due to properties of the presolver - or worse, performance drops. However, instances perform to the expected degree and can be solved within 24 hours.

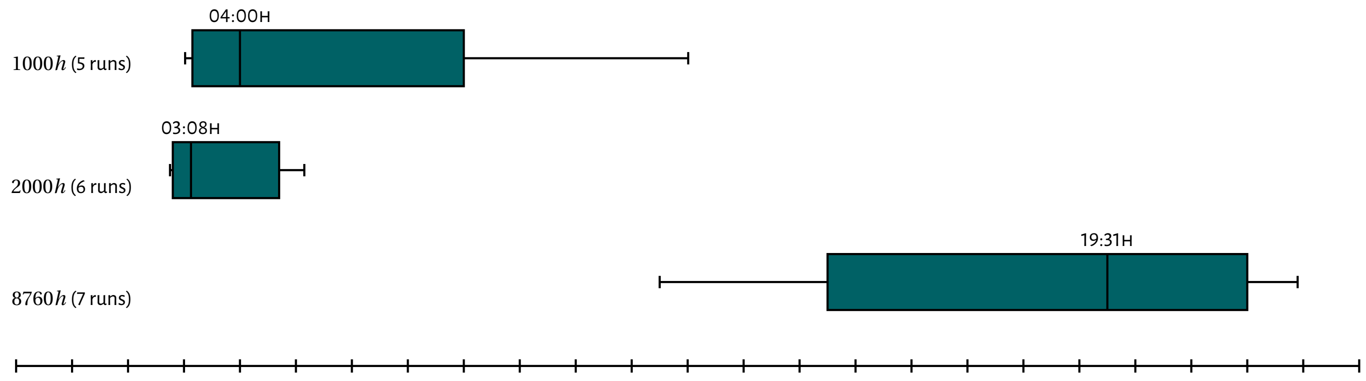


Figure 2.2: CES wall-clock run time of UC1-575 under several variations/sensitivities

Only assumptions can be made about the performance when scaling up the model size. Most likely and based on the constrained knowledge we have of the barrier algorithm, "increasing the model size by a factor of 2" should lead to an "increase of the solving time by a factor of 2". Unfortunately, we are not able to determine which model properties exactly contribute to increasing the model.

On multi-core performance

Several test runs had shown that Gurobi performed best with 16 cores available. Memory requirements and job-time usually increased with more than 16 cores, no matter the size of the instance. Up to 16 cores, adding more cores decreases run time and justifies additional memory requirements. Conformably, adding more than 16 cores results in higher memory requirements, but not necessarily in better job performance, hence higher energy consumption and waste of computing resources. Exemplary results are shown in Figure 2.3.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 863922.

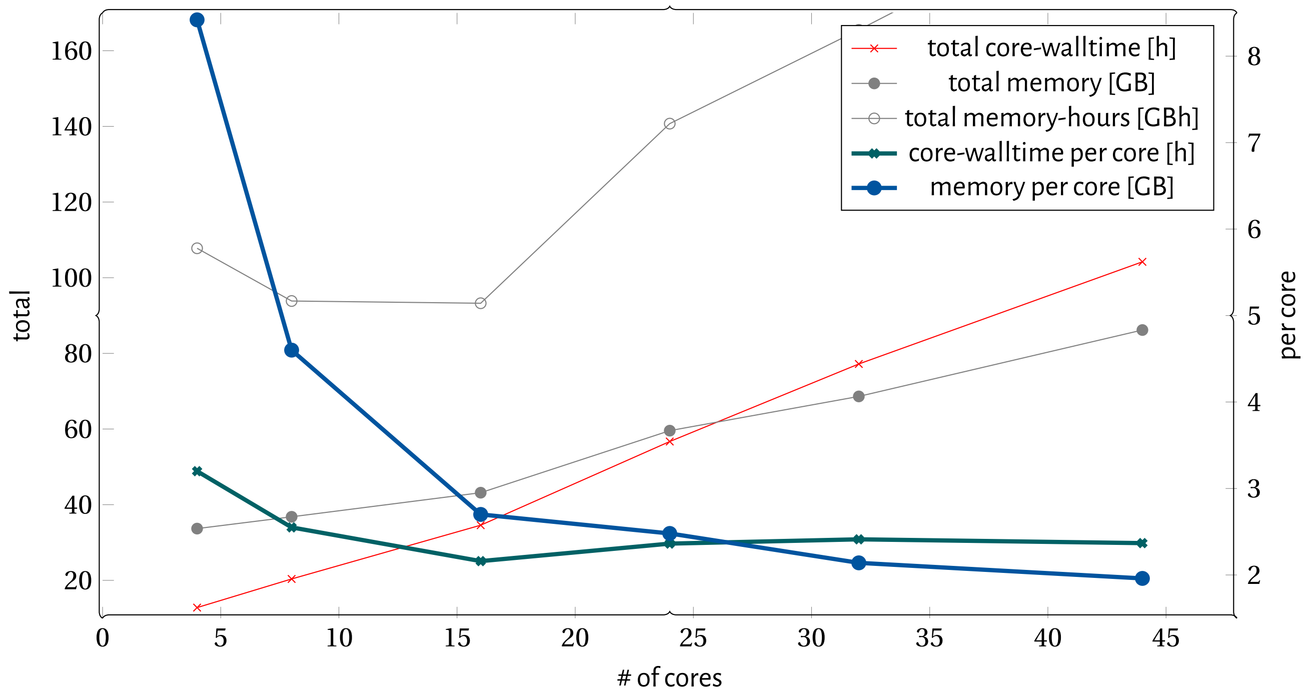


Figure 2.3: CES clock and memory performance of *IEEE-118* (8760h) on CLAIX2018

Assumably, Gurobi has been optimized to run with 16 cores and better performance is theoretically possible, especially with even larger instances. More research is required to understand how these operations are performed and how they can be improved in future instances.

2.2.4 Numerical issues

Variables and constraints in the model must be scaled to perform within the numerical boundaries of the solver. Gurobi suggests to keep the numerical range tight. Furthermore they should not exceed the numerical range of $1e-3$ and $1e6$ [4]. However, numerical performance cannot be guaranteed. Some constraints, e.g. Greenhouse gas emissions are measured in million tons. If the maximal limit was at 100 million tons or $1e8$ tons, the maximal CO2 emission that can be considered per variable is $1e-1$ tons or 100 kg; hence, Greenhouse gas emissions below 100 kg per MWh cannot be considered without disregarding numerical advice. When scaling up the model one has to decide whether such values should be neglected or whether numerical imprecision can be accepted. At this stage it is uncertain whether this will effectively become a problem or whether it can be ignored. In any case, although technically not state-of-the-art, increased numerical precision (e.g. to 128-bit) will solve the problem in the long run.

2.2.5 Outlook on future performance

Several tests have been performed on several machines and under varying conditions. With the presented test cases, the absolute limits of information technology could not be reached. The larger instance can be solved in less than a day.

Without committing to details, the instances can still become several times larger without hitting memory-limits, especially when the number of cores are being reduced. However, predicting the solving duration is more difficult. Solving duration is affected by the solution space, the machines, the number of cores and the overall performance of the solver. It is also unknown how numerical issues affect the solving duration of the model and their overall impact at larger scale. In any case, the shown instances have not been reduced in time or aggregated to a smaller set of nodes. Hence, using established reduction measures should allow the model to easily scale up to an European observation area.

[4] `https://www.gurobi.com/documentation/9.1/refman/guidelines_for_numerical_i.html`

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 863922.

2.3 Transmission Expansion Planning

Besides the generation aspect of a multi-energy system the transmission of the energy is of great importance. Hence, the Transmission Expansion Problem (TEP) determines a cost efficient way for grid expansion respecting expansion measures as well as remedial measures such as redispatch. The extensive expansion problem is based on the results of the CES model and the resulting power flows. The TEP is implemented as a mixed-integer linear problem in MATLAB. Deliverable 2.2[5] and shown in Figure 1.1 outlines the objective of the TEP model to be the minimization of the investment and operating costs of the grid expansion.

2.3.1 Test cases

Due to the sequential nature of the CES and the TEP model, we use the same two instances for performing calculations of the TEP as for the CES. They differ in various parameters, described in Table 2.4.

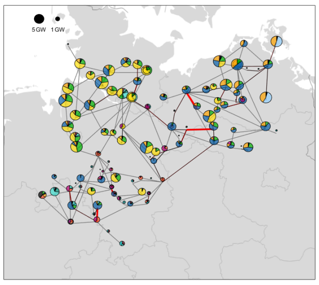
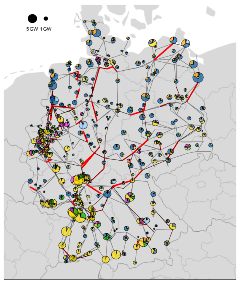
	IEEE-118	**UC1-575**
		
# of electrical nodes	118	575
# of electrical branches	186	802
# of expansion variables	872	3984
max # of variables	16047	60040
max # of constraints	40185	126434

Table 2.4: Various test case properties of the TEP instances

2.3.2 Performance results and comparison

Finding an optimal solution depends highly on the chosen machine and the test case. Especially, the number of cores used for the solving process as well as the number of nodes in the used case impacts the solving time. Furthermore, the number of included grid snapshots influence the solving time of the model. The performance under the best-known conditions are displayed in Table 2.5. Although the small IEEE-118 instance is calculated on the local machine with 4 cores as well as on the CLAIX2018 high performance cluster with 24 cores within acceptable time, the bigger instance with 575 nodes cannot be solved on the local machine. It also requires about 13 hours to determine a solution on the high performance cluster.

Instance	**IEEE-118**	**UC1-575**
LOCAL (4 cores)	**2:18h** (8310 sec)	Does not compute
CLAIX2018 (24 cores)	**0:43h** (2622 sec)	**12:54 h** (46498 sec)

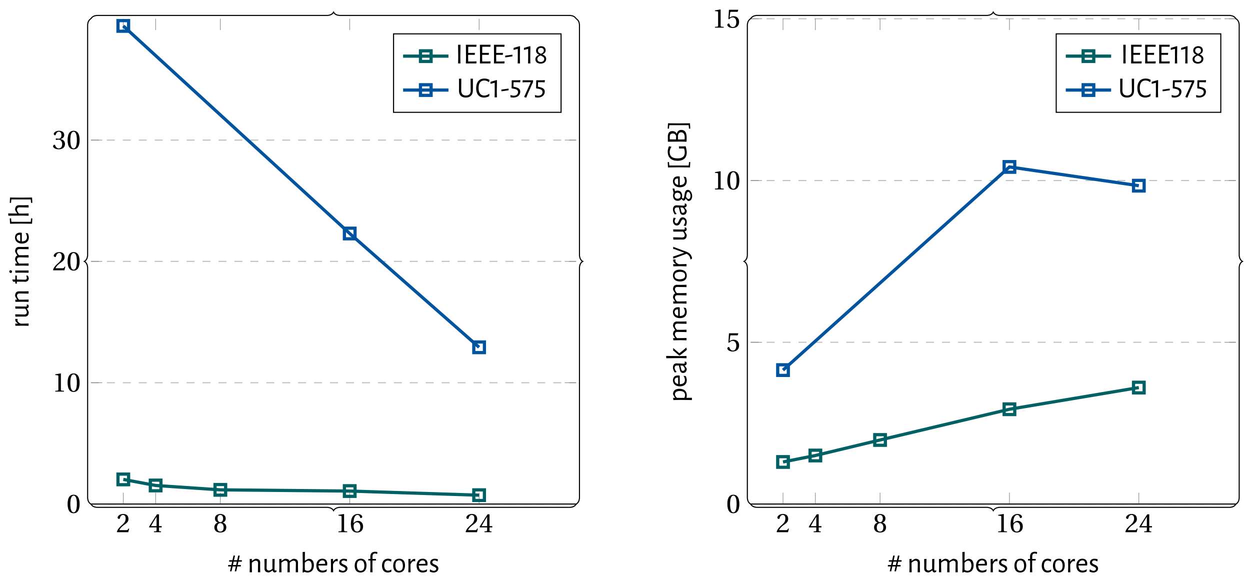
Table 2.5: TEP best-case wall-clock run time for finding an optimal solution respecting one grid snapshot

[5] Mathematical formulation of the model (Deliverable 2.2): `https://cordis.europa.eu/project/id/863922/results`

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 863922.

Computing Time Performance and Required Memory

Figure 2.4 illustrates the run time requirement and memory usage for solving the expansion problem on the IEEE-118 grid model and the exemplary central use case data with 575 nodes. In both cases, the expansion model respects the grid utilisation of one grid snapshot. The solvability of the expansion model using more grid snapshots is not necessarily given. Due to numerical issues explained in Section 2.3.3 the feasibility of these instances is not stable . The figure points out, that the impact of the acquired cores is significant. Looking at the bigger use case, about two thirds of the run time can be saved using 24 instead of two cores. The allocation of more cores comes with the need of more peak memory usage. However, it can be seen that the overall required memory is not so high for the transmission expansion model that it reaches limitations of the high performance cluster.



(a) Run time on CLAIX2018

(b) Memory usage on CLAIX2018

Figure 2.4: TEP run time and Memory usage on CLAIX2018

2.3.3 Numerical issues

As described in Section 2.2.4, the numerical limits for optimisation problems are tight using state-of-the-art solvers (e.g. Gurobi). Expansion models implemented as mixed-integer linear have to regard constraints which only arise in case an expansion measure is chosen. For example, if a new transmission line is built the power flow over this line must not exceed the limit of the new transmission line. Such constraints are implemented using the so-called Big M method, introducing a big value which activates and deactivates the constraint depending on the binary decision variable. This results in comparison of big values in the order of $1e6$ with values in the order of $1e-2$. By determining individual big M values and choosing the value only as high as it has to be this problem can be mitigated but not necessarily be solved.

2.3.4 Outlook on future performance

Due to the high complexity of the TEP model, the limits of solving the problem in a reasonable time are reached quite fast. The potential computational gain from the solver developed by the University of Bologna within the PlaMES project is not easy to determine. However, the target of PlaMES remains to be able to solve large instances that cannot be solved with state-of-the-art solvers.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 863922.

Chapter 3

Decentral Use Case Benchmarks

While the 'Central Use Case' deals with very large systems leading to large models, the 'Decentral Use Case' is simulated in very high granularity. The models DESD and DESOP simulate on building-level considering each technology of the buildings (generators, loads and flexibilities) for different commodities. DNEP can also simulate on building level when LV-grids should be expanded. This high granularity can also lead to large models. To enable decision makers to base decisions on the results of the Decentral Use Case, multiple parameter variations should be simulated. Therefore, the tool performance will be assessed in the following chapter.

3.1 Decentral Energy System Disaggregation

The DESD model disaggregates the expansion planning results of the CES model to building level. This is done by assigning technologies to buildings and dimensioning them according to the building's attributes and load profile. The model consists of different algorithms for data preparation and distribution of technologies to buildings.

While the number of buildings can grow very large, the distribution is performed sequentially which yields good scalability. The decision to assign a technology to a building does not affect previous assignments and only affects the remaining capacity of that technology to be installed. This allows most of the distribution to be handled through matrix operations, for which MATLAB offers very good performance, and ensures a linear run time complexity.

After assigning a technology, the installed capacity of that technology has to be determined according to certain building attributes. For example solar panels are dimensioned according to the roof size and batteries in return are dimensioned according to the previously installed capacity of solar panels, both of which are mathematically simple conditions. However, other technologies can be dependent on more complex constraints, for example heating solutions in households. A typical heating solution could consists of two different technologies, one for covering the base heat load and another for peak load. What makes this complex is the fact that in households, heating technologies do not only have to cover the room heat demand, but also warm water demand. And while the room heat demand is responsible for most of the thermal energy demand of households, the warm water demand is responsible for demand peaks. In order to distribute realistic capacities for base and peak heating, the specific combination of room heat and warm water demand profiles need to be considered with the respective annual thermal energy demand for scaling of the time series, which can be unique for each household. This could lead to calculations involving large time series for each building and drive up run time considerably.

For these cases some data preparation can lead to drastically improved performance. In the case of household heating solutions, the model uses the fact that only a couple hundred of time series profiles exists to its advantage. By pre-calculating specific values based on all profile combinations, we only have to access these values when distributing a technology to a building based on its room heat and warm water profile ID. These time series profiles are based on user behavior in a specific country, for example Germany. This means that while the scope of a use case can change - only a region of a country or the whole country - and therefore the number of buildings, the number of profiles stays the same. These data preparation results can also be cached to be used in further model runs to reduce run time even further.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 863922.

In the following benchmarks, only the number of buildings will be varied, since this is the parameter which is expected to vary the most between different use cases and distribution results cannot be pre-calculated and cached.

3.1.1 Benchmark setup

Base test case

The base test case scenario consists of 3.000 buildings, of which 1.000 buildings are part of the three sectors household, commerce, trade and services (CTS) and industry each. Technologies to be distributed include but are not limited to PV, electrical batteries, heatpumps, CHPs and other heating technologies. Target values for expansion planning results are chosen arbitrarily, but realistically, and are not based on calculated results by the CES model, as the specific values should not have a significant impact on the run time.

Test machine

Since the hardware requirements of DESD are expected to be comparatively low, the tests are performed locally on a laptop with the following specifications:

	Local Laptop
CPU	Intel Core i7-8665U @ 2,11GHz (4 cores)
RAM	32 GB
OS	Windows 10

Table 3.1: DESD: computing unit used to run benchmarks

3.1.2 Performance results

To test the performance of the DESD model, the amount of buildings is increased significantly from 3.000 up to 3.000.000 buildings in steps of 600.000 (or in other words, a factor of 200). The target values are scaled accordingly. The results are displayed in Figure 3.1, which shows that the model scales linearly with the amount of buildings. Even at 3 million buildings the run time stays below 2min on local hardware. The RAM requirements are quite low compared to modern hardware with around 2GB of peak RAM usage for the 3 million building case.

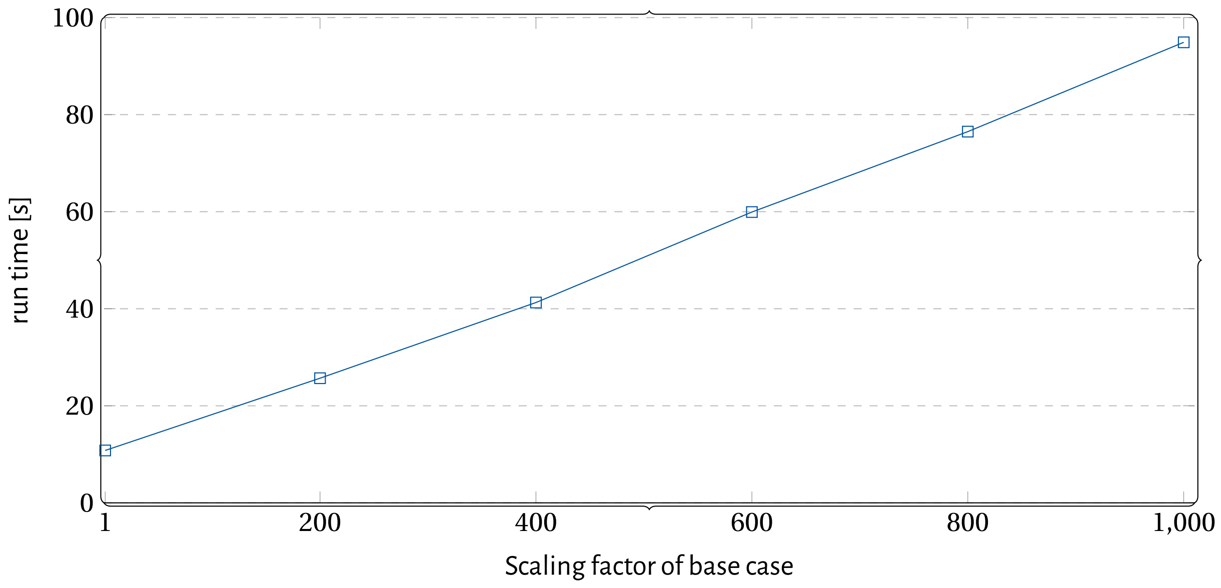


Figure 3.1: DESD run time on local hardware

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 863922.

3.1.3 Outlook on future performance

Due to the already short run time, the near-term development of the DESD model does not need to focus on increasing performance but rather expanding and improving the functionalities. Nonetheless, there are two major options which could improve the performance even further. The first option is implementing parallel computing. Since the target values are given by region, for example postal code, the disaggregation could be performed in multiple regions in parallel, without them affecting each other. The second option would be to run the model on a high performance cluster as demonstrated with the other models, which is also planned as part of the PlaMES project.

3.2 Decentral Energy System Operational Planning

The DESOP is a linear optimization problem performing the scheduling for customers, generation plants and flexibilities in decentral energy systems allowing to coordinate them. The coordination can be performed through energy sharing. While the target of the model is to simulate whole years, the model can be decomposed using the rolling-time-horizon-simulation. Thereby, each day is simulated on its own with an overlap that ensures the consideration of the next day for flexibility scheduling. The mathematical formulation of DESOP is described in Deliverable 2.2[1].

3.2.1 Benchmark setup

Test case

The test case is a set of multiple buildings from the sector households that have been generated with the model presented in [2]. For each building, its assets consisting of generation plants (PV), loads and flexibilities (battery storages, electric vehicles) are considered. The buildings can be coordinated with energy sharing, thus adding a coupling constraint between all buildings (comp. Deliverable 2.2).

Test machines

Similar to the central energy system, the decentral energy system is benchmarked on CLAIX2018 machines at the RWTH High Performance Cluster (comp. section 2.2.2).

3.2.2 Performance results and comparison

The model size of DESOP mainly depends on two parameters: The amount of time steps and the amount of buildings simulated. Besides those parameters, the solving time also depends on the algorithm used and the number of cores used. Therefore, we will first compare the solvers and amount of cores before investigating the impact of the amount of time steps and buildings.

Solving algorithm

Since DESOP is formulated as linear program, we compare the simplex algorithm, barrier algorithm with crossover and barrier without cross-over activated. As discussed in section 2.2.1, deactivating the cross-over does not lead to basic solutions. Besides varying the algorithms, we compare the commercial solvers CPLEX and Gurobi. The comparison of solving time can be seen in Figure 3.2. We simulated 5,000 buildings and 24 time steps using 4 cores. Since the solving time of the barrier algorithm with no cross-over is below one minute, we continue to use the barrier algorithm in the following simulations.

Amount of cores and required memory

After defining Gurobi Barrier as default algorithm, the performance dependency on the number of cores used needs to be tested. Figure 3.3 depicts the solving time and maximal memory usage for a simulation of $10,000$ buildings and 196 time steps with between 2 and 24 cores used. While increasing the number of cores used from 2 to 4 cores can improve the run

[1]D2.2: "Mathematical formulation of the model" (Link: `https://cordis.europa.eu/project/id/863922/results`)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 863922.

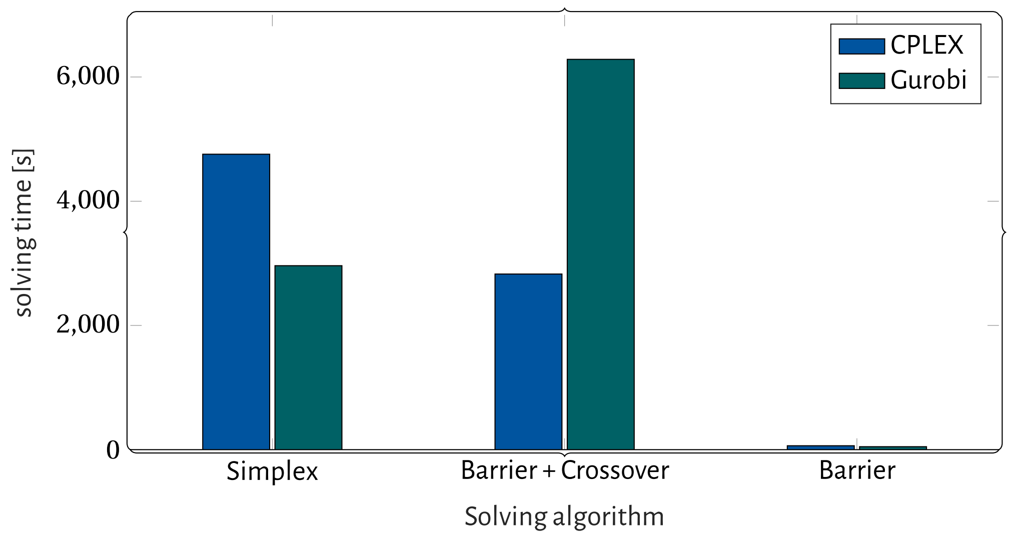


Figure 3.2: DESOP solving time of different solver algorithms

time significantly, further changes do not have significant impact on the solving time. The maximal memory usage does not change with the number of cores.

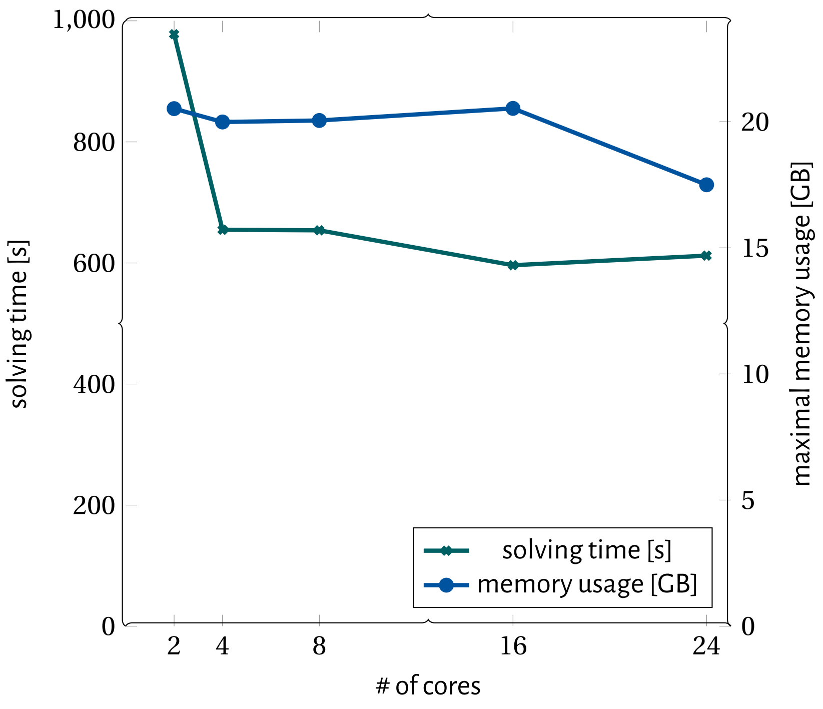


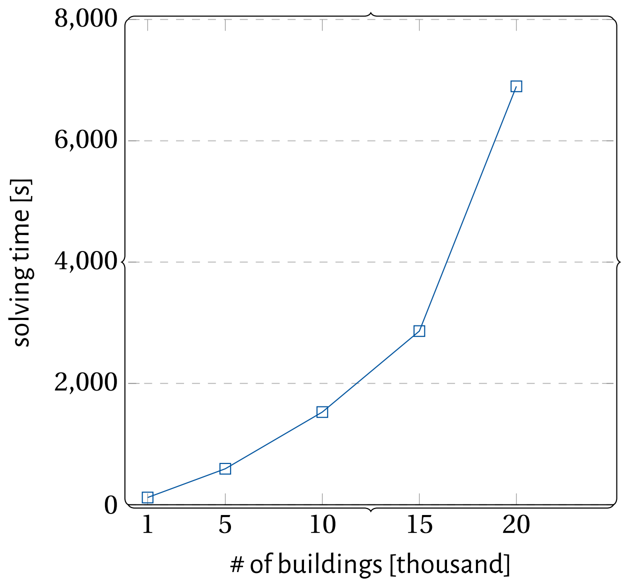
Figure 3.3: DESOP run time and memory tests

Varying the model size

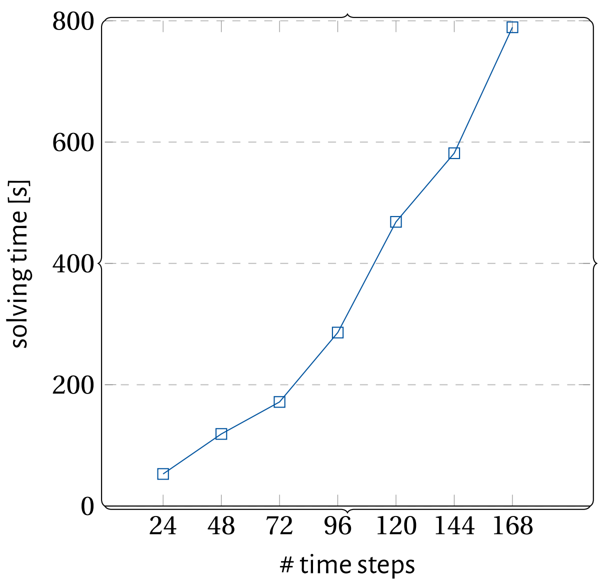
Finally, we test the influence of the model size on the run time. The model size depends linearly on the number of time steps and participants. Doubling the number of time steps leads to a doubling of the number of variables and constraints. Likewise, a doubling of the number of buildings - depending on the equipment of the buildings - leads approximately to doubling the variables and constraints. The results of the testing are depicted in Figure 3.4. The simulations in the left sub plot (varying the number of buildings) have been performed with 168 time steps. The simulations in the right sub plot (varying the number of time steps) have been performed with 10,000 buildings. For both simulations, 8 cores were used.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 863922.

The test results show that the required solving time increases disproportionately with the model size.



(a) Solving time at different numbers of buildings



(b) Solving time at different numbers time steps

Figure 3.4: Solving time test for different model sizes of DESOP

Applying white noise to improve run times

DESOP's objective function consists of the fuel costs for the technologies and the costs to procure or market energy sources (electricity, district heating, gas). If the same prices are used as a basis for the participants, this can lead to the optimization problem not having a unique optimal solution. This can explain the high run time differences between the Simplex algorithm and the Barrier algorithm. One solution could be to apply white noise to the prices in the objective function. Figure 3.5 depicts how the run time of the use case behaves with 4,000 buildings and 24 time steps using Gurobi for different algorithms with and without white noise. As can be seen, the delay of Simplex and Barrier + Crossover with white noise drops into the similar range as for Barrier. The Barrier algorithm also becomes about 25% faster with white noise. In summary, Simplex and Barrier + Crossover still solve slower than Barrier, but with white noise they need only 160% of the run time of Barrier with white noise.

3.2.3 Outlook on future performance

As shown in the previous chapter, the runtime of DESOP is already in ranges that make simulation of large use cases (up to tens of thousands of buildings) possible when using rolling-time-horizon. However, it might also be possible to reduce the runtime further by using appropriate decomposition methods. This could allow a simulation of even larger areas and suspend the incipient overlinear behavior from 20,000 buildings.

3.3 Distribution Network Expansion Planning

The Distribution Network Expansion Planning (DNEP) determines a network expansion solution for medium and low voltage electrical distribution grids. The necessary measures to be integrated to the distribution grid are based on the operational planning defined by DESOP. As solution methodology a hill climbing algorithm determines the most cost effective measures to be implemented. Deliverable 2.2[2] outlines the mathematical formulation taken into account to the minimisation of the investment and operating costs in the distribution grid.

[2] D2.2: "Mathematical formulation of the model" (Link: `https://cordis.europa.eu/project/id/863922/results`)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 863922.

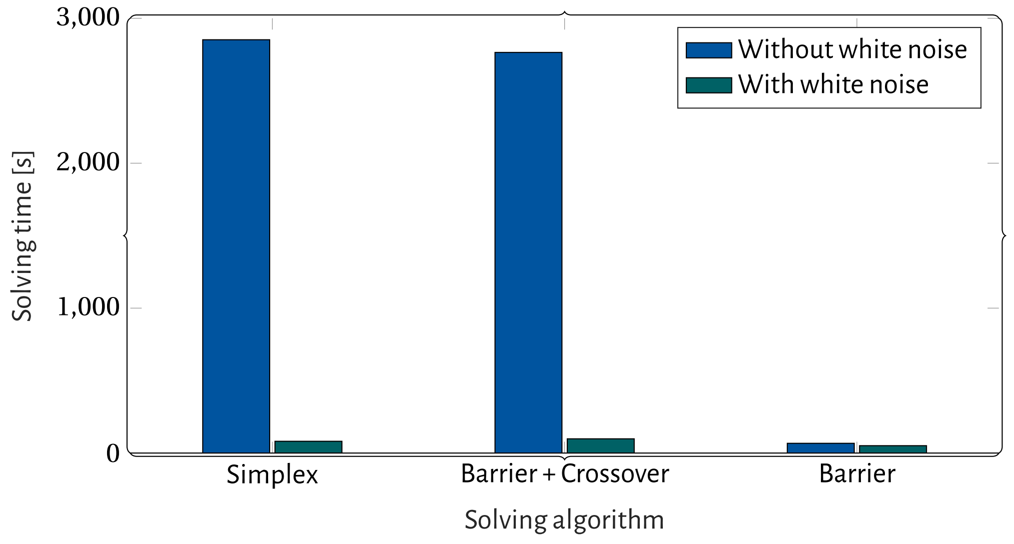


Figure 3.5: DESOP solving time with and without white noise

3.3.1 Benchmark Setup

Benchmark networks

For the testing of the implemented model two benchmark networks were determined. For testing the grid expansion model on medium voltage grids, one synthetic network from the SimBench [5] data set was chosen. To have a more realistic case in outlook to the second use case in PlaMES the pandapower [7] MV-Oberrhein network was chosen as a second grid to be expanded by the DNEP model.

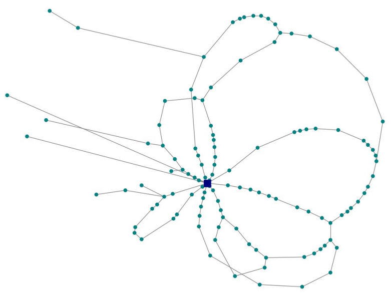
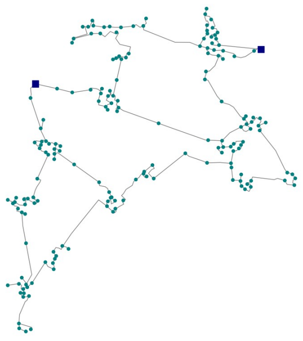
	SimBench MV-Semiurban	**MV-Oberrhein**
		
# of busses	120	179
# of loads	122	147
# of generators	123	153
# of storages	114	-
# of switches	8	-
# of lines	126	181
# of transformers	2	2

Table 3.2: Benchmark Distribution Grids

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 863922.

Possible Expansion Measures

In order to expand the networks, a solution space must first be defined. A solution is a collection of measures that can be taken on the grid. First, the network is considered as a complete graph, which is defined by the fact that each node is connected via an edge. This results in a start node and an end node for each line. The shortest path between the two now defines the line. This can subsequently only be exchanged in its entirety. In addition, this consideration enables the separation of lines at, for example, half of the length.

Conventional expansion measures taken into account:

- Replacement of the entire line
- Parallel line for the entire line
- Separation of the line
- Replacement of the transformer
- Parallel transformer

These measures are based on typical DSO planning principles. For the type of exchange or reinforcement measures a choice of assets can be made from various standard types of assets.

Operational expansion measures taken into account:

- $cos\Phi$-fix control
- $Q(V)$ control
- $Q(P)$ control
- Battery Storage System
- Voltage Regulating Distribution Transformer

These operational measures are following the planning principle of network optimization before reinforcement and expansion.

The combinations of all possible measures in one grid then defines the solution space of DNEP. For the model performance assessment a standard set of measures was defined to be analysed. In this set we defined a classical network expansion approach in which we took three conventional expansion (replacement of entire line, separation of the line, replacement of the transformer) measures and one operational measure ($cos\Phi$-fix control) to be analysed in a timeframe of 24h on the SimBench MV-Semiurban grid and on 1h on the MV-Oberrhein grid.

3.3.2 Performance results of solution methodology

To give the reader an overview on the complexity of the solution methodology first the used approach is briefly presented before the performance of the model is evaluated.

Hill climbing solution methodology

The heuristic solution method for DNEP is the so-called hill climbing. With this approach a better solution is iteratively searched for in a defined neighbourhood using a starting solution [3]. Based on this new solution, a new neighbourhood is again examined for a better solution. An overview of the hill climbing process is given in **Algorithm 1**.

The solution space is defined by the set of measures. Since each measure represents a binary decision variable, the solution space S increases exponentially with the number of measures [8].

$$|S| = 2^{|M|} \tag{3.1}$$

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 863922.

Algorithm 1 Hill Climbing Solution Methodology

```
procedure HillClimbing($s_0, M$)
    $s^* \leftarrow s_0$
    repeat
        Choose $s^{'} \epsilon N(s^*, M)$
        if $c(s^{'}) < c(s^*)$ then
            $s^* = s^{'}$
        end if
    until $c(s^{'}) \leq c(s^*), \forall s \epsilon N(s^*, M)$
    return $s^*$
end procedure
```

The solution space is constrained by various boundary conditions, which are respective planning assumptions. For example, limits apply to maximum line loading or the number of unconnected stations. The objective function $c(s)$ is to be minimised. The neighbourhood search is represented by $N(s, M)$, where s represents a current solution, while M represents all possible measures. The neighbourhood of a current solution is created by three different operations:

(1) Adding a measure to the current solution

(2) Removing a measure from the current solution

(3) Exchange of a measure from the current solution

The totality of all possibilities that arises from these changes to the current solution is called neighbourhood.

In order to apply the hill climbing method, a solution space must first be defined. A solution is a collection of measures that can be integrated to the grid, therefore the solution space increases exponentially with the number of possible measures according to the formula 3.1.

Computing Time Performance and Required Memory

The performance of the tool was carried out on two machines to analyse the impact of computing power to the problem solution methodology.

Test machines
The tests were performed locally on a laptop and on a terminal server machine with the following specifications:

	Local Laptop	Terminal Server
CPU	Intel Core i7-8565U @ 1,99GHz (4 cores)	Intel Xeon Gold 6226 @2.70GHz
RAM	16 GB	255 GB
OS	Windows 10	Windows Server 2019 Standard

Table 3.3: Computing units used to run DNEP benchmarks

Run time
The performance of this approach highly depends on the test case. Especially, the number of measures taken into account and handed to the solution process as well as the number of nodes in examined grid has an impact on the solving time. Furthermore, the number of included time steps (grid snapshots) influence the solution time. The performance of the DNEP model is displayed in Table 3.4.

CPU and RAM
The use of CPU and RAM, measured in the execution phase, shows, that the hill climbing and neighbourhood search do not require large computing instances in the way the model is implemented at this moment. But tests on larger problem

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 863922.

Instance	**Simbench MV-Semiurban**	**MV-Oberrhein**
LOCAL (4 cores)	**387 sec**	**133 sec**
TERM (24 cores)	**404 sec**	**86 sec**

Table 3.4: DNEP run time for finding an optimal solution for same expansion configuration

instances and networks will certainly require a revision of the solution methodology as it is implemented right now. RAM and CPU usage for the two exemplary cases is shown in the following table.

Instance	**Simbench MV-Semiurban**		**MV-Oberrhein**	
	RAM	CPU	RAM	CPU
LOCAL (4 cores)	**1123MB**	**14%**	**236MB**	**26%**
TERM (24 cores)	**2423MB**	**5%**	**278MB**	**6%**

Table 3.5: DNEP CPU and RAM usage for finding an optimal solution for same expansion configuration

3.3.3 Outlook on future performance

Since the solution time of the DNEP model is highly dependent on the size of the problem, in future development of the tool the solution method should be revised. A parallel approach to the hill climbing algorithm or a genetic algorithm are commonly used in these modelling approaches. These approaches could enhance the tool performance in regards to more efficiency (CPU- and RAM-wise) and result in less computing time.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 863922.

Chapter 4

Conclusion

Out of the five benchmarked tools, three (CES, TEP, DESOP) are built around optimization problems that make use of linear programming. The approaches to finding a solution vary, but all in all, current solvers offer successful approaches to finding solutions. DESOP will not reach practical limits with respect to the expected case sizes. Then again, CES and TEP models could soon reach their computational or practical limit and require additional decomposition techniques, provided that model reduction is to be avoided. The other tools make use of a hill climbing heuristic (DNEP) or do not make use of optimization problems at all (DESD) and allow for extensive scaling without changing the methodological approach.

The most important result of the benchmarks shown is that all of the models developed are computable and should remain computable even after scaling to larger study areas. This is not to say that there is no further potential for improvement. However, at this stage of development the authors/developers of the tools and this publication are confident that the problems are predictable and can be solved. Required decomposition techniques for CES and TEP are in development and will be presented in the near future.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 863922.

Bibliography

[1] Karl-Kiên Cao et al. "Classification and Evaluation of Concepts for Improving the Performance of Applied Energy System Optimization Models". In: *Energies* 12.24 (2019), p. 4656. DOI: 10.3390/en12244656.

[2] Wilhelm Cramer et al. "A simulative framework for a multi-regional assessment of local energy markets – A case of large-scale electric vehicle deployment in Germany". In: *Applied Energy* 299 (2021), p. 117249. ISSN: 0306-2619. DOI: 10.1016/j.apenergy.2021.117249. URL: https://www.sciencedirect.com/science/article/pii/S0306261921006693?dgcid=coauthor.

[3] Michel Gendreau and Jean-Yves Potvin. *Handbook of Metaheuristics*. Vol. 146. Boston, MA: Springer US, 2010.

[4] Jonas Hörsch et al. "Linear optimal power flow using cycle flows". In: *Electric Power Systems Research* 158.3 (2018), pp. 126–135. ISSN: 03787796. DOI: 10.1016/j.epsr.2017.12.034.

[5] Steffen Meinecke et al. "SimBench—A Benchmark Dataset of Electric Power Systems to Compare Innovative Solutions based on Power Flow Analysis". In: *Energies* 13.12 (June 2020), p. 3290. DOI: https://doi.org/10.3390/en13123290.

[6] Henrik Schwaeppe et al. "Generation and Transmission Expansion Planning With Respect to Global Warming Potential". In: *2021 IEEE Madrid PowerTech*. IEEE, 2021, pp. 1–6. ISBN: 978-1-6654-3597-0. DOI: 10.1109/PowerTech46648.2021.9494990.

[7] L. Thurner et al. "pandapower — An Open-Source Python Tool for Convenient Modeling, Analysis, and Optimization of Electric Power Systems". In: *IEEE Transactions on Power Systems* 33.6 (Nov. 2018), pp. 6510–6521. ISSN: 0885-8950. DOI: 10.1109/TPWRS.2018.2829021.

[8] Leon Thurner. "Structural Optimizations in Strategic Medium Voltage Power System Planning". PhD thesis. Kassel: University of Kassel, 2018.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 863922.

Abbreviations

CES	Central Energy System
DESA	Decentral Energy System Aggregation
DESD	Decentral Energy System Disaggregation
DESOP	Decentral Energy System Operation
DNEP	Distribution Network Expansion Planning
TEP	Transmission Expansion Planning

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 863922.