IET
Institut für Energietechnik

TUHH
Technische Universität Hamburg

Bachelor's Thesis

Developing a Method for Estimating the Computational Resources Required for Optimising Energy Supply System Models When Using Different Software Tools Through a Framework for Input Data Harmonisation

Andreas Jessen

März 2023

IET
Institut für Energietechnik

TUHH
Technische Universität Hamburg

Bachelor's Thesis

Developing a Method for Estimating the Computational Resources Required for Optimising Energy Supply System Models When Using Different Software Tools Through a Framework for Input Data Harmonisation

Andreas Jessen

Matr.-Nr.: 52365

Erstprüfer:	Dr.-Ing. Kristin Abel-Günther
Zweitprüfer:	Mathias Ammon M.Sc.
Betreuer:	Mathias Ammon M.Sc.

Hamburg, 20. März 2023

TUHH
Technische Universität Hamburg

Institut für Energietechnik

Dr.-Ing. Kristin Abel-Günther

Bachelor's Thesis for Mr. Andreas Jessen

Matr.-Nr. 52365

Developing a Method for Estimating the Computational Resources Required for Optimising Energy Supply System Models When Using Different Software Tools Through a Framework for Input Data Harmonisation

Figure 1.: Logo of the TUHH

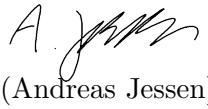
(Dr.-Ing. Kristin Abel-Günther)

Ich erkläre hiermit, dass die vorliegende Bachelorarbeit ohne fremde Hilfe selbstständig verfasst wurde und nur die angegebenen Quellen und Hilfsmittel benutzt worden sind. Wörtlich oder sinngemäß aus anderen Werken entnommene Stellen sind unter Angabe der Quelle kenntlich gemacht.

Alle Quellen, die dem World Wide Web entnommen oder in einer sonstigen digitalen Form verwendet wurden, sind der Arbeit beigefügt.

Diese Arbeit ist nach bestem Wissen erstellt worden. Für den Inhalt kann jedoch keine Gewähr übernommen werden.

Hamburg, 20. März 2023



(Andreas Jessen)

Abstract

This thesis describes and shows results of a method for comparing the computational resources required by different modelling tools to optimise energy system models. The method relies on a software called *Tessif*, a framework that unifies the data input and output of multiple Free and Open Source Software (FOSS) energy system modelling tools. Multiple energy system models are created in tessif through which they are then transformed to and optimised with each of the investigated modelling tools, which are *Calliope*, *Fine*, *Oemof* and *Pypsa*. The time elapsed while doing this and the memory usage are measured. Additionally the number of constraints that are used by the modelling tools to describe the models are recorded. The number of constraints can be used to assess the complexity of a model and is used in this thesis to allow the comparison of results of different energy system models.

The measurements show that with the exception of *Calliope* the resource usage for the investigated modelling tools scales linearly and is on a relatively similar level. Both *Calliope's* memory usage and the required time are significantly higher. The measurements further show that the difference in resource usage directly correlates with the difference in the number of constraints with the exception of *Calliope's* memory usage which increases quadratically while the corresponding number of constraints only increases linearly.

Kurzfassung

Diese Arbeit beschreibt und zeigt Ergebnisse einer Methode zum Vergleich der Rechenressourcen, die von verschiedenen Modellierungswerkzeugen zur Optimierung von Energiesystemmodellen benötigt werden. Die Methode stützt sich auf eine Software namens *Tessif*, ein Framework, das die Dateneingabe und -ausgabe mehrerer FOSS Energiesystemmodellierungswerkzeuge vereinheitlicht. In Tessif werden mehrere Energiesystemmodelle erstellt, die dann in jedes der untersuchten Modellierungswerkzeuge (*Calliope*, *Fine*, *Oemof* und *Pypsa*) transformiert und optimiert werden. Die dabei verstrichene Zeit und der Speicherverbrauch werden gemessen. Darüber hinaus wird die Anzahl der Constraints aufgezeichnet, die von den Modellierungswerkzeugen verwendet werden, um die Modelle zu beschreiben. Die Anzahl der Constraints kann zur Bewertung der Komplexität eines Modells herangezogen werden und wird in dieser Arbeit verwendet, um die Ergebnisse verschiedener Energiesystemmodelle miteinander vergleichen zu können.

Die Messungen zeigen, dass mit Ausnahme von *Calliope* der Ressourcenverbrauch für die untersuchten Modellierungswerkzeuge linear skaliert und auf einem relativ ähnlichen Niveau liegt. Sowohl der Speicherverbrauch von *Calliope* als auch die benötigte Zeit sind deutlich höher. Die Messungen zeigen ferner, dass der Unterschied in der Ressourcennutzung direkt mit dem Unterschied in der Anzahl der Constraints korreliert, mit Ausnahme des Speicherverbrauchs von *Calliope*, der quadratisch ansteigt, während die entsprechende Anzahl von Constraints nur linear zunimmt.

Contents

List of Figures

List of Tables

1. Introduction

Due to human activity the concentration of greenhouse gasses in the earth's atmosphere is increasing. This leads to a dangerous rise in its temperature. [1] Therefore the nations of the world have agreed in 2015 to decrease the emission of greenhouse gasses in such a way that earths average temperature rises by no more than 1.5 degrees compared to the average temperature before industrialisation. An effective measure to decrease the emission of additional greenhouse gasses into the atmosphere is to transition away from energy sources that burn fossil fuels. This is already happening. More and more renewable energy sources like solar panels or wind turbines are being used. Doing this leads to an increased demand for the ability to model energy systems and the impacts that changes have. Good modelling makes it possible to analyse conversion paths, compare them and thus to find the paths that are less expensive and easier to implement.

While modelling tools for energy systems do exist, most are closed source. [2] This is unfortunate as FOSS has multiple advantages. [3] These include

- The freedom to use the software without paying makes research using it more affordable.
- The freedom to see the source code makes it possible to have a better understanding of how the software works and to trust its results more.
- The freedom to change the software allows it to be adapted to ones own needs.
- The freedom to share the changes made with others allows to work collaboratively on the software instead of each making the same changes individually.

FOSS energy system modelling tools do also exist but it is not easy to choose a suited one as there are quite a lot and each have different focus areas, strengths and weaknesses. [4]

Tessif aims to help with these issues. It is a framework developed at the *Hamburg University of Technology* that provides a unified input and output for multiple energy system modelling tools, i.e. only one energy system model is written and the results are presented in a uniform way. [5] This allows for easy comparisons between different modelling tools. One possible comparison is to investigate the computational resources necessary to optimise energy system models in each tool.

Computational resources are a big driver in cost and also time spent when investigating energy systems and demand for them is rising with energy systems becoming more decentralised. [6] Obviously being able to tell which modelling tool is less computationally demanding helps to reduce the compute spent on a research task. This thesis aims to provide insight into this topic. Also it is meant as a showcase how this kind of research can be done with *tessif*. More concretely the following list of objectives is to be achieved:

- Develop and enhance *Tessif's* functionalities to measure its computational resources.
- Showcase workflow to compare computational resources of modelling tools while also considering the number of constraints used in the optimisation.
- Show and discuss results of said comparison.
- Assess the additional resource usage when using *Tessif* instead of the modelling tools directly.
- Assess which are the cases where using tessif makes sense in terms of additional resource usage.

2. Theory

2.1. Energy Systems

In its glossary the IPCC Sixth Assessment Report [7] defines an energy system as "all components related to the production, conversion, delivery, and use of energy". An energy system consists of sources and sinks, through which energy enters and leaves the system, as well as components to transport, transform or store it. An exemplary energy system would be the electricity grid of a region. Here, the sources are power plants and imports through connections to other grids and sinks are demands for electricity from households and industry as well as exports to other grids. The system also includes power lines to transport the energy, transformers to connect regional with local grids and storages such as batteries and hydro-storage reservoirs.

2.2. Energy System Modelling

Energy system modelling describes the effort to build computer models that represent and predict the behaviour of energy systems. They are often used to investigate and compare different scenarios. Results that are of interest include emissions, costs and efficiencies. A famous saying goes "all models are wrong but some are useful". Like a perfect map that includes every detail would need to be the same size as the place it depicts and therefore would be of no use, a useful energy system model can also not include every detail. Which parts a model focusses on depend on the intended use and the questions it is supposed to help to answer. [8] Therefore there are multiple model types in use. For example optimisation models that provide possible development scenarios, simulation models that forecast how a system behaves or models that include electricity market models to simulate or optimize power systems. [9]

Figure 2.1 shows the graph of an exemplary energy system model. Some simplifications that might be done in energy system models can be seen there. For example the way in which power is transported from the solar panel or the gas generator to the demand or from and to the battery is a single component called "powerline". The model does not contain inverters, transformers, the length or the thickness of the individual power lines and many other details about them but the parameters of the "powerline" component, for example its efficiency, its capacity, its operating costs and so on, are specified in such a way that it resembles the behaviour of its real world counterpart very closely. Models created this way will clearly always have some discrepancy between it and the real world but they can still be sufficiently precise to be of use.

There are not only different model types but also many different modelling tools. Each has different focus and functionality but there is some overlap and there might be synergy potential. Below the tools that this thesis focusses on are described. All of them have in common that they are FOSS written in *python* and use *pyomo* for optimisation.

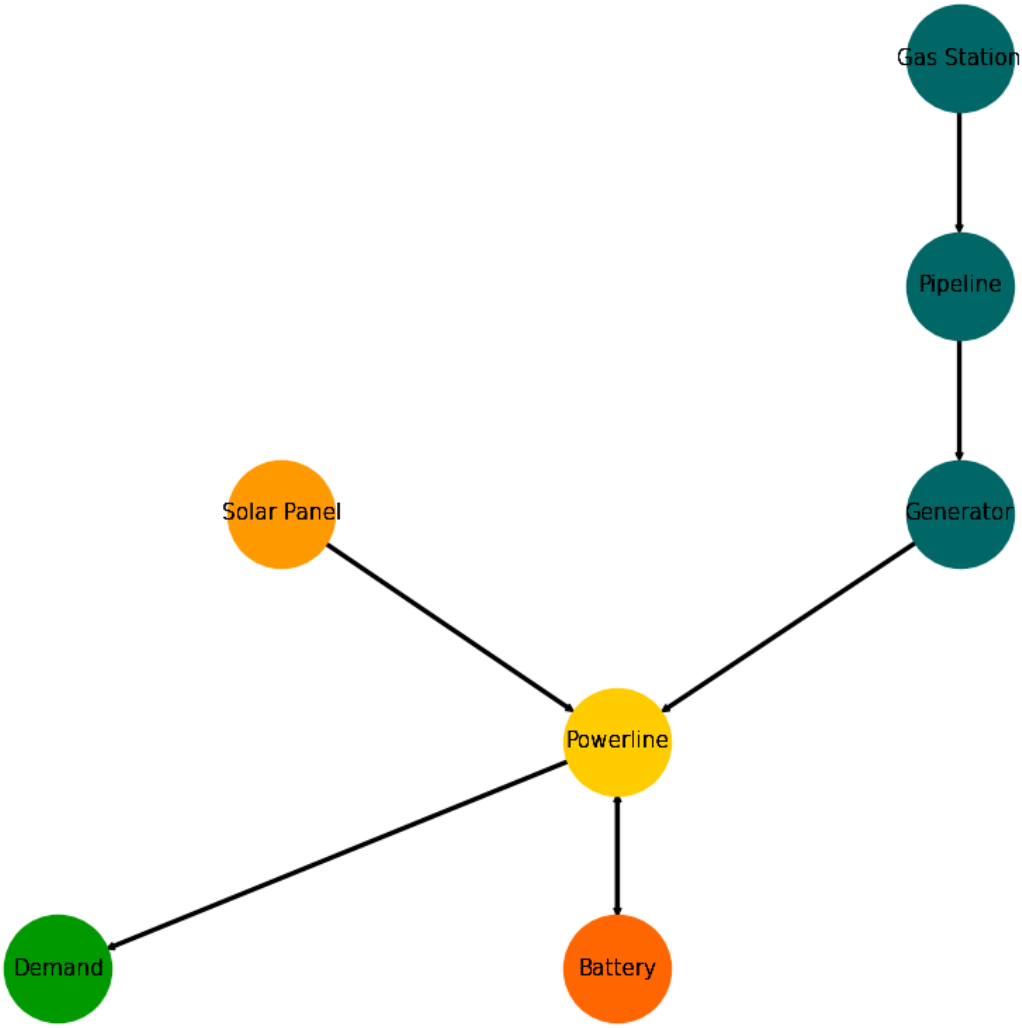


Figure 2.1.: Graph of an exemplary energy system model (From *Tessif's* documentation.[5] Figure made with *Tessif*)

2.2.1. Calliope

Calliope is being developed at the Department of Environmental Systems Science, ETH Zurich. It is designed to be able "to handle high spatial and temporal resolution and to easily run on high-performance computing systems. Its design cleanly separates the general framework (code) from the problem-specific model (data). It provides both a command-line interface and an API for programmatic use, to be useful both for users experienced with Python and those with no Python knowledge." [10] This thesis will analyse the version 0.6.6.post1.

2.2.2. Fine

Fine is an energy system modelling tool that supports that energy systems can be modelled with multiple locations, commodities and time steps. Also it doesn't necessarily

have to use the full temporal resolution but can also use an interconnected typical period storage formulation that reduces the complexity and computational time of the model. *Fine* optimises for minimal total annual cost while considering technical and environmental constraints. [11] This thesis will analyse the version 2.2.2.

2.2.3. Oemof

The *Oemof* project is managed by the Reiner Lemoine Institute, Berlin and the Center for Sustainable Energy Systems at the University of Flensburg and the Flensburg University of Applied Sciences. In this thesis oemof.solph, its model generator, will be investigated. Oemof.solph tries to simplify the data input by offering the ability to represent energy systems using spreadsheets. It can optimise for different targets, such as financial cost or CO2 emissions. [12] This thesis will analyse the version 0.4.4.

2.2.4. Pypsa

Pypsa is managed by the Institute for Automation and Applied Informatics, Karlsruhe Institute of Technology. Its main focus are electric power systems but it also supports allied sectors like coupling to the natural gas, hydrogen, heat and transport sectors. It is designed to be easily extensible and to scale well with large networks and long time series and is positioned as a bridge between traditional power flow analysis tools for steady-state analysis and full multi-period energy system models. [13] This thesis will analyse the version 0.19.3.

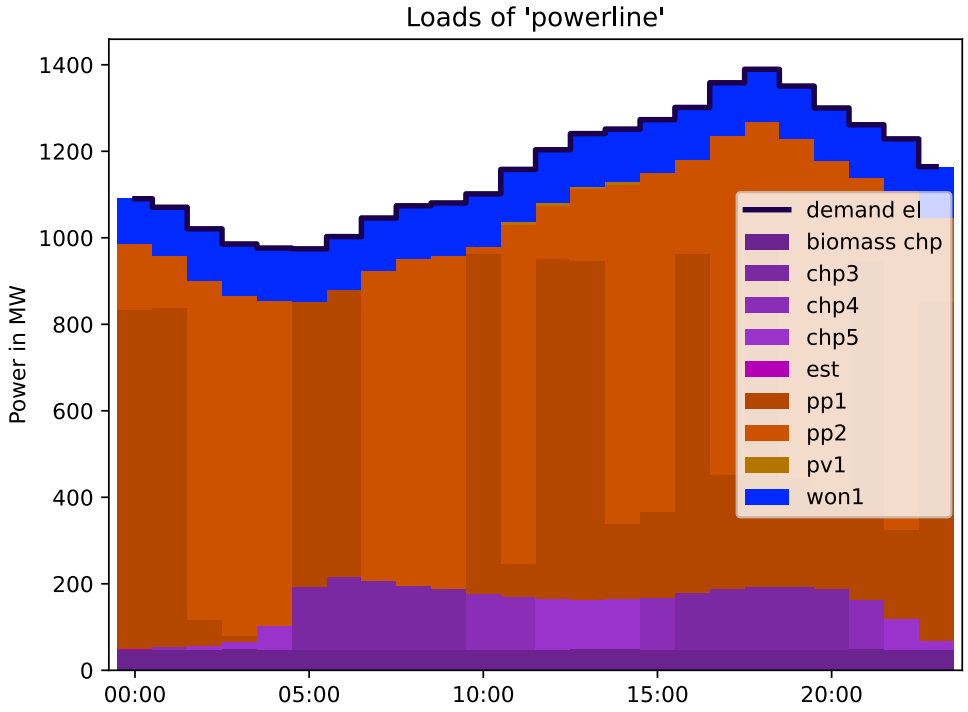


Figure 2.2.: Exemplary visualisation by *Tessif* showing the temporal development of the loads of a component. (own figure)

2.3. Tessif

Tessif is a framework written in *Python* by Ammon as part of a doctorate [14] at the Institute for Energy Technology, Hamburg University of Technology. It unifies inputs to and outputs from various FOSS energy system modelling tools. Energy system models can be defined in *Tessif* and it will then transform them into the way that model is represented in each tool. The results of optimisation are post-processed. There are multiple ways to visualise them but all look the same way for each tool. Figure 2.2 shows an example of results visualised with *Tessif*.

The Goal behind this is firstly to lower the threshold for engineers to use FOSS modelling tools by offering an easy to use and well documented framework that only has to be learnt once instead of once per tool. Secondly the goal is to allow for an easy comparison between tools for example to get insight into how good which tool might be suited for an application or to find scenarios in which the tools might give very different results. More info on *Tessif* can be found in its Documentation. [5] Reasons for and against using *Tessif* especially in relation to this thesis can be seen in Table **??**

Pro	Contra
+ Significantly reduced effort to create models that resemble each other in multiple tools	- Limited functionality compared to individual modelling tools
+ Unified API for running optimisations	- Not all modelling tools are available and a desired tool might need to be added manually
+ Results from optimisation available in uniform way	- Using *Tessif* increases overhead and might worsen computational resource usage
+ Wrappers for visualisation tools included	

Table 2.1.: Pros and contras of using Tessif

2.4. Python

Python is the programming language that *Tessif*, the investigated modelling tools and the author's own code for this thesis are written in. It is a high-level programming language, which means that it has strong abstraction from the details of the computer. *Python* puts great emphasis on code readability. For example many brackets that would be necessary in other languages can be omitted and instead the indentation is interpreted. Also *Python* is known for its comprehensive standard library and the abundance of external libraries that can be added. Examples for such libraries would be *Numpy*, *Scipy* and *Matplotlib*. *Numpy* expands pythons capabilities in numerical calculations. It is more computationally efficient in the handling of arrays and matrices and allows to handle larger datasets. *Scipy* adds more advanced mathematical operations and other tasks common in science. *Matplotlib* is a versatile plotting library. In conclusion it can be said that *Python* it a good choice and arguably the de facto standard for programming in scientific research. [15]

2.5. Computational Resources

Computational resources are the resources that are needed to run an algorithm. Two of the simplest computational resources are the computation time and the memory storage needed. Time in this case does not mean the amount of actual minutes and seconds passed but rather the number of operation that are executed to run the algorithm. This is because the amount of real time that elapses while an algorithm runs changes depending on the machine that is being used. This is not the case for the number of operations. [16] Another factor that is of interest in this context is scalability. Scalability describes how a system behaves when the load on it is increased. For example an algorithm with high scalability would be one where the resource consumption increases only moderately with increased input size. [17]

There are multiple approaches to determine the resources needed to run an algorithm. For smaller algorithms the bounds can be mathematically calculated or one could even simply count the number of operations necessary. Above a certain size this is no longer feasible. Optimisations of energy systems that take minutes to complete on a processor that does billions of operations per second certainly fall into this category. While the necessary memory storage can still be easily measured, for the computation time an other approach has to be chosen. One such approach would be to simply measure the time it takes to complete the algorithm on a certain machine. This can be improved by only measuring the time the machine's processor spends on the algorithm as modern machines usually have other tasks running in the background. But even then it is important to keep in mind that while the results obtained can be compared to others that were gathered on the same machine it is not trivial to transfer them to other machines. Absolute measurements of time are not easily transferable but relative differences between algorithms on the same machine do give meaningful insight for other machines too. So does examining scalability. Scalability can be examined by running the same algorithm multiple times with increasing input sizes and measuring the increase in time and used memory.

3. Methods

To get insight into how energy system modelling tools differ in terms of computational complexity and scalability, the chosen approach is the following: Multiple energy system models that are each both scalable in the energy system size and timeframe they represent are created in *Tessif* and transformed by it for each modelling tool. Then each tool optimises the systems and the computing time and the memory used are measured. To compare the results of the different models with each other the number of constraints that pyomo considers are also recorded. The measurements are then visualised and discussed. The steps will be described below in more detail.

3.1. The Energy System Models

Four energy system models are used. Each has a different focus as will be described below. Each of them can either be used on its own or as a unit of a larger energy system. Multiple different models are used to be able to see if some part of the results is a fluke or the effect of the behaviour of one particular model. Also, the measurement results for models with different focus might differ and might differ in different ways for each modelling tool. This could give insight into which part of a model is more or less demanding for optimisation and which modelling tool prefers which kind of model. It is important to note that the models differ in size, therefore the absolute results of the measurements (i.e. the seconds and the megabytes) can't be directly compared between models. It is however possible to compare the number of constraints of each system and to compare the time and memory in relation to the constraints.

3.1.1. Minimal Energy System

This is the smallest model examined. It has barely more components than is necessary to run a plausible optimisation and only consists of a single central bus with a renewable and a non-renewable sources plus fuel line and power generator, one sink and one storage. Those components connected to the central bus are all constrained in power so there is also a deficit source and an excess sink which are more expensive than the other components and only used if a limit in power is reached somewhere else. A graph of it can be seen in figure 3.1. In contrast to the graphs of the other energy system models, in this one every component has a zero at the end of its name. This is due to the fact that this model only exists as part of the larger model described in 3.2 in which every unit has a sequential number (zero in this case) which it passes to its components so that each component has a unique name. The other models also can be used independently and do not have a number in their components' names then. It is described in more detail by Emmel in his bachelor's thesis [18]. Its small size allows compared to the other models to have a wider range of scale. A smaller model can be optimised for a longer timeframe with the same complexity. Also multiple instances of it can be more easily

combined to a bigger system model. More on scaling the models can be found in section 3.2.

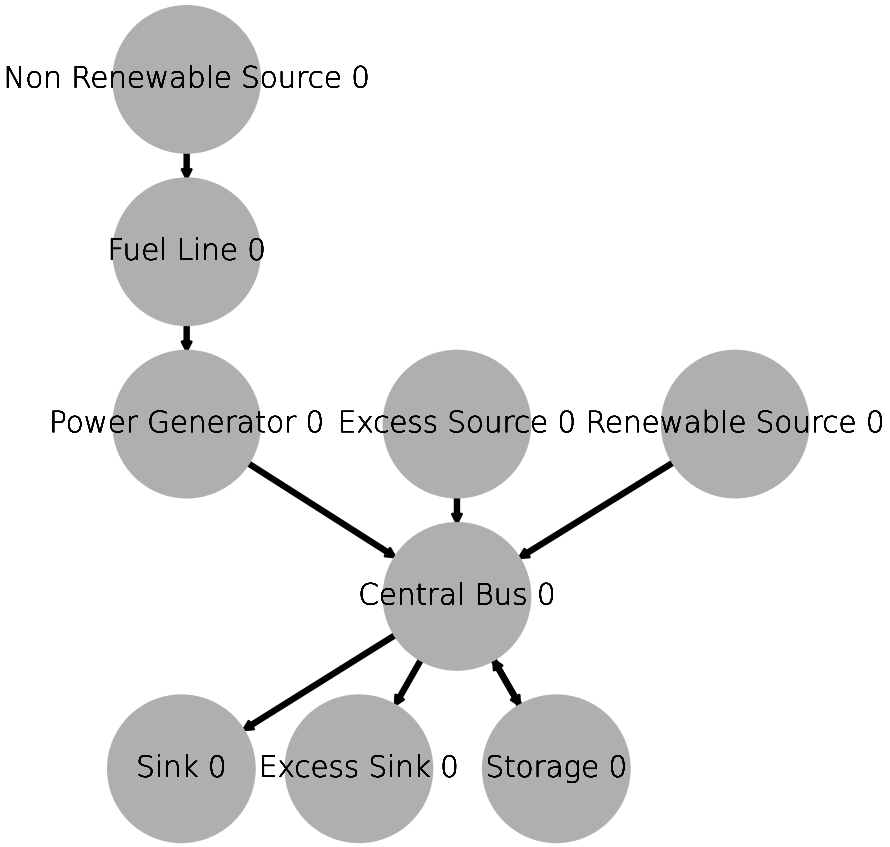


Figure 3.1.: Graph of the minimal energy system (own figure made with *Tessif*)

3.1.2. Component Focussed Energy System

This Model was written by Reimer as part of a project thesis. [19] A graph of it can be seen in figure 3.2. Its supply grids are modelled as ideal (i.e. they are unlimited in capacity and have no losses due to inefficiencies) and with only one component each for heat and power. They are connected by a single power to heat plant and have each one sink with a fixed demand and an energy storage. This part of the model is rather simple, as its focus lies in the components that represent the energy sources and the plants that transform their energy into power or heat. They depict a variety of renewable and non-renewable sources that generate either heat or power or both. Their capacities are chosen in a way that they could always supply more energy than is demanded so that there always is some optimisation to be made. Usually this means that the non-renewable sources are not used at full capacity to enable the renewables to be used at full capacity.

The model can either be optimised as a commitment problem where the goal is to reduce the costs as much as possible, while the CO_2 emissions are not limited and the installed capacities at the sources, transformers and storages are fixed, or as an expansion problem

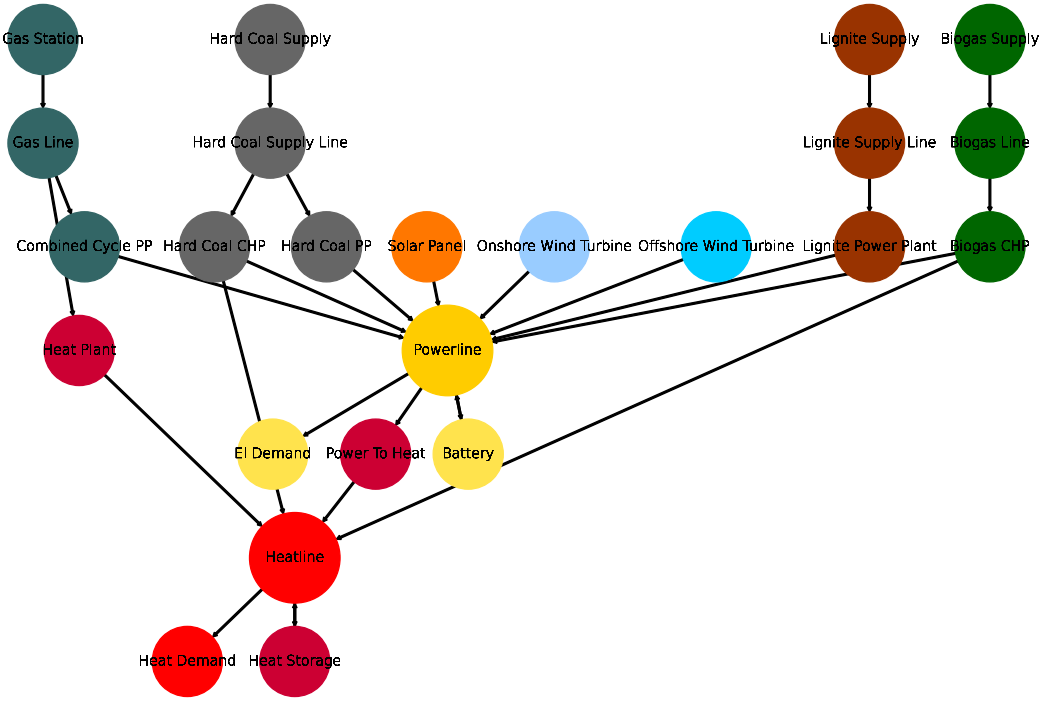


Figure 3.2.: Graph of the component focussed energy system (own figure made with *Tessif*)

where the goal still is to minimise costs but here the CO_2 emissions are constrained and the capacities have to be expanded to fulfil the constraints, thus the costs arise not only from the operation but also from the expansion of various components. In this thesis the commitment problem is used.

3.1.3. Grid Focussed Energy System

This Model was written by HANKE as part of a project thesis. [20] A graph of it can be seen in figure 3.3. In this model the power grid is represented in more detail when compared to the other models. It consists of a low, medium and a high voltage grid which are interconnected with transformers. Those are not perfectly efficient and have limited capacity but can be expanded if advantageous. It might seem obvious to choose the connector component to connect the different power grids but not all modelling tools support specifying a transfer efficiency for them. Therefore two transformer components, one for each direction are chosen. Additionally there is a district heating network which is coupled to the medium voltage grid by a power to heat plant. Each grid has its own energy demands and sources with some plants supplying multiple grids. When compared to the other models it is clear that while the other components are not as simple as possible. The goal of this model is not to investigate the behaviour of the supplies and demands but rather the way the grids interact with each other and the paths the energy takes. When measuring computational complexity, including models with grids that have different levels of detail allows to get insight what impact this part of the model has on complexity.

The original model only includes demand and renewable source data for 24 hours. To be able to investigate longer timeframes this is changed in such a way that the data

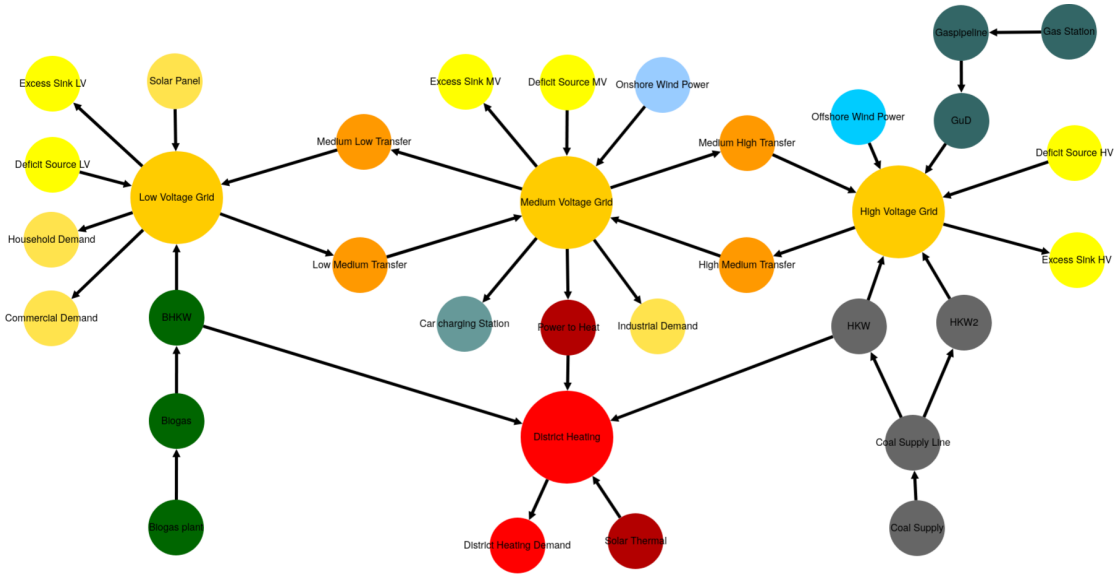


Figure 3.3.: Graph of the grid focussed energy system (From *Tessif's* documentation.[5] Figure made with *Tessif*)

is repeated multiple times. That means that every day has the same data. While this means that the model is now of not much use for anything else, measuring computational complexity is not impacted by this change. Just as the component focussed model, this model is optimised as a commitment problem but in contrast to the original grid focussed model where the capacity of the power grid transformers is fixed, this is changed and they are expandable. This was to fix a problem that occurred while optimisation when the system was made part of a larger model. But this too is unlikely to be an issue for measuring computational complexity as the model gives out plausible results no matter if the transformers' capacity is expandable or not.

3.1.4. Hamburg's Energy System

This Model was written by KÖRBER as part of a bachelor's thesis. [21] A graph of it can be seen in figure 3.4. Out of the four models it is the only one that is modelling a real world energy system. All power and heat plants and their parameters model the behaviour of their counterparts in Hamburg's power and heat grid. The data for the power and the heat demand as well as for the renewable energy sources match Hamburg in the year 2017 as this was the newest data available at the time of creation of the model. This model is also optimised as a commitment problem, where the capacity of the components is fixed and there is no constraint to the CO_2 emissions.

The grid structure is simpler than in the grid focussed model, but in total it is the most sophisticated and largest model. Having models in different sizes is not strictly necessary to investigate scalability as this thesis relies on a method to combine multiple instances of a model to a larger one. It can however still be beneficial to have a larger model to be able to compare it to a compound model consisting of multiple smaller systems. This way it can be verified that the approach of scaling small models by linking them together gives feasible results.

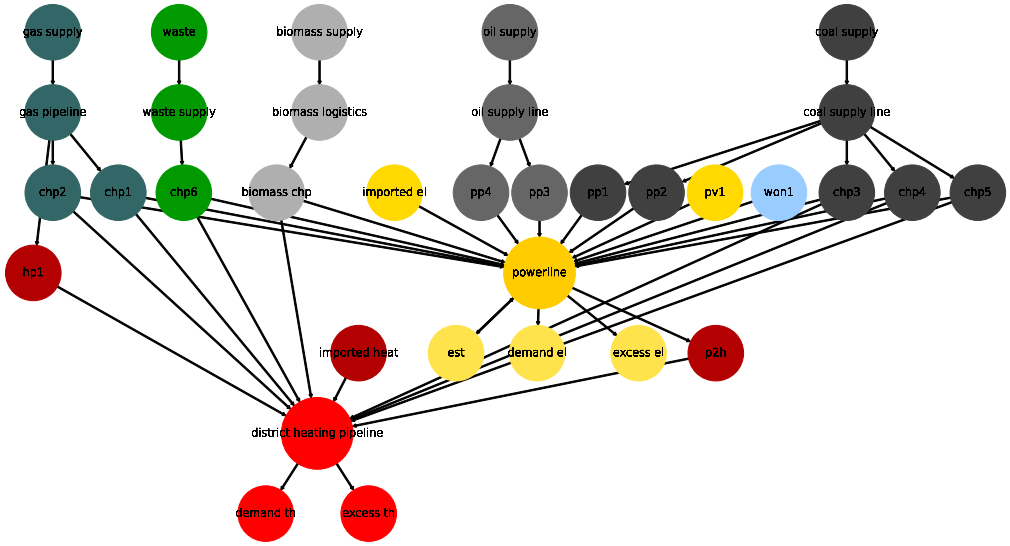


Figure 3.4.: Graph of Hamburg's energy system (own figure made with *Tessif*)

3.2. Scaling the Models

The models are scaled both in timeframe and in energy system size. Scaling the timeframe is straight forward. All the models can be optimised with arbitrary timeframes. Only slight modification was made so that in all of them the timeframe is defined in a uniform way to make integration into other functions easier and in the case of the component focussed energy system, the supported timeframe had to be extended which is described in more detail in section 3.1.2.

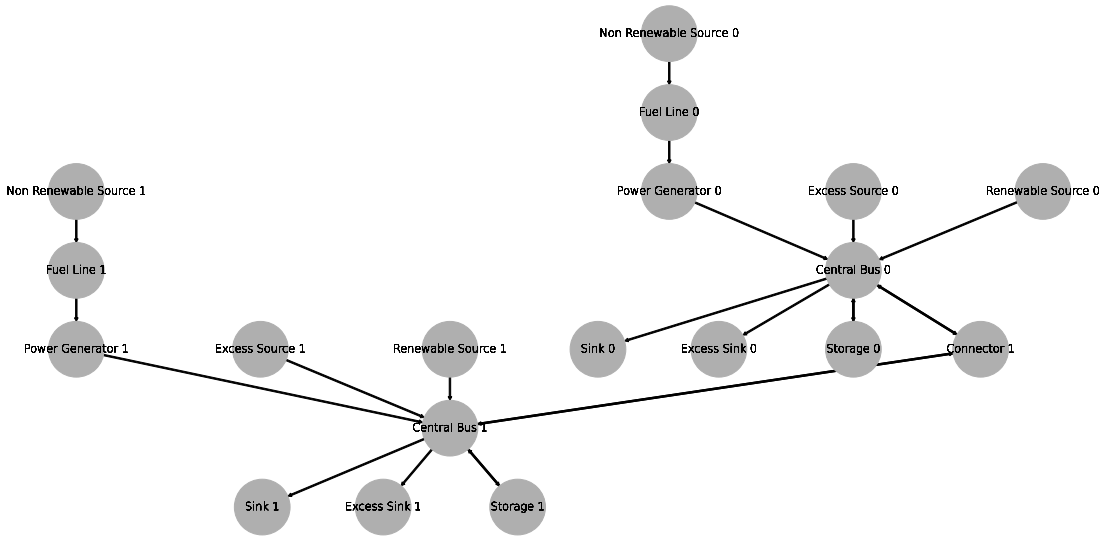


Figure 3.5.: Exemplary graph of a connected energy system model (own figure made with *Tessif*)

In contrast to the timeframe size, the energy system size is harder to scale. The approach chosen is to upgrade *Tessif's* so called self similar energy system model. It can consist of an arbitrary number of energy system units which are connected to each other at the

buses by a connector component. An exemplary graph of a system with two units can be seen in figure 3.5. Originally the model relied on a single unit that was hard-coded in but this has been adapted so that it is now possible to add other models to be used as units quite easily. To add a new model as a unit it has to be slightly adapted: The names of the components have to include a variable containing a number which is given to it as a parameter of the function creating the units. This is needed so that each component in a model with multiple units has a unique name. Additionally a connector component needs to be added for each bus to connect it to its counterpart in another unit. This has already been done for three models additionally to the already existing minimal unit, namely the energy system models described above.

3.3. The Workflow Using Tessif

For the measurements *Tessif's* `assess_scalability()` function is used. It creates an energy system model, transforms it into a model in the desired tool, runs the optimisation and lets *Tessif* post process the results. For every step it measures how much process time passed and how much memory was used, then a total amount for both is calculated. This process is then repeated for energy systems with different size and timeframe. The measurement results returned by the function are then stored as csv files. To visualise the data in the csv files *Matplotlib* is used. The `assess_scalability()` function was originally written for Emmel's bachelor's thesis [18] but has been modified and improved for this thesis. Added was the ability to also return the number of constraints that were used in *Pyomo*, to select the resolution (i.e. the number of intermediate steps used when scaling) of the measurement and to select which model shall be assessed. Also the code was simplified and its structure improved.

3.4. Counting the Constraints

For optimisation the modelling tools use *Pyomo* in which they formulate the energy system model as a mathematical optimisation problem which consists of variables, objectives and algebraic constraints. The larger an energy system or the timeframe of a model is the more constraints are needed to formulate it in *Pyomo*. Therefore the number of constraints can be used as an indicator of the complexity of a model. When *Pyomo* returns the results from optimisation, it also returns a variable that contains the number of constraints that were used in that optimisation. This variable is stored by the modelling tools inside of the energy system model object. To access it in a uniform way, a new resultier was added to *Tessif's* post-processing module called `es2mapping`. When the `assess_scalability()` function is called, all resultiers so now also the new constraints-resultier are run and the created mappings stored. So after that the resultiers can be parsed again and the number of constraints variable can be accessed.

3.5. Hardware and Software Used

Each of the investigated modelling tools relies on *Pyomo* for optimisation which in turn supports multiple solvers that can be used. This thesis will use the *COIN-OR branch*

and cut solver for every tool. Its version as well as the versions of other relevant software can be seen in table 3.1. *Tessif's* code can be found at [22] but the version used in and this thesis is *Tessif-PHD*. It has its own GitHub repository and includes the code for this thesis.

Software	Version
Calliope	0.6.6.post1
Coin-or-cbc	2.10.8-1
Fine	2.2.2
Ipython	8.7.0
Oemof.solph	0.4.4
Pyomo	5.7.2
Pypsa	0.19.3
Python	3.9.14
Tessif	pre-release version from sep. 22

Table 3.1.: Versions of Software Used

The hardware the measurements were made on is a Lenovo ThinkPad T420. It might not be the typical hardware where this kind of software usually runs on as it is quite old. It does however easily fulfil the requirements to be able to run the software. The only thing to consider is that measurement times will be longer than on today's hardware but as the absolute measurement values are not so much of interest as is how they compare to each other, this in not an issue. More details on the hardware and the operating system can be found in table 3.2.

Processors	4 × Intel Core i5-2520M CPU @ 2.50GHz
Memory	12 GB DDR3
Graphics Processor	Intel HD Graphics 3000
Hard Drive	SATA 3 SSD
Operating System	Manjaro Linux
KDE Plasma Version	5.26.5
Kernel Version	6.1.11-1-MANJARO (64-bit)
Graphics Platform	Wayland

Table 3.2.: Hardware Used

4. Results

This Chapter shows the results of the measurements as charts. Some of the following charts and texts include the Variables N or T. N stands for the number of energy system units (i.e. the size of the energy system). More on energy system units can be found in chapter 3.2. T stands for the number of time steps. One time step is equivalent to one hour. If not mentioned otherwise, the number of time steps is 24 and the number of energy system units is one. The csv files containing the data the following figures are based on are inside the digital Appendix of this thesis.

4.1. Number of Constraints

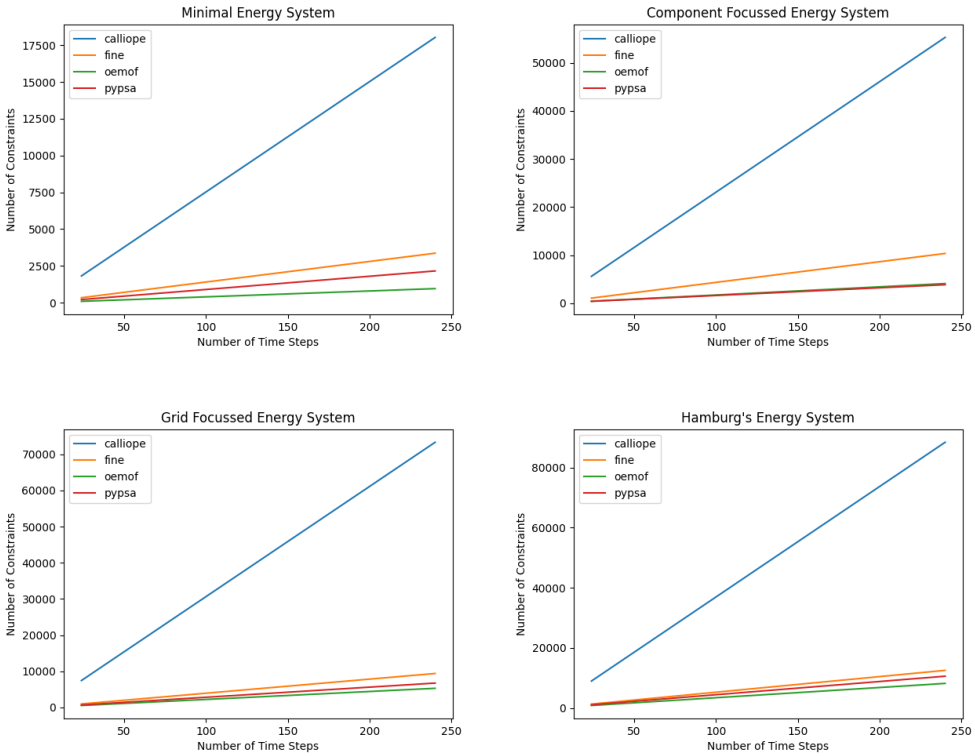


Figure 4.1.: Number of constraints over the number of time steps (own figure)

First of all the number of constraints that are used by *Pyomo* to optimise the model are shown. In figure 4.1 this is shown for scaling T. The figure shows four charts, each of them displaying the results of the measurement of one of the four energy system models for each modelling tool. Their x-axes show the number of time steps covered by each model's instance and their y-axes the number of constraints. Each of the coloured

lines shows the number of constraints for the number of time steps for one modelling tool. Each modelling tool has the same colour in every figure in this chapter with the exception of figure 4.5. Each model and each modelling tool in figure 4.1 were optimised with 24 time steps and then in 24 time step intervals up to 240 time steps. In figure 4.2 the same is shown for scaling N. Again the y-axes of the four charts for the four models show the number of constraints and the coloured lines stand for the individual modelling tools but this time the x-axis shows the number of energy system units that each model's instance consists of. Here models and tools were scaled in one unit steps from one to ten with the exception of Hamburg's Energy system which was only scaled up to seven units as the optimisation would not finish in a reasonable time frame with larger energy systems.

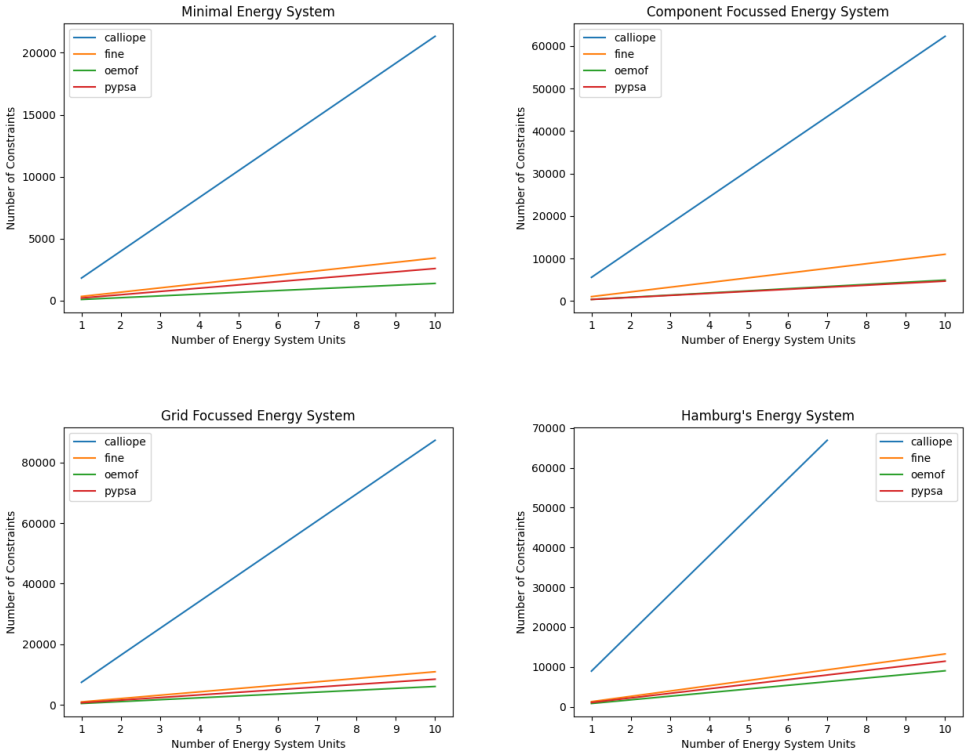


Figure 4.2.: Number of constraints over the number of energy system units (own figure)

In both cases it can be seen that the number of constraints rises linearly with T and N. It can also be seen that the line for *Calliope* is much higher, which means that with *Calliope* energy systems are optimised with a larger number of constraints than with other modelling tools. The lines of the other tools are comparatively close together. Their order is the same for all energy systems with the exception of the component focussed energy system where *Oemof* uses slightly more constraints than *Pypsa*. The models need different numbers of constraints when comparing the same T, N and tool, Hamburg's energy system needs the most followed by the grid and the component focussed model. The minimal energy system needs significantly less constraints. For example in figure 4.1 *Calliope* with T=240 uses between 80,000 and 100,000 constraints for Hamburg's, between 70,000 and 80,000 constraints for the grid focussed, between 50,000 and 60,000 constraints for the component focussed and between 17,500 and 20,000 constraints for the minimal energy system.

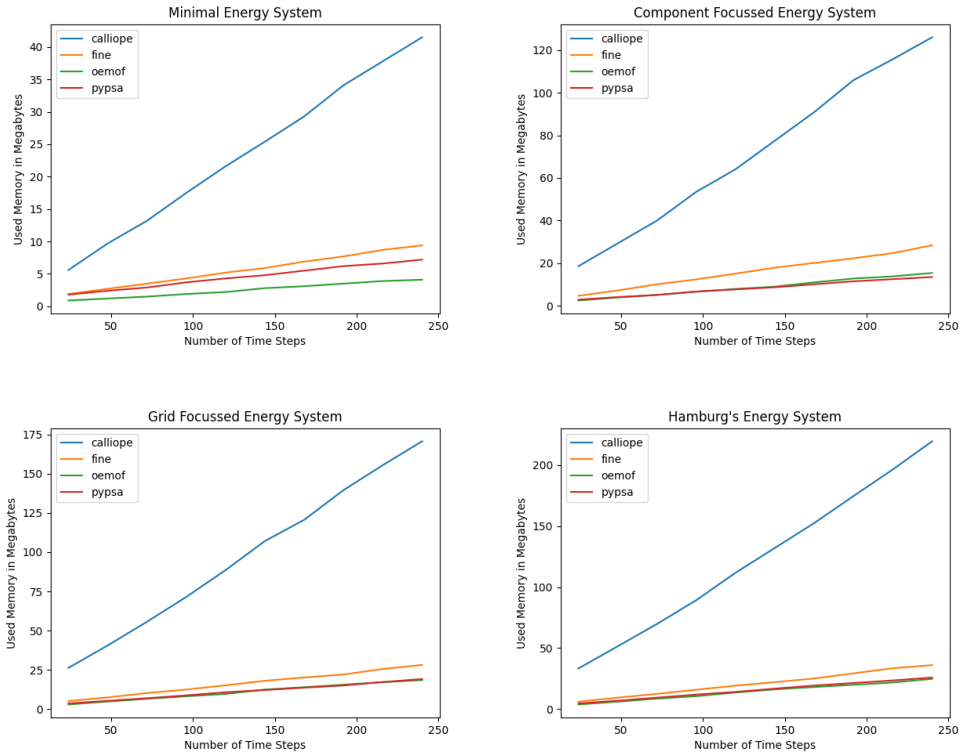


Figure 4.3.: Used Memory over the number of time steps (own figure)

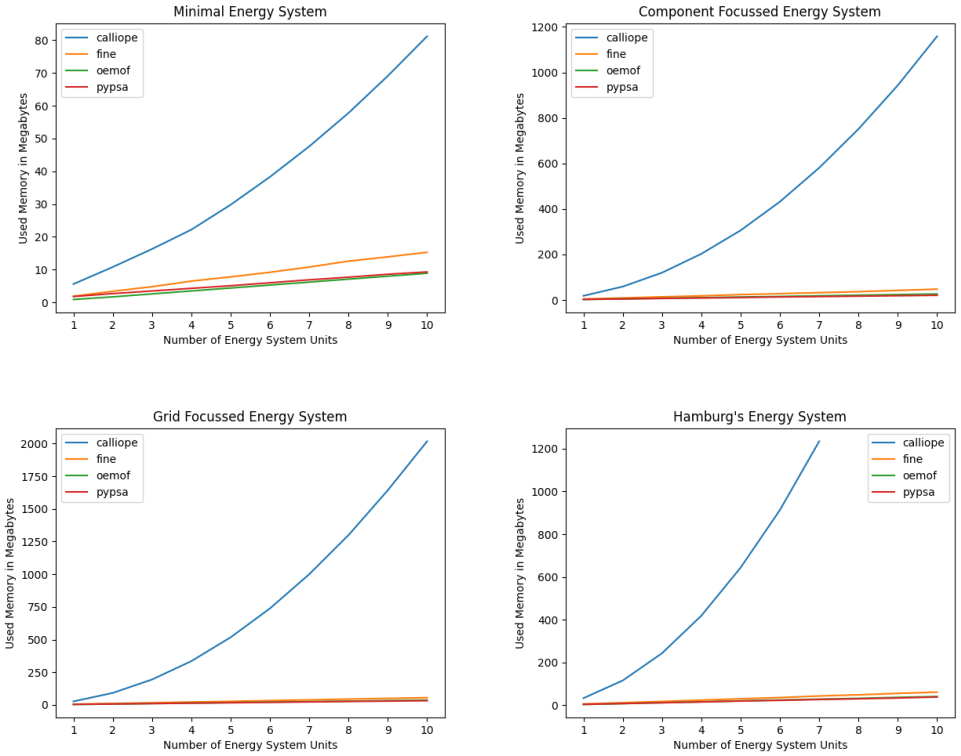


Figure 4.4.: Used Memory over the number of energy system units (own figure)

4.2. Memory

When used memory is mentioned in this paragraph, what is meant is the peak memory usage during the whole workflow from the start of parsing until the end of post-processing. Figure 4.3 shows how the amount of used memory scales over T. Again the four charts for the four models with the coloured lines for the individual modelling tools known from the number of constraints charts can be seen. Their x-axis shows the number of time steps (T) covered and their y-axis shows the peak memory usage in megabytes. T was scaled in 24 time step intervals from 24 to 240 time steps. Figure 4.4 shows how used memory scales over N. The y-axes of the four charts for the four models again show the memory usage in megabytes. Their x-axes show the number of energy system units (N) each model's instance consists of. N was scaled in steps of one from one to ten units with the exception of Hamburg's energy system which was only scaled up to seven units as the optimisation would not finish in a reasonable time frame with larger energy systems. Figure 4.5 shows the same as figure 4.4 but without the data for *Calliope* so that the other data can be seen in more detail. In this figure the lines for the modelling tools have different colours than in the other figures.

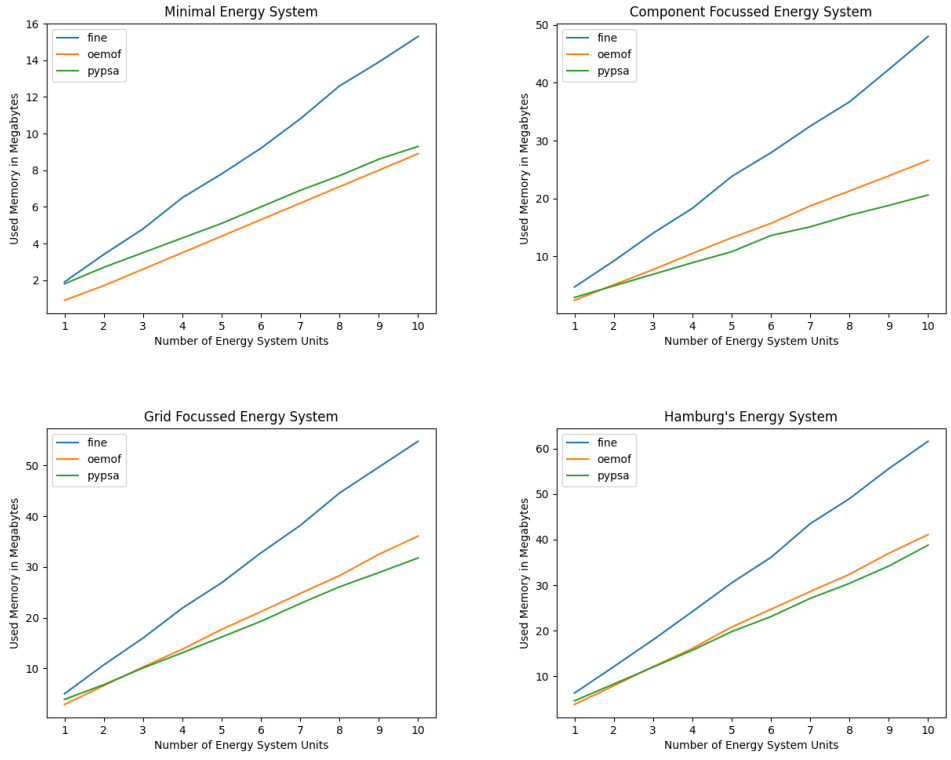


Figure 4.5.: Used Memory over the number of energy system units without *Calliope* (own figure)

The used memory scales linearly with T and N in all cases with the exception of *Calliope* which scales quadratically with N. Just as with the constraints it can be seen here too that *Calliope's* line is much higher, which means that *Calliope* needs significantly more memory than the other modelling tools. The lines of the other tools are again comparatively close together. Here however their order is not as clear. While *Fine's* line is always above the other two, it can be seen that for scaling N, *Pypsa's* has a flatter

slope than *Oemof's* and therefore needs more memory for smaller energy systems and less for larger ones. When comparing the different models' memory demand it can be seen that it is in the same order as the number of constraints, so Hamburg's energy system uses the most memory per time step an energy system unit and the minimal energy system the least.

In the four charts in figure 4.6 the y-axis shows memory usage in megabytes and the x-axis the number of constraints for each optimisation. The coloured lines again each stand for an individual modelling tool. The results displayed here are from measurements where T was scaled, like those in figure 4.1 and 4.3, but for the charts here and for those in figure 4.7 the non *Calliope* measurements have been done with larger scale if possible so that their numbers of constraints is similar to that of the models optimised with *Calliope*. This was done so that meaningful comparisons could be made between the modelling tools. Concretely, in the measurements of *Fine*, *Oemof* and *Pypsa* T was scaled from 240 to 2400 time steps in 240 time step intervals, and of *Calliope* from 24 to 240 time steps in 24 time step intervals. Most charts in figure 4.6 and 4.7 were cropped to only show the part where the data from multiple modelling tools can be seen.

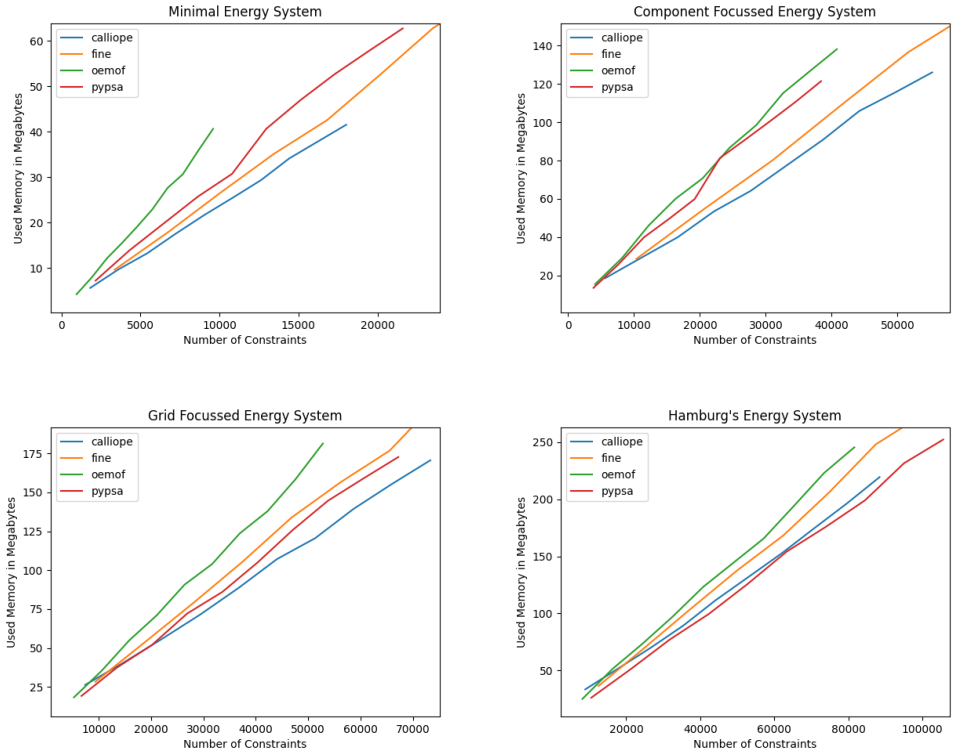


Figure 4.6.: Used Memory over the number of constraints when scaling T (own figure)

In the four charts in figure 4.7 the y-axis shows the memory usage in megabytes, the x-axis the number of constraints and the coloured lines each stand for an individual modelling tool. This time the results displayed are form measurements where N was scaled, like those in figure 4.2 and 4.4. In the measurements of *Fine* and *Pypsa* N was scaled from 5 to 50 units in 5 unit intervals for the minimal energy system and from 3 to 30 units in 3 unit intervals for the other models. The *Oemof* measurements while scaling N could not be done with more than ten units as there was an error in the program that

prevented this. Therefore not only for *Calliope* but also for *Oemof* T was scaled from one to ten units in one unit intervals.

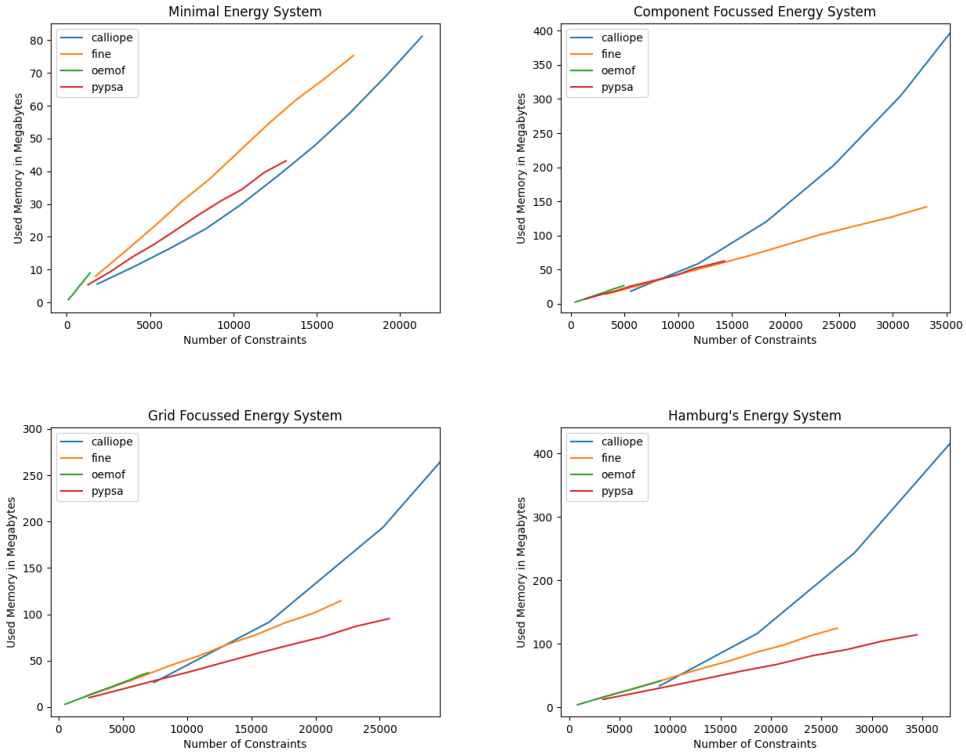


Figure 4.7.: Used memory over the number of constraints when scaling N (own figure)

Memory usage increases linearly with the number of constraints with the exception of *Calliope* which increases quadratically with the number of constraints when N is scaled. Apart from this the lines showing the measured values for each modelling tool are relatively close together but there are slight differences in memory per constraint. For scaling T (figure 4.6), *Oemof's* line is the highest, meaning that it always has the highest memory usage per constraint. For scaling N, this might be the case too but it is hard to judge with the data available. The order of the other modelling tools is less clear. When scaling T, *Calliope's* line is always the lowest, so it uses the least memory per constraint. This is with the exception of Hamburg's energy system where *Pypsa's* line is lower. In this model and in the grid focussed energy system *Pypsa* also uses less memory per constraint than *Fine* but not in the other two. When scaling N, the fact that *Calliope's* memory usage per constraint increases quadratically means that while it uses the least memory per constraint for small models, its memory usage increases faster than for the other tools and is the largest with large models. Without *Calliope*, *Pypsa* uses the least memory per constraint, followed by *Fine*. It can also be seen that memory per constraint is in a similar range across models with the exception of *Calliope* which uses less memory in the minimal energy system than in the other models when scaling N. For example at 20,000 constraints it uses 75 MB of memory for the minimal energy system and 150 MB for the other models. There are also differences between scaling T and scaling N. For example while 15,000 constraints need between 25 and 50 MB of memory when scaling T, they need between 50 and 100 MB when scaling N.

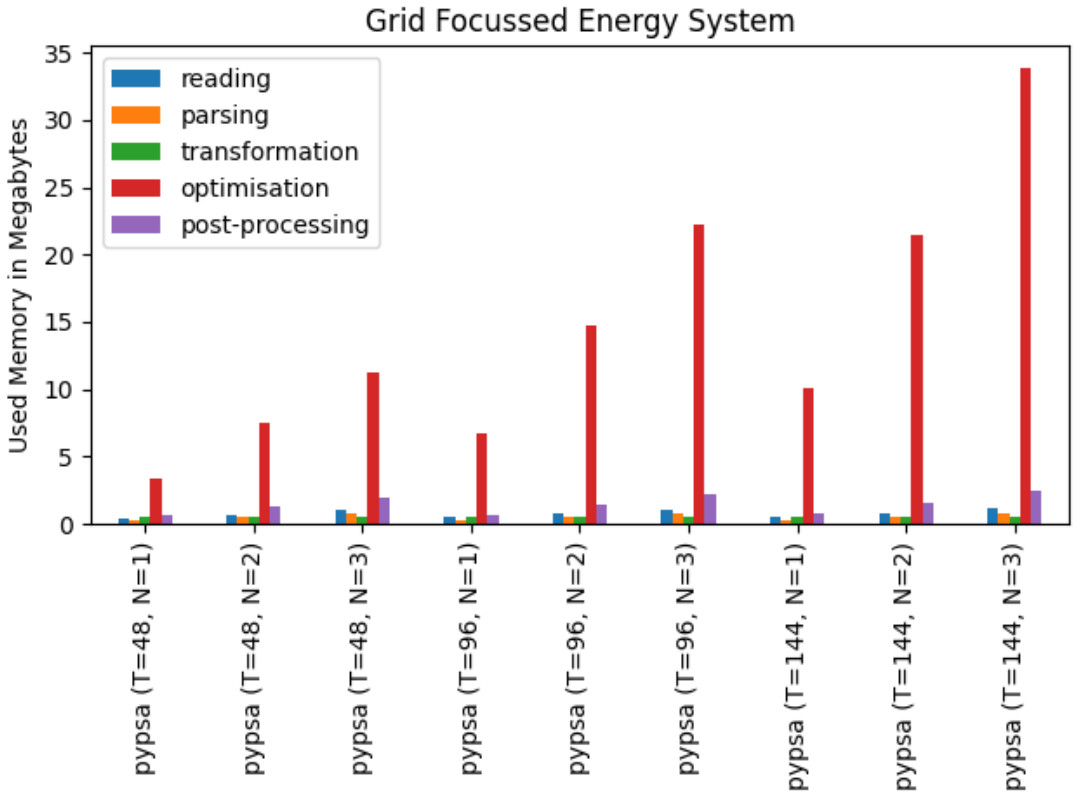


Figure 4.8.: Multiple memory measurements for *Pypsa* with different settings broken down into the individual steps (own figure)

Only the total value of memory usage for every step in the process (i.e. reading, parsing, transformation, optimisation, post-processing) was recorded, additionally an error in the code was found after the measurements were done: For the total value the measurement results of the individual steps were added together instead of making the maximal value of the individual steps the total value. This does however not lessen the insight the results discussed here provide. To show this, figure 4.8 can be used. It consists of a bar chart in which the y-axis shows the peak memory usage during the individual steps of the measurement process. Each group of five coloured bars belongs to one measurement process. Below each is written what the settings for that measurement were, so which modelling tool and which values for T and N were used. Each bar of a group stands for one step in the measurement process and the colour tells which it is. The figure clearly shows that the value for optimisation is far larger than the values for the other steps. Therefore the difference between the value for only optimisation and the summed up value is negligible. An other thing that can be seen in this figure is that memory usage for optimisation increases when N and also when T is scaled. Memory usage for the other steps however only increases when N is scaled and stays constant when only the value of T is changed. Figure 4.8 only shows results for *Pypsa* using the grid focussed energy system but the same was also done with the other models and modelling tools and their results look similar. In the interest of clarity those figures are not included here but can be found in the Appendix.

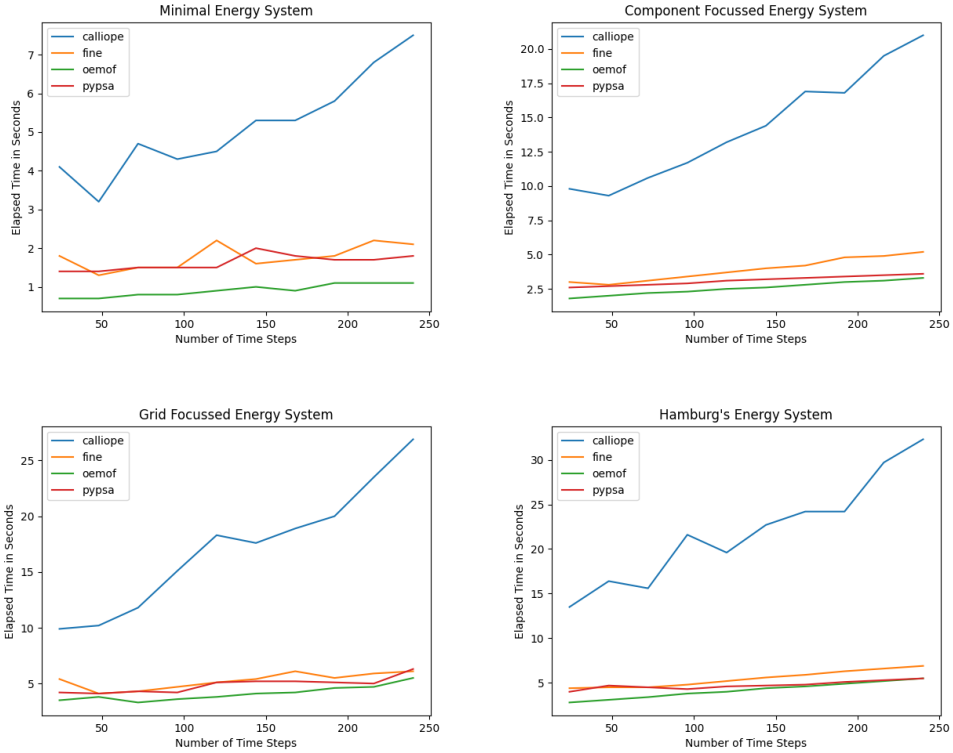


Figure 4.9.: Elapsed time over the number of time steps (own figure)

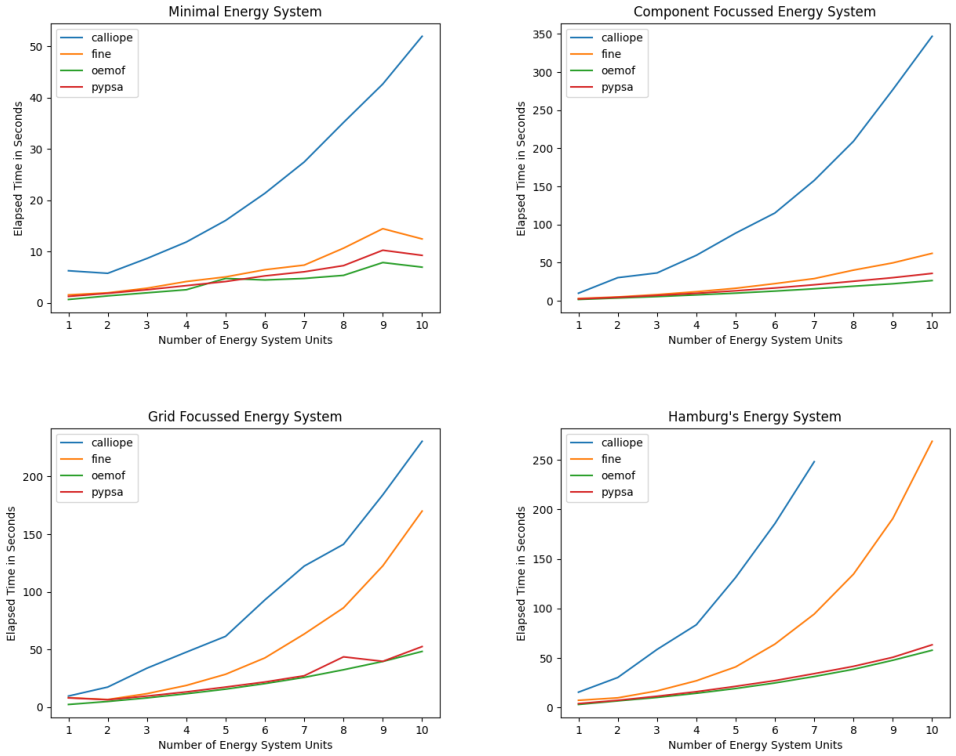


Figure 4.10.: Elapsed time over the number of energy system units (own figure)

4.3. Time

This section looks at the process time (i.e. the time a machine spends on operations for a certain process) needed to parse, transform, optimise and post-process models. First, the four charts in figure 4.9 show the elapsed time in seconds when scaling T in 24 time step intervals from 24 to 240 time steps. Their x-axes show the number of time steps covered by each model's instance, their y-axes the time that elapsed while the measurement process ran and the coloured lines each stand for an individual modelling tool.

The four charts in figure 4.10 show the elapsed time in seconds when scaling N in one unit steps form one to ten energy system units. This is with the exception of Hamburg's energy system which was only scaled up to seven units as the optimisation would not finish in a reasonable time frame with larger energy systems. Their x-axes show the number of energy system units the model consists of, their y-axes the time that elapsed while the measurement process ran and the coloured lines each stand for an individual modelling tool.

The elapsed time increases approximately linearly with T (figure 4.9) and quadratically with N (figure 4.10). The order of results for the elapsed time for each modelling tool is similar to that of the memory usage and the number of constraints. *Calliope* needs the most time by a clear margin followed by *Fine*, *Pypsa* and *Oemof* for which the results are closer together.

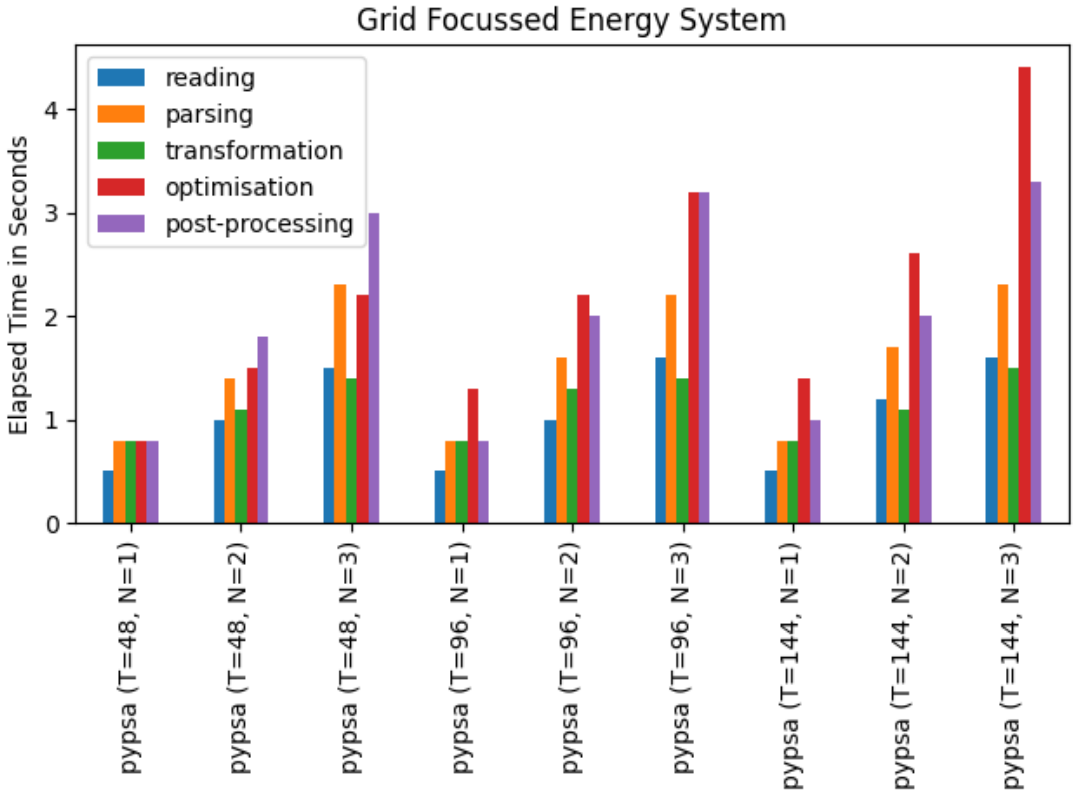


Figure 4.11.: Multiple time measurements for *Pypsa* with different settings broken down into the individual steps (own figure)

As with memory usage and number of constraints, Hamburg's energy system is also the most time consuming followed by the grid and component focussed and then the minimal energy system. It is however noticeable that *Fine* needs particularly much time when scaling the grid focussed and Hamburg's energy system in N.

Like memory usage, also the elapsed time was recorded only for the whole process and not for the individual steps (i.e. parsing, transformation, optimisation, post-processing) but in this case the difference between the total time and the time of optimisation is not negligible. This is shown in figure 4.11 for the minimal energy system in *Pypsa*. The other models and modelling tools have similar results and were not included here in the interest of clarity but can be found in the Appendix. Figure 4.11 shows on its y-axis the time that elapsed for each step. Each group of five coloured bars belongs to one measurement process. Below each is written what the settings for that measurement were, so which modelling tool and which values for T and N were used. Each bar of a group stands for one step in the measurement process and the colour tells which it is. It can be seen that optimisation clearly does not dominate the results. The other steps have similar or even higher results. It can also be seen that similarly to the behaviour of memory usage, elapsed time scales with T and with N for optimisation but only with N in the other steps. The other steps stay approximately constant when only T is changed. Therefore for scaling T using the total time does not distort the results much. As the time for the other steps stays constant, the lines slightly move up on the y-axis but remain unchanged otherwise. The figures still give insight into the scalability of optimisation times.

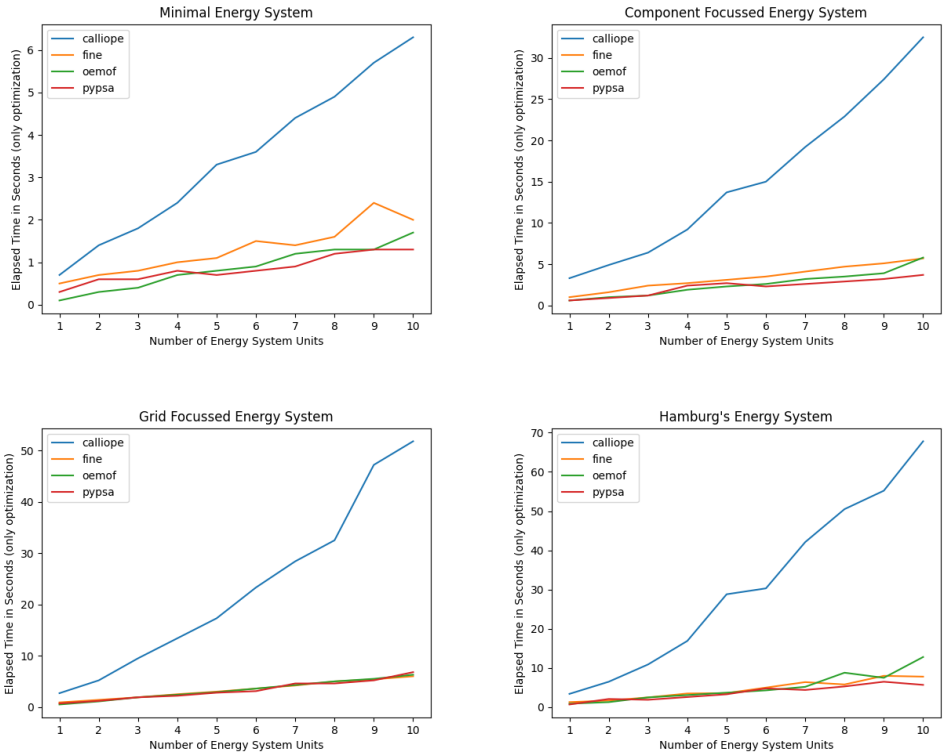


Figure 4.12.: Elapsed time for optimisation over the number of energy system units (own figure)

This does not apply to scaling N. Here it is not possible by looking at the figures to have a clear understanding of the optimisation times as both they and the other times change with scale. Therefore the measurements for scaling over N are repeated and also the times of the individual steps are recorded. The four charts in figure 4.12 shows the results from this. They are equivalent to those in figure 4.10 with the difference that they only show the time elapsed during optimisation. They clearly differ from those in figure 4.10. Here the elapsed time increases linearly with N. Also here *Fine's* results are much closer to the ones of the other modelling tools.

In the four charts in figure 4.13 the y-axis shows the elapsed time in seconds, the x-axis the number of constraints and the coloured lines each stand for an individual modelling tool. The results displayed here are from measurements where T was scaled, like those in figure 4.1, 4.3 and 4.9, but for the charts here and for those in figure 4.14 the non *Calliope* measurements have been done with larger scale if possible so that their numbers of constraints is similar to that of the models optimised with *Calliope*. Concretely, in the measurements of *Fine*, *Oemof* and *Pypsa* T was scaled from 240 to 2400 time steps in 240 time step intervals, and of *Calliope* from 24 to 240 time steps in 24 time step intervals. Most charts in figure 4.13 and 4.14 were cropped to only show the part where the data from multiple modelling tools can be seen.

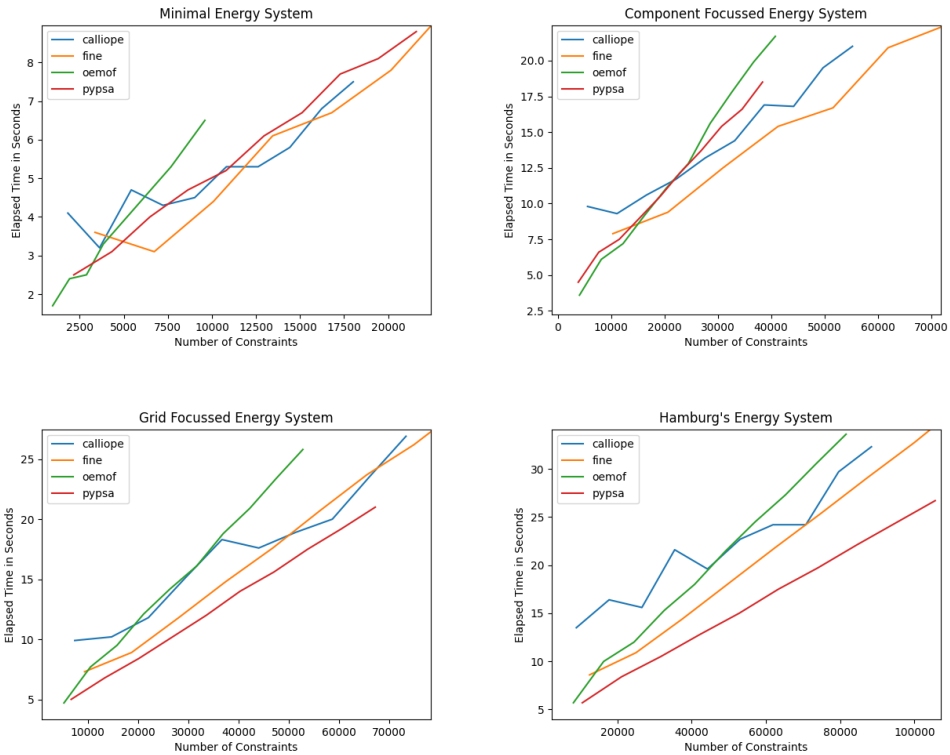


Figure 4.13.: Elapsed time over the number of time steps (own figure)

In the four charts in figure 4.14 the y-axis shows the elapsed time in seconds, the x-axis the number of constraints and the coloured lines each stand for an individual modelling tool. This time the results displayed are form measurements where N was scaled, like those in figure 4.2 and 4.10. In the measurements of *Fine* and *Pypsa* N was scaled from 5 to 50 units in 5 unit intervals for the minimal energy system and from 3 to 30 units in 3

unit intervals for the other models. The *Oemof* measurements while scaling N could not be done with more than ten units as there was an error in the program that prevented this. Therefore not only for *Calliope* but also for *Oemof* T was scaled from one to ten units in one unit intervals. For figure 4.13 the total time and for figure 4.14 the time for optimisation was used, so the absolute values are not directly comparable.

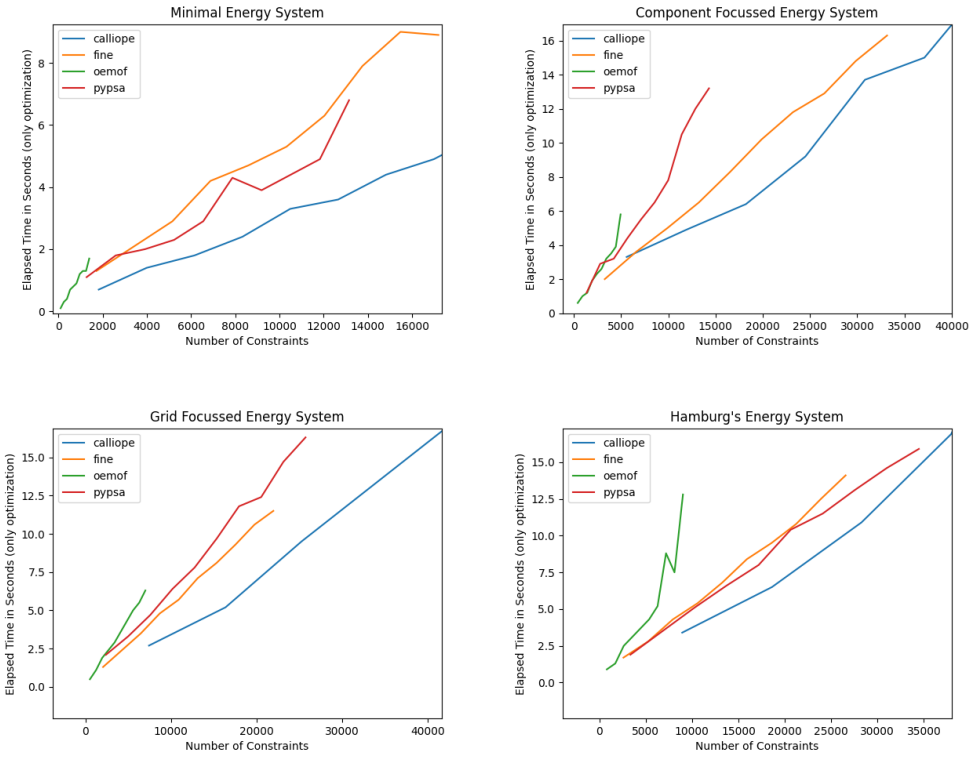


Figure 4.14.: Elapsed time over the number of energy system units (own figure)

The Elapsed time scales linearly over the number of constraints. The lines for the individual modelling tools are quite similar to each other but there are slight differences to be seen. When scaling over T, *Calliope's* line is more flat which means it needs more time per constraint than the other tools for smaller timeframes but less time for larger ones. Apart from this *Oemof's* result is always the highest per constraint. *Fine* needs more time per constraint than *Pypsa* for the minimal and component focussed energy system and needs less time for the grid focussed and Hamburg's energy system. When scaling over N, the the behaviour is similar with the difference that *Calliope* now needs the least time per constraint and *Fine* and *Pypsa* have swapped position when looking at their results for the minimal and Hamburg's energy system. The results for oemof however are hard to judge here with the data available.

When the results for the same number of constraints are compared across models it can be seen that they are in a similar range. For example for all models *Fine* needs around eight seconds with 20,000 constraints in figure 4.13 and around five seconds with 10,000 constraints in figure 4.14. An exception from this is *Calliope* when scaling T. It needs less time for the minimal energy system than for the other models. For example at 10,000 constraints it needs five seconds for the minimal energy system, ten seconds for the component and grid focussed and close to 15 seconds for Hamburg's energy system.

5. Discussion

The first thing to note is that, with the exception of *Calliope's* memory usage when scaling energy system size, all modelling tools scale well in terms of computational resources with both timeframe and energy system size as both memory usage and elapsed time only increase linearly. It is however notable that *Calliope* needs significantly more time and memory than the other modelling tools in all cases. This fits to the fact that *Calliope* also uses more constraints in *Pyomo* to describe the same model. In fact when looking at the figures that show memory and time over the number of constraints (figure 4.6, 4.7, 4.13 and 4.14) it can be seen that *Calliope* with the exception of memory usage when scaling energy system size is as good or even better than the other modelling tools in that regard, as its memory usage or elapsed time per constraint is not much higher or even lower. It therefore can be said with relative certainty that *Calliope's* higher number of constraints is the only reason for its higher demand for computational resources.

An obvious first guess as to why *Calliope's* number of constraints is higher would be that the way *Tessif* transforms the models for *Calliope* greatly differs from the approach chosen for the other modelling tools. Reimer's master's thesis [23] covers the integration of *Calliope* into *Tessif* in detail. In it he too finds that *Calliope* needs more time for optimisation than the other modelling tools and has the suspicion that this is due to the fact that *Calliope* models the impacts that the locations of components have. Tessif does not support having all components at the same location. Reimer therefore created a model directly in *Calliope* without using *Tessif* that was identical to the one he used to make his original comparison with the exception that here all components were in the same location. The results he got were however inconclusive as in one case the optimisation needed significantly more and in an other case significantly less time than for the model that was created and optimised through *Tessif*. Further research is needed to be able to judge if a difference in resource usage related to component location is there or not and also to evaluate if there is the possibility to improve either *Tessif* or *Calliope* in this regard. Also, in his comparison Reimer did not look at the memory usage and the number of constraints. In summary it can be said that it does not seem to be the case that *Tessif* transforms models for *Calliope* in a way that dramatically increases their complexity in comparison to the models for the other tools. There is however a particularity in the way that *Calliope* models energy systems that might be worth looking into as it is unclear if it might improve resource usage if taken into account by tessif. It seems however unlikely that this is the sole cause for the higher number of constraints when using *Calliope*.

The unusual behaviour of *Calliope's* memory when scaling the energy system size however can not just be explained with the relatively high number of constraints. Especially for large model sizes it uses significantly more memory per constraint than the other models. In figure 4.10 the results for the elapsed time also increase quadratically. At first this looks similar and one might guess it is caused by the same effect. The behaviour of the charts in figure 4.10 is caused by the fact that they show the time for the whole measurement process and not only for the optimisation and it turned out that for

models of large energy systems with short timeframes the time for optimisation does not dominate and the time spent before and after optimisation is not negligible. When only the optimisation is looked at in figure 4.12 the measured times differ significantly. They increase linearly instead of quadratically and are lower overall. This effect can not be used to explain *Calliope's* memory usage when scaling the energy system size. Figure 4.8 shows for *Pypsa* that in the case of memory usage optimisation clearly does dominate the result. The other models and modelling tools show the same behaviour. They can be found in the Appendix. *Calliope* does not differ in that regard. Additionally, another measurement was carried out with Hamburg's energy system in which only *Calliope's* memory usage for the optimisation was measured. Its result only slightly differed from those in figure 4.4 and 4.7.

Another thinkable explanation would be that *Calliope* reacts poorly to the way the models are scaled in energy system size. A way to investigate if this is the case is to compare the results of the different models with each other. The minimal energy system is less complex and needs fewer constraints per energy system unit. Therefore when looking at the same number of constraints, the minimal energy system consists of more units. If *Calliope* reacts poorly to the way the models are scaled the expected behaviour would be that when comparing models with the same number of constraints, the model with more energy system units uses more memory than the model that has the same number of constraints with fewer energy system units. This is not the case, the opposite is true. As can be seen in figure 4.7 the minimal energy system uses significantly less memory with the same number of constraints.

Intuitively this result seems logical as it would to make sense that optimising the minimal energy system needs less resources for the same number of constraints as it contains less variability but rather more instances of the same components or in the case of scaling the timeframe size fewer components over a longer timeframe but this effect can not be seen with the other modelling tools and also not for *Calliope's* memory usage when scaling the timeframe size in figure 4.6. It is however visible for *Calliope's* time usage when scaling the timeframe size in figure 4.13 but not the energy system size in figure 4.14.

An effect that is visible and that would fit the before mentioned intuition is that for memory usage it can be seen that the same number of constraints causes significantly less memory usage when scaling the timeframe size than scaling the energy system size (see figure 4.6 and 4.7). For the elapsed time this comparison can not be made as for scaling timeframe size the total time and for scaling energy system size the time for optimisation was used and the absolute values are therefore not directly comparable. Still is seems to be the case that there are constraints of different kind that impact resource usage differently.

One goal of this thesis was to use different energy system models to investigate how their differences impact resource usage. This turns out to be difficult. When the difference in size is compensated by plotting over the number of constraints, differences between the results of the different models are hard to find. Apart from the findings discussed earlier the only difference standing out is that when compared to the other modelling tools *Pypsa* needs less memory and time per constraint when scaling the timeframe size when the model is larger. This might indicate that *Pypsa* is comparatively better suited to optimise larger energy systems while the other modelling tools are in comparison better for optimising longer timeframes. While using multiple energy system models did not

bring much additional insight it did help to show that the results are reliable. When many models deliver the same result this proves that the result is not a fluke or due to some particularity of a certain model but is transferable to other models that were not investigated.

Another goal of this thesis is to showcase a scenario in which *Tessif* is used to do research. The topic of this thesis is quite suitable for this. Creating exactly the same energy system model in multiple modelling tools is an elaborate task especially when the way that each energy system component is modelled in a different way in each tool. With *Tessif* this is no longer an issue. A model is created once and is transformed with a single line of code. This reduced the effort to include multiple models significantly. Also wrappers for tools to measure the memory usage and the elapsed time are integrated in *Tessif*.

In the results it can also be seen what additional resources are used when a model is optimised through *Tessif* and not directly in the respective tool (see figure 4.8 and 4.11). For memory this is negligible, memory used for optimisation far exceeds that used for *Tessif's* code. If the additional time can be neglected depends on the model that is being optimised. Figure 4.11 shows that time spent running tessif code does not increase with timeframe size but only with energy system size. What this means is that when computing time is a critical resource it only makes sense to optimise models with longer timeframes through tessif and for example when many optimisations of models with smaller timeframes are to be performed it makes more sense to do so by hand with the individual modelling tools. From which length of timeframe on it make sense to use tessif depends on the size of the model. An indicator for this is the break even point, at which the time spent on optimisation is no longer smaller than any other step. In the example in figure 4.11 with *Pypsa* using the grid focussed energy system model for N=1 this is already the case at T=48. For N=2 this point is between T=48 and T=96 and for N=3 it is at T=96.

6. Conclusion and Outlook

The thesis shows how *Tessif* can be used to compare the computational resources required to optimise energy system models with different modelling tools. This is further reinforced by the fact that *Tessif's* functionalities have been improved for this. It is now relatively easy to add additional models as units *Tessif's* so called self similar energy system model. This has been done for three additional models. Also added was the ability to return the number of constraints that are used in the optimisation process and a parameter to select the resolution with which the measurements are to be made.

Also shown were the results of the comparison of computational resources that was undertaken. They show that with the exception of *Calliope* the resource usage for the investigated modelling tools scales linearly and is on a relatively similar level. *Calliope's* resource usage is significantly higher. This is true for both its memory and its time usage. It could be shown that the difference in resource usage directly correlates with the difference in the number of constraints that are used by *Pyomo* to describe the models. The number of constraints also is significantly higher for *Calliope*. An exception from this is *Calliope's* memory usage which increases quadratically while the corresponding number of constraints only increases linearly.

Further the additional resource usage when using *Tessif* instead of the modelling tools directly was shown. The additional memory used by *Tessif* is negligible, the additional computing time can however be considerable. It could be shown that the additional time increases with energy system size but does not increase with timeframe size. This means that using *Tessif* when time is a critical resource is only sensible when models with longer timeframes are optimised and not if for example many optimisations with smaller timeframes are to be performed.

6.1. Outlook

There are multiple areas where further research would be conceivable. Additional modelling tools could be added to *Tessif* and those could be investigated too. It might also be interesting to add a significantly larger energy system unit or to measure significantly larger timeframe and energy system sizes. Also the impact that different global constraints and optimisation goals for the same models have could be looked at. It could also be looked into what the reasons are that the number of constraints used by *Calliope* is so much higher, why its memory usage increases quadratically and if this is an intrinsic property of the software or if a workaround is technically feasible.

Bibliography

[1] V. Masson-Delmotte, P. Zhai and A. Pirani: *Climate Change 2021: The Physical Science Basis : Summary for Policymakers : Working Group I Contribution to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change*. Geneva, Switzerland: IPCC, 2021.

[2] M. Chang et al.: 'Trends in Tools and Approaches for Modelling the Energy Transition'. In: *Applied Energy* 290 (May 2021), p. 116731.

[3] *What Ist Free Software?* gnu.org. URL: `https://www.gnu.org/philosophy/free-sw.en.html` (visited on 10/02/2023).

[4] *Open Models*. openmod-initiative.org. URL: `https://wiki.openmod-initiative.org/wiki/Open_Models` (visited on 10/02/2023).

[5] M. Ammon: *Tessif. Documentation*. Insitute for Energy Technology, Hamburg University of Technology. URL: `https://tessif.readthedocs.io/en/stable/`.

[6] L. Kotzur et al.: 'A Modeler's Guide to Handle Complexity in Energy Systems Optimization'. In: *Advances in Applied Energy* 4 (Nov. 2021), p. 100063.

[7] V. Möller et al.: 'Annex II: Glossary'. In: *Climate Change 2022: Impacts, Adaptation, and Vulnerability. Contribution of Working Group II to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change*. Cambridge, UK and New York, NY, USA: IPCC, 2022, pp. 2897–2930.

[8] H.-M. Groscurth, T. Bruckner and R. Kümmel: 'Modeling of Energy-Services Supply Systems'. In: *Energy* 20.9 (Sept. 1995), pp. 941–958.

[9] S. Pfenninger, A. Hawkes and J. Keirstead: 'Energy Systems Modeling for Twenty-First Century Energy Challenges'. In: *Renewable and Sustainable Energy Reviews* 33 (May 2014), pp. 74–86.

[10] S. Pfenninger and B. Pickering: 'Calliope: A Multi-Scale Energy Systems Modelling Framework'. In: *Journal of Open Source Software* 3.29 (12th Sept. 2018), p. 825.

[11] T. Gross et al.: *Fine. Documentation*. URL: `https://vsa-fine.readthedocs.io/en/stable/` (visited on 15/02/2023).

[12] U. Krien et al.: 'Oemof.Solph—A Model Generator for Linear and Mixed-Integer Linear Optimisation of Energy Systems'. In: *Software Impacts* 6 (Nov. 2020), p. 100028.

[13] T. Brown, J. Hörsch and D. Schlachtberger: 'PyPSA: Python for Power System Analysis'. In: *Journal of Open Research Software* 6.1 (16th Jan. 2018), p. 4.

[14] M. Ammon: 'Developing a Method for Choosing and Comparing Energy Supply System Modelling Software'. Hamburg, Germany: TUHH, 2023.

[15] T. E. Oliphant: 'Python for Scientific Computing'. In: *Computing in Science & Engineering* 9.3 (2007), pp. 10–20.

[16] S. A. Cook: 'An Overview of Computational Complexity'. In: *ACM Turing Award Lectures*. New York: Association of Computing Machinery, 1st Jan. 2007, p. 1982.

[17] A. B. Bondi: 'Characteristics of Scalability and Their Impact on Performance'. In: *Proceedings of the 2nd International Workshop on Software and Performance*. WOSP00: 2nd International Workshop on Software and Performance. Ottawa Ontario Canada: ACM, Sept. 2000, pp. 195–203.

[18] F. Emmel: 'Developing a Method for Comparing Free Open Source Energy Supply System Modelling Software in the Context of Computational Complexity'. Hamburg, Germany: TUHH, 30th June 2022.

[19] M. Reimer: 'Entwicklung eines Komponenten basierten Szenarios zum Vergleich von Free and Open Source Energiesystemmodellierungssoftware in Python'. Hamburg, Germany: TUHH, 2022.

[20] T. J. Hanke: 'Entwickeln eines netzbasierten Energiesystemmodells zum Vergleich von Free und Open Source Energiesystemmodellierungssoftware in Python'. Hamburg, Germany: TUHH, 2022.

[21] C. Körber: 'Entwicklung eines methodischen Datenblatt generierenden Vergleichs für Free Open Source Energiesystem-Simulationssoftware anhand eines Fallbeispiels mit den Software-Tools Oemof und PyPSA'. Hamburg, Germany: TUHH, 2021.

[22] M. Ammon: *Tessif*. Version pre-release. Hamburg, Germany: TUHH, 27th Sept. 2022.

[23] M. Reimer: 'Vergleich von Modellierungsprogrammen zur Optimierung von Energiesystemen durch Integration in ein bestehendes Framework zur Transformation von Energiesystem-Modellen'. TUHH Universitätsbibliothek, Sept. 2022.

A. Appendix

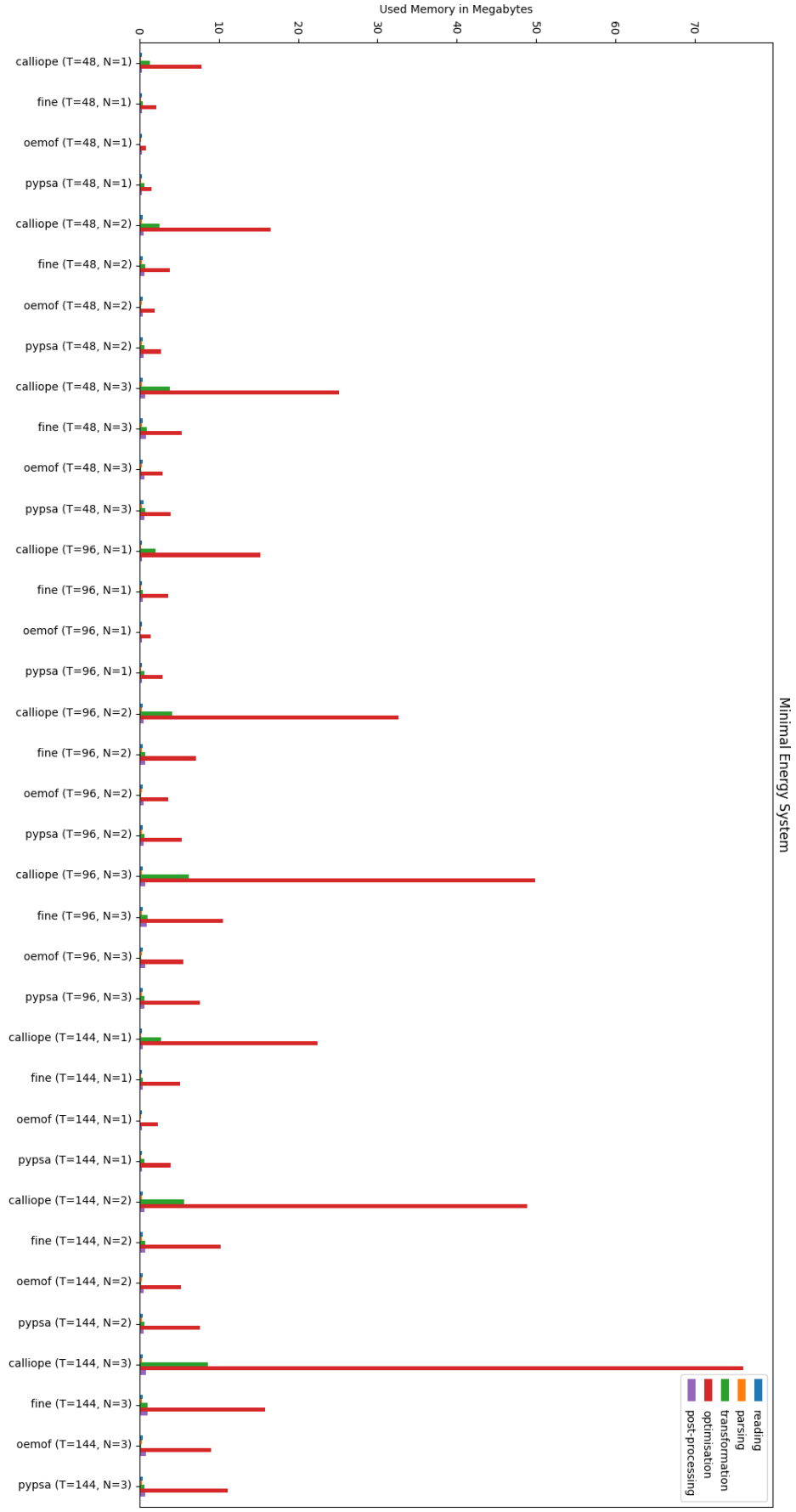


Figure A.1.: Memory usage of individual steps for the minimal energy system (own figure)

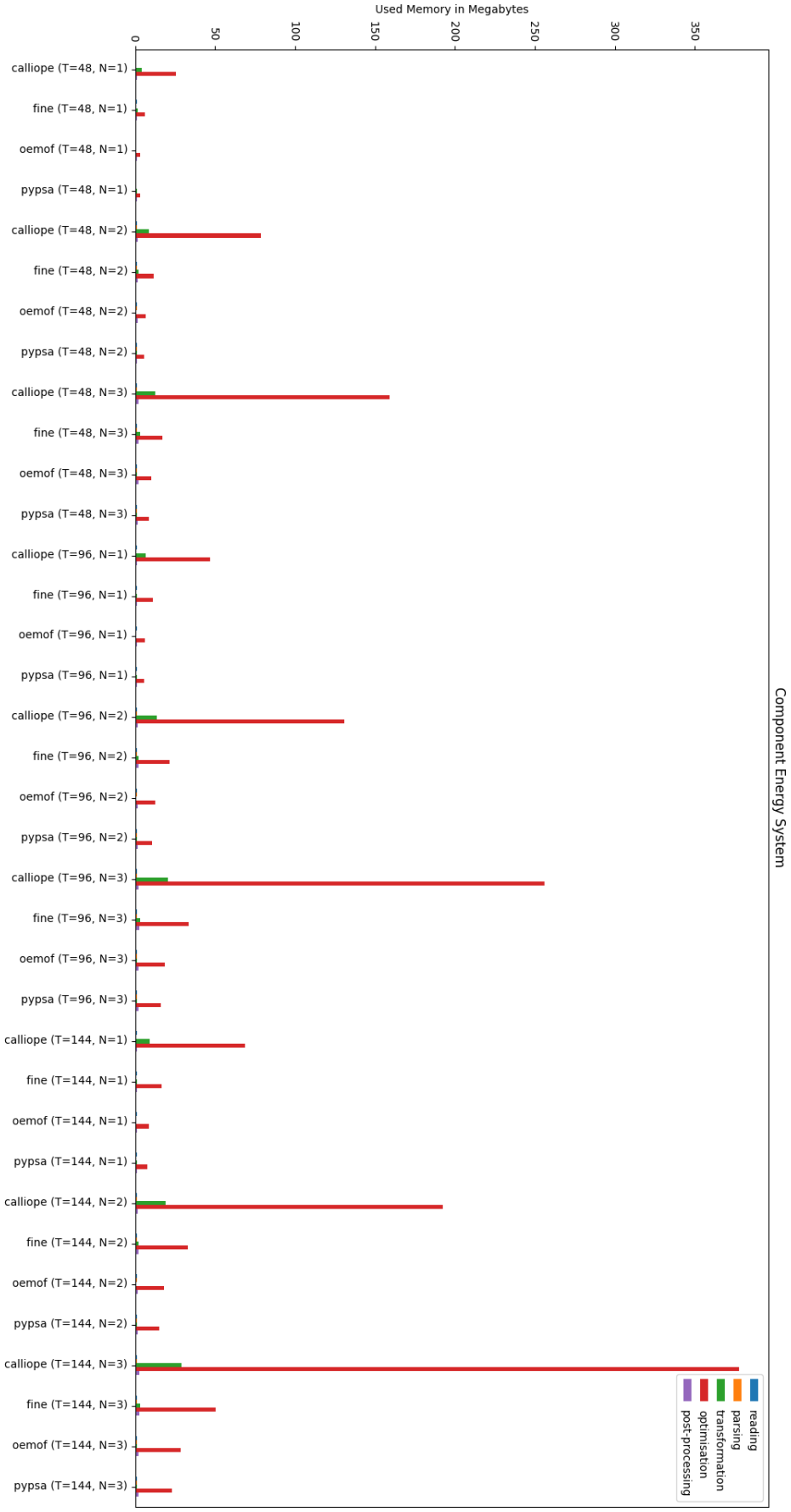


Figure A.2.: Memory usage of individual steps for the component focussed energy system (own figure)

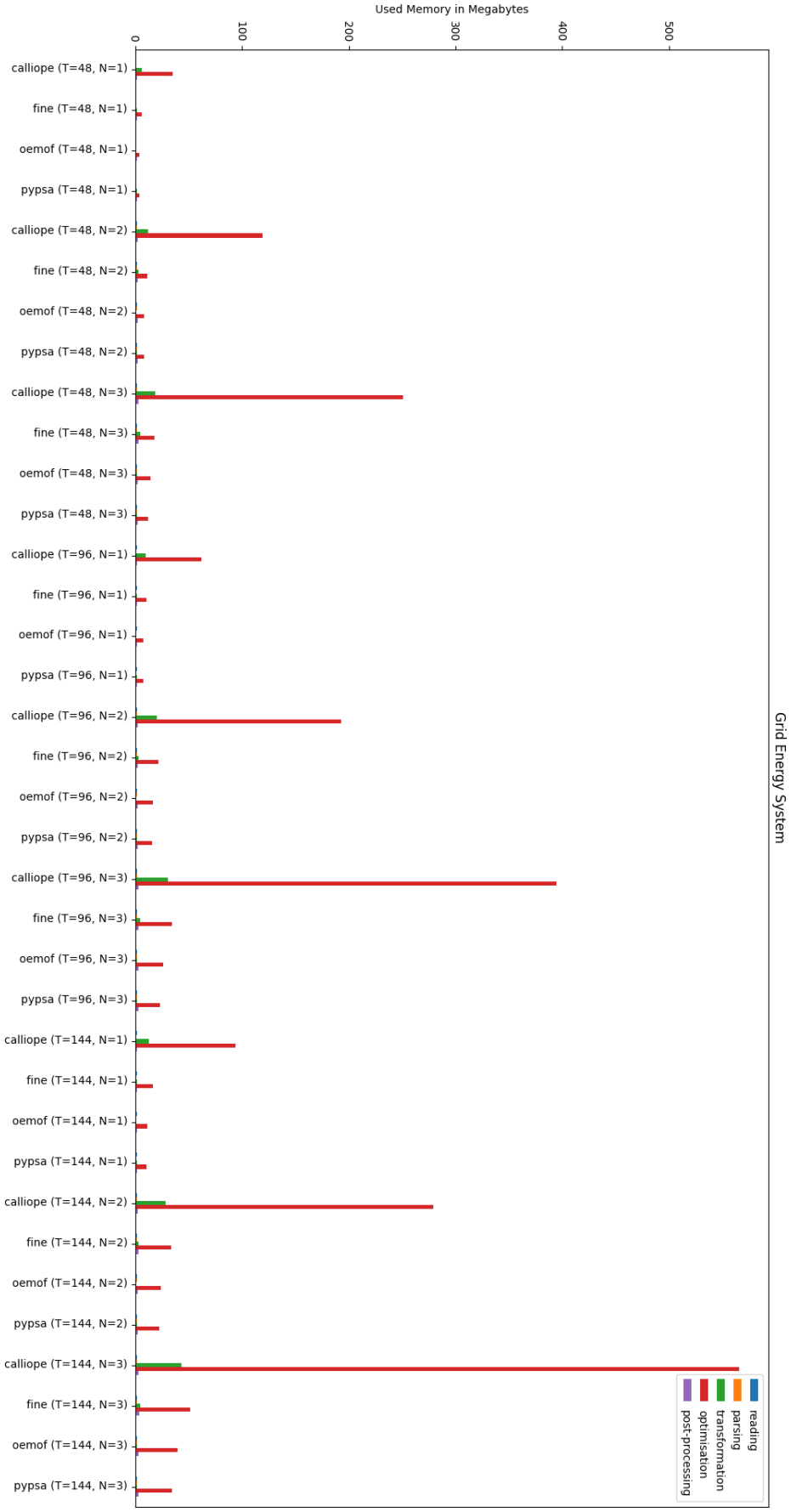


Figure A.3.: Memory usage of individual steps for the grid focussed energy system (own figure)

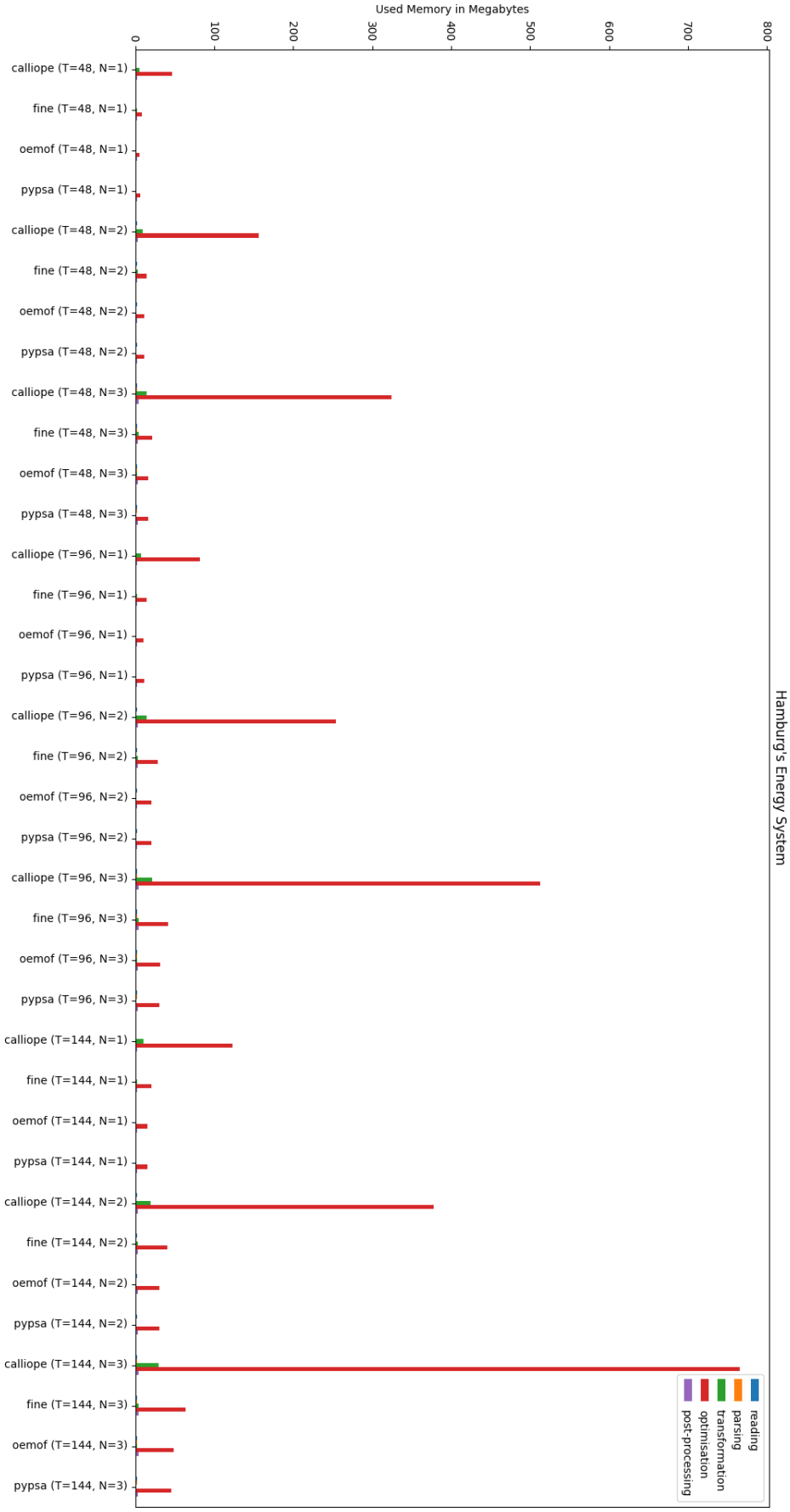


Figure A.4.: Memory usage of individual steps for Hamburg's energy system (own figure)

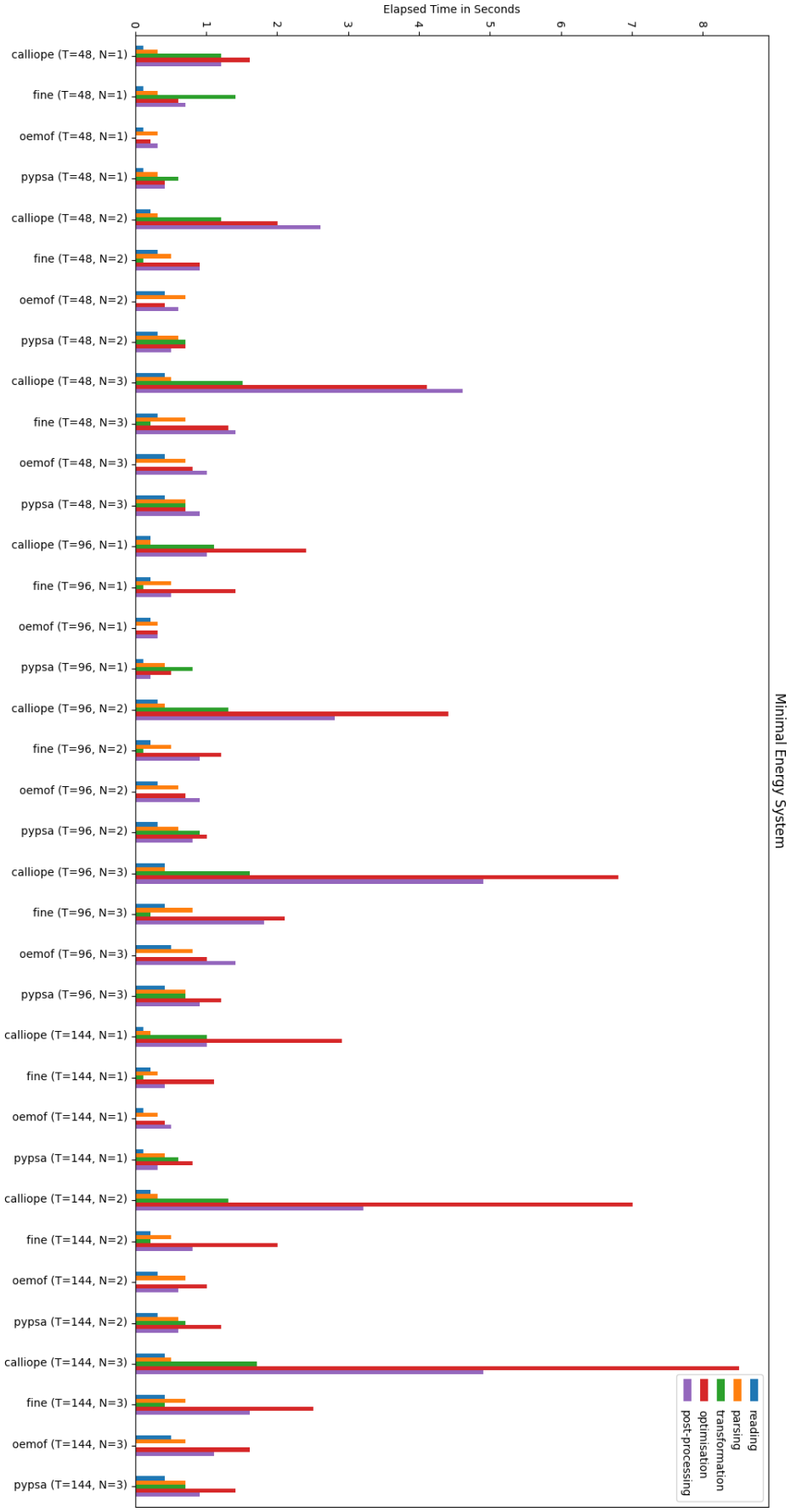


Figure A.5.: Elapsed Time of individual steps for the minimal energy system (own figure)

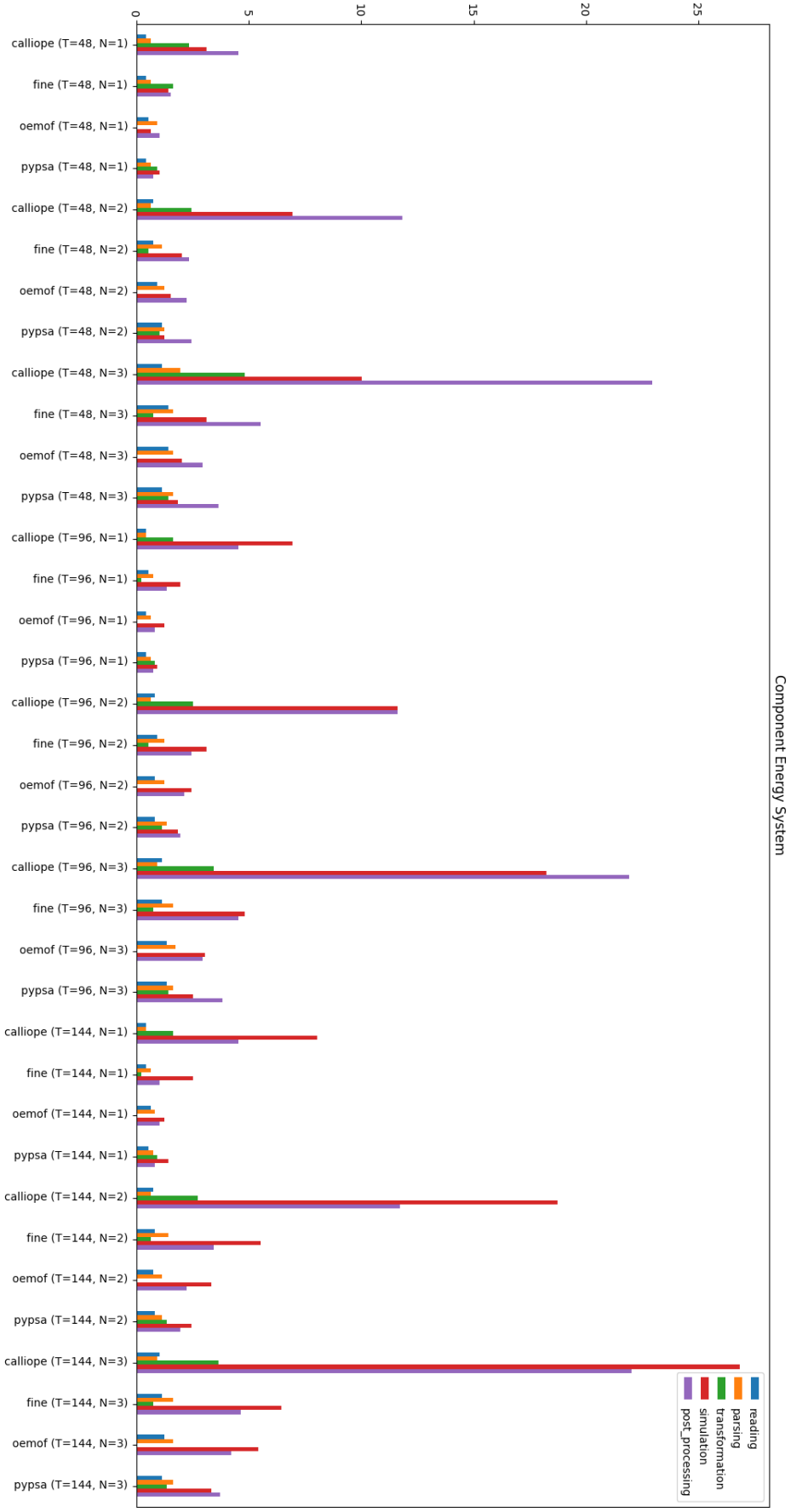


Figure A.6.: Elapsed Time of individual steps for the component focussed energy system (own figure)

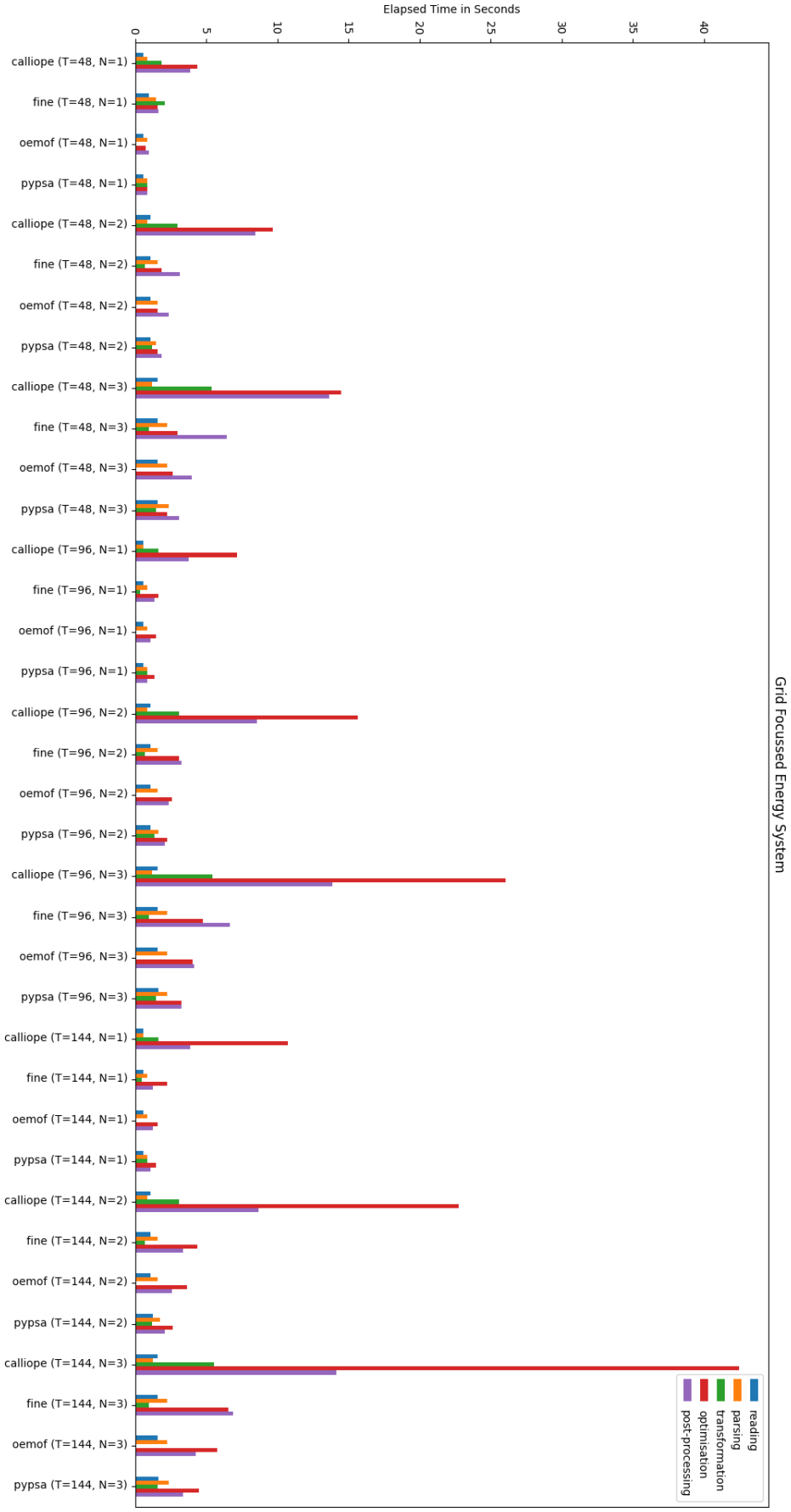


Figure A.7.: Elapsed Time of individual steps for the grid focussed energy system (own figure)

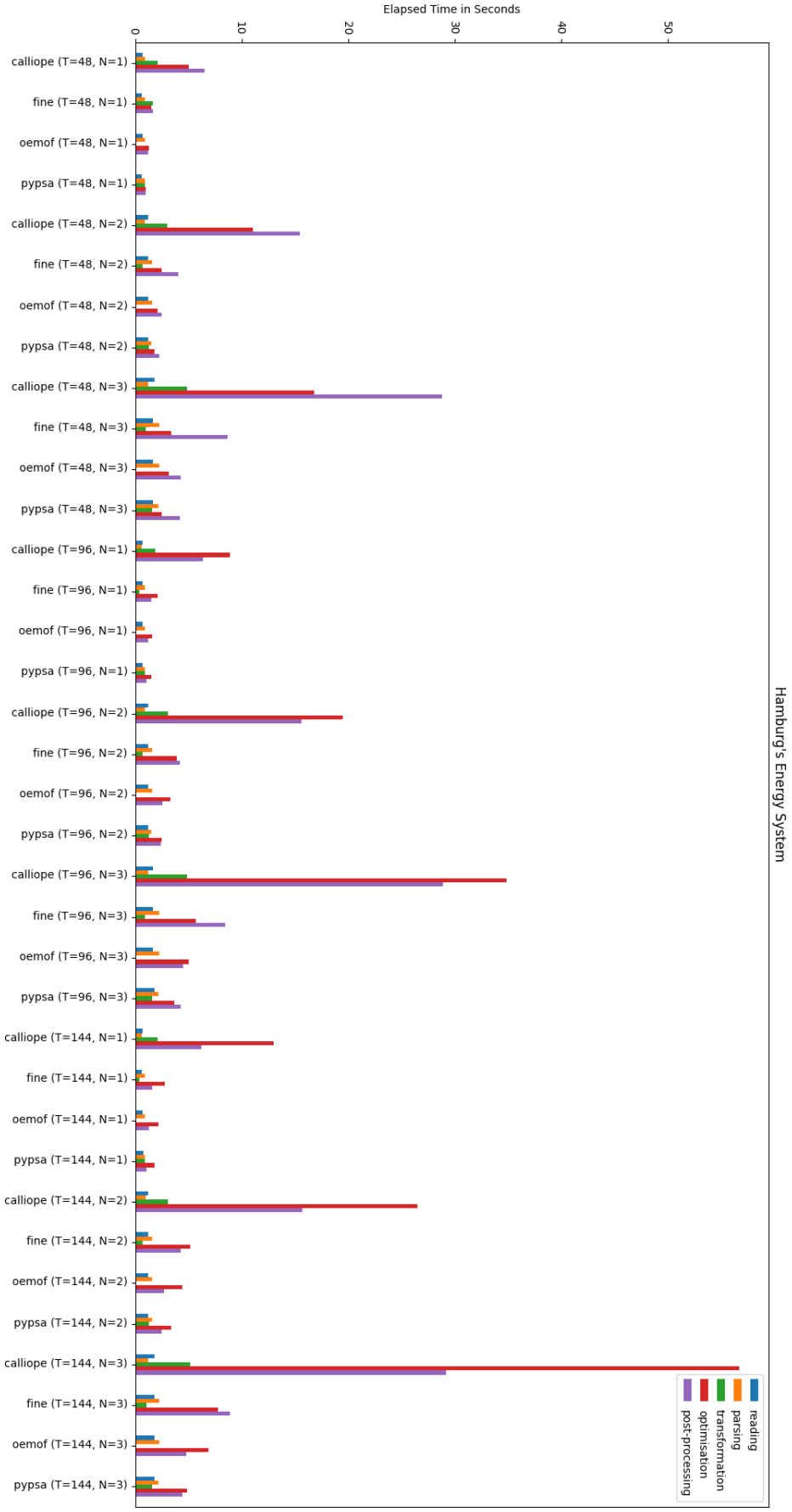


Figure A.8.: Elapsed Time of individual steps for Hamburg's energy system (own figure)