IET
Institut für Energietechnik

TUHH
Technische
Universität
Hamburg

Masterarbeit

Vergleich von Modellierungsprogrammen zur Optimierung von Energiesystemen durch Integration in ein bestehendes Framework zur Transformation von Energiesystem-Modellen

Max Reimer

September 2022

IET
Institut für Energietechnik

TUHH
Technische
Universität
Hamburg

Masterarbeit

Vergleich von Modellierungsprogrammen zur Optimierung von Energiesystemen durch Integration in ein bestehendes Framework zur Transformation von Energiesystem-Modellen

Max Reimer

Matr.-Nr.: 21510087

Erstprüferin:	Dr.-Ing. Kristin Abel-Günther
Zweitprüfer:	Prof. Dr.-Ing. Friedrich Wirz
Betreuer:	Mathias Ammon M.Sc.

Hamburg, 8. September 2022

TUHH

Institut für Energietechnik

Dr.-Ing. K. Abel-Günther

Masterarbeit für Herrn Max Reimer

Matr.-Nr. 21510087

Vergleich von Modellierungsprogrammen zur Optimierung von Energiesystemen durch Integration in ein bestehendes Framework zur Transformation von Energiesystem-Modellen

TUHH
Technische
Universität
Hamburg

Abbildung 0.1.: Logo der TUHH

(Dr.-Ing. Kristin Abel-Günther)

Ich erkläre hiermit, dass die vorliegende Masterarbeit 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, 8. September 2022

(Max Reimer)

Abstract

At the Institute of Energy Technology as part of the Technical University Hamburg the python based framework *Tessif*, which aims to be a platform for a uniform comparison of softwaretools for modelling and optimization of energy systems, is being developed. In this context a model build in *Tessif* is automatically transformed into a model of an integrated tool and then optimized before the results are displayed uniformly. Within this thesis work the list of integrated tools *FINE*, *Oemof* and *PyPSA* is going to be expanded by another appropriate tool. In a comparison of different Tools, *Calliope* is being identified as an adequate tool and going to be integrated in *Tessif*.

The main parts of the integration is the transformation of the *Tessif* model into a *Calliope* model as well as the restructuring of the results given by *Calliope* into *Tessif*'s uniform data output. Not all *Tessif* models can be perfectly transformed into *Calliope* models due to differences in model parameters. Differences are pointed out and their impact is analysed. Opportunities to handle these as well as ensure a proper integration are proposed.

With the help of chosen example models the correct integration of *Calliope* is ensured and differences between the compared tools are pointed out. The identified differences are going to be related to the already integrated tools, to derive specific use cases.

Kurzfassung

Am Institut für Energietechnik der Technischen Universität Hamburg wird das Python basierte Framework *Tessif*, welches eine Plattform für einen einheitlichen Vergleich von Softwaretools zur Modellierung und Optimierung von Energiesystemen darstellt, entwickelt. Hierbei wird ein in *Tessif* erstelltes Modell automatisiert in ein Modell eines der integrierten Tools umgewandelt und optimiert, ehe die Ergebnisse einheitlich ausgegeben werden. Zu den bisher in *Tessif* integrierten Tools *FINE*, *Oemof* und *PyPSA* wird im Rahmen dieser Arbeit ein weiteres geeignetes Tool integriert. Als ein solches Tool wird, aus einer Reihe potenzieller Kandidaten, *Calliope* als besonders gut geeignet identifiziert und in *Tessif* integriert.

Den Hauptbestandteil der Integration stellen die Übersetzung bzw. Transformation des *Tessif*-Modells in ein *Calliope*-Modell sowie die Umstrukturierung der Ergebnisse von *Calliope* in den einheitlichen Datenoutput *Tessif*'s dar. An dieser Stelle sind aufgrund von Unterschieden in der Modellierung einige Parameter des *Tessif*-Modells nicht in *Calliope* darstellbar. Die Unterschiede werden aufgezeigt sowie deren Einflüsse untersucht. Es werden außerdem Möglichkeiten vorgeschlagen, mit diesen umzugehen und eine erfolgreiche Integration sicherzustellen.

Anhand ausgewählter Beispielmodelle wird die korrekte Integration *Calliope*'s sichergestellt und Unterschiede zwischen den verschiedenen Tools ausgearbeitet. Die identifizierten Differenzen werden in Zusammenhang mit den bereits integrierten Softwaretools gebracht, um daraus spezifische Anwendungsfälle abzuleiten.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

Nomenklatur

Lateinische Symbole

Symbol	Einheit	Bedeutung
c	€	Kosten
r	-	Verhältnis der Energieträger

Griechische Symbole

Symbol	Einheit	Bedeutung
η	-	Wirkungsgrad

Indizes

Index	Bedeutung
average	Durchschnitt
carrier	Energieträger
cllp	Calliope
el	elektrisch
in	Eintritt
th	thermisch
out	Austritt
out1	Primärer Austritt
out2	Sekundärer Austritt

Abkürzungen

Abkürzungen

Capex	Investitionsausgaben; engl.: capital expenditure
CSV	Comma Separated Values
EE	Erneuerbare Energie
es2es	Energysystem to energysystem
es2mapping	Energysystem to mapping
FINE	Framework for Integrated Energy System Assessment
FOSS	Free and Open Source Software
IET	Institut für Energietechnik
KWK	Kraft-Wärme-Kopplung
LP	Lineare Programmierung; engl.: linear programming
MILP	Gemischt-ganzzahlige lineare Programmierung; engl.: mixed integer linear programming
NLP	Nichtlineare Programmierung; engl.: nonlinear programming
Oemof	Open energy modelling framework
Opex	Betriebskosten; engl.: Operating Expenses
PV	Photovoltaik
PyPSA	Python for Power System Analysis
spez.	spezifisch
Spores	Spatially-explicit Practically Optimal Results
Tessif	Transforming Energy Supply System (Modelling) Framework
TUHH	Technische Universität Hamburg
UID	Unique Identifier
YAML	YAML Ain't Markup Language

1. Einleitung

Im folgenden wird eine Einleitung der Arbeit vorgestellt. Dafür wird zunächst auf ihre Motivation eingegangen, indem die Thematik der Modellierung von Energiesystemen mit aktuellen politischen, gesellschaftlichen und wissenschaftlichen Aspekten in Verbindung gebracht wird. Anschließend wird daraus ein entsprechendes Ziel abgeleitet und vorgestellt.

1.1. Motivation

Eine weitgreifende Umstellung der Energieerzeugung ist zum Aufhalten des Klimawandels erforderlich. Ein Wechsel auf, zumeist fluktuierende, erneuerbare Energien von einem weitestgehend auf fossilen Brennstoffen beruhenden Energieversorgungssystem benötigt neben dem Ausbau von Leistungen ebenfalls einen Ausbau der Energietransportsysteme sowie der Speichermethoden [1].

Die Modellierung dieser Energiesysteme kann das Abschätzen und Bewerten von den damit einhergehenden Kosten und den erzielten Emissionseinsparungen unterstützen. Jedoch existiert eine Vielzahl von Modellierungstools, welche alle verschiedene Stärken und Schwächen aufweisen. Dementsprechend ist die Wahl eines, für das zu untersuchende System und die dazugehörige Problemstellung, geeigneten Tools von großer Bedeutung [2].

Eine Reihe von Arbeiten befasst sich mit dem Vergleich von Modellierungstools, wobei in der Regel das Ziel verfolgt wird, den Leser bei der Wahl eines Tools zu unterstützen [2–5]. Aufgrund dessen, dass diese Vergleiche selten über ein Gegenüberstellen von Ziel, Methodik, Zeithorizont, sektoraler Abdeckung u.ä. hinaus gehen, besteht ein Mangel an tiefergreifenden Vergleichen, welche eine Nutzung der Tools sowie einen Vergleich der erzielten Ergebnisse einschließen.

Im Rahmen dieser Arbeit wird das Softwaretool *Tessif* mit der Integration eines weiteren Modellierungstools erweitert. *Tessif* ist ein am Institut für Energietechnik (IET) der Technischen Universität Hamburg (TUHH) entwickeltes Framework, welches durch eine einheitliche Modellformulierung, sowie einen einheitlichen Datenoutput eine Vergleichsplattform für verschiedene Modellierungstools darstellt.

1.2. Ziel der Arbeit

Ziel dieser Arbeit ist es *Tessif* um die Unterstützung eines zusätzlichen Modellierungstools zu erweitern und einen Vergleich dieses Tools mit den bereits integrierten durchzuführen. Dieser Vergleich strebt es an, die Wahl eines geeigneten Tools, abhängig von der jeweiligen Problemstellung, zu unterstützen.

Hierfür wird in Kapitel 2 zunächst die Bedeutung der Modellierung von Energiesystemen sowie der Wahl eines passenden Modellierungstools erörtert. Zudem wird das Framework *Tessif*, welches einen einfachen Vergleich verschiedener Tools ermöglicht, genauer vorgestellt. Daraufhin erfolgt ein Gegenüberstellen verschiedener zur Integration in *Tessif* geeigneter Tools, sowie die Wahl eines dieser. Das gewählte Tool wird daraufhin genauer vorgestellt.

In Kapitel 3 wird die Integration des zuvor ausgewählten Tools *Calliope* in *Tessif* durchgeführt. Aufgrund von elementaren Unterschieden in der Definition der Modellierung einiger Komponenten innerhalb des Energiesystem-Modells, lässt sich nicht jedes in *Tessif* erstellte Modell detailgetreu in *Calliope* übertragen. Die Unterschiede werden in diesem Kapitel besonders hervorgehoben.

Daraufhin erfolgt in Kapitel 4, anhand einer Reihe von *Tessif* Beispielmodellen, eine Überprüfung der Implementierung von *Calliope*, sowie ein Vergleich zu den anderen integrierten Tools. Diese Modelle sind zum Teil so stark vereinfacht, dass die Nutzung eines Optimierungstools trivial erscheint. Dennoch eignen sie sich zur schnellen Ermittlung von grundlegenden Fehlern in der Übersetzung des Modells sowie zur eindeutigen Identifikation von Unterschieden zwischen den Tools.

In Kapitel 5 werden die Ergebnisse bezüglich der Integration von *Calliope* in *Tessif* und des Vergleichs zu den anderen Tools mittels *Tessif* ausgewertet. Darüber hinaus werden einige Grenzen des Vergleichs der Softwaretools mithilfe von *Tessif* aufgeführt.

Abschließend werden in Kapitel 6 die wesentlichen Erkenntnisse in einem Fazit zusammengefasst. Zudem wird ein Überblick gegeben, unter welchen Voraussetzungen welches Tool, auf Grundlage der Ergebnisse dieser Arbeit, zu empfehlen ist. Des Weiteren erfolgt ein Ausblick für die Zukunft von *Tessif*, den Nutzen der *Calliope* Integration und der Ergebnisse des Vergleichs.

2. Grundlagen

Dieses Kapitel vermittelt die wesentlichen Grundlagen, auf welchen die darauf folgenden Kapitel aufbauen. So wird zunächst auf die Bedeutung und die, in dieser Arbeit untersuchten, Methoden der Modellierung von Energieversorgungssystemen eingegangen. Anschließend wird das Framework *Tessif* vorgestellt und die einzelnen Details beschrieben. Ein in dieser Arbeit in *Tessif* zu implementierendes Softwaretool wird daraufhin aus verschiedene potenzielle Kandidaten ausgewählt und genauer vorgestellt.

2.1. Modellierung von Energieversorgungssystemen

Mit dem Pariser Abkommen von 2015 haben sich 196 Staaten dazu entschlossen dem Klimawandel entgegenzuwirken und den Anstieg der anthropogen verursachten, mittleren, globalen Temperatur auf deutlich unter 2 °C gegenüber vorindustriellen Werten zu beschränken. Dies soll vor allem durch Reduktion der Treibhausgasemissionen gelingen [6]. Hierbei spielt der Stromsektor eine führende Rolle. Deshalb gewinnen vermehrt erneuerbare Energien (EE), in der sonst auf fossilen Brennstoffen aufbauenden Stromerzeugung, an Bedeutung. Im Jahr 2020 war an der Stromerzeugung in Europa erstmals der Anteil an EE größer als jener der fossilen Brennstoffe [7].

Der Ausbau der EE ist jedoch nicht der einzige notwendige Schritt im Vorgehen gegen den Klimawandel. So gelten der Ausbau des elektrischen Übertragungsnetzes, der Ausbau von Energiespeichern und die Kopplung der Sektoren Strom, Wärme und Mobilität ebenfalls als notwendig [1]. Durch diesen Wandel der Energiesysteme entstehen nicht nur neue Herausforderungen sondern ebenso ein steigendes öffentliches Interesse [8]. So steigt das Interesse an der Analye von Energiesystemen seit zwei Jahrzehnten stark an, was auch an der Anzahl an entsprechenden Veröffentlichungen sowie an Zitierungen von Veröffentlichungen deutlich zu erkennen ist. Bei der Suche nach Schlagwörtern von Veröffentlichungen und aufstellen von Trends zwischen den Jahren 2015 und 2019 zeigen Dominković et al. einen steigenden Trend bei den Schlagwörtern „Renewable energy“, „Energy system modelling“, „Optimisation“ sowie vielen weiteren Schlagwörtern mit einem Bezug zum Energiesektor [8].

Bei der Untersuchung von Energiesystemen, sowie zum Abschätzen eines Rahmens in welchem sich zukünftige Energiesysteme unter verschiedenen Annahmen befinden könnten, sind Modelle von großer Bedeutung. So können gewonnene Erkenntnisse bei den letztlich in der Energiepolitik getroffenen Entscheidung eine unterstützende Rollen einnehmen [9–13].

Diese Modelle sind vereinfachte und abstrahierte Darstellungen von Energiesystemen, welche erstellt werden um spezifische Untersuchungen durchzuführen und beinhalten für gewöhnlich Code und Daten. Code in diesem Sinne steht für den Programm-Code zum Lesen der Daten sowie zum Konstruieren und Lösen von Gleichungen. Daten stehen dabei unter anderem für Input- und Output-Daten, wie spezifische Technologiekosten oder zeitlich aufgelöster Erzeugungs- und Verbrauchsprofile [14].

Modelle sind hierbei von Frameworks zu unterscheiden. Frameworks bilden ein Gerüst in Form von Code, welches durch hinzufügen von Daten automatisiert Modelle erstellt [14]. Diese Unterscheidung ist in der Literatur jedoch nicht immer gegeben. So werden die Frameworks beispielsweise von Lopion et al. [2] sowie Ringkjøb et al. [5] ebenfalls als Modelle bezeichnet. Als eine weitere Bezeichnung eines Framworks existiert zudem die Bezeichnung als Tool [11].

Das Erstellen eines Modells bedeutet vor allem die Faktoren zu identifizieren, welche in Anbetracht der Problemstellung von größter Bedeutung sind. Für unbekannte Parameter gilt es oftmals begründete Annahmen zu treffen [9].

Sollen verschiedene Modelle miteinander verglichen werden, so ist zu beachten, dass aufgrund der Verschiedenheiten einzelner Modelle es nicht möglich ist alle direkt miteinander zu vergleichen. Stattdessen gilt es den Fokus auf jene vergleichbaren Modelle zu beschränken [9].

Als grundlegender Unterschied ist beispielsweise der methodische Ansatz der Simulation gegenüber der Optimierung zu nennen. Während bei der Simulation die Ausführung eines vorgegeben Systems berechnet wird, liefert eine Optimierung den optimalen Systementwurf [9]. In dieser Arbeit werden ausschließlich Modellierungstools zur Optimierung von Energiesystemen betrachtet.

Genauer wird ausschließlich die mathematische Programmierung betrachtet. Diese lässt sich unterteilen in lineare Programmierung (linear programming; LP), gemischt-ganzzahlig lineare Programmierung (mixed integer linear programming; MILP[1]) und nichtlineare Programmierung (nonlinear programming; NLP) [15]. Der Fokus liegt hierbei vor allem auf der linearen Programmierung, welche der am häufigsten verwendete Ansatz in der Optimierung von Energiesystemen ist. Zudem ist dies jener Ansatz mit den wenigsten Nebenbedingung und dementsprechend der am wenigsten Rechenaufwendige [16].

[1] Oft auch nur als MIP (mixed integer programming) bezeichnet.

Im Rahmen der Modellierung und Optimierung von Energiesystemen werden in der Regel die Gesamtkosten des Systems minimiert [17]. Bei der Optimierung sind zudem eine Reihe von Nebenbedingungen einzuhalten, welche sich in die Kategorien

- Gleichungsnebenbedingungen,
- Ungleichungsnebenbedingungen und
- Parametergrenzen

unterteilen lassen. Bei der linearen Programmierung beschränken sich diese Nebenbedingungen ebenso wie die Optimierungsfunktion selbst auf lineare Zusammenhänge [15].

Als eine Gleichungsnebenbedingung könnte beispielsweise die Vorgabe von Null Emissionen vorliegen. Werden die Emissionen stattdessen als eine obere Grenze angegeben, so handelt es sich um eine Ungleichungsnebenbedingungen. Parametergrenzen sind in jeder Komponente eines Modells, beispielsweise in Form von Begrenzungen der Kapazität, der Leistungssprünge oder anderen Parametern, gegeben.

Die verschiedenen Modellierungstools bieten eine Auswahl von Basiskomponenten bzw. Technologien, welche genutzt werden um ein Energiesystem zu erstellen. Die Komponenten stellen dabei eine anwendungsorientierte Abstraktion der mathematischen Formulierung dar. So werden durch das Hinzufügen und Verknüpfen der einzelnen Komponenten automatisch die entsprechenden mathematischen Zusammenhänge einem Gleichungssystem hinzugefügt, für das anschließend eine (quasi) optimale Lösung gesucht wird [18, 19].

Es existieren sowohl frei zugängliche als auch kostenpflichtige Tools zur Modellierung von Energiesystemen [5]. Ist bei einem Tool auch der Quellcode frei zugänglich, so werden diese im Weiteren als FOSS (free and open source software) Tools bezeichnet. FOSS Tools sind somit jene Tools welche unter die *Open Definition* der Open Knowledge Foundation fallen und somit frei zugänglich sind und von jedem und für jeden Zweck genutzt, angepasst und geteilt werden können [20].

FOSS Tools haben den Vorteil, das die Grundsätze des wissenschaftlichen Arbeitens wie Transparenz, Reproduzierbarkeit und Nachvollziehbarkeit sowie die Möglichkeit von Peer-Reviews gewährleistet sind. Dies sorgt nach Pfenninger et al. für eine erhöhte Qualität und höhere Produktivität [12].

Der freie Zugang zu leistungsfähigen Tools führt jedoch ebenso zum Risiko von Fehlinterpretationen der Ergebnisse durch Nutzer deren Hintergrundwissen zu Energiesystemen für eine fundierte Bewertung der Modellergebnisse zu gering ist [13].

Allgemein gilt, dass die Komplexität von frei zugänglichen Tools geringer als jene kommerzieller Tools ist. Dies ist unter anderem auf die Tatsache zurückzuführen, dass FOSS Tools erst am Anfang ihrer Entwicklung sind, während kostenpflichtige bereits lange etabliert sind [13].

Im Bereich der Modellierung von Energiesystemen sind drei aktuelle Trends zu erkennen [11]:

- Vermehrte Modellierung von Wechselwirkungen bei Sektorkopplung
- Ansteigender Fokus auf freie Verfügbarkeit
- Verbesserte zeitliche Auflösung um variable erneuerbare Energien detailliert abbilden zu können

Weitere Trends haben Lopion et al. [2] in einer Betrachtung von 24 Tools ausgearbeitet. So existiert seit 2010 ein Trend weg von Tools basierend auf der Simulation und hin zur Optimierung, sowie der Trend zur Programmiersprache Python. Darüber hinaus ist seit 2000 ein Anstieg an frei zugänglichen Tools zu vermerken [2].

Die Anzahl verglichener Tools ist jedoch mit Vorsicht zu genießen, in Betracht dessen, dass deutlich mehr Tools existieren. So werden 24 Tools von Lopion et al. [2] betrachtet, während die Arbeit von Chang et al. [11] eine Summe von 137 Tools identifiziert. Ein Fazit über aktuelle Trends ist folglich bei einer Betrachtung von lediglich 24 Tools schwierig.

Die Wahl des für die gegebene Problemstellung am Besten geeigneten Tools ist von großer Bedeutung, da jedes Tool verschieden ist und damit einhergehend auch verschiedene Stärken und Schwächen aufweist und keines existiert, welches jeder Problemstellung gerecht wird [2–4]. Dementsprechend ist auch der Vergleich von Modellen und Modellierungstools von ebenso großer Bedeutung. Wie Chang et al. [11] aufzeigen existiert eine Vielzahl von solchen Vergleichen. Viele dieser Arbeiten verfolgen das Ziel ihren Lesern bei der Wahl eines geeigneten Tools zur Modellierung von Energiesystemen zu unterstützen [2–5].

Diesem Ziel, welches folglich ein über Jahre hinweg relevantes Thema im Bereich der Modellierung von Energiesystemen darstellt, widmet sich auch das Framework *Tessif*, welches im Folgenden genauer beschrieben wird und die Grundlage des in dieser Arbeit erfolgenden Vergleichs von Modellierungstools bildet.

2.2. Transforming Energy Supply System (Modelling) Framework

Tessif (*Transforming Energy Supply System (Modelling) Framework*) wird am Institut für Energietechnik (IET) der Technischen Universität Hamburg (TUHH) entwickelt und ist ein Framework zum einfachen Vergleich verschiedener Energiesystem Modellierungstools. *Tessif* beschränkt sich hierbei auf „free and open source software“ (FOSS). Das Modell eines Energiesystems wird in *Tessif* erstellt

und daraufhin in ein Modell eines der integrierten Softwaretools übersetzt, sodass dieses die Optimierung durchführen kann. Die Optimierungsergebnisse werden daraufhin von *Tessif* einheitlich aufbereitet. Dieser Vorgang ist schematisch in Abbildung 2.1 dargestellt. Die *Tessif* Funktion zum Übersetzen des *Tessif*-Modells in ein Modell der spezifisch zu untersuchenden Software befindet sich im *Tessif* Unterordner `es2es` (energysystem to energysystem) und die Aufbereitung der Ergebnisse im `es2mapping` (energysystem to mapping). Darüber hinaus bietet *Tessif* eine Funktion namens *Comparatier*, in welcher mehrere zu untersuchende in *Tessif* implementierte Tools in einem Befehl aufgerufen werden können. Das Modell wird daraufhin nacheinander in die jeweiligen spezifischen Modelle übersetzt und optimiert. Das Ergebnis wird letztlich für alle Tools parallel ausgegeben und ermöglicht einen direkten Vergleich [21].

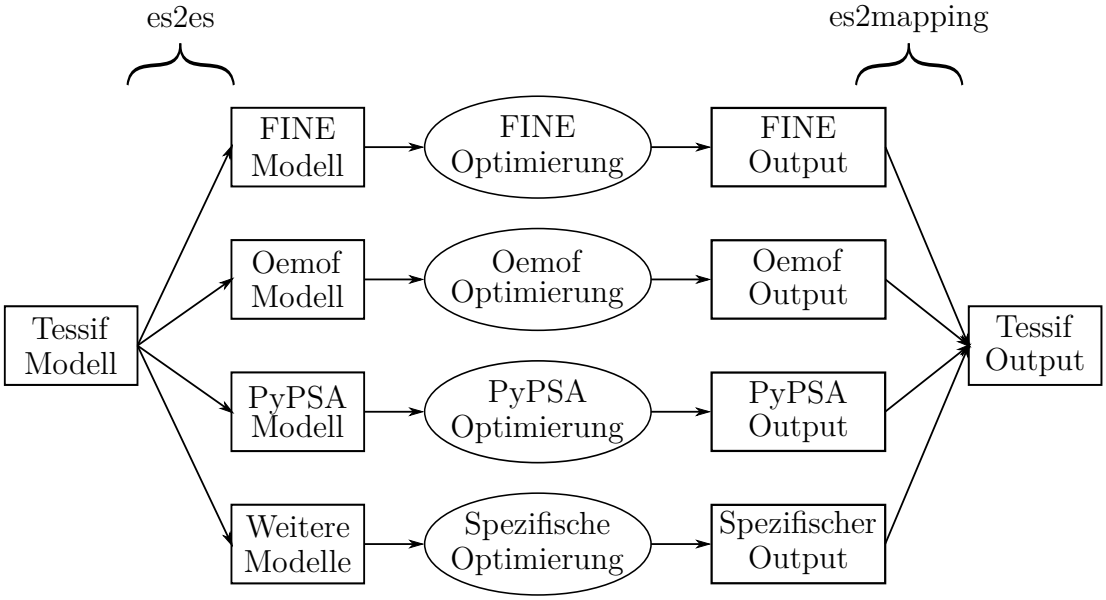

Abbildung 2.1.: Schematische Darstellung der Funktionsweise von *Tessif* (in Anlehnung an [21])

In *Tessif* wird jedes Energiesystem als Graph angesehen. Für die Formulierung des Modells eines Energiesystems stehen in *Tessif* sechs Komponenten zur Verfügung, welche parametrisiert werden können. Erzeuger (*Source*), welche den Eintrittspunkt für Energie in das Energiesystem darstellen, Verbraucher (*Sink*), welche als Austrittspunkt für Energie aus dem Energiesystem hinaus das Gegenstück zum Erzeuger darstellen, sowie eine Komponente zum Energietransport (*Bus*) bilden das Grundgerüst und finden sich in jedem Modell mindestens einmal wieder. Darüber hinaus gibt es Energiespeicher (*Storage*), welche überschüssige Energie in einem Zeitpunkt aufnehmen und bei Bedarf zu einem anderen Zeitpunkt abgeben, Energiewandler (*Transformer*), welche eine Energieform in eine andere wandeln, sowie Konnektoren (*Connector*), welche eine Energieform von einem *Bus* verlustbehaftet zu einem anderen *Bus* übertragen. Auch wenn die Energieverbraucher die einzige Komponente darstellen, welche die Aufgabe haben Energie aus dem System abzuführen, so wird an jeder Komponente mit einem Wirkungsgrad von kleiner als

eins Energie abgeführt. In der Betrachtung als Graph werden alle Komponenten als Knoten dargestellt und mit gerichteten Kanten, welche den Energiefluss darstellen, verbunden [21].

Eine solche grafische Darstellung mit allen *Tessif* Komponenten ist in Abbildung 2.2 zu sehen.

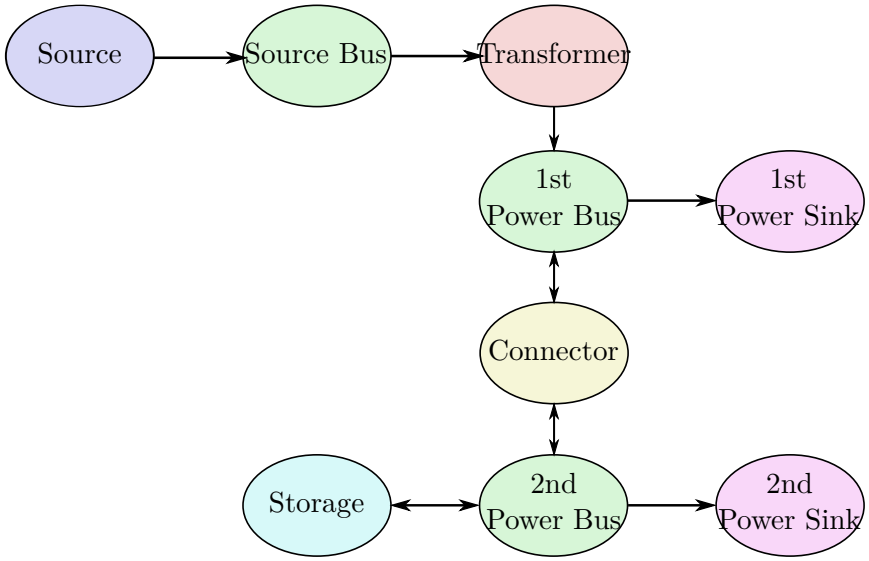

Abbildung 2.2.: Graphische Darstellung eines Energiesystems mit allen *Tessif* Komponenten (eigene Darstellung)

Die Optimierungsergebnisse bereitet *Tessif* in tabellarischer Form auf, sodass ein einfacher Vergleich der Ergebnisse verschiedener Modellierungstools gewährleistet ist. Neben der tabellarischen Darstellung, welche sowohl für jeden Zeitschritt als auch in Summe ausgegeben werden kann, bietet *Tessif* eine Vielzahl von visuellen Möglichkeiten die Ergebnisse darzustellen. So können die Leistungen über ein Energienetz für jeden Zeitschritt oder die Leistung jeder Komponente aufgeteilt in die verschiedenen Sektoren abgebildet werden. Ebenso lassen sich Lastdauerlinien erstellen, sowie Graphen welche an den Kanten zwischen zwei Knoten jeweils den gesamten Energiefluss, sowie die spezifischen Kosten und Emissionen aufführen. In einer weiteren grafischen Darstellung sind diese Zahlenwerte durch variable Kantenlänge, -breite und -graustufe in Abhängigkeit von Kosten, Energiefluss und Emissionen abgebildet [21].

Im Folgenden wird ein kurzer Überblick über die aktuell in Tessif implementierten Tools gegeben. Diese sind allesamt in Python geschriebene FOSS Tools, verwenden das Optimierungspaket *Pyomo*[2] und arbeiten somit auch mit allen mit *Pyomo* kompatiblen Solvern.

[2] `https://www.pyomo.org/` (besucht am 08.06.2022)

2.2.1. Framework for Integrated Energy System Assessment

Das „*Framework for Integrated Energy System Assessment*“ - *FINE* [22] ist ein Open Source Framework zum Modellieren und Optimieren von Energiesystemen. Sektoren übergreifende Energiesysteme können mit *FINE* erstellt und hinsichtlich ihrer Kosten unter Berücksichtigung von technischen und ökologischen Beschränkungen optimiert werden [22]. Im Rahmen dieser Arbeit wird die *FINE* Version 2.2.1 verwendet, welche von Schnute [23] in *Tessif* integriert wurde.

2.2.2. Open energy modelling framework

„*Open energy modelling framework*“ - *Oemof* [24] bezeichnet eine Reihe von Open Source Python Bibliotheken im Bereich der Modellierung von Energiesystemen. Im Entwicklungsprozess hat sich *Oemof* von einer einzigen Bibliothek zu mehreren Paketen mit speizifschen Funktionen entwickelt. [24]. In dieser Arbeit bezieht sich die Bezeichnung *Oemof* auf das *Oemof* Paket *Solph* [25], welches ein Modellgenerator für lineare sowie gemischt ganzzahlig lineare Optimierungsaufgaben von Energiesystemen ist. Die Kopplung verschiedener Sektoren sowie die Beschränkung von Emissionen sind mit *Oemof* ebenfalls möglich [18]. In dieser Arbeit wird die Version 0.4.4 von Oemof.Solph verwendet.

2.2.3. Python for Power System Analysis

„*Python for Power System Analysis*“ - *PyPSA* [26] ist ein Open Source Framework zum Simulieren und Optimieren von Energiesystemen. *PyPSA* ist entworfen um große Energiesysteme und große Zeithorizonte gut darzustellen. Der Stromsektor, welcher detailliert analysiert werden kann, steht bei *PyPSA* besonders im Fokus. So können Wechsel und Gleichstromnetze im Modell abgebildet und optimiert werden. Die Modellierung weiterer Sektoren so wie deren Kopplung und die Limitierung von Emissionen sind ebenfalls möglich [19]. Die *PyPSA* Version 0.19.3 findet in dieser Arbeit Anwendung.

2.3. Auswahl einer geeigneten Software zur Implementierung in *Tessif*

Bevor ein in *Tessif* zu implementierendes Softwaretool ausgewählt wird, werden zunächst einige Arbeiten betrachtet, welche sich mit einem Vergleich von Energiesystem Modellierungstools befassen. Aus dieser Betrachtung werden Tools, welche sich dazu eignen in *Tessif* integriert zu werden, ausgewählt und genauer betrachtet, ehe sie in einem Ranking entsprechend ihrer Eignung zur Integration sortiert werden und hieraus eine finale Wahl abgeleitet wird. Aufgrund dessen, dass alle in

Tessif integrierten Tools Optimierungstools sind, werden im Weiteren ausschließlich Tools zur Optimierung betrachtet.

Connolly et al. [3] vergleichen 37 Tools, welche sich dazu eignen die Integration von erneuerbaren Energien zu analysieren. Neben einem tabellarischen Vergleich von Verfügbarkeit, Downloads bzw. Verkäufen, Softwaretypen, der geografischen und sektoralen Abdeckung und der Integration von erneuerbaren Energien, ist zu jedem der aufgeführten Tools ein Abschnitt mit Hintergrundinformationen, der Funktionalität und Arbeiten mit diesem Tool verfasst [3].

Allegrini et al. [27] vergleichen 24 Tools zum Modellieren von Energiesystemen in Stadtgebieten. Beide Arbeiten kommen zu dem bereits in Abschnitt 2.1 aufgeführten Schluss, dass das passende Tool stets entsprechend der Problemstellung zu wählen ist, da kein Tool jeder Problemstellung gerecht wird [3, 27].

Beim Vergleich von 24 Modellierungstools zeigen Lopion et al. [2], dass seit dem Jahr 2000 vermehrt Open Source bzw. Open Access Software zur Modellierung von Energiesystemen entwickelt wird. Zudem ist seit dem Jahr 2010 eine starke Zunahme an Software, welche in Python programmiert ist, zu erkennen [2]. Aufgrund dessen, dass *Tessif* ein in Python geschriebenes Framework ist, bietet es sich an ein Tool zu wählen, welche ebenfalls in Python geschrieben ist. Dies ist zwar nicht zwingend notwendig, vereinfacht jedoch zum einen den Prozess der Implementierung, da keine Schnittstelle zwischen verschiedenen Programmiersprachen verfasst werden muss und des Weiteren bleibt der Aufwand für jene, die sich mit *Tessif* auseinander setzen auf das Arbeiten mit Python beschränkt. Ein ansteigender Trend zu mehr Python basierten Softwares bekräftigt die Wahl eines Python basierten Tools zur Integration in *Tessif*.

Ringkjøb et al. [5] haben ebenfalls aufgrund der steigenden Anzahl an Modellen und Modellierungstools einen Vergleich ausgearbeitet, um einen aktualisierten Überblick über die verfügbaren Modelle und Tools zu geben und somit dem Leser bei der Wahl eines geeigneten Modells oder Tools zu helfen. Eine explizite Unterscheidung zwischen Modellen und Tools ist hierbei nicht vorgenommen und stattdessen beides als Modell bezeichnet. Modelle, welche nach 2012 in keiner Veröffentlichung auftauchen, sind prinzipiell ausgeschlossen. Von den betrachteten 75 Modellen sind darüber hinaus die Information von 71 durch die Entwickler oder Beteiligte am Modell bestätigt [5].

Chang et al. [11] vergleichen verschiedene Veröffentlichungen, welche sich damit befassen Energiesystem Modelle und Frameworks zu vergleichen. In diesem Vergleich sticht die Arbeit von Ringkjøb et al. [5] besonders hervor. Der Inhalt verschiedener Arbeiten wird hierbei in sieben Kategorien unterteilt. Ringkjøb et al. [5] bieten demnach, mit einer Abdeckung von fünf der sieben Kategorien, den tiefgreifensten Vergleich der 42 betrachteten Veröffentlichungen [11]. Darüber hinaus ist es eine der wenigen Arbeiten, die neben einer Literaturrecherche auf Informationen aus dem direkten Austausch mit den Entwicklern der *Tools* entstanden ist [11]. Aus diesen Gründen wird angenommen, dass sich der Vergleich von Ringkjøb et al. [5] als Basis zur Auswahl eines *Tools* gut eignet.

In Anlehnung an die Ergebnisse von Ringkjøb et al. [5] ist letztlich Tabelle 2.1 entstanden, welche die bereits in *Tessif* integrierten Tools mit einer Auswahl von potenziellen Kandidaten zur Implementierung vergleicht. Die Auswahl dieser Tools ist auf die ausschließliche Betrachtung von FOSS Tools, welche mit Python auszuführen sind, zurückzuführen. *OSeMOSYS* ist ebenso in die Auswahl aufgenommen, da die Recherche aus Abschnitt 2.1 zeigte, dass *OSeMOSYS*, neben der von Ringkjøb et al. [5] gelisteten GNU MathProg Version, auch eine Python basierte Version bietet [14]. Die Python basierten FOSS Tools *PowerGAMA* und *Temoa* hingegen werden nicht betrachtet. *Temoa* ist zur Szenarioanalyse gedacht [5] und eignet sich somit nicht für eine Implementierung in *Tessif* und aller Voraussicht nach ebenso wenig für einen Vergleich mit Optimierungstools. *PowerGAMA* berücksichtigt keine Emissionen [5]. Da die Frage nach umweltschonender und dennoch zuverlässiger Energieerzeugung ein wesentlicher Grund für die steigende Relevanz von Energiesystem Modellierungstools darstellt [2, 5, 13], erscheint ein Tool, welches keine emissionsbehafteten Zusammenhänge darstellen kann, keine sinnvolle Option zu sein. Somit bleiben schließlich sechs Kandidaten, welche in Tabelle 2.1 aufgelistet sind, übrig.

Ringkjøb et al. [5] führt auf, dass sich die Formulierung von MILP Problemen für *Calliope* in der Entwicklung befindet. Zusätzlich wird angegeben, dass *PyPSA* lediglich LP Probleme formulieren kann [5]. Da diese beiden Daten nicht mehr aktuell sind, wurde die Problemformulierung für alle Tools, wenn möglich, in der jeweiligen Software Dokumentation mithilfe der dort vorhandenen Suchfunktion recherchiert.

Neben den Daten, welche von Ringkjøb et al. [5] übernommen sind, ist zusätzlich eine Recherche über die Aktivität der Entwicklung mittels des netbasierten Dienstes zur Versionsverwaltung für Software-Entwicklungsprojekte GitHub[3] sowie die Software Dokumentation für die potenziellen Kandidaten durchgeführt worden. Eine ausführliche Software Dokumentation ist für die korrekte Integration in *Tessif* essenziell, da diese zusammen mit der *Tessif* Dokumentation [21] die Basis der zu erstellenden Funktionen bildet.

[3] `https://github.com/`

Tabelle 2.1.: Vergleich ausgewählter FOSS Tools.
Abkürzungen: BD - Benutzerdefiniert, CMA-ES - Covariance Matrix Adaption Evolution Strategy, LP - Linear programming, MILP - Mixed integer linear programming, NLP - Non linear programming, S - Simulation

Vergleichskriterien	Bereits integriert			Kandidaten zur Implementierung					
	FINE	*Oemof*	*PyPSA*	*Calliope*	*ficus*	*NEMO*	*OSeMOSYS*	*SWITCH*	*Urbs*
Programmierumgebung	Python[a]	Python[b]	Python[b]	Python[b]	Python[b]	Python[b]	GAMS, GNU-MathProg und Python[a]	Python[b]	Python[b]
GitHub	[32]	[33]	[34]	[35]	[36]	[37]	[38]	[39]	[40]
- Erster Commit[c]	02.07.2018	18.12.2014	12.10.2015	12.11.2013	07.09.2015	06.05.2013	03.10.2016	08.04.2015	24.01.2014
- Letzter Commit[c]	15.02.2022	03.06.2021	25.04.2022	10.09.2021	31.01.2018	23.01.2022	21.05.2021	09.03.2021	07.02.2022
- Anzahl Commits[c]	1528	5512	1299	1097	160	1308	335	579	823
Dokumentation	Ja[a]	Ja[a]	Ja[a]	Ja[a]	Ja[a]	(Ja)[d]	Ja[a]	(Ja)[e]	Ja[a]
Geografische-Abdeckung	BD[a]	BD[b]	Lokal bis Kontinental[b]	BD[b]	Lokal bis National[b]	National[a]	Gemeinde bis Kontinental[b]	Regional bis National[b]	Lokal bis National[b]
Sektorale Abdeckung	Alle[a]	Alle[b]	Alle[b]	Alle[b]	Alle[b]	Strom[b]	Strom[b]	Strom[b]	Alle[b]
Zeithorizont	BD[a]	BD[b]	1 Jahr[bf]	BD[b]	1 Jahr[b]	1 Jahr[b]	BD[b]	BD[b]	BD[b]
Zeitschritte	BD[a]	BD[b]	1 Stunde[b]	BD[b]	15 min[b]	1 Stunde[b]	BD[b]	1 Stunde[b]	BD[b]
Problemformulierung	LP, MILP, NLP[a]	LP, MILP[a]	LP, MILP, NLP[a]	LP, MILP[a]	LP, MILP[a]	CMA-ES, S[b]	LP[a]	MILP[b]	LP[a]
Emissionen	BD[a]	BD[b]	CO_2[b]	BD[b]	BD[b]	CO_2[b]	BD[b]	CO2[b]	BD[b]

[a]Spezifische Dokumentation: FINE: [22], Oemof: [25], PyPSA: [26], Calliope: [28], ficus: [29], OSeMOSYS: [30], Urbs: [31],
[b]Ringkjøb et al. [5]
[c]Master Branch Stand: 26.04.2022
[d]Aufgeteilt in API Dokumentation und einen User-Guide. `https://nemo.ozlabs.org/`
[e]Muss selbst erstellt werden [38]
[f]Die Modelle in Kapitel 4 zeigen, dass der Zeithorizont frei wählbar ist. An dieser Stelle ist voraussichtlich eine Auslegung von *PyPSA* auf einen Horizont von einem Jahr gemeint.

Anhand von Tabelle 2.1 wird im Weiteren ein Ranking aufgestellt, welches die einzelnen Tools angesichts der aufgeführten Kriterien als „besser“ bzw. „schlechter“ geeignet für eine Integration in *Tessif* sortiert.

Da die bisher in *Tessif* implementierten Tools mehrere Sektoren abdecken und der Sektorkopplung eine große Rolle zum Aufhalten des Klimawandel zugesprochen wird [1, 41], werden *NEMO*, *OSeMOSYS* und *SWITCH*, welche alle ausschließlich den Stromsektor darstellen, als die am wenigsten geeigneten Tools angesehen. *OSeMOSYS* liegt aufgrund der tiefgreifenderen und einfacher aufzurufenden Software Dokumentation, dennoch vor *NEMO* und *SWITCH*. Die Dokumentation von *SWITCH* ist nach dem Download der Software über das GitHub [39] lokal zu erstellen und somit nicht wie die restlichen unter einem Link online aufzurufen. *NEMO* bietet online eine API Dokumentation sowie einen User Guide [42]. Beides jedoch in einem deutlich geringeren Umfang als die Dokumentationen der weiteren Tools. Vor allem hinsichtlich der von den anderen Tools abweichenden Problemformulierung mit CMA-ES[4], wäre eine ausführliche Software Dokumentation voraussichtlich für die Analyse von Abweichungen in den Optimierungsergebnissen von Bedeutung. Oberle et al. [13] beispielsweise kritisieren die Dokumentation des Modells *DESSTinEE*, dessen Mängel keine genaue Erklärung zum Zustandekommen der Ergebnisse erlaubt.

Aufgrund der geringen Aktivität stellt *ficus*, welches im GitHub [36] zuletzt am 10.09.2018 aktualisiert wurde (Stand 26.04.2022), ebenfalls keine ideale Option für die Implementierung in *Tessif* dar. Sowohl *Calliope* als auch *Urbs* sind mit ihren letzten Updates deutlich aktueller.

Die beiden verbleibenden Tools *Calliope* und *Urbs* sind beide nahezu gleich alt und beide noch immer aktiv[5]. Darüber hinaus stimmen viele der aufgeführten Kriterien überein. So können verschiedene Sektoren abgebildet werden und Zeithorizont, Zeitschritte sowie die Emissionen frei gewählt werden. *Calliope* jedoch beruht neben der linearen Programmierung (LP) ebenso auf der gemischt-ganzzahlig linearen Programmierung (MILP), weshalb der Umfang *Calliope*'s als größer eingestuft und *Calliope* daher als am geeignetsten angesehen wird. Das Ranking ist zur Übersicht in Tabelle 2.2 dargestellt.

Aus einer tieferen Einarbeitung in *Calliope*, anhand der klar strukturierten und ausführlichen Software Dokumentation [28], ergibt sich vermehrt die Wahrnehmung, dass sich *Calliope* ideal für eine Integration in *Tessif* eignet. Daher wird auf eine Einarbeitung in *Urbs* verzichtet und im Weiteren *Calliope* genauer vorgestellt, ehe die Integration dessen präsentiert wird.

[4] `https://cma-es.github.io/`

[5] Bei *Calliope* ist die aktuellste Version 0.6.8 vom 07.02.2022 (noch) nicht auf dem Master Branch, sondern nur im Branch der Version 0.6 [35]. Somit sind die wenigen Monate unterschied des letzten Commits des Master Branches vernachlässigbar.

Tabelle 2.2.: Ranking der Tools zur Eignung einer Integration in *Tessif*, wobei 1 „am besten“ und 6 „am schlechtesten“ geeignet bedeutet.

Platzierung	Software
1	*Calliope*
2	*Urbs*
3	*ficus*
4	*OSeMOSYS*
5	*SWITCH*
6	*NEMO*

2.4. *Calliope*

Calliope ist ein Python basiertes Open Source Framework zum Erstellen von Modellen von Energiesystemen. Die Größe der Modelle sowie die zeitliche Betrachtung sind beliebig. Das Modell wird in *Calliope* klar von dem Framework getrennt. So wird es nicht ebenso in Python erstellt, sondern von *Calliope* aus einer Reihe von `YAML`[6] (YAML Ain't Markup Language) und `CSV`[7] (Comma Separated Values) Dateien gebildet. Das Modell wird daraufhin mithilfe von *Pyomo* als Backend optimiert. Sowohl lineare als auch gemischt-ganzzahlig lineare Probleme werden von *Calliope* unterstützt [43].

Wie ein Modell in *Calliope* erstellt wird ist in Abschnitt 2.4.1 und 2.4.2 genauer beschrieben. Abschnitt 2.4.3 umfasst eine Reihe von verschiedenen Verfahren, mit welchen ein *Calliope*-Modell optimiert werden kann. Letztlich werden in Abschnitt 2.4.4 Arbeiten vorgestellt, in welchen *Calliope* Anwendung fand.

2.4.1. Modellformulierung

Calliope-Modelle werden nicht als Python Code erstellt, sondern als `YAML`-Files in Kombination mit `CSV`-Files für die Zeitreihen. Die einzelnen Dateien werden in einem Hauptordner mit zwei Unterordnern erstellt. Einer mit den Modell Konfigurationen und einer mit den Zeitreihen (vgl. Abb. 2.3). Das übergeordnete File `model.yaml`, welches grundlegende Informationen wie den zu verwendenden Solver oder dem globalen Emissionslimit enthält, importiert die Daten aus dem Ordner `model_config` und gibt ebenfalls den Ordner der Zeitreihen für Energieerzeuger und -verbraucher an [28].

In dem Ordner `model_config` befinden sich die Dateien `locations.yaml` und `techs.yaml`. Die Techs-Datei beinhaltet die einzelnen Technologien (*Techs*) und deren Parametrisierung. In der Locations-Datei wird auf diese Technologien zugegriffen, indem diese verschiedenen oder gleichen Standorten (*Locations*) zugeordnet

[6] `https://docs.fileformat.com/programming/yaml/`
[7] `https://docs.fileformat.com/spreadsheet/csv/`

werden. Darüber hinaus werden mit Verbindungen (*Links*) verschiedene Standorte miteinander verknüpft. Dieser Aufbau wird gegenüber einer einzelnen `YAML`-Datei mit sämtlichen Modelldaten bevorzugt, da er eine bessere Lesbarkeit gewährleistet. So können beispielsweise alle Standorte, Technologien und Verbindungen in der `locations.yaml` Datei eingesehen werden, ohne dass die einzelnen Technologieparameter die Lesbarkeit beeinflussen [28].

```
+ urban_scale
    + model_config
        - locations.yaml
        - techs.yaml
    + timeseries_data
        - demand_heat.csv
        - demand_power.csv
        - export_power.csv
        - pv_resource.csv
    - model.yaml
    - scenarios.yaml
```

Abbildung 2.3.: Darstellung der Ordnerstruktur des *Calliope*-Beispielmodells „Urban Scale“ [35]. Ein „+“ steht für einen Ordner, während ein „-“ eine Datei darstellt. (in Anlehnung an [28])

Diese Struktur ist jene, welche in der *Calliope* Dokumentation empfohlen und in den in *Calliope* implementierten Beispielmodellen verwendet wird [28]. Bei der genaueren Betrachtung einiger Arbeiten mit *Calliope* fällt auf, dass es sehr wohl auch weitere Möglichkeiten gibt, eine übersichtliche Dateistruktur abweichend von der Empfehlung zu verwenden. So sind bei der Modellierung eines Bezirks der Stadt Bangalore[8] (Indien) die Standorte von den Verbindungen getrennt in einer `locations.yaml` und einer `links.yaml` Datei aufgeführt (vgl. Anhang A.1). Bei der Modellierung von Großbritannien[9] wird auf den Ordner `model_config` verzichtet. Stattdessen befinden sich die Standorte und Technologien in separierten Ordnern. Darüber hinaus sind die einzelnen Standorte in verschiedenen Dateien und die Technologien befinden sich ebenfalls je nach Typ und Energieträger in verschiedenen Dateien. So sind beispielsweise Speicher in einer eigenen Datei ebenso wie fossile Erzeuger getrennt von nuklearen Erzeugern (vgl. Anhang A.2). Solch ein Aufbau verfügt aller Voraussicht nach bei besonders großen Modellen für eine bessere Lesbarkeit.

In der `scenarios.yaml` Datei werden verschiedene zu optimierende Szenarien festgelegt. So kann der zu betrachtende Zeitraum angepasst werden oder jeglicher Parameter, wie beispielsweise das einzuhaltende Emissionsziel oder spezifische Kosten, speziell für jedes Szenario verändert werden. Dies ermögilcht es verschiedene

[8] `https://github.com/brynpickering/bangalore-calliope`
[9] `https://github.com/calliope-project/uk-calliope`

Szenarien vorab zu definieren und schnell drauf zuzugreifen. Hierbei werden unter dem Schlüsselwort *Overrides* jene Anpassungen des Modells definiert und unter dem Schlüsselwort *Scenario* auf beliebig viele Anpassungen verwiesen, welche dieses Szenario definieren [28].

2.4.2. Technologien

Vergleichbar mit den verschiedenen *Tessif* Komponenten (vgl. Abschnitt 2.2) werden in *Calliope* verschiedene Technologien (*techs*) parametrisiert. Grundlegende Eigenschaften erbt eine Technologie von einer Technologie-Gruppe (*tech_groups*), welche über die Standardgruppen hinaus erweitert werden können, um das Erstellen von Technologien mit gleichen Eigenschaften zu vereinfachen [28].

Die Standard Technologie-Gruppen lauten:

- *Supply*: Energieerzeuger, welcher dem System Energie zuführt.
- *Supply_plus*: Energieerzeuger, welcher im Unterschied zum *Supply* zusätzlich einen Speicher beinhaltet.
- *Demand*: Energieverbraucher, welcher aus dem System Energie abführt.
- *Storage*: Energiespeicher, welcher Energie in einem Zeitschritt aufnimmt, um diese in einem anderen Zeitschritt abzugeben.
- *Conversion*: Energiewandler, welcher Energie von einem Energieträger in einen anderen wandelt.
- *Conversion_plus*: Energiewandler, welcher einen oder mehreren Energieträgern in einen oder mehrere Energieträger wandelt.
- *Transmission*: Energietransport, welcher Energie von einem Standort zu einem anderen überträgt.

Die Technologien besitzen eine Vielzahl von gleichen sowie verschiedenen Möglichkeiten der Parametrisierung abhängig von ihrer zugehörigen Technologie-Gruppe. Genauer wird auf diese Parameter in Abschnitt 3.1 eingegangen, wo die Übersetzung der *Tessif* Komponenten in *Calliope* Technologien erläutert wird.

Die Technologien innerhalb eines Standorts stehen in einem verlustfreien Energieaustausch. Unter *Links* werden die Verbindungen verschiedener Standorte durch eine *Transmission* Technologie aufgeführt. Eine Technologie kann zudem mehrmals innerhalb eines Modells an verschiedenen Standorten verwendet werden. Hierbei können beliebige Parameter der Technologie innerhalb eines jeden Standortes bei Bedarf angepasst werden [28].

2.4.3. Optimierungsverfahren

Calliope beinhaltet eine Reihe von Funktionen, welche nicht in *Tessif* vorhanden sind und dementsprechend in der Integration und dem Vergleich mit den anderen integrierten Tools innerhalb *Tessif*'s wegfallen. Grundsätzlich lassen diese sich zwar zu *Tessif* hinzufügen, jedoch bringen Optionen, welche von nur einem der integrierten Tools verwendet werden könnnen, *Tessif* als Plattform zum Vergleich der Optimierungsergebnisse verschiedener Tools keinen Mehrwert.

In *Calliope* existieren drei verschiedene Modi, in welchen die Optimierung ablaufen kann [28].

- Plan,
- Operate und
- Spores

Der „plan“ Modus bezeichnet den Fall, dass keine vorab installierten Leistungen und Kapazitäten vorliegen und diese sich durch die Optimierung, innerhalb des vorgegebenen Minimum und Maximum, ergeben. Somit werden die mit Kosten behafteten Leistungen und Kapazitäten sowie der Betrieb der Technologien optimiert. Dies ist die Standardeinstellung in *Calliope* [28], obwohl die meisten Optimierungstools ein aktuelles System als Ausgangspunkt nehmen und von dort aus den optimalen Weg für die Zukunft planen [9].

Im „operate“ Modus sind die Leistungen und Kapazitäten allesamt vorab festgelegt und ausschließlich der Betrieb wird optimiert. Diese Optimierung geschieht zudem mit einem rollierenden Horizont Algorithmus, während die anderen Modi eine perfekte Vorhersage nutzen. Hierbei wird ein Zeithorizont festgelegt, welcher bestimmt wie weit das System optimiert wird, sowie ein Zeitfenster welches tatsächlich genutzt wird. So kann beispielsweise ein Horizont von 48 Stunden optimiert werden. Das Ergebnis der ersten 24 Stunden wird übernommen und die nächsten 24 Stunden verworfen. Nach 24 Stunden beginnt der nächste Zeitabschnitt welcher optimiert wird (vgl. Abb. 2.4).

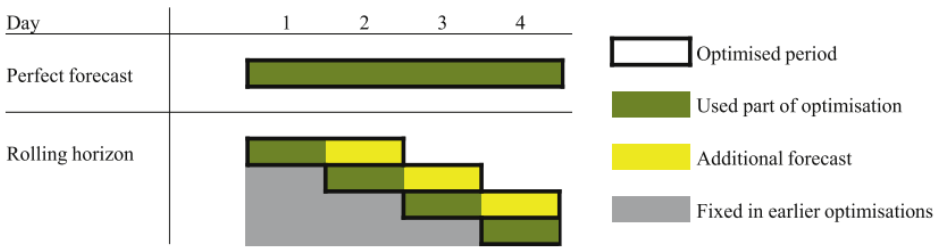

Abbildung 2.4.: Vergleich einer perfekten Vorhersage mit einem rollierenden Horizont (in Anlehnung an [16])

Der „Spores“ (Spatially-explicit Practically Optimal REsultS) Modus ermöglicht es eine beliebige Anzahl an Optimierungsergebnissen zu generieren, welche in einem festgelegten Bereich von dem Optimum abweichen. Mithilfe von Gewichtungsfaktoren werden einzelne Technologien, abhängig von ihrer bisherigen Nutzung, bevorzugt und andere nachrangig betrachtet. Die Bezeichnung dieses Gewichtungsfaktors, welcher wie die Kosten und Emissionen als eine Kostenklasse definiert ist, kann frei gewählt werden, jedoch wird „spores_score“ empfohlen [28].

Sind beispielsweise fünf zusätzliche Ergebnisse erwünscht, welche bis zu 10 % vom Kostenoptimum abweichen können, so wird zunächst in einer ersten Optimierung im „plan“ Modus das Kostenoptimum ermittelt. In den darauffolgenden Optimierungen wird der „spores_score“ anstelle der Kosten minimiert. Dieser bleibt stets für jede zuvor nicht genutzte Technologie unverändert und steigt für jede genutzte Technologie automatisch an. Als zusätzliche Bedingung ist ein Kostenmaximum von 10 % über dem Optimum einzuhalten. Die Optimierung wird daraufhin automatisch mehrfach mit angepassten „spores_scores“ wiederholt.

Somit entstehen Ergebnisse, welche im vorgegeben Bereich vom Optimum abweichen und in der Verwendung der Technologien möglichst stark variieren [28]. Durch die Möglichkeit mehrere nahezu optimale Ergebnisse zu betrachten, ermöglicht *Calliope* den Entscheidungsträgern eine gesellschaftlich und politisch akzeptable Option zu identifizieren [17].

Calliope beinhaltet darüber hinaus mit der Option des Erstellens verschiedener Szenarien eine Funktion, welche *Tessif* bereits in ähnlicher Form ermöglicht und somit keiner Integration in *Tessif* bedarf. So können in *Tessif* mit einer Funktion jegliche Parameter des Modells verändert werden [21]. Mit dieser Funktion kann somit ein Script erstellt werden, in welchem verschiedene Anpassungen durchgeführt werden, das Modell optimiert wird und die Ergebnisse abgespeichert werden, ehe die nächsten vorab definierten Anpassungen vorgenommen werden und der Vorgang wiederholt wird. Die bereits in *Tessif* vorhandene Funktion hat zudem den Vorteil, dass diese Anpassung auf alle integrierten Tools angewendet werden kann.

Solche verschiedenen Szenarien ermöglichen es den Einfluss einzelner Parameterunsicherheiten, wie zukünftiger Rohstoffpreise oder Innovationen in Technologien, zu identifizieren [44]. Optimierungsmodelle haben in der Regel Schwierigkeiten Technologien realitätsnah zu nutzen, welche noch nicht auf dem Markt etabliert sind und deren Entwicklungstempo schwer abschätzbar ist [9].

Zuletzt bietet *Calliope* Optionen zur Aggregation von Zeitreihen. Diese ermöglichen es die zeitliche Betrachtung für den gesamten zu optimierenden Zeitraum oder altnerativ für Teilabschnitte vereinfacht darzustellen. Hierbei stehen beispielsweise die Möglichkeit einer Zusammenfassung von beliebig vielen Zeitschritten zu einem Zeitschritt, sowie die Clustermethode „k-means“, welche beliebig viele typische Perioden identifiziert und Mittelwerte jener bildet [45], zur Verfügung [28]. Diese Optionen erlauben es eine, auf Kosten von schwindender Exaktheit, vereinfachte und schnellere Optimierung des Modells durchzuführen.

2.4.4. Anwendungsbeispiele

In einer Vielzahl von Arbeiten, welche sich mit der Modellierung und Analyse von Energiesystemen befassen, diente *Calliope* als Grundlage. Eine Auswahl dieser Arbeiten wird im Folgenden vorgestellt, sodass ein Eindruck des Nutzens der Möglichkeiten von *Calliope* gewonnen werden kann. Dieser dient einer besseren Abschätzung des Mehrwertes, welchen eine Integration von *Calliope* für *Tessif* mitbringt.

Calliope wurde beispielsweise von Díaz Redondo et al. [46] zur Betrachtung der Realisierbarkeit von Szenarien der zukünftigen Stromversorgung der Schweiz genutzt. Hierbei liegt der Fokus auf dem Atomausstieg und hohen Anteilen an erneuerbaren Energien an der Stromversorgung. Zwei Szenarien werden betrachtet und für die Jahre 2035 und 2050 optimiert. Eine Optimierung des Jahres 2014 dient hierbei als Referenzwert. Ein Szenario beinhaltet keine Stromimporte, sodass folglich die gesamte Stromerzeugung innerhalb der Schweiz erfolgt. Das zweite Szenario behält die Importe im Jahr 2050 auf gleichem Level wie 2014, sowie einen erhöhten Importbedarf im Jahr 2035. Trotz der Annahme, dass der Verbrauch in Zukunft sinkt, zeigt die Analyse mit *Calliope*, dass es der Schweiz, unter den getroffenen Annahmen, nicht möglich ist den Verbrauch ausschließlich mit der Energieerzeugung im Inland zu decken [46].

In einer weiteren Arbeit dient ein Modell der Schweizer Stromversorgung der Beantwortung der Frage, ob Erdgas als Lösung zum Ausgleich der schwankenden Einspeisung durch Wind und Sonne in einem überwiegend auf erneuerbaren Energien beruhenden Energiesystem notwendig oder lediglich die kostengünstigste Lösung ist. Das Ergebnis dieser Untersuchung zeigt, dass die Kostenersparnisse durch die Verwendung von Erdgas minimal ausfallen und ebenso wenig eine Notwendigkeit aufgrund von Problemen der Netzstabilität vorliegt. Dies ist vor allem auf die günstige Lage der Schweiz in Bezug auf eine Nutzung von Wasserenergie in Form von Pumpspeichern oder Dämmen zurückzuführen [47].

Zur Analyse von Möglichkeiten zur Nutzung von konzentrierter Sonnenenergie in Kombination mit Wärmespeichern in den Wüsten Chinas und der USA wurde *Calliope* von Labordena et al. [48] verwendet. Ein Problem stellt hierbei jedoch die große Entfernung zwischen Energieerzeuger und Verbraucher dar. Die Untersuchungen zeigen, dass es technisch in beiden Staaten möglich ist mit solchen Anlagen grundlastfähige sowie flexible erneuerbare Energie zu nutzen. Jedoch ist dies nur in China auch wirtschaftlich. In den USA sind die Witterungsbedingungen aufgrund der Monsune nicht ausreichend gut geeignet um als einziger grundlastfähiger Erzeuger zu agieren. Wird die Verfügbarkeit jedoch nicht durchgehend gefordert, so reduzieren sich die Kosten und es wird wirtschaftlich interessant [48].

Zur Untersuchung verschiedener Methoden zur Reduzierung der zeitlichen Auflösungen von erneuerbaren Energieerzeugern wurde ein Modell Großbritanniens in *Calliope* von Pfenninger [49] verwendet. Hierbei zeigt sich, dass keine der

untersuchten Methoden grundlegend besser geeignet ist als die anderen verglichenen Methoden. Demnach gilt es stets entsprechend der gegebenen Fragestellung abzuwägen und zu begründen, welche zu verwenden ist [49].

Eine vollständig auf erneuerbaren Energien beruhende Stromversorgung Europas wurde mithilfe von *Calliope* von Tröndle et al. [50] hinsichtlich der variierenden Kosten einer regionalen, nationalen oder kontinentalen Versorgung untersucht. Hierbei konnte gezeigt werden, dass eine kontinentale Versorgung am kostengünstigsten ausfällt, eine regionale und nationale Versorgung, wenn auch mit höher ausfallenden Kosten, jedoch ebenso möglich ist [50].

Das gleiche Modell der europäischen Stromversorgung wurde darüber hinaus genutzt, um die notwendige Landfläche sowie die Korrelation zu den damit einhergehenden Kosten verschieder Szenarien zu vergleichen. So konnte gezeigt werden, dass beispielsweise eine vermehrte Nutzung von offshore Windenergie, anstelle von onshore Anlagen, die benötigte Landfläche deutlich reduziert, während dies nur geringfügig höhere Kosten zur Folge hat [51].

Im Rahmen einer Untersuchung verschiedener nahezu optimaler Ergebnisse eines Modells der italienischen Stromversorgung von Lombardi et al. [17] wurde der, in Abschnitt 2.4.3 erläuterte, „Spores“ Modus *Calliope*'s entwickelt und verwendet. Mithilfe dieses Modus konnte gezeigt werden, dass Photovoltaik (PV) Anlagen sowie Speichertechnologien zur Dekarbonisierung notwendig sind, da diese in allen Ergebnissen auftauchen. Die Nutzung von beispielsweise Windenergie und Biogasenergie hingegen erlaubt, unter geringer Abweichung vom Kostenoptimum, einiges an Flexibilität [17].

Die Möglichkeit die zu minimierende Variable in *Calliope* einfach zu verändern, wurde in einer Arbeit zur Modellierung des Energiesystems eines Bezirks in Bangalore verwendet. So wurden beispielsweise neben einer Minimierung der Gesamtkosten ebenso die Minimierung der Emissionen, der Gesamtkosten unter Einhaltung eines Emissionslimits sowie der Betriebskosten unter Einhaltung eines Investitionskostenlimits untersucht [44].

In weiteren Arbeiten mit *Calliope* sind verschiedenste Energiesysteme modelliert, optimiert und ausgewertet worden [52–54], welche an dieser Stelle jedoch nicht weiter ausgeführt werden. Die beschriebenen Arbeiten zeigen bereits, dass *Calliope* über Jahre hinweg und vor allem auch global Anwendung findet. Zusammen mit der aktiven Weiterentwicklung von *Calliope* lässt sich vermuten, dass *Calliope* auch in Zukunft zur Modellierung und Analyse von Energiesystemen verwendet wird. Somit ist davon auszugehen, dass eine erfolgreiche Implementierung in *Tessif* eine wertvolle Erweiterung *Tessif*'s darstellt.

3. Integration in *Tessif*

Die Integration von *Calliope* in *Tessif* beinhaltet, wie in Abbildung 2.1 dargestellt, hauptsächlich die folgenden zwei Funktionen:

- `es2es`: Die Übersetzung des Energiesystems als *Tessif*-Modell in ein *Calliope*-Modell
- `es2mapping`: Das Übertragen der Optimierungsergebnisse als Output von *Calliope* in den vereinheitlichten Output von *Tessif*

Abschnitt 3.1 beschreibt die Energiesystem zu Energiesystem Funktion, während Abschnitt 3.2 die Energiesystem zu Mapping Funktion erörtert. Beide Funktionen wurden mithilfe der *Calliope* Dokumentation [28] und der *Tessif* Dokumentation [21] erstellt.

Beide Funktionen sind zudem in Anlehnung an die Funktionen der bereits implementierten Tools *Oemof*, *PyPSA* und *FINE* programmiert, sodass ein weitestgehend einheitlicher Aufbau in der Struktur der einzelnen Scripte in *Tessif* vorliegt. Die Einheitlichkeit *Tessif*'s hat darüber hinaus zur Folge, dass spezifische Optimierungsverfahren wie der „Operate“ und „Spores“ Modus, nicht direkt in der gleichen Form bezüglich ihrer Parametrierung in *Tessif* abgebildet werden[1].

Eine dritte Funktion zum Optimieren des Energiesystems ist ebenfalls zu integrieren. Deren Inhalt beschränkt sich jedoch auf den *Calliope* Befehl der Durchführung einer Optimierung und dient lediglich dazu, dass die Optimierung eines *Calliope*-Modells in *Tessif* mit einem Befehl durchgeführt werden kann, dessen Aufruf analog zu den anderen Tools ist.

Aufgrund dessen, dass *Oemof* mit einer älteren *Pyomo* Version arbeitet, als die aktuellsten *Calliope* Versionen, wird in *Tessif* die *Calliope* Version „v0.6.6-post1“ verwendet. Die `es2es` und `es2mapping` Funktion sind jedoch so konzipiert, dass Modelle der aktuellsten *Calliope* Version (V.0.6.8 [28]) ebenfalls erstellt und ausgewertet werden können. Dies wurde anhand von den in *Tessif* vorhandenen Tests in einer neuen *Tessif* Installation mit der *Pyomo* Version V.6.2 untersucht und sichergestellt[2]. Somit wird eine Aktualisierung der *Calliope* und *Pyomo* Versionen nicht durch die in dieser Arbeit erstellten *Tessif* Funktionen behindert.

[1] Die Eigenschaft des „Operate“ Modus installierte Leistungen ohne die Möglichkeit eines Leistungsausbaus zu berücksichtigen, ist in *Tessif* ebenfalls möglich. Eine Optimierung mit einem rollierenden Horizont sieht *Tessif* jedoch nicht vor.

[2] Die Tests können mit einer aktuellen *Pyomo* und *Calliope* Version nicht automatisiert durchgeführt werden, da dies eine Optimierung mit *Oemof* voraussetzt. Daher wurden diese manuell und ausschließlich mit *Calliope* durchgeführt.

3.1. Energiesystem zu Energiesystem

Der erste Schritt um ein in *Tessif* erstelltes Modell mit *Calliope* zu optimieren, geht über das `es2es`. In diesem Script wird das *Tessif*-Modell in ein *Calliope*-Modell umgewandelt, welches dann bereit ist mit den üblichen *Calliope* Mitteln optimiert zu werden (vgl. Abb. 2.1).

Die *Tessif* Komponenten und ihre Parameter werden zur Übersetzung in *Calliope* zunächst in die, wie in Abschnitt 2.4.1 beschrieben, für *Calliope*-Modelle gängigen Art und Weise in Form von `YAML`-Dateien gespeichert. Die `YAML`-Dateien werden noch in der `es2es` Funktion wieder aufgerufen und das *Calliope*-Modell erstellt. Das Abspeichern der `YAML` Dateien ermöglicht es zudem das Modell, falls erwünscht, unabhängig von *Tessif* zu nutzen und *Calliope* spezifische Anpassungen vorzunehmen. Hierfür wird für jedes übersetzte Modell ein Ordner mit dem Namen des *Tessif*-Modells in dem extra dafür vorgesehenen *Calliope* Ordner im `write` Ordner in *Tessif* angelegt.

In *Calliope* stehen alle Technologien eines Standorts, wie in Abschnitt 2.4.2 aufgeführt, in einem verlustfreien Energieaustausch. Ist jedoch ein Konnektor im Energiesystem so entstehen zwei separate Bereiche eines gleichen Energieträgers, welche nur über den verlustbehafteten Konnektor im Austausch stehen. Des Weiteren könnte ein Energiesystem beispielsweise mehrere Gasquellen vorliegen haben, welche jedoch separate Kraftwerke speisen. In solch einem Fall ist ebenso sicherzustellen dass die Kraftwerke auch in *Calliope* ausschließlich Gas aus der in *Tessif* ihnen zugeordneten Quelle erlangen. Aus diesen Gründen wird das *Tessif*-Modell so übersetzt, dass jede aus den *Tessif* Komponenten erstellte *Calliope* Technologie in einem eigenen Standort vorliegt. Diese Standorte sind mit einer verlustfreien *Transmission* miteinander verknüpft. Lediglich die Konnektoren werden in *Calliope* mithilfe von *Transmissions*, welche mit einem Wirkungsgrad behaftet sind, abgebildet. Hierzu mehr in Abschnitt 3.1.7.

Tabelle 3.1 fasst die in Kapitel 2.2 ausgeführten *Tessif* Komponenten sowie ihre äquivalente in *Calliope* zusammen. Bei der Übersetzung der Energiewandler wird zwischen jenen mit einem Ausgang und welchen mit mehreren Ausgängen unterschieden. *Transformer* mit einfachem output werden als *Conversion* dargestellt und *Transformer* mit mehreren outputs (z.B. Kraftwärmekopplung) als *Conversion_plus*.

In *Calliope* ist es stets notwendig, dass in jedem Modell mindestens eine Technologie mit einer Zeitreihe vorgegeben ist. Darüber hinaus hat der zu betrachtende Zeitraum mindestens eine Länge von zwei Zeitschritten. Sind diese Bedingungen nicht gegeben, so kann das System mit *Calliope* nicht optimiert werden und eine Fehlermeldung wird ausgegeben.

In *Tessif* kann mit dem Parameter „Expansion“ der Wahrheitswert (True oder False) eines erlaubten Ausbaus der Komponente festgelegt werden. Ist dieser Wert „False“, so wird keiner der Ausbauparameter an das *Calliope*-Modell übergeben. Dies ermöglicht es in *Tessif* beispielsweise Leistungsgrenzen und Ausbaukosten zu

definieren, aber dennoch mit nur der Änderung eines Parameters zu wechseln, ob dieser Ausbau nun zu berücksichtigen ist oder nicht. Dadurch lässt sich ein „Plan“ Modus de facto ohne Ausbau von Leistungen und Kapazitäten optimieren, welcher einem „Operate“ Modus ohne rollierenden Horizont gleicht.

Tabelle 3.1.: *Tessif* Komponenten und ihr Äquivalent in *Calliope*

Tessif	*Calliope*
Bus	leere Location
Sink	Demand
Source	Supply
Transformer	Conversion bzw. Conversion_plus
Storage	Storage
Connector	leere Location

Da *Calliope* im für die Implementierung gewählten „Plan“ Modus im Gegensatz zu *Tessif* grundsätzlich keine installierten Leistungen vorsieht, führen diese in *Tessif* vorgegebenen Parameter zu einigen Komplikationen. Um diese vorab installierten Leistungen dennoch, wenn auch über Umwege, in *Calliope* darstellen zu können, werden folgende drei alternative Lösungsansätze untersucht:

1. Das Minimum des Ausbaus wird mit der installierten Leistung gleichgesetzt, wodurch unerwünschte Kosten entstehen. Mit Hilfe von Anpassungen des Ergebnisses im `es2mapping` werden diese wieder eliminiert.
2. Jede Komponente wird zweifach modelliert, wobei eine Technologie keine Investitionskosten aber die installierte Leistung als Maximum hat und die zweite keine installierte Leistung, aber einen kostenaufwendigen Ausbau erlaubt.
3. Der *Calliope* Parameter „purchase“, welcher vorgegebene Kosten unabhängig von der Größe des Ausbaus beaufschlägt, wird verwendet. Diese werden den bis zum Expansionsminimum anfallenden Kosten gleichgesetzt, jedoch mit einem negative Vorzeichen versehen und gleichen somit diese Kosten umgehend aus.

Der erste Ansatz ist jener, welcher in der Integration von *FINE*, welches ebenso keine installierten Leistungen vorsieht, wiederzufinden ist [23]. Allgemein gilt, dass mehr Komponenten den Rechenaufwand erhöhen, weshalb davon auszugehen ist, dass der zweite Ansatz nicht der schnellste ist. Der dritte Ansatz mithilfe des „purchase“ Parameters, welcher ein MILP Parameter ist und somit die Optimierungsdauer erheblich erhöht, sollte ebenso längere Optimierungszeiten gegenüber dem ersten Ansatz aufweisen. Interessant ist die Untersuchung dennoch, da die Größe dieses zeitlichen Mehraufwands unter Umständen dem Vorteil, dass die `YAML`-Files problemlos im nativen *Calliope* genutzt und ohne die Anpassung des `es2mapping` ausgewertet werden können, unterliegt.

Zur Ermittlung, welcher dieser Ansätze am geeignetsten ist, wird das in der *Tessif* Bibliothek vorhandene Beispielmodell „Component Example“ als Expansionsproblem verwendet. Das Modell ist im Abschnitt 4.5 genauer beschrieben. Dieses Beispielmodell ist mit 24 Komponenten ein relativ großes Modell, welches über einen Zeitraum von einem Jahr betrachtet wird und somit durch eine lange Optimierungsdauer klare Tendenzen aufzeigen sollte. Das Energiesystem ist manuell in `YAML` Dateien für *Calliope* umgeschrieben worden und wurde einer Optimierung der drei aufgeführt Variationen unterzogen.

Jede Komponente doppelt zu erstellen führt zu einer Optimierungsdauer von etwa einer Stunde. Der Ansatz mithilfe des „purchase“ Parameter wurde nach 2 Stunden und 30 Minuten abgebrochen, da an dieser Stelle bereits eindeutig zu erkennen ist, dass der vorherige Ansatz merklich schneller zu einem Ergebnis führt. Der erste Ansatz ist mit einer Optimierungsdauer von etwa 15 Minuten deutlich schneller zu einem Ergebnis gekommen und ist deshalb jener, welcher in der *Tessif* Implementierung verfolgt wird. Hierbei ist es essenziell, dass diese `es2mapping` Ergebniskorrekturen in der Dokumentation von *Tessif* [21] festgehalten werden, sodass für jeden Nutzer, welcher ein von *Tessif* in *Calliope* umgewandeltes Modell direkt mit nativen *Calliope* Mitteln auswertet, nachvollziehbar ist, weshalb Unterschiede in den Kosten vorliegen.

Calliope ermittelt außerdem die Investitionskosten stets mithilfe einer Lebensdauer und einem Zinssatz. Sollen diese keinen Einfluss haben, was im Kontext der *Tessif* Integration der Fall ist, wird der Zinssatz auf null gesetzt. Die Lebensdauer entspricht dem betrachteten Zeitraum, indem die Anzahl an Zeitschritten durch 8760 geteilt wird. Folglich liegt bei der Optimierung eines Jahres eine Lebensdauer von eins vor. Ungenauigkeiten aufgrund von Rundungen können an dieser Stelle zu geringen Abweichungen des *Calliope* Ergebnisses im Vergleich zu *Tessif* führen.

Das in *Tessif* vorgegebene Expansionsminimum kann somit nicht an *Calliope* übergeben werden, da dieses durch die zu Beginn installierte Leistung ersetzt wird. Darüber hinaus gibt es in *Calliope* keine Kosten, welche von dem Gradienten des Energieflusses abhängen. Zudem ist lediglich ein Parameter vorgesehen, welcher sowohl die positiven als auch negativen Leistungsgradienten beschreibt. An dieser Stelle ist stets der in *Tessif* als negativ angegebene Wert gewählt. Zuletzt finden sich ebenso wenig die MILP Parameter von *Tessif* in *Calliope* wieder. Auch wenn *Calliope* mit dem „purchase“ und einigen weiteren Parametern MILP Probleme erstellen kann, so gleicht keiner dieser Parameter jenen, welche in *Tessif* vorzufinden sind. Ist in *Tessif* einer der Parameter

- „expansion_limits.min“,
- „flow_gradients.positive“,
- „gradient_costs“ oder
- jegliche unter *Tessif* als „MILP“ aufgeführte Parameter

definiert, so wird eine Warnung bei der Übersetzung in ein *Calliope*-Modell ausgegeben, welche den Nutzer darüber informiert, dass dieser Parameter im Weiteren ignoriert wird. Warnungen in Python beeinflussen das Script nicht weiter, als dass sie in der Standard-Ausgabe (z.B der aktuell ausgeführten Konsole) ausgegeben werden. Somit kann das Energiesystem umgewandelt, optimiert und, mit dem Wissen welche Parameter nicht übernommen wurden, analysiert werden. In der genaueren Beschreibung der Übersetzung der einzelnen Komponenten in den folgenden Abschnitten werden die hier gelisteten Parameter nicht weiter aufgeführt. Werden in diesen Abschnitten weitere komponentenspezifische Parameter aufgeführt, welche sich nicht in *Calliope* abbilden lassen, so sind diese im `es2es` stets mit einer, den Nutzer aufklärenden, Warnung versehen.

Calliope betrachtet außerdem Emissionen als eine Klasse von Kosten. Dem Solver kann vor der Optimierung mit *Calliope* eine oder auch mehrere Kostenklassen als zu minimierendes Ziel angegeben werden. In *Calliope* und ebenso in der *Tessif* Integration werden als Standard die finanziellen, in *Calliope* als „monetary“ bezeichneten, Kosten minimiert. Eine Minimierung der Emissionen, wie *Calliope* sie ebenso ermöglicht, ist in *Tessif* nicht vorgesehen und dementsprechend nicht implementiert. Jedoch kann eine Obergrenze an Emissionen angegeben werden, welche bei der Optimierung einzuhalten ist und in Abschnitt 3.1.8 erörtert wird.

Für den Fall dass die Transformation des Energiesystems nicht wie erhofft funktioniert, lassen sich die *Calliope* spezifischen Fehlermeldungen von „Error“ auf „Warning“ umstellen, um somit mehr Informationen über mögliche Fehlerquellen zu erhalten.

Zudem wird über die `es2es` Funktion angegeben, ob eine Zeitreihenaggregation vorgenommen werden soll. In der Standardeinstellung wird die Zeitreihe nicht aggregiert. Durch Setzen dieses Parameters auf eine ganze Zahl „X“, werden X Zeitschritte zu einem Zeitschritt zusammengefasst. Diese simple Methode der Aggregation von Zeitreihen wird implementiert da sie, im Gegensatz zur Clustermethode „k-means“, keine Gewichtung einzelner Zeitschritte vornimmt [28] und dementsprechend in Kombination mit dem Datenoutput von *Tessif* harmoniert.

Der Quellcode 3.1 zeigt den Aufruf des „minimum working examples“ der *Tessif* Bibliothek sowie die Transformation in ein *Calliope*-Modell.

Quellcode 3.1: Beispiel der Transformation des „minimum working examples“

```
import tessif.examples.data.tsf.py_hard as tsf_examples
import tessif.transform.es2es.cllp as tessif_to_calliope

tsf_es = tsf_examples.create_mwe()
calliope_es = tessif_to_calliope.transform(
                  tsf_es, warnings=True, aggregate=None)
```

Da die *Calliope* Integration mit der Version „0.6.6-post1“ durchgeführt wird, wird diese Information ebenso in der Datei `model.yaml` gespeichert. Zukünftige sowie

ältere *Calliope* Versionen arbeiten unter Umständen sowohl im Erstellen als auch im Auswerten des Modells verschieden. Sollte beispielsweise *Tessif* in Zukunft eine aktuellere Version unterstützen, jedoch in den Beispielmodellen noch ältere Modelle vorliegen, so kann dies Anhand dieser Datei ausgelesen werden[3].

3.1.1. Unique Identifier (UID)

Um eine Komponente eineindeutig zu beschreiben, verwendet *Tessif* einen sogenannten *Unique Identifier* (UID), welcher aus sieben Parametern[4] besteht und sich für jede Komponente unterscheiden muss. Die Parameter der UID sind in Tabelle 3.2 zusammengefasst. Um was für eine Komponente es sich handelt, kann aus der *Calliope* Technologie, wie in Tabelle 3.1 aufgeführt, geschlossen werden. Der Großteil der UID Parameter ist in *Calliope* jedoch nicht vorhanden. Deshalb werden die verbleibenden Parameter, bis auf die Koordinaten deren Darstellung in *Calliope* in Abschnitt 3.1.2 beschrieben wird, im Namen der Technologie übergeben und jeder Parameter mit einem Punkt „ . “ getrennt. Somit kann im `es2mapping` in *Tessif* jeder dieser Parameter aus dem Namen ausgelesen werden. Der Name der *Calliope* Technologie des Beispiels aus Tabelle 3.2 ergibt sich somit zu:
„Solar Panel.Here.Power.Electricity.Renewable“.

Tabelle 3.2.: Unique Identifier (UID)

UID	Beschreibung	Beispiel[a]
name	Name	Solar Panel
latitude	Breitengrad	42
longitude	Längengrad	42
region	Region	Here
sector	Sektor	Power
carrier	Energieträger	Electricity
component	Komponente	Source
node_type	Typ	Renewable

[a] Aus dem „fully parameterized working example“ der *Tessif* Bibliothek

Voraussetzung für ein korrektes Ergebnis ist, dass keiner dieser Parameter vom vornherein einen Punkt in der Bezeichnung enthält, da das `es2mapping` darauf angewiesen ist die Punkte im Namen zu zählen und die Informationen an entsprechender Stelle zu liegen haben. Für den Sektor beispielsweise liegt die Information hinter dem zweiten Punkt im Namen und der Energieträger hinter dem dritten. Ein Punkt in der Bezeichnung des Sektors selbst, würde zu falschen Ausgaben führen.

[3] Voraussetzung hierfür ist es, dass dieser Parameter bei einem Update der *Calliope* Unterstützung ebenso angepasst wird.
[4] Die beiden Koordinaten werden in einem Parameter aufgeführt.

Der Energieträger ist in dem Namen nicht zwingend notwendig, da jede Technologie in *Calliope* ohnehin einen entsprechenden Parameter aufweist. Bei einer Energiequelle oder einem Verbraucher ist der Energieträger eindeutig, bei einem Energiewandler jedoch ist nicht klar, ob in der UID nun der Eingang oder Ausgang angegeben ist. Daher ist der Energieträger zusätzlich im Namen angegeben und wird in solch einem Fall im `es2mapping` ausgelesen. So ist sichergestellt, dass bei einer Sortierung der Komponenten nach dem Energieträger auch die erwünschte Ausgabe vorliegt.

Für *Tessif* Energienetze und Konnektoren wird die UID nicht im Namen übergeben, da diese beiden Komponenten in *Calliope* als „leere“ Standorte dargestellt werden (siehe entsprechend Abschnitt 3.1.2 sowie 3.1.7), welche keinen Namens Parameter besitzen. Der Energieträger wird über die angebundenen Komponenten bestimmt, da weder das Energienetz noch der Konnektor sich im Energieträger von den an ihnen angebundenen Komponenten unterscheiden. Der Komponenten Typ kann eindeutig bestimmt werden, anhand der Tatsache, dass es sich um leere *Locations* handelt und den weiteren angebundenen Komponenten. Verloren gehen an dieser Stelle somit die Informationen:

- „region“,
- „sector“ und
- „node_type“.

3.1.2. Energienetze

Der Transport von Energie wird in *Tessif* in den *Busses* beschrieben. Die Übersetzung in ein Äquivalent in *Calliope* unterscheidet sich von jener der anderen *Tessif* Komponenten. Der *Bus* wird in *Calliope* durch einen Standort dargestellt, welcher keine Technologie beinhaltet. Dieser „leere“ Standort dient somit lediglich dazu einen Knotenpunkt bei der Verbindung weiterer Standorte miteinander darzustellen und das Auslesen der Energieflüsse des *Busses* im `es2mapping` zu erlauben.

Da jede *Tessif* Komponente, welche kein *Bus* ist, ausschließlich mit *Busses* verbunden ist und umgekehrt auch jeder *Bus* ausschließlich mit Komponenten, welche keine *Busses* sind, reicht es für die Verknüpfungen aller Komponenten die *Busses* auszulesen. Diese Verbindungen werden in *Calliope* in der `locations.yaml` als *Links* aufgeführt. Hierbei beinhalten die *Links* die Informationen welche Standorte miteinander verbunden sind, welche *Transmission* Technologie diese Standorte verbindet und ob es sich um einen einseitigen oder beidseitigen Energieaustausch handelt. Liegt ein Energiefluss nur in eine Richtung vor, so gibt im *Link* der erst genannte Standort an von wo der Energiefluss stammt und der zweit genannte Standort wohin die Energie fließt. Da jede *Transmission* Technologie nur einen Energieträger transportieren kann, wird zu jedem Energieträger eine eigene verlustfreie *Transmission* erstellt.

Die *Busses* beinhalten in *Tessif* keine Informationen zu möglicherweise angebundenen Konnektoren. Somit sind die entsprechenden *Links* und *Transmission* Technologien für diese separat zu erstellen. Dies kommt der Integration der Konnektoren in *Calliope* sehr gelegen und wird im Abschnitt 3.1.7 genauer beschrieben.

In *Tessif* ist ein *Bus* wie folgt formuliert:

Quellcode 3.2: Beispiel eines *Busses* in *Tessif*

```
powerline = components.Bus(
    name='Powerline',
    inputs=('Power Source.electricity',),
    outputs=('Power Demand.electricity',),
    latitude=0,
    longitude=0,
)
```

In *Calliope* übersetzt verteilen sich die Information wie folgt[5]:

Quellcode 3.3: Beispiel einer `locations.yaml` entsprechend des Beispiel Busses 3.2

```
locations:
  Power Source location:
    techs:
      Power Source:

  Power Demand location:
    techs:
      Power Demand:

  Powerline:
    coordinates:
      lat: 0.0
      lon: 0.0
links:
  Power Source location,Powerline:
    techs:
      power transmission:
        constraints:
          one_way: true

  Powerline,Power Demand Location:
    techs:
      power transmission:
        constraints:
          one_way: true
```

[5]Für eine bessere Lesbarkeit ist im Quellcode 3.3 auf die Koordinaten der weiteren Standorte verzichtet. An dieser Stelle sei jedoch angemerkt, dass es für *Calliope* bei den verschiedenen Standorten nicht von Belangen ist, ob mehrere die gleichen Koordinaten besitzen oder sich alle unterscheiden.

3.1.3. Verbraucher

Als nächstes werden die Energieverbraucher betrachtet. Diese werden in *Tessif* als *Sink* und in *Calliope* als *Demand* bezeichnet. Während *Tessif* prinzipiell mehrere Energieträger als Eingänge in einen Verbraucher gestattet, kann *Calliope* dies nicht. Ein Verbraucher hat in *Calliope* einen festen Energieträger und kann keine weiteren aufnehmen. Der Verbrauch, in *Calliope* im Parameter „resource“ dargestellt, wird entweder über einen festen Wert oder eine Zeitreihe vorgegeben, wobei ein negatives Vorzeichen klar signalisiert, dass es sich um einen Verbrauch handelt.

Darüber hinaus kann in *Calliope* festgelegt werden, ob ein Verbrauch erzwungen werden soll oder es im Rahmen des Optimierungsproblems abzuwägen ist, ob es kostengünstiger ist diesen Verbrauch einzuhalten oder nicht. In *Tessif* können dem Verbrauch ein Minimum sowie ein Maximum vorgegeben werden. Da dies in *Calliope* nicht möglich ist, wird je nach Szenario abgewogen. Berücksichtigt werden dabei folgende Faktoren:

- Ist eine Zeitreihe vorgegeben, so wird die als Maximum angegebene Zeitreihe übernommen und der Verbrauch erzwungen.
- Ist keine Zeitreihe gegeben und das „flow_rates“ Maximum nicht unendlich, so wird dieser Wert für jeden Zeitschritt als Verbrauch festgelegt und erzwungen.
- Ist keine Zeitreihe gegeben und das „flow_rates“ Maximum unendlich, so wird dieser Wert für jeden Zeitschritt als Verbrauch festgelegt, jedoch nicht erzwungen.

Die Festlegung dieser Reihenfolge beruht zunächst auf der Präferenz der Zeitreihe. Die Darstellung des Verbrauchs über eine Zeitreihe ist die exakteste Möglichkeit und somit die zu bevorzugende. Dass die maximal Größe des Verbrauchs als maßgebender Parameter gewählt wird, hängt damit zusammen, dass somit der, aus energietechnischer und ökonomischer Sicht, pessimistischste Fall abgedeckt wird. Außerdem stellt die Vorgabe von Zeitreihen einen der häufigsten Anwendungsfälle dar. So werden z.B. oft Lastgänge, basierend auf historischen Daten für den Strom- und Wärmebedarf, verwendet.

Ebenso wird bei dem *Tessif* Parameter „flow_rates“ verfahren, um so den pessimistischen Fall zu betrachten. Wird das „flow_rates“ Maximum zu unendlich groß parametriert, so ergibt es keinen Sinn dies in *Calliope* zu erzwingen. Entsprechend wird der Verbrauch dann als optional dargestellt. Unendlich große Verbraucher und Quellen dienen dazu, sicherzustellen, dass das Optimierungsproblem stets lösbar ist.

Sind summierte Grenzen als „accumulated_ammounts“ im *Tessif*-Modell enthalten, welche von null als untere Grenze und unendlich als obere Grenze abweichen, so werden diese in einem von der Technologie separierten Abschnitt als

„group_constraints“ übernommen. Eine Randbedingung dieser Art kann in *Calliope* einen Parameter für eine Reihe von Technologien, oder aber wie hier, im Fall der Beschreibung einer einzelnen Komponente, nur diese einen Technologie begrenzen. So wird an dieser Stelle entsprechend ein summiertes Minimum oder Maximum für die Energieaufnahme des Verbrauchers vorgegeben. Eine genauere Beschreibung dieser Beschränkungen für eine Reihe von Technologie erfolgt in Abschnitt 3.1.8.

Kosten haben in der Beschreibung einer Technologie in *Calliope* eine eigene Kategorie. Diese ist wiederum in die Unterkategorien „monetary“ und „emissions“ gegliedert. Da es sich bei einem Verbraucher um einen Energiekonsum handelt, werden die Betriebskosten und Emissionen als „om_con“ angegeben. Analog dazu werden bei den produzierenden Technologien im Folgenden diese als „om_prod“ aufgeführt.

Bei den Verbrauchern in *Calliope* können, im Gegensatz zu allen weiteren Technologien, keine Leistungssprünge begrenzt werden. Außerdem können Verbraucher in *Calliope* nicht ausgebaut werden. Daher entfallen jegliche *Tessif* Expansionsparameter an dieser Stelle. Zusammenfassend sind die Parameter des Verbrauchers in *Tessif* sowie in *Calliope* in Tabelle 3.3 gegenübergestellt.

Tabelle 3.3.: Parameter der Energieverbraucher

Tessif	*Calliope*
inputs	carrier
accumulated_amounts	group_constraints carrier_con
flow_rates	resource
flow_costs	monetary.om_con
flow_emissions	emissions.om_con
flow_gradients	-
timeseries	resource
expansion_costs	-
expansion_limits	-

Ein Beispiel eines Verbrauchers in *Tessif* sieht wie folgt aus:

Quellcode 3.4: Beispiel eines *Tessif* Verbrauchers

```
demand=components.Sink(
    name='sink',
    inputs=('electricity',),
    flow_rates={'electricity': nts.MinMax(min=10, max=10)},
    flow_costs={'electricity': 2},
    flow_emissions={'electricity': 1},
    timeseries=None,
)
```

In *Calliope* sieht der Verbraucher aus Quellcode 3.4 wie folgt aus:

Quellcode 3.5: Beschreibung des Verbrauchers aus Quellcode 3.4 in *Calliope*

```
techs:
  sink:
    essentials:
      name: sink.None.None.electricity.None
      parent: demand
      carrier: electricity
    constraints:
      energy_con: true
      resource_unit: energy
      resource: file=sink.csv:sink
      force_resource: true
    costs:
      monetary:
        om_con: 2
      emissions:
        om_con: 1
```

Die Zeitreihe wird in einer `CSV` Datei gespeichert und der Parameter „resource“ beinhaltet in diesem Fall den Namen der `CSV` Datei im Format „`Datei.csv:Spalte`“. In dem dargestellten Beispiel ist dies eine Zeitreihe, welche zu jedem Zeitschritt einen Verbrauch von -10 Einheiten[6] angibt.

3.1.4. Erzeuger

Ein Energieerzeuger wird in *Tessif* durch eine *Source* und in *Calliope* durch einen *Supply* dargestellt. Wie bei den Verbrauchern kann auch ein Erzeuger in *Calliope* lediglich einen Energieträger erzeugen. Während in *Tessif* eine Zeitreihe mit Minimum und Maximum versehen werden kann, steht in *Calliope* an dieser Stelle, analog zu den Verbrauchern, lediglich der Parameter „resource“ zur Verfügung. Ist eine Zeitreihe gegeben, so wird diese in *Calliope* relativ zum maximalen Energiefluss, welcher als „energy_cap“ übergeben wird, dargestellt (vgl. Abb. 3.1). Somit ist gewährleistet, dass, falls die Komponente ausgebaut werden kann, die Zeitreihe nach dem Ausbau der Kapazität korrekt verrechnet wird. Hierzu ist es zudem notwendig neben der Zeitreihe den Parameter „resource_unit“ auf „energy_per_cap“ anstelle von „energy“ zu stellen.

Ist darüber hinaus in *Tessif* der Parameter „accumulated_amounts“ gegeben, so wird dieser, wie bei den Energieverbrauchern, in einer „group_constraints“ realisiert, welche lediglich diese eine Technologie beschränkt. Da Erzeugern in *Tessif*

[6] Megawatt (MW), Tonnen (t) und Euro (€) sind die Standard Einheit in *Tessif*. Da jedoch keine Umrechnung der Einheiten, weder in *Tessif* noch in *Calliope*, vorgenommen wird, kann jede beliebige Einheit gewählt werden, solange diese in dem Modell konsequent verwendet wird.

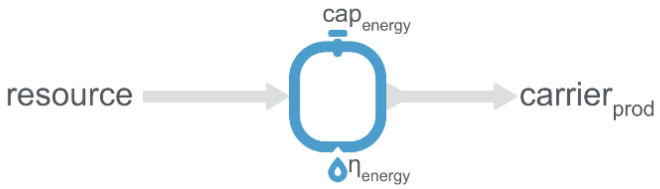

Abbildung 3.1.: Darstellung einer *Calliope Supply* Technologie [28]

kein Wirkungsgrad zugeordnet werden, haben diese in der Übersetzung in *Calliope* stets einen Wirkungsgrad von eins. Folglich kann als „group_constraints“ der aus der Komponente ausströmende Energiefluss äquivalent zu jenem, welcher aus der „resource“ zur Komponente strömt, beschränkt werden. Eine genauere Beschreibung einer solchen Beschränkung erfolgt in Abschnitt 3.1.8.

Der maximale Energiefluss ist wie in Abschnitt 3.1 ausgeführt in *Calliope* als minimale Leistung angegeben. Ist ein minimaler Energiefluss in *Tessif* gegeben, so wird dieser in *Calliope* mithilfe des Parameters „energy_cap_min_use“ realisiert.

Die obere Expansionsgrenze wird in *Calliope* als „energy_cap_max“ angegeben. Die untere Expansionsgrenze wird, aus den in Abschnitt 3.1 aufgeführten Gründen, nicht übergeben. Die Expansionskosten sind in *Calliope* als „energy_cap“ in dem Abschnitt der „monetary“ Kosten aufgeführt.

Bei den Gradienten handelt es sich in *Calliope* um relative Werte, welche somit bei der Expansion mit der Leistung mit skalieren. Sind Leistungssprünge von 100 % möglich, so kann alternativ auch „true“ anstelle von „1“ angegeben werden.

Die restlichen Parameter werden analog zu jenen der Energieverbraucher übersetzt (vgl. Abschnitt 3.1.3). Tabelle 3.4 fasst die Übersetzung der Energieerzeuger von *Tessif* zu *Calliope* zusammen.

Tabelle 3.4.: Parameter der Energieerzeuger

Tessif	*Calliope*
outputs	carrier
accumulated_amounts	group_constraints carrier_prod
flow_rates.max	energy_cap_min
flow_rates.min	energy_cap_min_use
flow_costs	monetary.om_prod
flow_emissions	emissions.om_prod
flow_gradients.negative	energy_ramping
timeseries	resource
-	resource_unit
expansion_costs	monetary.energy_cap
expansion_limits.max	energy_cap_max

Der prinzipielle Aufbau der Definition verschiedener Komponenten in *Tessif* sowie in *Calliope* sind mit Quellcode 3.4 und 3.5 ausreichend aufgeführt. Die restlichen Parameter werden analog zu den bereits vorhandenen angegeben, weshalb auf weitere Beispiele bezüglich der Energieerzeuger sowie der restlichen Komponenten verzichtet wird.

3.1.5. Speicher

Sowohl in *Tessif* als auch in *Calliope* werden Energiespeicher als *Storage* bezeichnet. Der Speicherstand zum Beginn der Optimierung, welcher in *Tessif* entweder vorgegeben oder als Optimierungsvariable dem Solver überlassen werden kann, ist in *Calliope* immer anzugeben. So wird dieser, sollte er in *Tessif* nicht festgelegt sein, zu null angenommen. Ein zu erzielender Speicherstand zum Ende der Optimierung kann in *Calliope* nicht direkt angegeben werden. Jedoch ist es möglich vorzugeben, dass der Speicher zyklisch verwendet werden muss. Dies bedeutet, dass der finale Speicherstand dem initial Speicherstand zu entsprechen hat. Dieser Parameter ist nicht für jeden Speicher separat zu erstellen, sondern als ein Parameter, welcher für alle Speicher des Systems gilt. Für Systeme mit mehr als einem Speicher ist die Konvention gewählt, dass sobald ein Speicher im *Tessif*-Modell nicht zyklisch ist, kein Speicher im *Calliope*-Modell gezwungen ist zyklisch zu agieren.

Standverluste werden in *Calliope* ebenso wie in *Tessif* dargestellt. Jedoch kann neben den Verlusten in *Tessif* auch angegeben werden, dass der Speicher im Stillstand ansteigt. Dies ist in *Calliope* nicht möglich.

Der Wirkungsgrad, welcher in *Tessif* für den Ein- und Austritt separat gehandhabt wird, ist in *Calliope* durch einen Wert in beide Richtungen gegeben (vgl. Abb 3.2). Um im Gesamtwirkungsgrad nicht abzuweichen, werden die beiden *Tessif* Wirkungsgrade miteinander multipliziert und daraus die Wurzel gezogen um den *Calliope* Wirkungsgrad zu bestimmen.

$$\eta_{average} = \sqrt{\eta_{in} \cdot \eta_{out}} \tag{3.1}$$

Da jedoch der Speicherstand, wenn Ein- und Austrittswirkungsgrad verschieden sind, aufgrund des in *Calliope* abweichenden Eingangswirkungsgrads verschieden ist, ist auch der absolute Standverlust in diesem Fall verschieden. Daher weist eine Warnung auf die Handhabung der Wirkungsgrade in *Calliope* hin.

Der maximale und minimale Energiefluss wird in *Calliope*, damit dieser Wert bei einem Ausbau der Speicher Kapazität mit skaliert, als „energy_cap_per_storage_cap“ angegeben. Die Optimierung eines Beispielmodells (detailliert ausgeführt in Abschnitt 4.2) zeigt, dass dieser Wert in *Calliope* nicht, wie zunächst aus Abbildung 3.2 anzunehmen, für den Eintritt wie den Austritt gilt, sondern die maximale Differenz zweier aufeinander folgender Lade und Entlade Zeitschritte beschränkt.

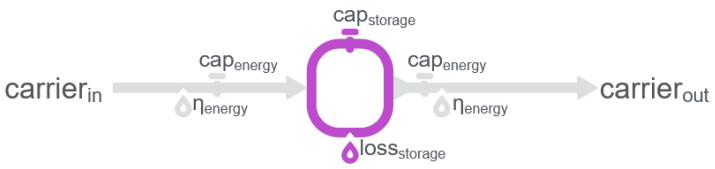

Abbildung 3.2.: Darstellung einer *Calliope Storage* Technologie [28]

Tabelle 3.5 zeigt beispielhaft das Lade- und Entladeprofil des Speichers aus dem in Abschnitt 4.2 untersuchten Modell. Während der Parameter „flow_rates“ in *Tessif* in diesem Fall 12 (oder größer) zu sein hat, da dies die maximale Ladung bzw. Entladung beschreibt, ist in *Calliope* der Parameter „energy_cap“ mit mindestens 17 festzulegen, da die maximale Differenz vom Laden zum darauffolgenden Entladen, oder vice versa, bei 17 liegt. Dies kann, wie Abschnitt 4.2 zeigt, zu Abweichungen von der ursprünglichen Bedeutung dieses *Tessif* Parameters führen, sobald ein Speicher im Zeitschritt nach dem Laden wieder entladen wird oder umgekehrt direkt nach dem Entladen wieder geladen wird.

Tabelle 3.5.: Beispiel des Lade- und Entladeprofils eines Speichers

Zeitschritte	1	2	3	4	5
Laden	9	9	12	0	0
Entladen	0	0	0	-5	-10

Abgesehen von der Zeitreihe werden die weiteren Parameter analog zu den bereits behandelten Komponenten übergeben, wobei die Angabe eines einzigen Energieträgers ausreicht, da ohnehin keine Energieumwandlung stattfindet. Die Zeitreihe des *Tessif* Speichers kann in *Calliope* nicht dargestellt werden. Grundsätzlich ist dies in *Calliope* möglich, jedoch ist dies ausschließlich für den „resource“ Parameter ausführlich dokumentiert [28]. Aufgrund dessen, dass die Speicher diesen Parameter nicht aufweisen, die auf der *Calliope* Homepage aufgeführten Beispielmodelle[7] ausschließlich Zeitreihen im „resource“ und „export“ Parameter verwenden und verschiedene Versuche Zeitreihen in Speichern abzubilden ohne Erfolg blieben, wird auf die Verwendung dieser verzichtet. Tabelle 3.6 fasst die Parameter eines Energiespeichers von *Tessif*, sowie die entsprechenden Pendants in *Calliope* zusammen.

[7] https://www.callio.pe/model-gallery/

Tabelle 3.6.: Parameter der Energiespeicher

Tessif	*Calliope*
input	carrier
output	-
capacity	storage_cap_min
initial_soc	storage_initial
final_soc	-
idle_changes.negative	storage_loss
idle_changes.positive	-
flow_efficiencies	energy_eff
flow_rates.max	energy_cap_per_storage_cap_max
flow_rates.min	energy_cap_min_use
flow_costs	monetary.om_prod
flow_emissions	emissions.om_prod
flow_gradients.negative	energy_ramping
timeseries	-
expansion_costs	monetary.storage_cap
expansion_limits.max	storage_cap_max

3.1.6. Energiewandler

Die Übersetzung der Energiewandler von *Tessif* zu *Calliope* ist in zwei Teile im `es2es` unterteilt. Ein Teil zum Erstellen einer *Conversion*, welcher zunächst einige allgemeine Parameter definiert und daraufhin entweder die restlichen Parameter füllt, oder in den Teil des Erstellens einer *Conversion_plus* springt, sollte es sich um einen Energiewandler handeln, welcher mehr als einen Energieträger erzeugt. In diesem Fall ist einer der Ausgänge als primärer Energieträger zu definieren. Die weiteren Parameter wie Kosten und Emissionen sind nicht separat für jeden Ausgang zu erstellen, sondern beziehen sich stets auf den primären Ausgang. Die Umrechnung der einzelnen Kostenfaktoren ist beispielhaft für die Kosten einer *Conversion_plus* Technologie mit zwei Ausgängen in Gleichung (3.2) aufgeführt. Die Kosten der einzelnen Ausgänge des *Tessif*-Modells sind hierbei als c_{out1} bzw. c_{out2} bezeichnet und die Kosten des *Calliope*-Modells als c_{cllp}. Die Bezeichnung $r_{carrier}$ beschreibt die „carrier_ratios“, welche in *Calliope* das Verhältnis der Erzeugung der verschiedenen Energieträger zum primären Energieträger beschreibt.

$$c_{cllp} = c_{out1} + c_{out2} \cdot r_{carrier} \tag{3.2}$$

mit

$$r_{carrier} = \eta_{out2}/\eta_{out1} \tag{3.3}$$

Der in *Calliope* angegebene Wirkungsgrad bezieht sich ebenso ausschließlich auf den primären Ausgang. In welchem Maße die restlichen Energieträger erzeugt werden, ist mit dem Parameter „carrier_ratios“ eindeutig festgelegt.

Aufgrund dessen, dass Energiewandler mit mehreren Eingängen lediglich für *Oemof* in *Tessif* implementiert sind und eine Implementierung für *Calliope* mit einem großen Aufwand und letztlich wenig Ertrag einhergehen würde, wird auf diese verzichtet.

Die *Conversion_plus* Technologien können mit beliebig vielen Ein- und Ausgängen erstellt werden. Ein Beispiel einer solchen Technologie aus der *Calliope* Dokumentation [28] ist in Abbildung 3.3 gegeben. In diesem Beispiel ist in der Parametrisierung ein festes Verhältnis der Eingänge „$\text{carrier}_{\text{in}}$“ zu „$\text{carrier}_{\text{in2}}$“ vorgegeben. Ob jedoch der Anteil, welchen „$\text{carrier}_{\text{in}}$“ zu liefern hat, in Form von Kohle, Gas oder Öl bereit gestellt wird, kann nicht beeinflusst werden und wird durch den Optimierungsvorgang bestimmt. Analog dazu gilt das selbe für die Ausgänge.

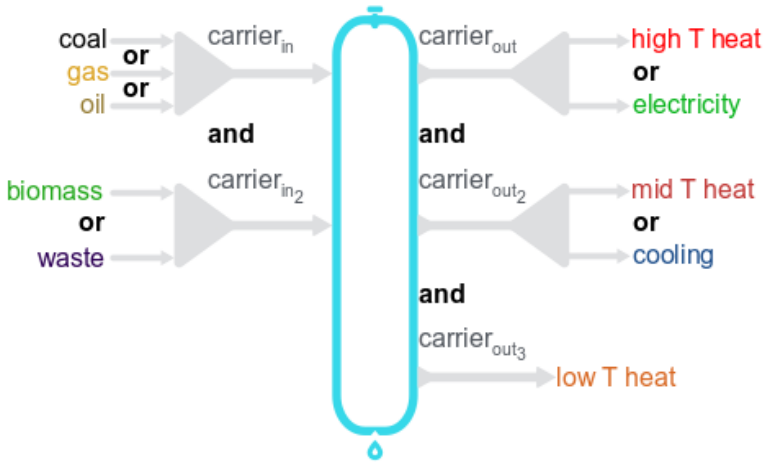

Abbildung 3.3.: Beispiel einer komplexen *Conversion_plus* Technologie [28]

Da in *Tessif* die Verhältnisse zwischen den einzelnen Ein- und Ausgängen stets festgelegt sind, wird in der Implementierung lediglich die Ebene der „$\text{carrier}_{\text{in}}$“ und „$\text{carrier}_{\text{out}}$“, nicht aber die dahinter liegende Ebene, verwendet. Diese ist auf maximal drei Ein- und Ausgänge beschränkt.

Aus der Dokumentation von *Calliope* [28] geht nicht eindeutig hervor, ob sich die vorgegebene Leistung auf den Ein- oder Ausgang bezieht und ebenso wenig ob es sich auf die Summe der Ein- oder Ausgänge oder lediglich auf den Primären bezieht. Aus diesem Grund wurde für diesen Fall ein Beispielmodell erstellt. Dieses Modell beinhaltet einen Strom sowie einen Wärme Verbraucher und Erzeuger. Dazu sind eine Gasquelle und die im Fokus stehende Komponente, welche aus Gas sowohl Strom als auch Wärme erzeugt, enthalten. Die Kosten für die gekoppelte Erzeugung sind deutlich geringer als die separate Erzeugung in den Quellen, sodass der Ausbau der *Conversion_plus* Technologie eindeutig die kostengünstigste Variante darstellt. Aufgrund der Vernachlässigung von Energiewandlern mit mehreren Eingängen ist keine zusätzliche Betrachtung eines entsprechenden Beispiels (oder

potenziell weiterer Beispiele, aufgrund dessen dass eine Komponente mit sowohl mehreren Eingängen als auch mehreren Ausgängen eventuell anders agiert, als eine welche lediglich einseitig mehrere Energieträger berücksichtigt) notwendig.

Dieses Beispielsystem, welches in der Dokumentation von *Tessif* [21] festgehalten ist, zeigt dass die Expansionskosten sowie das Expansionslimit in *Calliope* auf den primären Energieträger des Ausgangs bezogen sind. Somit sind die in *Tessif* in einem Energiewandler festgelegten Werte, welche separat für jeden Ein- und Ausgang beschrieben werden können, für *Calliope* auf diesen primären Energieträger umgerechnet. Beim Expansionslimit wird der Wert übernommen, welcher nach dieser Umrechnung der geringste ist. Als Beispiel zur Illustration sei eine Kohle Anlage mit Kraft-Wärme-Kopplung (KWK) betrachtet. Hat diese in *Tessif* ein Expansionslimit von 100 MW für Kohle, Strom und Wärme, bei Wirkungsgraden von 45 % für die Stromerzeugung und 40 % für die Wärmeerzeugung, so ist ohnehin das Expansionslimit der Kohle der limitierende Faktor, da ein Ausbau der Stromerzeugung wegen des Kohlelimits bei 45 MW liegt. Somit reicht es aus diese Grenze zu betrachten. In *Tessif* kann zudem einzeln festgelegt werden, welcher Ein- und Ausgang ausgebaut werden kann. Da die einzelnen Energieflüsse direkt voneinander abhängig sind, ist ein Ausbau in *Calliope* möglich sobald im *Tessif*-Modell der Ausbau eines Energieflusses eingestellt ist.

Aus den gleichen Gründen wie bei den Speichern, aufgeführt in Abschnitt 3.1.5, werden den Energiewandlern in *Calliope* keine Zeitreihen des *Tessif*-Modells übergeben. Zur Übersicht sind alle Parameter des *Tessif* Energiewandlers, sowie ihre Darstellung in *Calliope*, in Tabelle 3.7 aufgeführt. In Klammern sind hierbei jene Parameter aufgeführt, welche verwendet werden, wenn eine *conversion_plus* Technologie anstelle einer *conversion* erstellt wird.

Tabelle 3.7.: Parameter der Energiewandler. In Klammern sind jene Parameter aufgeführt, welche ausschließlich bei der *conversion_plus* Technologie verwendet werden.

Tessif	*Calliope*
inputs	carrier_in
outputs	carrier_out
	(carrier_out1), (carrier_out2),...
-	(primary_carrier)
conversions	energy_eff
-	(carrier_ratios)
flow_rates.max	energy_cap_min
flow_costs	om_con bzw. prod
flow_emissions	om_con bzw. prod
flow_gradients.negative	energy_ramping
timeseries	-
expansion_costs	energy_cap
expansion_limits.max	energy_cap_max

3.1.7. Konnektoren

Ein Konnektor verbindet zwei Energienetze und berücksichtigt dabei vorgegebene Wirkungsgrade. Im Grunde ist dies nicht mehr als ein Energiewandler, welcher als Ein- und Ausgang den gleichen Energieträger aufweist und in beide Richtungen Energieflüsse gestattet. Der beidseitige Energiefluss ließe sich somit durch zwei separate Energiewandler realisieren, welche im `es2mapping` für die einheitlichen Ergebnisse als ein einzelner betrachtet werden. Jedoch ist es in *Calliope* nicht möglich in einer *Conversion* Technologie den gleichen Ein- und Ausgangsenergieträger zu definieren.

Wie auch schon die Energienetze (vgl. Abschnitt 3.1.2) werden auch die Konnektoren somit ohne eine Technologie an ihrem Standort dargestellt. Die Information zu den zu verbindenden Energienetzen, welche in *Calliope* Standorte sind, werden in *Links* angegeben. Im Unterschied zu allen anderen *Links*, kommen an dieser Stelle jedoch *Transmission* Technologien mit einem Wirkungsgrad zum Einsatz. Da ein Wirkungsgrad einer *Transmission* stets für beide Richtungen des Energieflusses gilt und keine zwei separaten, unterschiedlich gerichteten *Transmissions* die beiden gleichen Standorte miteinander verbinden können, wird der Konnektor in zwei Standorte aufgeteilt und ein Energiefluss je Standort nur in eine Richtung gestattet. Dies ist in Abbildung 3.4 dargestellt. Der Energiefluss vom *Bus* zum Konnektor wird als verlustfrei dargestellt, während vom Konnektor zum *Bus* der Wirkungsgrad berücksichtigt wird. Im `es2mapping` wird der Energiefluss über diese beiden Standorte zusammengetragen und als ein Konnektor ausgegeben. Darüber hinaus wird auch bei dem Abrufen des Energieflusses, der aus dem Konnektor ausströmt und in dieser Darstellung jenem entspricht der einströmt, nicht am Konnektor selbst sondern am angebundenen *Bus* ausgewertet. Somit zeigen die Ergebnisse nicht, dass der Wirkungsgrad erst nach dem Konnektor in Betracht gezogen wird, sondern lassen es so erscheinen, als würde dieser im Konnektor selbst berücksichtigt.

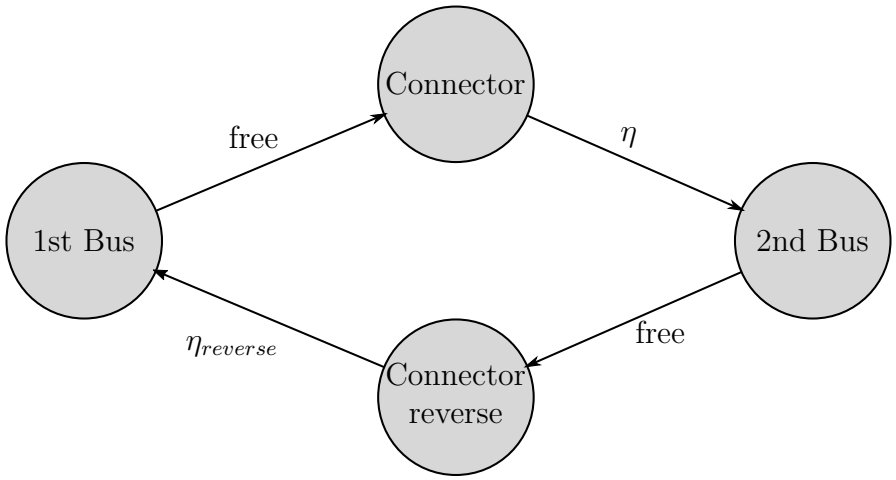

Abbildung 3.4.: Darstellung eines *Tessif Connector* in *Calliope*

3.1.8. Globale Randbedingungen

Ein wichtiger Teil der Modellierung von Energiesystemen ist die Betrachtung der entstehenden Emissionen sowie das Optimieren des Energiesystems bei gleichzeitigem Einhalten eines Emissionslimits. Auch in *Tessif* und in *Calliope* lässt sich solch ein globales Emissionslimit für das Gesamtsystem erstellen. In *Calliope*, wo die Emissionen auf gleiche Weise betrachtet werden wie Kosten, lässt sich für die Kostenklasse der Emissionen ein Limit als „group_constraints“ für eine Gruppe von Technologien und Standorten definieren. Werden weder Technologien noch Standorte angegeben, so werden alle betrachtet.

Wie in der Beschreibung der Verbraucher 3.1.3 und Erzeuger 3.1.4 beschrieben, werden die für jede Technologie vorgegebenen „accumulated_amounts“ ebenfalls als „group_constraints“ realisiert. Hierbei wird lediglich diese eine Technologie anstelle einer Reihe von Technologien angegeben. Neben des zu begrenzenden Parameters, welcher entweder der Verbrauch oder die Erzeugung ist, sowie der Angabe ob es sich um ein Maximum oder Minimum handelt, ist der entsprechende Energieträger anzugeben.

Der Quellcode 3.6 zeigt eine globale Emissionsauflage sowie die Beschränkung der minimalen Erzeugung eines einzelnen Energieerzeugers. Die Bezeichnung der Auflage ist automatisiert so erstellt, dass eindeutig zu erkennen ist was an dieser Stelle für welche Technologien beschränkt ist. Daraufhin folgt die Liste der mit dieser Auflage beschränkten Technologien, sowie des Parameters und der Information, ob dieses Limit ein Maximum oder Minimum darstellt. Bei der Emissionsauflage sind keine Technologien genannt, somit gilt diese für alle Technologien des Modells. Nachdem die Beschränkung der Emissionen als ein Kostenmaximum festgelegt ist, sind die Kostengruppe sowie das entsprechende Limit anzugeben. Analog dazu ist bei dem Energieerzeuger nach der Angabe des Limits als Minimum (oder analog Maximum) der Energieträger sowie das Limit zu nennen.

Quellcode 3.6: Beispiel der Darstellung von globalen Randbedingungen in *Calliope*

```
group_constraints:

  systemwide_emission_cap:
    cost_max:
      emissions: 1000

  Gas Source_accumulated_amounts_min:
    techs:
    - Gas Source
    carrier_prod_min:
      gas: 20
```

Anders als in *Tessif* wird in *Calliope* wie in Abschnitt 3.1.5 aufgeführt die Beschreibung von Speichern als zyklisch oder nicht zyklisch global für alle Speicher festgelegt. Sind in *Tessif* entweder alle Speicher zyklisch oder alle Speicher nicht

zyklisch, so wird dies in *Calliope* übernommen. Ist jedoch ein Speicher in *Tessif* nicht zyklisch, so ist in *Calliope* kein Speicher zyklisch. Eine entsprechende Warnung weist den Nutzer auf diese Konvention hin, sobald diese Einfluss auf die Transformation des Modells nimmt.

3.2. Energiesystem zu Mapping

In der Energiesystem zu Mapping (`es2mapping`) Funktion werden die Ergebnisse der *Calliope* Optimierung in die Struktur des einheitlichen *Tessif* Datenoutputs konvertiert. Die Einheitlichkeit des Datenoutputs erlaubt einen einfachen und direkten Vergleich der Optimierungsergebnisse. Der *Tessif* Datenoutput lässt sich in die Kategorien

- *CalliopeResultier*,
- *LoadResultier*,
- *CapacityResultier*,
- *StorageResultier*,
- *NodeCategorizer*,
- *FlowResultier* und
- *IntegratedGlobalResultier*

unterteilen. Diese Verarbeitung der Ergebnisse funktioniert in Kombination mit der `es2es` Funktion, um von *Tessif* in *Calliope* umgewandelte Modelle zu Analysieren. Ein manuell in *Calliope* erstelltes Modell, oder eines welches von *Tessif* umgewandelt und daraufhin mit *Calliope* spezifischen Parametern angepasst wurde, ist nicht zwangsläufig auch mit den hier vorgestellten Funktionen kompatibel. Ebenso wenig funktioniert das `es2mapping` mit einem *Calliope*-Modell, wenn mehrere Technologien in einem gemeinsamen Standort vorliegen.

Wird beispielsweise die zu optimierende Variable von den Gesamtkosten zu den Emissionen verändert, was in *Calliope* einfach vorzunehmen ist, in *Tessif* jedoch nicht vorgesehen, so wird ein falsches Gesamtergebnis ausgegeben, da davon ausgegangen wird, dass die Kosten optimiert werden.

Der *CalliopeResultier* (analog z.B. für *PyPSA* als *PypsaResultier* bezeichnet) ermöglicht eine Auflistung der im Modell enthaltenen Komponenten sowie derer UID's. Aufgrund dessen, dass im `es2es` Teile der UID der Energienetze und von Konnektoren nicht in das *Calliope*-Modell übergeben werden (vgl. Abschnitt 3.1), ist auch die Ausgabe jener Komponenten an dieser Stelle unvollständig. Mit dem Energieträger sowie dem Komponententyp sind jedoch die wichtigsten Parameter vorhanden.

Darüber hinaus ist es in *Calliope* nicht möglich eine Technologie mit der gleichen Bezeichnung wie einer Technologie-Gruppe zu erstellen. Damit in *Tessif* dennoch ein Verbraucher beispielsweise als „demand“ bezeichnet werden kann und auch in der Ausgabe der Ergebnisse als solcher angegeben ist, wird dieser im `es2es` automatisch umbenannt und diese Umbenennung im *CalliopeResultier* rückgängig gemacht. Dies erfordert jedoch, auch in den weiteren Funktionen des `es2mapping`, einen regelmäßigen Abgleich der Bezeichnung in *Calliope* mit jener in *Tessif*.

Eine Liste der gerichteten Kanten des Graphen des Energiesystems wird ebenso im *CalliopeResultier* erstellt. Diese beinhaltet die Informationen welche Knoten miteinander verknüpft sind, sowie in welche Richtung der Energiefluss möglich ist. Für den beidseitigen Energieaustausch werden zwei Kanten erstellt, welche jeweils in die entgegengesetzte Richtung zeigen.

Die Erzeugungs- und Verbrauchsprofile jeder Komponente lassen sich mit dem *LoadResultier* abrufen. Dieser gibt für jeden Zeitpunkt, oder wahlweise über alle Zeitschritte aufsummiert, und für jede angebundene Komponente den Eintritt von Energie negativ und den Austritt positiv an.

Dies ist im Output 3.7 für ein Modell der *Tessif* Beispielbibliothek dargestellt. Die Batterie ist vorab voll geladen mit 10 Einheiten, welche im ersten Zeitschritt genutzt werden um den ebenso großen Bedarf zu decken. In den darauffolgenden Zeitschritten wird der Bedarf durch den Generator gedeckt. In die Batterie wird zu keiner Zeit Energie eingespeist. Der Ein- und Austritt von Speichern und Konnektoren wird der Übersicht halber in *Tessif* getrennt aufgeführt und auch wenn keine Energie in die betrachtete Komponente (hier die Powerline) fließt, wird die Null mit einem negativen Vorzeichen versehen, sodass eindeutig ist in welche Richtung der Energiefluss möglich wäre.

Quellcode 3.7: Ergebnisse der Energieflüsse für die *Powerline* des Beispielmodells „minimum working example“

```
Powerline             Battery  Generator  Battery  Demand
2019-01-01 00:00:00    -10.0       -0.0      0.0    10.0
2019-01-01 01:00:00     -0.0      -10.0      0.0    10.0
2019-01-01 02:00:00     -0.0      -10.0      0.0    10.0
2019-01-01 03:00:00     -0.0      -10.0      0.0    10.0
```

Darüber hinaus ist es möglich den Eintritt und Austritt separiert voneinander auszugeben, wobei in diesem Fall alle Energieflüsse positiv dargestellt werden. Ebenso ist es möglich den Eintritt sowie den Austritt von Energie als Summe über alle angebundenen Komponenten auszugeben.

Die vorab installierten Leistungen und Kapazitäten, die zum Ende der Optimierung installierten Leistungen und Kapazitäten sowie die Investitionskosten werden im *CapacityResultier* aufgeführt. Ein Energienetz sowie ein Konnektor haben in *Tessif* keine installierten Leistungen und können somit auch nicht ausgebaut werden. Der Vollständigkeit halber werden diese dennoch, wie in der Ausgabe 3.8

dargestellt, mit aufgeführt. Als vorab installierte Leistung wird die minimale Leistung aus dem *Calliope*-Modell übernommen. Dies beruht auf der Tatsache, dass *Calliope* grundsätzlich davon ausgeht, dass vorab keine Leistungen installiert sind und ist in Abschnitt 3.1 genauer erörtert. Die Investitionskosten fallen somit bereits beim Ausbau von null zu diesem Minimum an. Diese gilt es somit in dem Gesamtergebnis der Kosten zu eliminieren.

Quellcode 3.8: Ergebnisse der installierten Leistungen des Beispielmodells „minimum working example“

```
{'Battery': 20.0,
 'Demand': 10.0,
 'Gas Station': 23.809524,
 'Generator': 10.0,
 'Pipeline': None,
 'Powerline': None}
```

Außerdem kann eine Referenzleistung bzw. -kapazität des Energiesystems ausgegeben werden. Dies ist die höchste Leistung bzw. Kapazität des Energiesystems, welche vor allem für die visuellen Darstellungen der Ergebnisse von Bedeutung ist. So kann diese beispielsweise dazu dienen um die Größe der Knoten in der graphischen Darstellung mit ihrer Leistung bzw. Kapazität zu skalieren. Ebenfalls für graphische Darstellungen von Bedeutung ist der „Characteristic Value“, welcher den durchschnittlichen Nutzen der Komponente ins Verhältnis zur Leistung bzw. Kapazität setzt und somit in etwa der im deutschsprachigen Raum verwendeten Größe „Volllaststundenzahl“ entspricht. Dieser kann im Graphen verwendet werden um die Füllung der Knoten anzupassen.

Der *StorageResultier* gibt für jeden Speicher den Ladezustand zu jedem Zeitpunkt an. Zusammen mit den Ergebnissen des *LoadResultier* für einen Speicher kann somit überprüft werden, dass die Wirkungsgrade sowie der Stillstandverlust korrekt berücksichtigt werden.

Zur Sortierung der Komponenten nach einem beliebigen UID Parameter dient der *NodeCategorizer*. An dieser Stelle ist erneut zu beachten, dass für ein Energienetz sowie einen Konnektor ein Teil der UID in *Calliope* nicht vorhanden ist. Somit werden diese Parameter im *NodeCategorizer* als „Unspecified“ ausgegeben.

Der *FlowResultier* gibt die spezifischen Parameter des Energieflusses jeder Kante an. Neben der Summe an Energie, welche über diese Kante fließt, sind dies zudem die spezifischen Kosten und Emissionen. Diese lassen sich erneut visuell durch angepasste Kantenlänge, Kantenbreite oder Graustufen im Energiesystemgraphen nutzen. Die Ausgabe 3.9 zeigt ein Beispiel einer Ausgabe des Energieflusses über den *FlowResultier*.

Quellcode 3.9: Energiefluss Ergebnisse des Beispielmodells „minimum working example“

```
{Edge(source='Powerline', target='Battery'): 0.0,
 Edge(source='Powerline', target='Demand'): 40.0,
 Edge(source='Pipeline', target='Generator'): 71.43,
 Edge(source='Gas Station', target='Pipeline'): 71.43,
 Edge(source='Battery', target='Powerline'): 10.0,
 Edge(source='Generator', target='Powerline'): 30.0}
```

Der *IntegratedGlobalResultier* liefert die Gesamtergebnisse der Optimierung. Diese bestehen aus:

- Emissionen,
- Gesamtkosten,
- Opex und
- Capex

Der Capex (Capital Expenditures) bezeichnet die Investitionskosten und der Opex (Operational Expenditures) die Betriebskosten. Die Ergebnisse des *IntegratedGlobalResultier* werden in den bisher in *Tessif* integrierten Tools auf ganze Zahlen gerundet, weshalb dies für *Calliope* ebenso gehandhabt wird.

Zum Erstellen der Ergebnisse bezüglich der Emissionen, des Opex und des Capex, greift der *IntegratedGlobalResultier* auf die Ergebnisse des *CapacityResultier* und des *FlowResultier* zu und summiert diese für jede Komponente bzw. jede Kante auf. Diese Werte sind somit keine direkte Ausgabe des jeweiligen Tools. Die Gesamtkosten hingegen werden von jedem Tool direkt als ermitteltes Optimum ausgegeben. Jedoch gilt es für *Calliope* dieses dennoch anzupassen. Wie in Abschnitt 3.1 aufgeführt geht *Calliope* grundsätzlich davon aus dass vorab keine Leistungen bzw. Kapazitäten installiert sind. Da dies in *Tessif* jedoch der Fall ist und diese in *Calliope* über das Minimum des Ausbaus dargestellt werden, entstehen in der *Calliope* Optimierung Kosten bis zu der als vorab installiert angenommenen Leistung bzw. Kapazität. Diese Kosten werden dementsprechend vom durch *Calliope* ermittelten Optimum abgezogen. Aus dem gleichen Grund ist auf gleiche Weise das Gesamtergebnis der *FINE* Optimierung angepasst [23].

Für die graphischen Darstellungen der Ergebnisse sind keine spezifischen Anpassungen zu machen, da diese Anhand der einheitlichen Ergebnisse der verschiedenen *Resultier* der einzelnen in *Tessif* integrierten Tools in einer einzelnen Funktion für alle Tools bereits formuliert sind.

4. Überprüfung der Integration

In diesem Kapitel wird die Integration von *Calliope* in *Tessif* überprüft und mit den Ergebnissen der bereits integrierten Tools verglichen. Hierfür bietet *Tessif* eine Bibliothek mit Beispielmodellen, welche von Modellen mit vier Komponenten und einer Optimierung über wenige Stunden bis hin zu Modellen mit über 30 Komponenten, welche über ein Jahr optimiert werden, reichen [21]. Besonders die kleineren Modelle bieten eine solide Grundlage zur Überprüfung der korrekten Integration, da diese Teils so trivial sind, dass bei einer korrekten Integration zwangsläufig das gleiche Ergebnis in allen Tools zu erwarten ist. Sollten die Ergebnisse doch voneinander abweichen, so ist es aufgrund der geringen Anzahl an Komponenten möglich die Ursache zu identifizieren.

Die Überprüfung der Integration anhand dieser Beispielmodelle fand im Entwicklungsprozess der `es2es` und der `es2mapping` Funktion iterativ statt und führte regelmäßig zu Anpassungen. Nichtsdestotrotz werden die Ergebnisse an dieser Stelle der Vollständigkeit halber für die finale Version des `es2es` und des `es2mapping` aufgeführt. Zusätzlich zur Überprüfung der Integration dient diese Untersuchung der Auffindung von grundlegenden Verschiedenheiten der Tools in der Betrachtung einzelner Komponenten oder Parameter. Die Optimierung eines größeren Modells dient zum Abschluss zusätzlich einem Vergleich der Optimierungsdauer. Bei allen aufgeführten Rechenzeiten handelt es sich um Mittelwerte von drei durchgeführten Optimierungen, welche durchgeführt werden um Störungen durch Hintergrundprozesse zu reduzieren.

Die Optimierungen werden in dieser Arbeit alle mit dem *CBC*[1] Solver durchgeführt. Dies ist jener kostenfreie Solver, welcher von *Oemof* und *Calliope* empfohlen wird[2] [25, 28] und der Standardsolver in *Tessif* ist [21]. Darüber hinaus wird die Python Version 3.8.9 sowie die *Pyomo* Version 5.7.2 verwendet. Die Optimierungen werden auf einem Windows 10 Rechner mit einem AMD Ryzen 5 1600 @3.20 GHz und 16 GB RAM durchgeführt.

Die Modelle werden alle mit der „Comparatier“ Funktion *Tessif*'s optimiert, mit welcher mehrere Tools automatisch nacheinander das Modell optimieren und die Ergebnisse gemeinsam ausgegeben werden. Hierbei ist es zusätzlich möglich die Optimierungsdauer genauer zu analysieren. Neben der Angabe, welche Tools zu untersuchen sind, kann ebenso angegeben werden, ob für einzelne Tools Anpassungen an dem Modell vorzunehmen sind. Eine solche Anpassung wird im Abschnitt 4.5.2 für die Optimierung mit *PyPSA* durchgeführt.

[1] `https://www.coin-or.org/`

[2] *PyPSA* und *FINE* führen keine Empfehlung bezüglich eines kostenfreien Solvers auf [22, 26].

Zunächst werden in Abschnitt 4.1 die Erzeuger, Energienetze und Verbraucher betrachtet, welche die Grundlage eines jeden Modells in *Tessif* bilden. Daraufhin werden die verbleibenden Komponenten beginnend mit den Speichern, gefolgt von den Energiewandlern und abgeschlossen mit den Konnektoren separat untersucht. Nachdem die Funktionalitäten der einzelnen Komponenten geklärt sind, wird in Abschnitt 4.5 eines der komplexeren in *Tessif* vorliegenden Modelle optimiert und ein Vergleich der Ergebnisse sowie der Optimierungsdauer der Tools durchgeführt. Das gleiche Modell wird zudem direkt als ein *Calliope*-Modell erstellt, welches nicht der für die Automatisierbarkeit der *Tessif* Übersetzung notwendigen Konventionen folgt. Dieses wird folglich unabhängig von *Tessif* untersucht und mit der Optimierung über *Tessif* verglichen.

4.1. Untersuchung der Energienetze, Erzeuger und Verbaucher

In diesem Abschnitt wird anhand des in der *Tessif* Bibliothek hinterlegten „Expansion Plan Examples“ die Verwendung von Erzeugern, Verbrauchern und Energienetzen überprüft und über die verschiedenen Tools verglichen. Dies sind die wichtigsten Komponenten innerhalb eines *Tessif*-Modells, da jedes beliebige Modell jede dieser Komponenten mindestens einmal aufzuweisen hat. Das Modell, welches im Folgenden aufgrund des Fokus auf die Erzeuger als „Source Example“ bezeichnet wird, ist in Abbildung 4.1 graphisch dargestellt und beinhaltet einen Verbraucher, ein Energienetz sowie drei verschiedene Erzeuger.

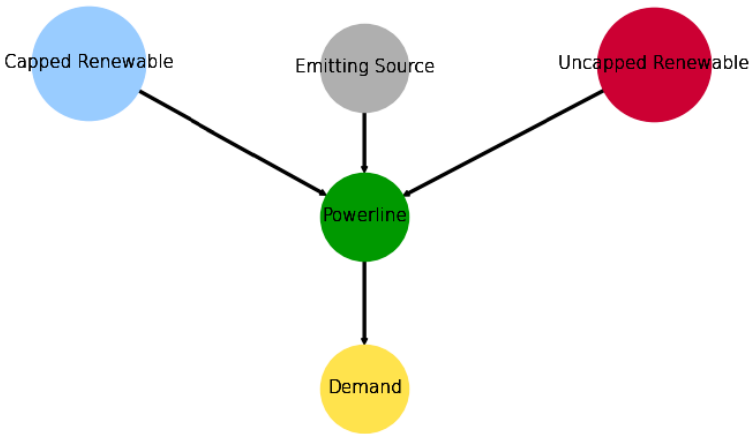

Abbildung 4.1.: Graphische Darstellung des „Source Example“ [21]

Die drei Erzeuger unterscheiden sich grundlegend in ihrer Parametrisierung. So ist die „Emitting Source“ unendlich groß und kostenfrei, jedoch mit Emissionen von 1 t/MWh behaftet. Der Erzeuger „Capped Renewable“ hat eine installierte Leistung von 2 MW und spezifische Kosten von 2 €/MWh. Zusätzlich ist ein Ausbau

bis zu einer Leistung von 4 MW unter einem Kostenaufwand von 1 €/MW möglich. Dem dritten Erzeuger „Uncapped Renewable“ ist ein festes Erzeugungsprofil vorgegeben. Unter einem Kostenaufwand von 2 €/MW kann dieses ausgebaut werden. Zusätzlich liegen spezifische Betriebskosten von 2 €/MWh vor. Der Verbrauch beträgt konstant 10 MW. Zusätzlich ist dem gesamten System ein Emissionslimit von 20 Tonnen vorgegeben. Die entsprechenden Werte der einzelnen Komponenten sind in den Tabellen 4.2 und 4.3 neben den Optimierungsergebnissen aufgeführt.

Aus Tabelle 4.1 geht hervor, dass die Gesamtergebnisse bei allen Optimierungen übereinstimmen. Das Emissionslimit wird zudem in allen Tools berücksichtigt, da eine abweichende Nutzung der Komponenten eindeutig kostengünstiger wäre, jedoch aufgrund des vorgegebenen Emissionslimits nicht ausführbar ist.

Tabelle 4.1.: Gesamtergebnis des „Source Examples“

	Calliope	*FINE*	*Oemof*	*PyPSA*
Emissionen [t]	20,0	20,0	20,0	20,0
Kosten [€]	41,0	41,0	41,0	41,0
Opex [€]	40,0	40,0	40,0	40,0
Capex [€]	1,0	1,0	1,0	1,0

Anhand der Energieströme aus Tabelle 4.2 sowie den aufgeführten spezifischen Kosten und Emissionen ist zu erkennen, dass die Emissionen ebenso wie der Opex in Tabelle 4.1 korrekt ermittelt sind. Bei dem Capex, welcher sich aus den in Tabelle 4.3 aufgeführten Leistungen und Kosten ergibt, und folglich ebenso bei den Gesamtkosten, kommt es aufgrund der Rundung auf ganze Zahlen im `es2mapping` (vgl. Abschnitt 3.2) zu einer geringen Abweichung zum zu erwartenden Ergebnis. So wären ein Capex von 1,25 € sowie Gesamtkosten von 41,25 € zu erwarten.

Tabelle 4.2.: Summierte Energieströme und spezifische Kosten und Emissionen des „Source Examples“

	Summierte Energieströme				spez.	spez.
	Calliope	*FINE*	*Oemof*	*PyPSA*	Kosten	Emissionen
Capped Renewable	13,0	13,0	13,0	13,0	2	0
Emitting Source	20,0	20,0	20,0	20,0	0	1
Uncapped Renewable	7,0	7,0	7,0	7,0	2	0
Demand	40,0	40,0	40,0	40,0	0	0

Tabelle 4.3.: Leistungen und Investitionskosten der Erzeuger des „Source Examples“

		Calliope	*FINE*	*Oemof*	*PyPSA*
initiale Leistung	Emitting Source	0,0	0,0	0,0	0,0
	Capped Renewable	2,0	2,0	2,0	2,0
	Uncapped Renewable	3,0	3,0	3,0	3,0
optimale Leistung	Emitting Source	5,75	5,75	5,75	5,75
	Capped Renewable	3.25	3.25	3.25	3.25
	Uncapped Renewable	3,00	3,00	3,00	3,00
Investitionskosten	Emitting Source	0,0	0,0	0,0	0,0
	Capped Renewable	1,0	1,0	1,0	1,0
	Uncapped Renewable	2,0	2,0	2,0	2,0

Tabelle 4.4 zeigt die Optimierungsdauer sowie die Dauer der einzelnen *Tessif* Funktionen vor und nach der Optimierung. „Transformation“ bezeichnet das Umwandeln des *Tessif*-Modells in ein Modell eines der Tools über die entsprechende `es2es` Funktion. Daraufhin wird das Modell optimiert (Optimization[3]) und die Ergebnisse im `es2mapping` in die einheitliche Form umgewandelt (Post Processing).

Tabelle 4.4.: Zeiten des „Source Examples“ in Sekunden (gerundet auf 2 Nachkommastellen)

	Calliope	*FINE*	*Oemof*	*PyPSA*
Transformation	0,52	0,09	0,05	0,35
Optimization	0,31	0,22	0,06	0,18
Post Processing	0,30	0,07	0,07	0,05
Summe	1,22	0,47	0,27	0,68

Zu erkennen ist, dass *Calliope* stets am meisten Zeit benötigt. Eine genauere Analyse der Zeiten wird für ein größeres Modell in Abschnitt 4.5 und 4.6 durchgeführt. Bedeutend ist an dieser Stelle vor allem, dass wegen des *Calliope* spezifischen Schreibens und Lesens der Modell Daten in `YAML` und `CSV` Dateien bei einem kleinen Modell keine deutlich höheren Gesamtzeiten zu erwarten sind.

Auf Grundlage der Ergebnisse dieses Modells lassen sich im folgenden die einzelnen Komponenten Speicher, Energiewandler sowie Konnektor individuell untersuchen, da davon auszugehen ist, dass die Erzeuger, Verbraucher und Energienetze korrekt integriert sind.

[3] In *Tessif* sind zum aktuellen Zeitpunkt die Begriffe Optimierung und Simulation nicht klar getrennt und dieser Teil als „Simulate“ bezeichnet. Eine entsprechende Korrektur ist jedoch geplant.

4.2. Untersuchung der Speicher

Das „Storage Example“ dient dazu die Integration der Energiespeicher genauer zu betrachten. Das Modell besteht neben dem Speicher aus einem Verbraucher, einem Erzeuger sowie einem Stromnetz (vgl. Abb. 4.2). Die geringe Anzahl an Komponenten hat den Vorteil, dass Abweichungen der Ergebnisse aufgrund anderer Komponenten minimiert werden. Dem Verbraucher sowie dem Erzeuger liegen zudem ein festes Profil vor, welches so gewählt ist, dass eine Verwendung des Speichers zwingend notwendig ist.

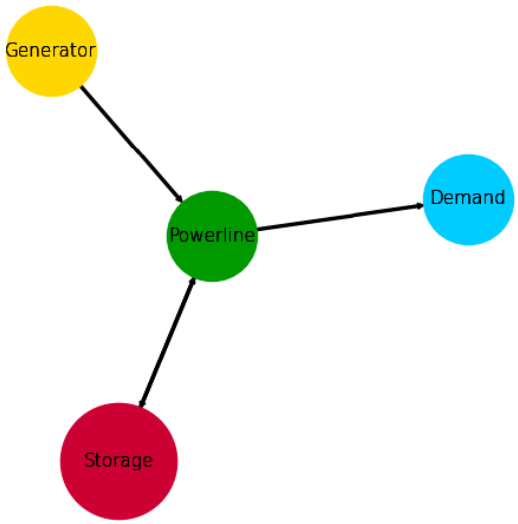

Abbildung 4.2.: Graphische Darstellung des „Storage Examples“ [21]

Da bereits bekannt ist, dass *Calliope* beim Ein- und Austrittswirkungsgrad eines Speichers nicht differenziert (vgl. Abschnitt 3.1.5) und aus den Dokumentationen der anderen Tools [22, 25, 26] hervorgeht, dass diese dies tun, ist es nicht notwendig das Herausfinden anderer Unterschiede der Tools durch diesen bereits bekannten Unterschied zu erschweren. Deshalb sind die Wirkungsgrade in *Tessif* beide auf 90 % festgelegt. An dieser Stelle werden einige weitere Anpassungen an dem Modell der *Tessif* Bibliothek vorgenommen, sodass die Auswirkungen vieler Parameter des Speichers erkennbar sind. Ohnehin ist die vorab installierte Kapazität als 1 MWh sowie ein maximaler Energiefluss, relativ zur Kapazität, von 0,1 MW/MWh gegeben. In der hier angepassten Variante des Modells ist der Speicher zu Beginn (vor dem ersten zu optimierenden Zeitschritt) zur Hälfte gefüllt. Zusätzlich wird ein Stillstandverlust von 1 % angegeben. Ein Ausbau der Kapazität ist unter einem Kostenaufwand von 2 €/MWh möglich und in der Konstellation des Modells zwingend notwendig. Die Kosten für die Energieerzeugung liegen bei 1 €/MWh und die Emissionen bei 0,5 t/MWh. Diese stehen jedoch in dieser Untersuchung nicht im Mittelpunkt, da der Fokus auf die für einen Speicher spezifischen Parameter gerichtet ist und die korrekte interne Berechnung der Emissionen in den verschiedenen Tools nur mit einem Emissionslimit bewertet werden kann. In Tabelle 4.5 sind die einzelnen Parameter des Speichers der Übersicht halber zusammengefasst.

Tabelle 4.5.: Parameter des Speichers des „Storage Examples“

Parameter	zugewiesener Wert
installierte Kapazität	1 MWh
max. Energiefluss[a]	0,1 MW/MWh
initialer Speicherstand	50 %
Wirkungsgrad[b]	90 %
Standverlust	1 %
Investitionskosten	2 €/MWh
Betriebskosten	1 €/MWh
Emissionen	0,5 t/MWh

[a]Relativ zur Kapazität
[b]Bezogen auf den Ein- und Austritt

Zuletzt wird der Verbrauch im vierten Zeitschritt auf fünf geändert, da dies die Definition des *Calliope* Parameters „energy_cap_per_storage_cap“ aufzeigt, welcher stark von der ursprünglichen Definition im *Tessif*-Modell abweicht. Da der Verbrauch ebenso wie die Erzeugung der Energiequelle erzwungen sind, gehen die entsprechenden Profile aus dem Ergebnis in Tabelle 4.6 hervor.

Die Ergebnisse des *LoadResultier* stimmen für die Optimierung mit *Calliope* mit jenen von *Oemof*, *PyPSA* und *FINE* überein. Tabelle 4.6 führt dieses Ergebnis für das Stromnetz auf, wobei auf die mehrfache Wiederholung des gleichen Ergebnisses der verschiedenen Tools verzichtet wird und die *Calliope* Ergebnisse stellvertretend für alle verwendeten Tools stehen. Dieses Ergebnis ist, aufgrund dessen, dass die Erzeugung und der Verbrauch fest vorgegeben sind, ohnehin zu erwarten. Somit hat der Speicher zwangsläufig die überschüssige Erzeugung der ersten drei Zeitschritte aufzunehmen und den Bedarf in den verbleibenden beiden Zeitschritten zu decken.

Tabelle 4.6.: Lastprofil der *Powerline* des „Storage Examples“

Zeit	Generator	Storage	Demand	Storage
00:00:00	-19,0	-0,0	10,0	9,0
01:00:00	-19,0	-0,0	10,0	9,0
02:00:00	-19,0	-0,0	7,0	12,0
03:00:00	-0,0	-5,0	5,0	0,0
04:00:00	-0,0	-10,0	10,0	0,0

Tabelle 4.7 zeigt den aktuellen Speicherstand zu jedem Zeitschritt für die verschiedenen Optimierungen an. Diese unterscheiden sich allesamt voneinander und bedarfen demnach einer genaueren Analyse. Die Werte der *Calliope* und *Oemof* Optimierung erschließen sich erst nach einer Betrachtung der installierten Kapazität, weshalb zunächst die *FINE* und *PyPSA* Ergebnisse erläutert werden.

Tabelle 4.7.: Speicherstand des Speichers des „Storage Examples“ (gerundet auf 4 Nachkommastellen)

Zeit	*Calliope*	*FINE*	*Oemof*	*PyPSA*
00:00:00	93,1000	0,0000	67,5000	8,5950
01:00:00	100,2690	8,1000	74,9250	16,6091
02:00:00	110,0663	16,1190	84,9758	27,2430
03:00:00	103,4101	26,7578	78,5704	21,4150
04:00:00	91,2649	20,9347	66,6736	10,0897

Aus der Integration von *FINE* ist bekannt, dass der initiale Speicherstand in *FINE* nicht abzubilden ist [23]. Darüber hinaus ist in Tabelle 4.7 der Ladezustand von *FINE* stets um einen Zeitschritt versetzt ausgegeben. In *PyPSA* wird der initiale Speicherstand berücksichtigt und bereits vom initialen zum ersten abgebildeten Zeitschritt der Stillstandverlust berücksichtigt. Die darauffolgenden Speicherstände ergeben sich in beiden Tools stets korrekt aus dem vorhergehenden, abzüglich des Standverlustes, sowie der dazu kommenden Ladung bzw. Entladung unter Berücksichtigung des Wirkungsgrads.

Die Ergebnisse von *Calliope* und *Oemof* weichen bereits im ersten Zeitschritt stark von den anderen beiden Ergebnissen ab. Dies ist darauf zurückzuführen, dass der initiale Speicherstand von 50 % auf die optimale Kapazität, bis zu welcher der Speicher ausgebaut wurde, bezogen ist (vgl. Tab. 4.8) und nicht auf die ursprünglich installierte Kapazität von 1 MWh wie bei *PyPSA*. Darüber hinaus berücksichtigt *Calliope* den Stillstandverlust noch nicht vom initialen Zeitschritt aus, sondern erst ab dem darauffolgenden.

Tabelle 4.8.: Optimale Kapazität des Speichers des „Storage Examples“

Tool	Kapazität
Calliope	170
FINE	120
Oemof	120
PyPSA	120

Die in *Calliope* abweichende Kapazität des Speichers aus Tabelle 4.8 ergibt sich aus der Definition der Begrenzung des Energieflusses. Während in den anderen Tools der Energiefluss für das maximale Laden bzw. Entladen begrenzt wird, wird dieser in *Calliope* mit einem Wert, welcher die maximale Differenz zweier aufeinander folgender Lade und Entlade Zeitschritte begrenzt, angegeben. In diesem Modell ist somit ein Ausbau auf eine Kapazität von 170 MWh vonnöten, sodass die Differenz des Ladens und Entladens der Zeitschritte drei und vier von 17 MW mit dem vorgegebenen „energy_cap_per_storage_cap“ möglich ist (vgl. Tab. 4.6). Da die anderen Tools diese separat betrachten, wie es auch in dem *Tessif*-Modell vorgesehen ist, ist dort ein Ausbau auf 120 MWh ausreichend.

Grundsätzlich ist es möglich einen Energiefluss in *Calliope* auch über die *Transmission* Technologie zu beschränken, welche die Standorte bzw. Technologien miteinander verbindet. Jedoch ist diese Beschränkung nicht mit dem Ausbau des Speichers skalierbar und insofern nur möglich für Speicher, welche nicht ausgebaut werden können. Da eine verschiedene Definition der Energieflussbeschränkung abhängig von der Möglichkeit eines Ausbaus der Kapazität die einfache Analyse der Ergebnisse behindert, wird hierauf verzichtet.

Der Speicher ist die einzige Komponente dieses Modells, welcher Kosten und Emissionen zugewiesen sind. Demnach reicht es, zur Überprüfung der korrekten Übertragung der Parameter des *Tessif*-Modells zu den jeweiligen spezifischen Modellen, aus das Gesamtergebnis, mit Blick auf die vorgegebenen Werte sowie den zuvor aufgeführten Leistungen (Tab. 4.6) und Kapazitäten (Tab. 4.8), zu betrachten. Das Gesamtergebnis aus Tabelle 4.9 zeigt, dass die Kosten und Emissionen in allen Optimierungen, bis auf den Unterschied in den Kosten der *Calliope* Optimierung aufgrund der Speicherkapazität wegen der abweichenden Definition der Energieflussbeschränkung, gleich sind.

Tabelle 4.9.: Gesamtergebnis des „Storage Examples“

	Calliope	*FINE*	*Oemof*	*PyPSA*
Emissionen [t]	8,0	8,0	8,0	8,0
Kosten [€]	353,0	253,0	253,0	253,0
Opex [€]	15,0	15,0	15,0	15,0
Capex [€]	338,0	238,0	238,0	238,0

Die korrekte Berücksichtigung der Emissionen lässt sich erst eindeutig feststellen, wenn ein einzuhaltendes Emissionslimit vorgeschrieben ist, da dieser Wert von *Tessif* ausgerechnet wird. Aktuell ist lediglich festzuhalten, dass die Emissionen wie erwartet an *Calliope* übergeben werden und von *Tessif* ausgelesen werden. Dies ist beispielsweise ebenso für *PyPSA* der Fall, während *PyPSA* jedoch diese vollkommen verschieden verrechnet [26, 55]. Jedoch ist zu erwarten dass die Emissionen in *Calliope* korrekt aufgefasst werden, da sie als weitere Kostengruppe gehandhabt werden und dementsprechend genauso verrechnet werden sollten.

Insgesamt lässt sich festhalten, dass die Speicher in allen Tools wesentliche Unterschiede aufweisen. In dem hier untersuchten Modell jedoch, bei einer ausschließlichen Betrachtung des Gesamtergebnisses aus Tabelle 4.9, drei der vier Tools gleiche Ergebnisse liefern und lediglich *Calliope* von den anderen Tools abweicht. Dies zeigt ebenso, dass eine detaillierte Analyse der einzelnen Ergebnisse, wie in diesem Fall dem Speicherstand in Tabelle 4.7, trotz eines gleichen Gesamtergebnisses, weitere Erkenntnisse liefern kann.

4.3. Untersuchung der Energiewandler

Als nächstes wird ein Energiewandler mit zwei verschiedenen Ausgangsenergieträgern, anhand des in Abbildung 4.3 dargestellten „CHP Examples“ aus der *Tessif* Bibliothek, genauer betrachtet. Wie auch das Modell in Abschnitt 4.2 wird dieses Modell einigen Anpassungen unterworfen, sodass möglichst viele Parameter berücksichtigt werden.

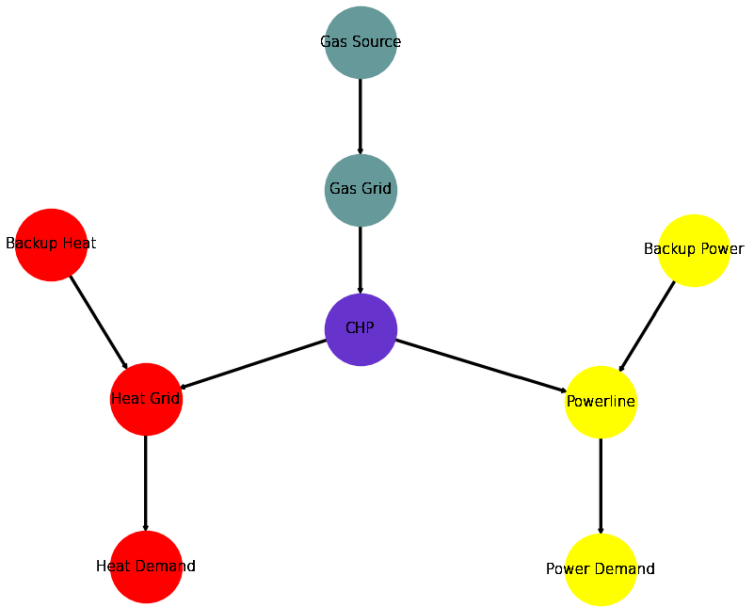

Abbildung 4.3.: Graphische Darstellung des „CHP Examples“ [21]

Das im Fokus stehende Gaskraftwerk mit Kraft-Wärme-Kopplung (KWK) wird aus einer idealen, unendlich großen Gasquelle gespeist und weist einen Wirkungsgrad von 30 % zur Stromerzeugung und 20 % zur Wärmeerzeugung auf. Jeweils ein Verbraucher im Wärme- und Stromnetz fordert konstant 10 MW Leistung. Neben dem KWK Gaskraftwerk, kann diese Leistung alternativ durch eine Wärme- und Stromquelle bereitgestellt werden. Diese sind jedoch mit Betriebskosten von 100 €/MWh deutlich teurer als das Gaskraftwerk, welchem Kosten von 3 €/MWh für die Stromerzeugung und 2 €/MWh für die Wärmeerzeugung vorgegeben sind. Emissionen fallen ausschließlich im Gaskraftwerk an und diese belaufen sich auf $2\,\mathrm{t/MWh_{Strom}}$ und $3\,\mathrm{t/MWh_{Wärme}}$. Vorab installiert ist ein Gaskraftwerk, welches $3\,\mathrm{MWh_{Strom}}$ und $2\,\mathrm{MWh_{Wärme}}$ erzeugen kann. Dieses kann unter einem Kostenaufwand von $2\,€/\mathrm{MW_{Strom}}$ und $1\,€/\mathrm{MW_{Wärme}}$ ausgebaut werden. Somit stellt das Gaskraftwerk trotz der Notwendigkeit des Ausbaus die deutlich kostengünstigere Strom- und Wärmeerzeugung dar.

Die Optimierungsergebnisse stimmen in allen in *Tessif* integrierten Tools überein. Als einziger minimaler Unterschied, welcher keinerlei Auswirkungen auf das

Gesamtergebnis hat (vgl. Tab. 4.11), ist die Ausgabe der spezifischen Kosten zu nennen. Diese sind in Tabelle 4.10 aufgeführt. In *Calliope* ist, wie in Abschnitt 3.1.6 ausgeführt, keine Unterteilung der Kosten auf die verschiedenen Energieträger möglich, weshalb diese zusammengefasst und auf den primären Energieträger umgerechnet werden (vgl. Gleichung (3.2)). Aus diesem einzelnen Wert lässt sich im Nachhinein nicht auf die beiden im *Tessif*-Modell vorgegeben Werte schließen.

Tabelle 4.10.: Kosten und Emissionen der KWK Komponente des „CHP Examples“

		Calliope	*FINE*	*Oemof*	*PyPSA*
Betriebskosten	Powerline	4,33	3,00	3,00	3,00
	Heat Grid	0,00	2,00	2,00	2,00
Emissionen	Powerline	4,00	2,00	2,00	2,00
	Heat Grid	0,00	3,00	3,00	3,00
Investitionskosten	Powerline	2,67	1,00	1,00	1,00
	Heat Grid	0,00	2,00	2,00	2,00

Tabelle 4.11.: Gesamtergebnis des „CHP Examples“

	Calliope	*FINE*	*Oemof*	*PyPSA*
Emissionen [t]	160,0	160,0	160,0	160,0
Kosten [€]	1525,0	1525,0	1525,0	1525,0
Opex [€]	1506,0	1506,0	1506,0	1506,0
Capex [€]	19,0	19,0	19,0	19,0

Durch die Hinzunahme von Tabelle 4.12 sowie Tabelle 4.13 lassen sich analog zu Abschnitt 4.2 die Ergebnisse aus Tabelle 4.11 auf Plausibilität prüfen. Auf eine genauere Ausführung dieser Prüfung wird an dieser Stelle aufgrund der Einheitlichkeit sämtlicher Ergebnisse verzichtet und festgehalten, dass die Energiewandler in *Calliope* korrekt implementiert zu sein scheinen und nicht wesentlich verschieden gehandhabt werden als in den anderen Tools.

Tabelle 4.12.: Lastprofil der KWK Komponente des „CHP Examples“

Zeit	Gas Grid	Heat Grid	Powerline
00:00:00	-33,33	6,67	10,0
01:00:00	-33,33	6,67	10,0
02:00:00	-33,33	6,67	10,0
03:00:00	-33,33	6,67	10,0
04:00:00	-33,33	6,67	10,0

Tabelle 4.13.: Installierte Leistungen der KWK Komponente des „CHP Examples“

		Calliope	*FINE*	*Oemof*	*PyPSA*
initiale Leistung	Powerline	3,0	3,0	3,0	3,0
	Heat Grid	2,0	2,0	2,0	2,0
optimale Leistung	Powerline	10,00	10,00	10,00	10,00
	Heat Grid	6,67	6,67	6,67	6,67

4.4. Untersuchung der Konnektoren

Ein Minimalbeispiel, in welchem Konnektoren verwendet werden, stellt das „Connected Example“ dar. Dieses ist in Abbildung 4.4 graphisch dargestellt.

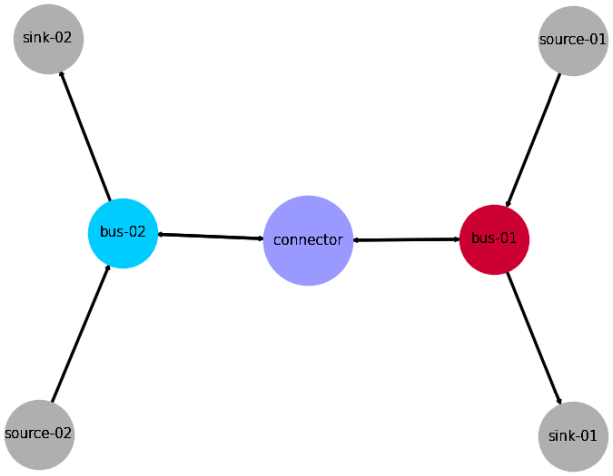

Abbildung 4.4.: Graphische Darstellung des „Connected Examples“ [21]

Das „Connected Example“ beinhaltet neben dem Konnektor zwei Verbraucher, zwei Erzeuger sowie zwei Energienetze des gleichen Energieträgers. Während die Erzeuger zu jedem Zeitpunkt 10 MW Energie bereitstellen können, ist der Verbrauch so gewählt, dass die Nutzung des Konnektors in beide Richtungen zu verschiedenen Zeitpunkten vonnöten ist.

Der Energiefluss über den Konnektor von Bus-01 zu Bus-02 ist mit einem Wirkungsgrad von 90 % beaufschlagt, während der entgegengesetzte Energiefluss einen Wirkungsgrad von 80 % aufweist. Die Ergebnisse werden in diesem Beispiel nicht mit *PyPSA* verglichen, da die Übersetzung der *Tessif* Konnektoren in *PyPSA* keinen Wirkungsgrad übergibt und dementsprechend von den anderen Ergebnissen abweicht [21]. Da dieser Unterschied bereits bekannt ist, die Nutzung von Konnektoren für *Oemof* und *PyPSA* bereits verglichen wurde [56] und die Konnektoren neben dem Wirkungsgrad keine weiteren Parameter aufweisen, ist ein Vergleich mit *PyPSA* nicht notwendig.

Die Ergebnisse in *Calliope*, *FINE* und *Oemof* stimmen allesamt überein, weshalb auf die Darstellung der Ergebnisse für jedes Tool verzichtet wird und die Tabellen 4.14 und 4.15 stellvertretend für alle Optimierungen stehen. Zu erkennen ist, dass die Wirkungsgrade korrekt berücksichtigt werden und folglich die *Tessif* Konnektoren korrekt in *Calliope* integriert sind.

Tabelle 4.14.: Lastprofil von bus-01 des „Connected Examples“

Zeit	connector	source-01	connector	sink-01
00:00:00	-0,0	-5,56	5,56	0,00
01:00:00	-5,0	-10,00	0,00	15,00
02:00:00	-0,0	-10,00	0,00	10,00

Tabelle 4.15.: Lastprofil von bus-02 des „Connected Examples“

Zeit	connector	source-02	connector	sink-02
00:00:00	-5,0	-10,00	0,00	15,00
01:00:00	-0,0	-6,25	6,25	0,00
02:00:00	-0,0	-10,00	0,00	10,00

Zudem ist die komplexe Darstellung in *Calliope* über zwei Standorte und Wirkungsgrad behafteten *Transmission* Technologien aus den Ergebnissen von *Tessif* nicht ersichtlich. Die Ergebnisse *Calliope*'s sind für die *Tessif* Ausgabe so verarbeitet, dass der Konnektor wie eine eigene Technologie zu fungieren scheint.

4.5. Untersuchung eines Referenzmodells

In diesem Abschnitt soll ein Referenzmodell Gemeinsamkeiten und Unterschiede der Tools bei der Optimierung eines komplexeren Modells aufzeigen. Als Referenzmodell wird das in der *Tessif* Bibliothek vorliegende „Component Example“ verwendet, welches von Reimer [55] in Anlehnung an einen realen Modellierungsfall entwickelt und mit *Oemof* und *PyPSA* untersucht wurde. Darüber hinaus wurde dieses Modell von Schnute [23] dazu verwendet die Integration von *FINE* in *Tessif* mit *Oemof* und *PyPSA* zu vergleichen.

Das Modell beinhaltet 24 Komponenten auf zwei Sektoren und ist graphisch in Abbildung 4.5 dargestellt. Eine Zusammenfassung der Parameter des Modells findet sich im Anhang A.1. Für eine genauere Ausführung zur Zusammenstellung des Modells sowie der Wahl der Parameter sei auf Reimer [55] verwiesen.

Das Modell ist so konstruiert, dass zwei verschiedene Szenarien durch Anpassen eines Parameters untersucht werden können. Das erste Szenario, welches als „Commitment Example“ bezeichnet und in Abschnitt 4.5.1 untersucht wird, gestattet keinen Ausbau der installierten Leistungen. Das Modell wird folglich hinsichtlich

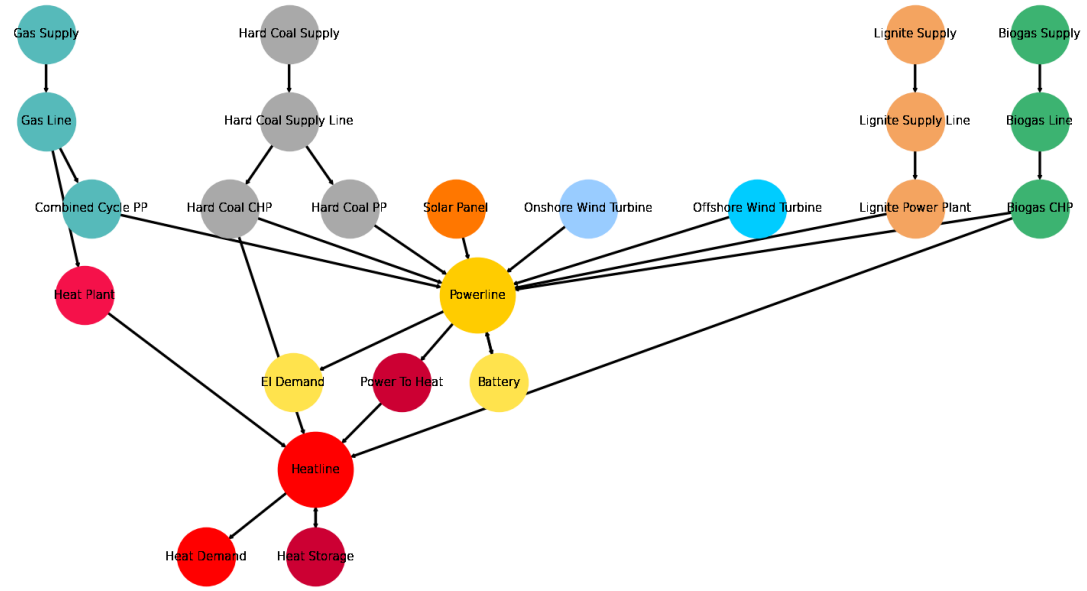

Abbildung 4.5.: Graphische Darstellung des „Component Examples“ [21]

des kostengünstigsten Betriebs optimiert und ist somit als klassische Einsatzplanoptimierung zu verstehen. Das zweite Szenario wird im weiteren als „Expansion Example“ bezeichnet. Es erlaubt einen kostenaufwendigen Leistungsausbau jeder Komponente und beschränkt zusätzlich die Emissionen des Gesamtsystems. Es kann somit als typisches emissionsbeschränktes Ausbauproblem interpretiert werden. Dieses zweite Szenario wird in Abschnitt 4.5.2 untersucht.

Aus Reimer [55] geht hervor, dass *PyPSA* intern die Emissionen von Speichern anders ermittelt als *Tessif* in der Aufbereitung der Ergebnisse annimmt. Zudem werden die Emissionen der Power-to-Heat Anlage nicht von *PyPSA* berücksichtigt, da *PyPSA* ausschließlich Emissionen von Erzeugern und Speichern verrechnet. Einfache Energiewandler, welche aus idealen Quellen gespeist werden, werden daher in der Übersetzung des *Tessif*-Modells für *PyPSA* automatisch zu Erzeugern umformuliert. Für die Energiewandler mit mehreren Ausgängen, gilt es die Emissionen manuell für die *PyPSA* Optimierung in die davor liegenden Erzeuger zu überführen. Da *PyPSA* grundsätzlich jedoch die Parametrisierung aller Komponenten mit Emissionen gestattet, auch wenn diese intern nicht verrechnet werden, ist die abweichende Berechnung von *PyPSA* erst bei einem gegebenen Emissionslimit von Bedeutung.

4.5.1. Commitment Example

Abbildung 4.6 fasst die Abweichungen des Gesamtergebnisses der verschiedenen Optimierungen des „Commitment Examples“ zusammen. Hierbei dient das Ergebnis der Optimierung mit *Calliope* als Referenzwert. Die exakten Ergebnisse sind im Anhang in Tabelle A.2 aufgeführt. Im weiteren sind jedoch vielmehr die

Unterschiede der Tools, als die exakten Ergebnisse von Bedeutung. Optimierungsergebnisse sehen ohnehin lediglich exakt aus, sind dies aufgrund der Annahmen und Vereinfachungen in der Formulierung des Modells jedoch nicht [9]. Dies bedeutet, dass die Ergebnisse, welche auf den Euro genau berechnet und ausgegeben werden, nicht direkt auf die Realität übertragbar sind, sondern stets unter Berücksichtigung der Vereinfachungen des Modells zu interpretieren sind.

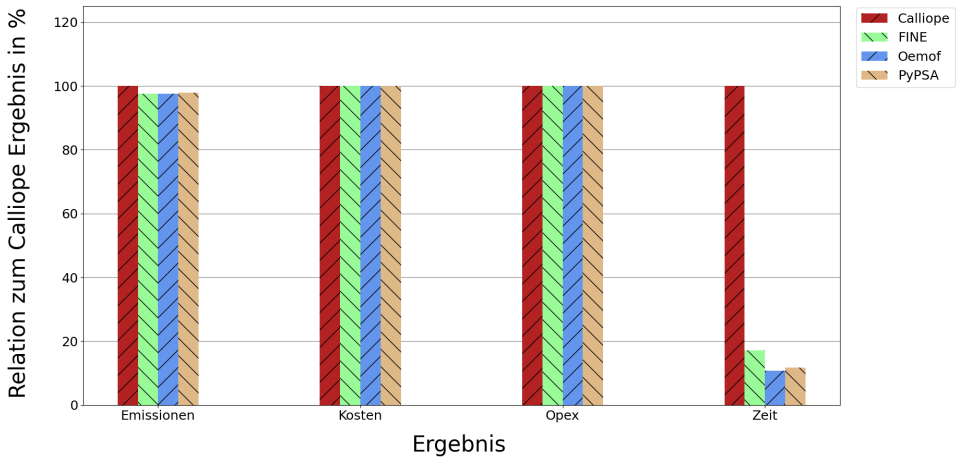

Abbildung 4.6.: Gesamtergebnis des „Commitment Examples“ der verschiedenen Tools in Relation zum Ergebnis von *Calliope*

Zu erkennen ist, dass die Unterschiede bei den Kosten mit Abweichungen von kleiner als 0,001 % vernachlässigbar gering sind. Aufgrund der gleichen Betriebskosten bei gleichzeitig verschiedenen Emissionen des Steinkohle Kraftwerks und der PV-Anlage entstehen die größeren Abweichungen in den Gesamtemissionen in Abbildung 4.6. Dies zeigt sich bei einer Betrachtung der Stromerzeugung in Abbildung 4.7 sowie der Wärmeerzeugung in Abbildung 4.8. Aufgeführt sind an dieser Stelle nur jene Komponenten, welche einen Teil zur Energieerzeugung dieser Sektoren beitragen. Die Ergebnisse der meisten Komponenten haben lediglich Abweichungen in einem Bereich von kleiner als 0,001 % und sind daher zu vernachlässigen.

In Anbetracht dessen dass keine gesellschaftlichen und politischen Aspekte (wie z.B. Arbeitsplätze oder Landflächennutzung) modelliert werden, ist bei der ausschließlichen Betrachtung von Kosten und Emissionen grundsätzlich zu erwarten, dass in der Realität bei gleichen Kosten eine emissionsarme Variante zu bevorzugen wäre. In allen Tools gilt, dass immer nur dann sowohl die PV-Anlage als auch das Steinkohlekraftwerk aktiv sind, wenn eine dieser Komponenten voll ausgelastet ist. Für *Oemof* und *PyPSA* wurde dies bereits genauer untersucht [55] und zeigt sich für *Calliope* und *FINE* ebenso. Dies ist in Abbildung 4.9 für einen ausgewählten Zeitbereich zu erkennen und gilt ebenso für den restlichen Optimierungszeitraum.

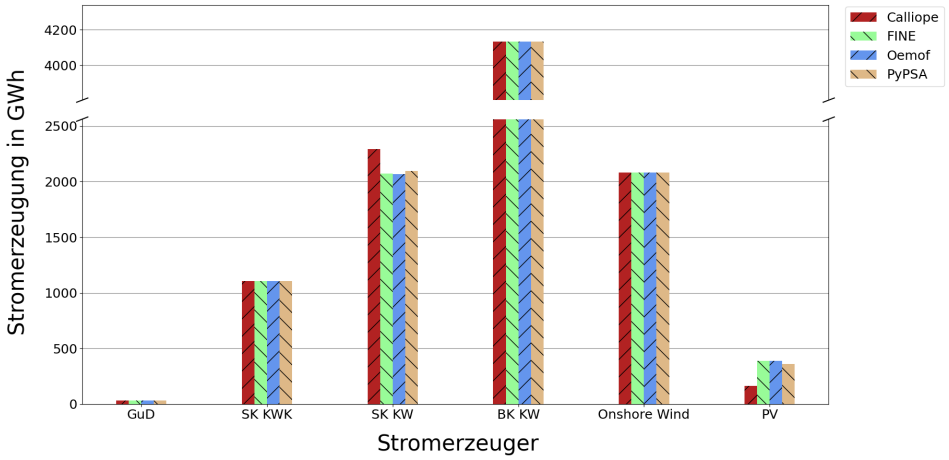

Abbildung 4.7.: Stromerzeugung des „Commitment Examples“

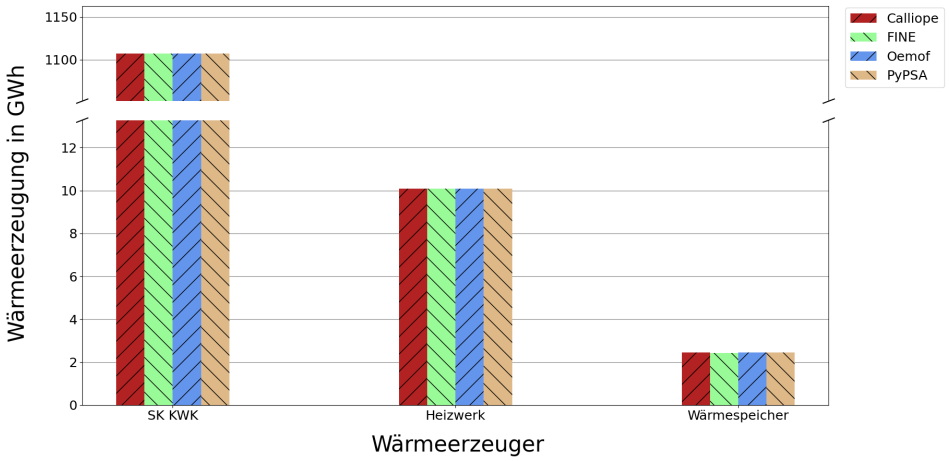

Abbildung 4.8.: Wärmeerzeugung des „Commitment Examples“

Calliope ist jedoch das einzige Tool, welches eine klare Priorisierung vornimmt. Und zwar eine Priorisierung des emissionsreichen Steinkohlekraftwerks. So wird in *Calliope* die PV-Anlage immer nur dann genutzt, wenn das Steinkohlekraftwerk bereits auf Volllast gefahren wird. Dies ist in 1280 Zeitschritten der Fall, von welchen in 39 Fällen gar eine erhöhte Nutzung der PV-Anlage ein Abschalten des Steinkohlekraftwerks erlauben würde. In keinem einzigen Zeitschritt liegt eine Nutzung der PV-Anlage vor, ohne dass das Steinkohlekraftwerk voll ausgelastet ist. In den anderen Tools existieren Zeitschritte in welchen die PV-Anlage auf Volllast gefahren wird und das Steinkohlekraftwerk in Teillast, jedoch eine Erhöhung der Leistung des Steinkohlekraftwerks ein Abschalten der PV-Anlage ermöglichen würde, und vice versa (vgl. Abb. 4.9).

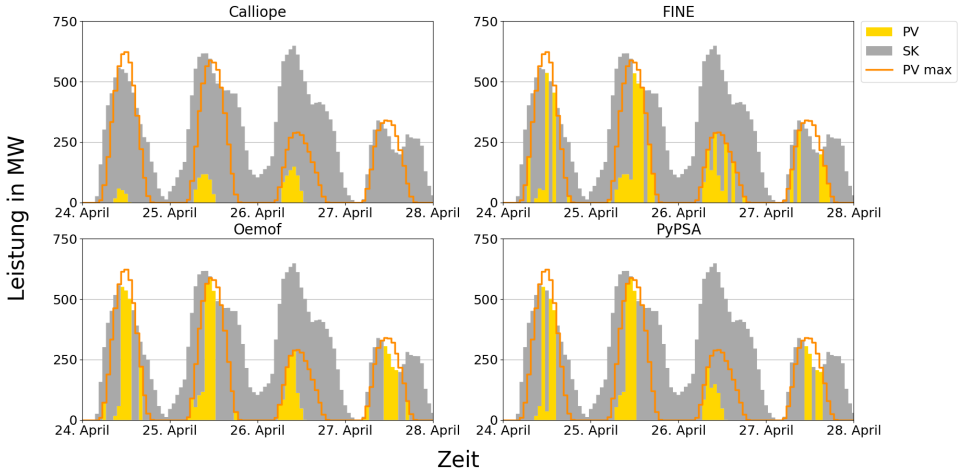

Abbildung 4.9.: Vergleich der Stromerzeugung durch Steinkohle und Photovoltaik des „Commitment Examples“ (in Anlehnung an [55])

Ein weiterer Test zeigt, dass die Priorisierung von *Calliope* unter anderem in der Bezeichnung der Komponenten sowie weiteren unbekannten Einflüssen begründet ist. So ist durch das Anpassen der Bezeichnung der PV-Anlage von „Solar Panel“ zu „A Solar Panel“ zu erkennen, dass diese nun gegenüber der „Hard Coal PP“ Komponente zum Großteil priorisiert wird. Die Emissionen belaufen sich in dieser Optimierung auf 6.555.940 Tonnen bei unveränderten Kosten. Die Optimierungsergebnisse der anderen Tools verändern sich im Vergleich zu Tabelle A.2 ebenfalls innerhalb der Emissionen, wobei, wie aus Abbildung 4.10 hervorgeht, auch in dieser Optimierung kein Schema der Auswahl der Komponente zu erkennen ist.

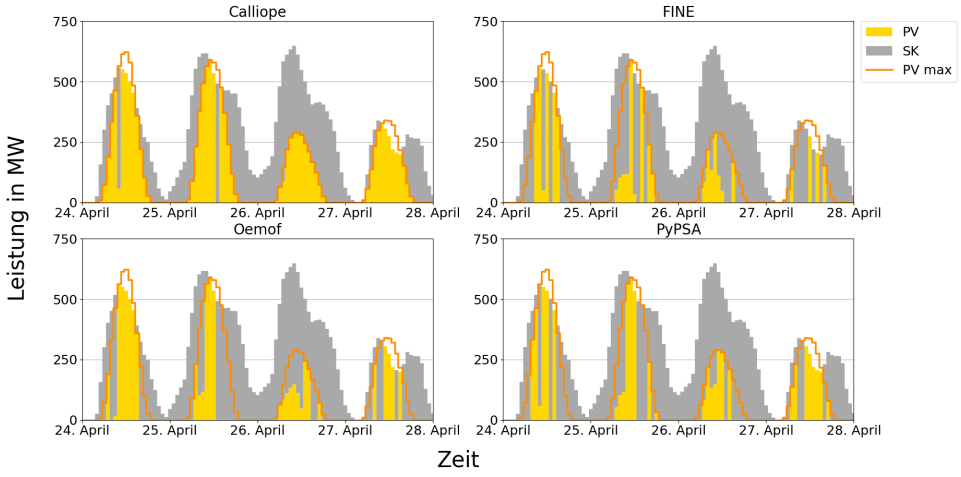

Abbildung 4.10.: Vergleich der Stromerzeugung durch Steinkohle und Photovoltaik des „Commitment Examples“ mit angepasster Komponentenbezeichnung (in Anlehnung an [55])

Die Art und Weise der Bevorzugung einer Komponente gegenüber einer kostengleichen ist durchaus interessant. Denn diese Priorisierung in *Calliope* ist nicht konsequent. Während in der ursprünglichen Optimierung nicht ein einziger Zeitschritt existiert, in welchem die PV-Anlage Leistung bereitstellt, welche die Steinkohleanlage hätte bereit stellen können, existieren in der Optimierung mit einer angepassten Bezeichnung der PV-Anlage 317 Zeitschritte, in welchen dennoch das Steinkohekraftwerk bevorzugt wird. Von diesen sind drei Zeitschritte in Abbildung 4.10 ersichtlich, wobei in einem eine ausschließliche Nutzung der PV-Anlage ein Abschalten des Steinkohlekraftwerks zur Folge hätte. Insgesamt existieren sechs Zeitschritte in welchen eine erhöhte Nutzung der PV-Anlage ein Abschalten des Steinkohekraftwerks ermöglichen würde. Somit scheint *Calliope* bei Komponenten mit gleichen Kosten die Auswahl der Leistungsbereitstellung überwiegend wie folgt zu wählen:

1. Ausschöpfen des Leistungsvermögens einer Komponente ehe die zweite Aktiv wird.
2. Mindestens ein unbekannter Einfluss.
3. Verwendung der Komponenten in alphabetischer Reihenfolge.

Nichtsdestotrotz zeigen mehrere Optimierungen mit angepassten Bezeichnungen, dass die Bezeichnung durchaus bei kostengleichen Komponenten in allen Tools, wenn auch in verschiedenem Ausmaß, einen Einfluss haben. Beispielsweise sind in einem Test mit der Bezeichnung „Z Solar Panel“ alle Ergebnisse der Tools unverändert zu der ursprünglichen Optimierung, während die Bezeichnung „A Solar Panel“ in allen Tools geringe und *Calliope* gar große Änderungen zur Folge hat. Eine zufällige Auswahl der Leistungsbereitstellung bei Kostengleichheit ist ohnehin auszuschließen, da im Zuge der Analyse der Optimierungsdauer drei Optimierungen durchgeführt sind und die Ergebnisse allesamt in jedem Durchlauf identisch ausfallen.

Einen deutlichen Unterschied stellt zudem die Optimierungsdauer dar. Diese ist aufgeführt in Tabelle 4.16. Sowohl die Modellumwandlung, als auch die Optimierung und die Aufbereitung der Ergebnisse benötigen mit *Calliope* am meisten Zeit. Ein Teil der für die Modellumwandlung benötigten Zeit ist aller Voraussicht nach der Tatsache geschuldet, dass das Modell in Form von `YAML` und `CSV` Dateien gespeichert wird. Der Vorteil, diese Dateien nutzen zu können um *Calliope* spezifische Anpassungen vorzunehmen, wird an dieser Stelle jedoch höher gewichtetet als der geringe Mehraufwand. Zumal das Schreiben dieser Dateien für kleinere Modelle deutlich schneller abläuft (vgl. Tab. 4.4) und bei größeren Modellen die Optimierungsdauer den Großteil der benötigten Zeit einnimmt.

Die Optimierung benötigt in *Calliope* merklich mehr Zeit als in den anderen Tools. Die Vermutung liegt nahe, dass der Grund hierfür in der Darstellung jeder Komponente als eigener Standort mit einer Technologie liegt. Dies sorgt mit den *Transmission* Technologien, welche die einzelnen Standorte verbinden, für ein komplexeres System mit mehr Parametern, als es in den restlichen Tools der Fall ist. So wird beispielsweise die optimale Kapazität jeder Technologie, aber auch jeder *Tranmission*

Tabelle 4.16.: Zeiten des „Commitment Examples“ in Sekunden (gerundet auf 2 Nachkommastellen)

	Calliope	*FINE*	*Oemof*	*PyPSA*
Transformation	8,19	0,39	0,22	2,76
Optimization	394,73	65,58	39,51	42,38
Post Processing	7,31	4,17	3,97	3,06
Summe	410,23	70,14	43,70	48,20

und jeder Kombination aus Standort und Technologie ermittelt und abgespeichert. Dies führt zu 72 Kapazitäten in *Calliope*, während lediglich 24 Komponenten im *Tessif*-Modell vorliegen.

Um diese These bestätigen oder widerlegen zu können, werden die `YAML` Dateien so angepasst, dass alle Technologien in einem Standort vorliegen und die Ergebnisse sowie die Optimierungsdauer mit den hier vorliegenden verglichen. Die Darstellung in einem einzelnen Standort ist für dieses Modell möglich, da kein Konnektor vorliegt und zudem keine zwei (oder mehr) Komponenten vorliegen, welche den gleichen Energieträger erzeugen und diesen jedoch an verschiedene Energienetze übergeben. Somit ist ein verlustfreier Energieaustausch aller Komponenten gleichwertig. Für die Auswertung der Ergebnisse ist es nicht möglich *Tessif* zu nutzen, da die `es2mapping` Funktion auf die Darstellung der Komponenten in einzelnen Standorten angewiesen ist. Die Ergebnisse dieser Untersuchung werden in Abschnitt 4.6 diskutiert.

Die Dauer der Ergebnisaufbereitung ist ebenfalls in *Calliope* am höchsten. Ein Grund hierfür ist erneut die Darstellung jeder Komponente an einem separaten Standort. Dies erzeugt in den *Calliope* Ergebnissen deutlich mehr Daten und sorgt dementsprechend in der Aufbereitung dieser Daten für größeren Aufwand sobald Iterationen notwendig sind. Darüber hinaus führt die im Abschnitt 3.2 aufgeführte Umbenennung der Bezeichnung der Komponenten dazu, dass selbst wenn alle Bezeichnungen in *Calliope* jenen in *Tessif* gleichen, aufgrund der erforderten automatischen Handhabung eines jeden Modells, ein Abgleich der Bezeichnungen vorgenommen wird.

4.5.2. Expansion Example

Vor der Optimierung des „Expansion Examples“ ist eine Anpassung für die Optimierung mit *PyPSA* vorzunehmen, in welcher die Emissionen der KWK Anlagen in die davor liegende Energiequelle verschoben werden. Dies beruht auf der Tatsache dass Emissionen innerhalb *PyPSAs* nur bei Erzeugern und Speichern berücksichtigt werden [26]. Für die weiteren Energiewandler ist dies nicht vonnöten, da im `es2es` für *PyPSA* automatisch Energiewandler, welche nur einen Energieträger erzeugen und aus einer Quelle gespeist werden, als Quellen dargestellt werden [21].

Die Emissionen der Power-to-Heat Anlage werden jedoch von *PyPSA* intern ignoriert, während sie jedoch im Modell vorhanden und auch von *Tessif* ausgelesen werden. Darüber hinaus liegt eine grundlegend verschiedene Definition der Emissionsberechnung bei Speichern im Vergleich zu *Tessif* vor. Während die Speicher in *Tessif* auschließlich während des Entladens emittieren, wird in *PyPSA* beim Laden CO_2 aufgenommen und beim Entladen wieder abgeben. Dies hat zur Folge dass lediglich der initiale Speicherstand mit dem finalen Speicherstand zu vergleichen ist, um die Emission, welche dementsprechend negativ ausfallen können, zu ermitteln [26]. Die daraus resultierenden Abweichungen sind von Reimer [55] und Schnute [23] genauer untersucht worden.

Abbildung 4.11 zeigt, dass die Optimierungsergebnisse über *Calliope*, *FINE* und *Oemof* nur minimal voneinander abweichen. Die größeren Differenzen zu den Ergebnissen von *PyPSA* beruhen auf der zuvor aufgeführten abweichenden Definition der Emissionen. Das geforderte Emissionslimit wird von *Calliope*, *FINE* und *Oemof* eingehalten (vgl. Anhang A.3), woraus zu schließen ist, dass diese Tools die Emissionen ebenso wie *Tessif* ermitteln.

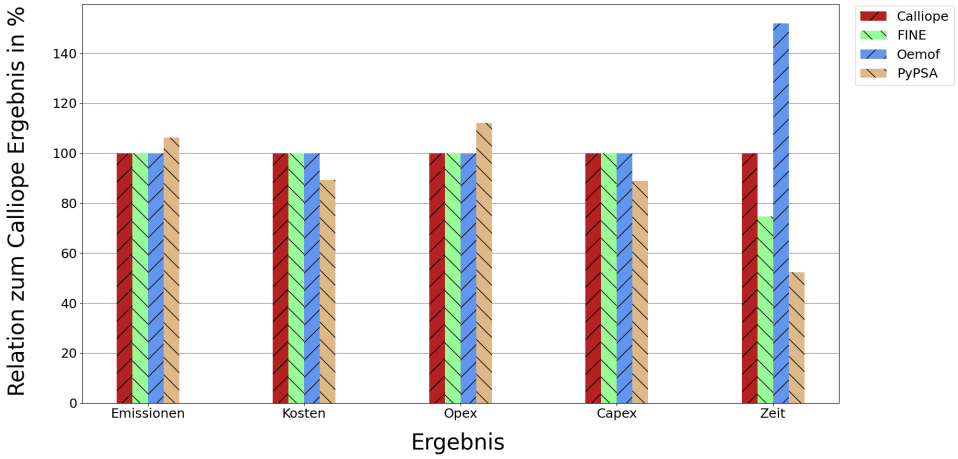

Abbildung 4.11.: Gesamtergebnis des „Expansion Examples“ der verschiedenen Tools in Relation zum Ergebnis von *Calliope*

Dass die Unterschiede der Ergebnisse von *Calliope*, *FINE* und *Oemof* vernachlässigbar gering ausfallen zeigt auch die Strom- bzw. Wärmeerzeugung der einzelnen Komponenten in Abbildung 4.12 und 4.13. Aufgeführt sind, wie in Abschnitt 4.5.1, nur jene Komponenten welche einen Teil zur Strom- bzw. Wärmeerzeugung beitragen. Erneut weicht nur *PyPSA* von den restlichen Tools nennenswert ab.

Die Leistung der einzelnen Komponenten wird ebenfalls in allen Tools außer *PyPSA* nahezu identisch ausgebaut. Die optimale Leistung aller Komponenten ist in Abbildung 4.14, für jene Komponenten deren optimale Leistung von der zu Beginn installierten Leistung abweicht, dargestellt. Aufgrund der Definition der Speicheremissionen in *PyPSA* ist ein erhöhter Nutzen und Ausbau der Batterie im Vergleich

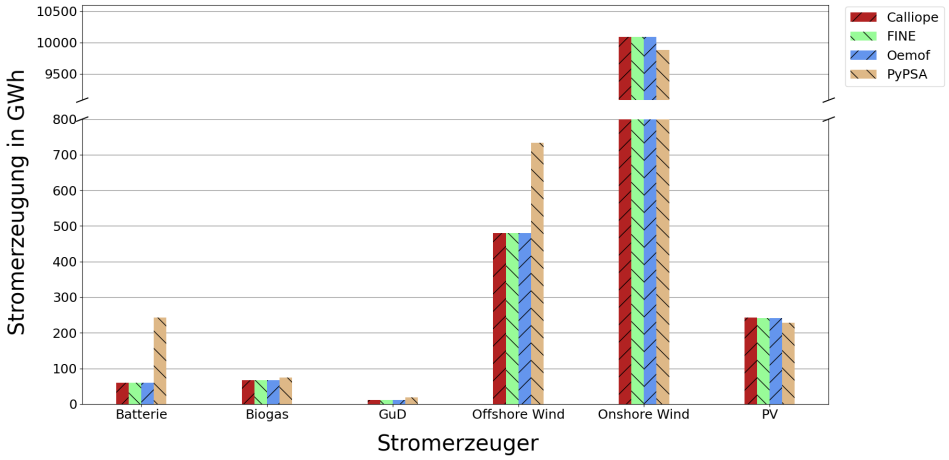

Abbildung 4.12.: Stromerzeugung des „Expansion Examples“

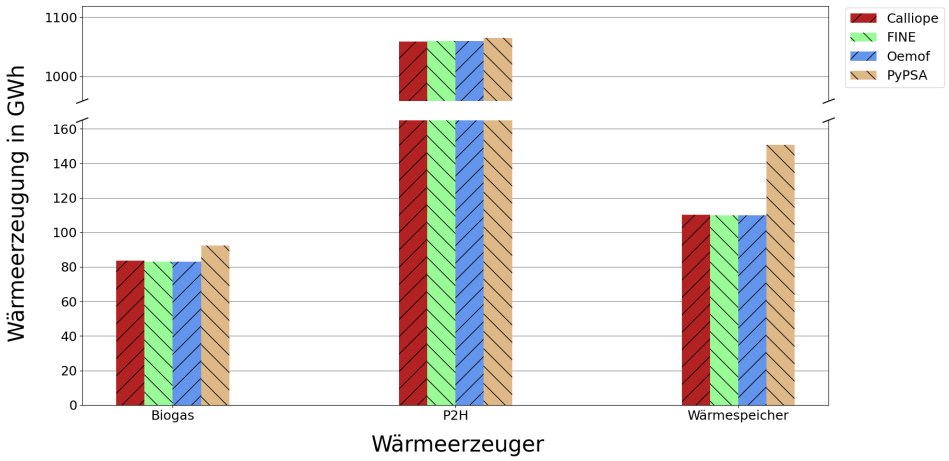

Abbildung 4.13.: Wärmeerzeugung des „Expansion Examples“

zu den anderen Tools möglich, welcher einen geringeren Ausbau der meisten anderen Komponenten erlaubt.

Bei der Betrachtung der Optimierungsdauern der einzelnen Tools in Tabelle 4.17 zeigt sich, dass *Calliope* bei der Optimierung dieses Modells nicht das langsamste Tool ist. *Oemof*, das beim „Commitment Example“ das schnellste Tool ist (vgl. Tab. 4.16), benötigt für die Optimierung des „Expansion Example“ am meisten Zeit, während *PyPSA* am wenigsten Zeit benötigt. Die geringe Optimierungszeit von *PyPSA* ist jedoch vor allem dem geschuldet dass aufgrund der abweichenden bzw. fehlenden Berücksichtigung der Emissionen einiger Komponenten ein im Vergleich zu den anderen Tools vereinfachtes Modell entsteht [55].

Da die vereinfachte Abbildung der Speicheremissionen sowie das Vernachlässigen der Emissionen der Power-to-Heat Anlage in *PyPSA* zu einem weniger komplexen Problem führt, bietet sich eine Anpassungen dieser Parameter des Modells an, um

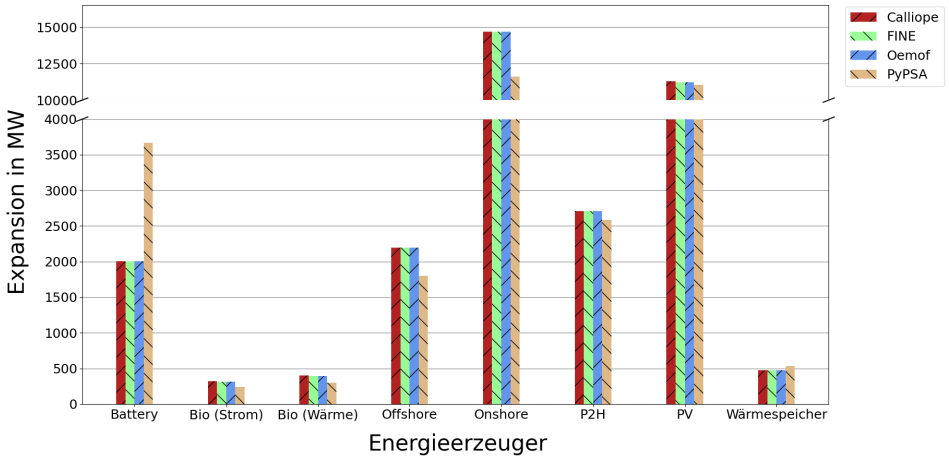

Abbildung 4.14.: Leistungsausbau des „Expansion Examples“

Tabelle 4.17.: Zeiten des „Expansion Examples“ in Sekunden (gerundet auf 2 Nachkommastellen)

	Calliope	*FINE*	*Oemof*	*PyPSA*
Transformation	8,42	0,38	0,21	2,75
Optimization	967,03	715,81	1459,80	499,41
Post Processing	7,59	4,22	4,03	3,19
Summe	983,04	720,41	1464,04	505,35

eine bessere Vergleichbarkeit der Optimierungszeiten zu gewinnen [55]. Die Emissionen dieser Komponenten werden somit auf Null gesetzt, wodurch das Modell soweit angepasst ist, dass auch *PyPSA* die Emissionen des Gesamtsystems ebenso wie alle anderen Tools betrachtet.

Das Ergebnis dieser Optimierung ist in Abbildung 4.15, bzw. genauer ausgeführt im Anhang in Tabelle A.4, dargestellt.

Abbildung 4.15 zeigt, dass die Ergebnisse der Optimierung aller Tools übereinstimmen. Nach der Anpassung der Emissionen ist *Oemof* das schnellste Tool. Entgegen der Erwartungen liegt die Optimierungsdauer von *Calliope* und *FINE* in dieser angepassten Variante des Modells höher als die ursprüngliche Variante (vgl. Tabelle A.3 und A.4). Eine Erklärung hierfür ergibt sich aus den Ergebnissen jedoch nicht. Aufgrund dessen, dass die Optimierungen drei mal durchgeführt wurden, um Unsicherheiten aufgrund von Hintergrundprozessen zu reduzieren, ist davon auszugehen, dass diese Optimierungszeiten keine zufälligen Abweichungen darstellen. Der vermutete Vorteil von *PyPSA* in der besseren Optimierungszeit, aufgrund eines vereinfachten Gesamtsystems in Tabelle 4.17, besteht demnach lediglich im Vergleich zu *Oemof*.

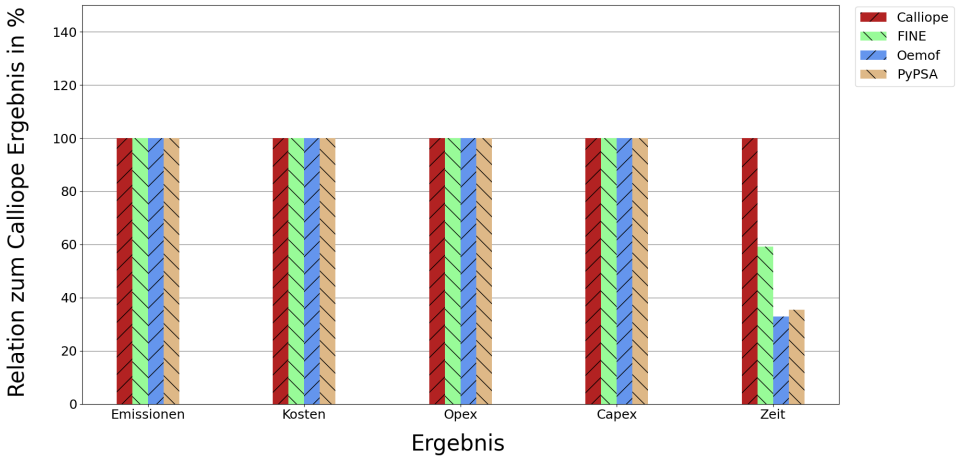

Abbildung 4.15.: Gesamtergebnis des „Expansion Examples“ mit angepassten Emissionen in Relation zum Ergebnis von *Calliope*

4.5.3. Untersuchung der Zeitreihenaggregation von *Calliope*

Das Referenzmodell wird in diesem Abschnitt über eine spezielle *Calliope* Funktion mit einer Zeitreihenaggregation versehen und optimiert. Von Zeitschritten von einer Stunde bis hin zu 24 Stunden, werden alle Schrittgrößen verwendet durch welche sich die gesamte Anzahl von 8760 Zeitschritten zu ganzen Zahlen teilen lässt. Die Rechenzeit zum Erstellen des „Commitment Examples“, welche die Zeit zum Aggregieren der Daten beinhaltet, sowie die Optimierungsdauer sind in Abbildung 4.16 neben der Anzahl an Zeitschritten dargestellt. Die Anzahl der Zeitschritte, welche in orange aufgetragen ist, ist hierbei rechts abzulesen, während die Optimierungszeit (blau) links abzulesen ist.

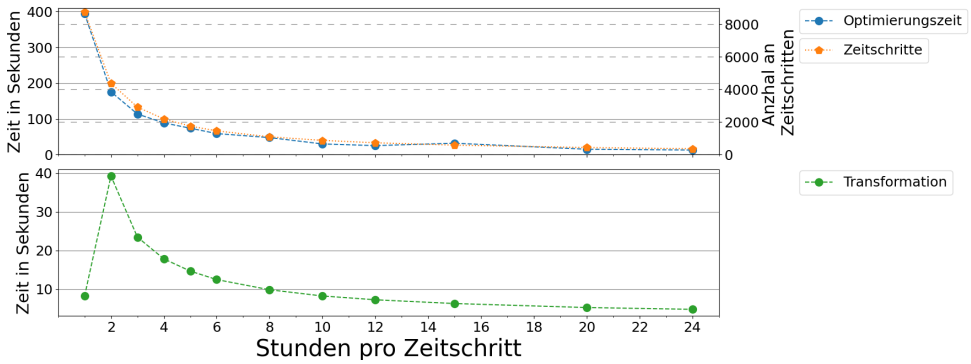

Abbildung 4.16.: Vergleich der Rechenzeiten des „Commitment Examples“ bei verschiedenen Zeitreihenaggregationen

Die Optimierungsdauer zeigt einen eindeutigen Abwärtstrend bei steigender Schrittweite, welche in etwa analog zu der Anzahl an zu optimierenden Zeitschritten

verläuft. Die Rechenzeit zum Erstellen des Modells verläuft ebenfalls vergleichbar. Die Dauer wenn keine Aggregation vorgenommen wird weicht jedoch von dem grundsätzlichen Verlauf ab. Dies ergibt sich daraus, dass dem Modell in allen Optimierungen die gesamte Datenlage zu Grunde liegt und dementsprechend keine Anpassungen vonnöten sind, wenn keine Aggregation vorgenommen wird. Dem gleichen Verlauf folgen zudem die Rechenzeiten des mit einer Aggregation versehenen „Expansion Examples“ (vgl. Abbildung 4.17).

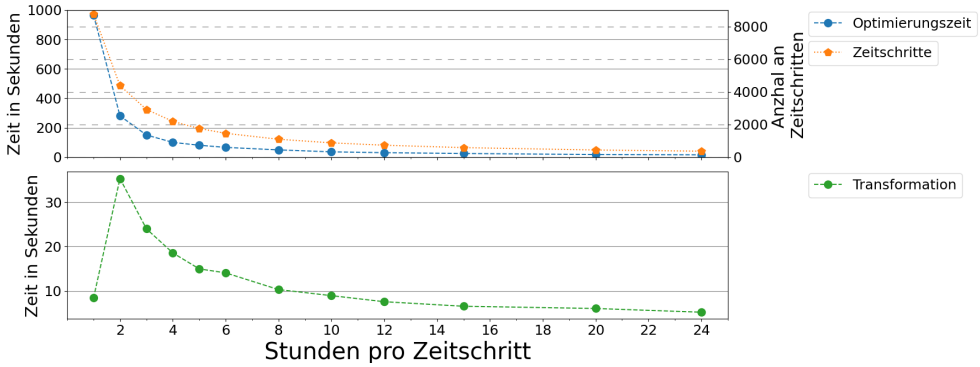

Abbildung 4.17.: Vergleich der Rechenzeiten des „Expansion Examples“ bei verschiedenen Zeitreihenaggregationen

Die minimalen Kosten der jeweiligen Optimierungen des „Commitment Examples“ sind in Abbildung 4.18, neben den entstehenden Emissionen, dargestellt. Neben *Calliope* ist ebenfalls eine Aggregation der Zeitreihen mit *FINE* über *Tessif* möglich, welche repräsentative Perioden identifiziert und aggregiert. Für *FINE* wurde diese Option anhand des gleichen Modells von Schnute [23] untersucht. Dennoch werden diese Optimierungen erneut durchgeführt, sodass alle Ergebnisse mit dem gleichen Rechner ermittelt sind und somit die Vergleichbarkeit gewährleistet ist. Das Ergebnis von *FINE* ist in Abbildung 4.18 rechts aufgetragen.

Die Kosten der Optimierung mit *Calliope* fallen mit zunehmender Aggregation. Eine geringere Optimierungsdauer sowie eine Unterschätzung der Kosten bei einer Aggregation von Optimierungsdaten stimmt mit den Erwatungen überein [45]. Die Emissionen hingegen steigen an. Dies ist aller Voraussicht nach auf die reduzierten Spitzenlasten sowie den erhöhten minimalen Lasten bei steigender Aggregation zurückzuführen (vgl. Abb. A.3 und A.4). Diese ermöglichen eine erhöhte Nutzung der kostengünstigeren Erzeuger, die zugleich mehr emittieren (vgl. Tab. A.1).

Dass die Gesamtkosten auch bei relativ starker Aggregation, mit etwa 0,25 % Abweichung, sehr nahe am Optimum liegen, ist vor allem auf das Modell und dessen Parametrisierung zurückzuführen. Dieses beinhaltet keine Begrenzung der Lastwechsel, welche vor allem bei kleineren Schrittweiten interessanter werden, sowie keine detaillierte Netzstruktur. Die Aggregation mit *FINE* führt zu einem ähnlichen Kostenoptimum, jedoch zu deutlich geringeren Emissionen. Zudem ist anzumerken, dass die Optimierung mit einer Aggregation der Zeitreihen über *FINE* zu

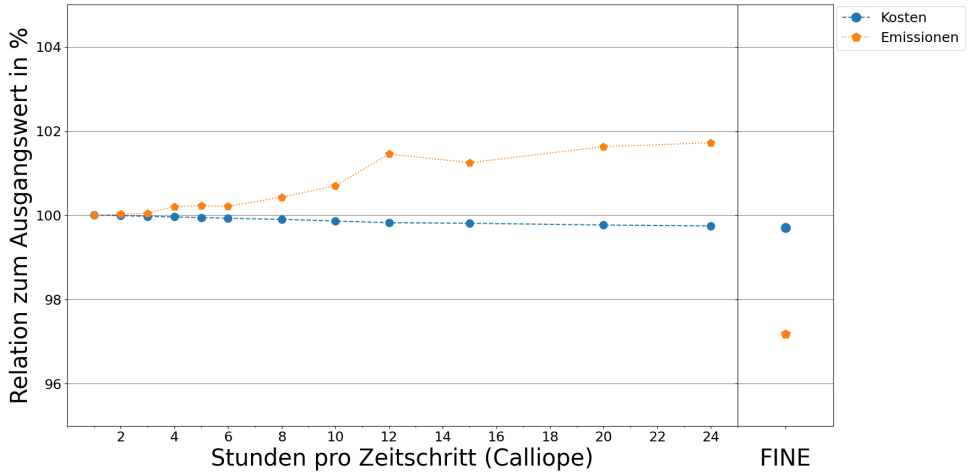

Abbildung 4.18.: Vergleich der Kosten und Emissionen des „Commitment Examples“ bei verschiedenen Zeitreihenaggregationen im Verhältnis zum Ergebnis ohne Aggregation

einer Optimierungsdauer von 4 Sekunden führt, während die Optimierungsdauer von *Calliope* bei einer Schrittweite von 24 Stunden bei 13 Sekunden liegt.

Bei einer Aggregation der Zeitreihen des „Expansion Examples“ fällt die Kostenabweichung deutlich größer aus. Dies zeigt Abbildung 4.19. Das Ergebnis durch *FINE* ist erneut rechts aufgetragen und die Ergebnisse *Calliope*'s links. Aufgrund des vorgegebenen Emissionslimits liegen keine nennenswerten Abweichungen in den entstehenden Emissionen vor. Die Unterschätzung der Gesamtkosten durch *Calliope* ist in diesem Beispiel vor allem auf die reduzierten maximalen Lasten zurückzuführen (vgl. Abb. A.5 und A.6) welche einen geringeren Leistungsausbau ermöglichen. So sind die Kosten bei starker Aggregation nur noch etwa zwei Drittel der ursprünglich ermittelten Kosten.

Die Kosten der Optimierung mit *FINE* liegen zwischen den *Calliope* Ergebnissen der Schrittweiten 10 und 12. Für diese Optimierungen benötigt *Calliope* jedoch eine Rechenzeit von 36 bzw. 30 Sekunden, während *FINE* die Optimierung in 5 Sekunden durchführt.

Insgesamt lässt sich festhalten, dass eine Verbesserung der Optimierungszeiten, auf Kosten der zeitlichen Auflösung und einer damit einhergehenden Unterschätzung des ermittelten Optimums, sowohl mit *Calliope* als auch mit *FINE*, über *Tessif* möglich ist. *FINE* liefert jedoch, vor allem beim „Expansion Example“, sowohl schnellere als auch bessere Ergebnisse. Dieses Zwischenfazit bezieht sich jedoch ausschließlich auf die integrierte Variante der Aggregation von *Calliope* in *Tessif*. Wie in Abschnitt 3.1 aufgeführt, existieren in *Calliope* weitere Verfahren zur Aggregation der Zeitreihen, welche nicht in *Tessif* implementiert sind. Inwieweit diese andere Ergebnisse liefern wird im Rahmen dieser Arbeit nicht untersucht.

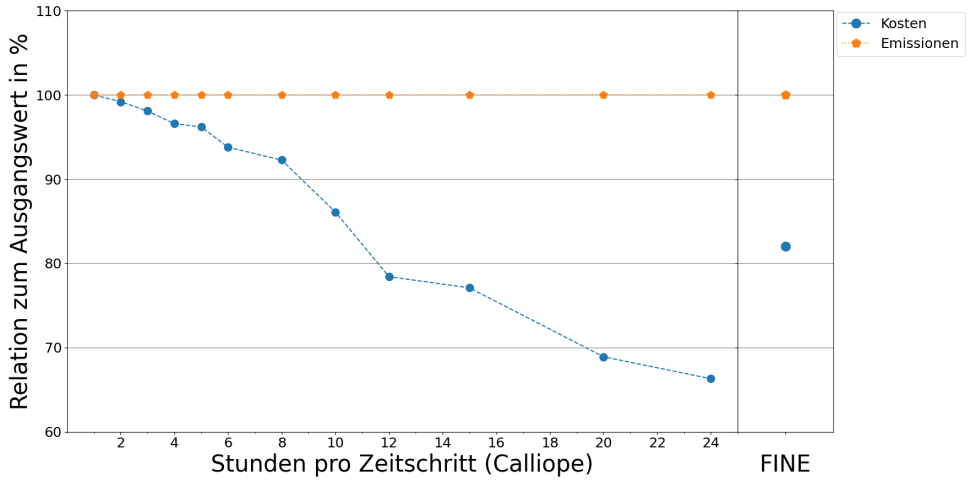

Abbildung 4.19.: Vergleich der Kosten und Emissionen des „Expansion Examples“ bei verschiedenen Zeitreihenaggregationen im Verhältnis zum Ergebnis ohne Aggregation

4.6. Vergleich des Referenzmodells über *Tessif* und nativem *Calliope*

Bei der Darstellung jeder Technologie in einem eigenen Standort liegt die Vermutung nahe, dass dies zu einem komplexeren System führt, als eines in welchem alle Technologien an einem Standort vorliegen. Aufgrund der erforderten Automatisierbarkeit jedes beliebigen *Tessif*-Modells ist die Nutzung einzelner Standorte für jede Technologie jedoch notwendig, da ansonsten unter Umständen Komponenten in einem Energieaustausch stehen, welche nicht in einem direkten Energieaustausch stehen sollten.

In diesem Abschnitt wird ein Test durchgeführt, in welchem die Optimierungsdauer eines manuell angepassten *Calliope*-Modells, welches nicht dieser Konvention folgt, mit einem *Calliope*-Modell welches aus einem *Tessif*-Modell erstellt wurde verglichen werden. Anhand des Vergleichs gilt es abzuschätzen, inwieweit die Vielzahl an Standorten die Optimierungsdauer beeinflusst.

Zum Vergleich dient das in Abschnitt 4.5 untersuchte Modell in beiden untersuchten Varianten. In den manuell angepassten Versionen sind alle *Transmission* Technologien und alle *Links* aus den `YAML` Dateien entfernt. Darüber hinaus werden alle Technologien in einer *Location* aufgerufen und alle weiteren *Locations* entfernt. Diese angepassten Modelle werden mit *Calliope* unabhängig von *Tessif* optimiert und ausgewertet.

Die Ergebnisse dieser Optimierung sowie die Optimierungsdauern sind den Ergebnissen aus Abschnitt 4.5 in Tabelle 4.18 gegenüber gestellt. Die Reduzierung der Standorte auf einen Standort für alle Technologien führt demnach beinahe zu

einer Halbierung der Optimierungsdauer beim „Commitment Example“, während das gefundene Optimum das gleiche bleibt. Ein Vergleich zu den weiteren Tools aus Tabelle 4.16 zeigt jedoch, dass selbst die Darstellung aller Technologien in einem Standort zeitaufwendiger ist als die Optimierung über ein anderes Tool.

Tabelle 4.18.: Vergleich der Optimierung des Referenzmodells über *Tessif* und nativem *Calliope* (gerundet auf ganze Zahlen)

	„Commitment Example“		„Expansion Example“	
	Calliope ohne *Tessif*	*Calliope* über *Tessif*	*Calliope* ohne *Tessif*	*Calliope* über *Tessif*
Kosten [10^3 €]	688.509	688.509	49.922.846[a]	42.289.121
Zeit [s]	207	395	1.656	967

[a] In diesem Wert sind die Kosten des Leistungsausbaus bis zur installierten Leistung vorhanden, welche in der Aufbereitung der Ergebnisse in *Tessif* herausgerechnet werden.

Die Optimierungsdauer des „Expansion Examples“ hingegen liegt in der Darstellung aller Technologien in einem Standort höher als jene einer Optimierung über *Tessif*. Dies widerlegt die zuvor aufgestellte These, dass ein über *Tessif* erstelltes Modell aufgrund der Darstellung jeder Technologie in einem eigenen Standort zwangsläufig zu einem komplexeren System führt. Das von *Calliope* ermittelte Kostenoptimum wird von *Tessif* im `es2mapping` um die Kosten welche *Calliope* zum Ausbau bis zur installierten Leistung ermittelt reduziert (vgl. Abschnitt 3.2). Diese Kosten belaufen sich auf $7.633.725 \cdot 10^3$ € und lassen sich anhand von Tabelle A.1 durch Multiplikation der Anlagenkosten mit den jeweiligen installierten Leistungen bzw. Kapazitäten ermitteln. Werden diese vom nativen *Calliope* Ergebnis des „Expansion Examples“ abgezogen, so zeigt sich, dass die Kosten mit der Optimierung über *Tessif* übereinstimmen.

Letztlich lässt sich festhalten dass aus einem *Tessif*-Modell nicht grundsätzlich ein komplexeres *Calliope*-Modell erstellt wird als notwendig. Eine Erklärung für die abweichenden Optimierungsdauern bedarf voraussichtlich eine tiefere Einarbeitung in die Programmierung *Calliope*'s, welche im Rahmen dieser Arbeit in dieser Tiefe nicht durchgeführt wird.

5. Auswertung

In diesem Kapitel werden die bisher gewonnenen Erkenntnisse zusammengefasst und ausgewertet. Zunächst beginnend mit der Auswertung der Integration von *Calliope* in *Tessif*. Daraufhin erfolgt ein Vergleich der in *Tessif* implementierten Softwaretools anhand der erzielten Optimierungsergebnisse innerhalb *Tessifs*. Zuletzt werden einige Grenzen des einheitlichen Vergleichs der Tools über *Tessif* aufgeführt.

Allgemein ist die Qualität eines Modells schwierig zu bewerten, da es keinen Standard gibt [4]. Jedoch haben alle Modelle spezifische Vor- und Nachteile [4]. Dies ist auf die Modellierungstools übertragbar. Dementsprechend stellt sich im Folgenden weniger die Frage nach dem besten oder schlechtesten Tool, sondern vielmehr die Frage danach, unter welchen Voraussetzungen welches Tool entsprechende Vor- und Nachteile aufweist.

5.1. *Calliope* Integration in *Tessif*

Die Ergebnisse aus Kapitel 4 zeigen, dass die Integration von *Calliope* in *Tessif* gelungen ist. *Tessif*-Modelle lassen sich, analog zu den anderen integrierten Tools *FINE*, *Oemof* und *PyPSA*, mit *Calliope* untersuchen und, dank des automatischen Abspeicherns der Modelldaten in `YAML` und `CSV` Dateien, schnell und einfach mit *Calliope* spezifischen Anpassungen versehen. Die Vielzahl untersuchter Modelle aus der *Tessif* Bibliothek zeigen, dass mit *Calliope* sehr ähnliche bis gleiche Ergebnisse im Vergleich zu den anderen Tools erzielt werden. Der Ursprung auftretender Unterschiede konnte zudem stets identifiziert werden.

Bei der Definition der Parameter eines Energiespeichers liegen *Tessif* und *Calliope* jedoch weit auseinander. *Calliope*, ebenso wie *FINE*, geht in dem gewählten „plan“ Modus zudem davon aus dass in einem zu optimierenden Modell vorab keine Leistungen und Kapazitäten installiert sind. Um diese dennoch in *Calliope* abbilden zu können wird daher auf den in *Tessif* vorgegeben minimalen Leistungsausbau verzichtet. Kosten für den Lastwechsel einer Komponente lassen sich in *Calliope* ebenfalls nicht abbilden. Diese sind jedoch in der *Tessif* Integration von *FINE* und *PyPSA* ebenso wenig berücksichtigt und lassen sich folglich nur in *Oemof* abbilden, welches jenes Tool ist, in dessen Anlehnung das *Tessif*-Modell entwickelt wurde [21].

Darüber hinaus lassen sich die in *Tessif* definierten MILP Parameter allesamt nicht in *Calliope* abbilden. *Calliope* verfügt zwar über eine Reihe von MILP Parametern, jedoch gibt es zusammen mit *Tessif* keine Gemeinsamkeiten bezüglich dieser Parameter. Aufgrund dessen, dass *Tessif* ausschließlich FOSS Tools vergleicht und dementsprechend auch die Nutzung eines kostenfreien Solvers vorsieht, ist die fehlende Darstellung von MILP Problemen kein allzu großer Verlust in der *Tessif* Integration. Denn diese kostenfreien Solver führen gerade in Kombination mit MILP Problemen zu sehr hohen Optimierungsdauern, weshalb die Tools bei der Optimierung von MILP Problemen eine Warnung ausgeben, dass die Nutzung eines kommerziellen Solvers zu empfehlen ist.

Zudem zeigte die Auswahl eines geeigneten Tools in Abschnitt 2.3, dass bei der Modellierung und Optimierung von Energiesystemen LP Probleme üblich sind [16] und einige Tools grundsätzlich keine MILP Probleme darstellen können [5].

Dank des Ansatzes das *Calliope*-Modell in Form von `YAML` und `CSV` Dateien zu speichern, ist es problemlos möglich ein von *Tessif* transformiertes Modell mit *Calliope* spezifischen Parametern oder beispielsweise der „Spores“ Option zu erweitern und zu untersuchen. Hierbei ist jedoch zu beachten, dass eine Auswertung der Ergebnisse möglicherweise nicht mehr über *Tessif* möglich ist.

Die in *Tessif* formulierten Funktionen zum Übersetzen des *Tessif*-Modells in ein *Calliope*-Modell sowie der Aufbereitung der Optimierungsergebnisse beanspruchen mehr Zeit als jene für die weiteren in *Tessif* implementierten Tools. Die zusätzlich benötigte Zeit liegt bei kleinen Modellen in der Regel unter einer Sekunde. Bei komplexeren Modellen steigt diese auf wenige Sekunden an, fällt hierbei jedoch im Verhältnis zur Optimierungsdauer kaum ins Gewicht, weshalb dies kein allzu großen Makel darstellt.

Ob die Darstellung jeder Technologie in einem Standort für *Calliope* zu einem komplexeren Modell führt als notwendig, ist nicht eindeutig festzustellen. Dies wäre zwar anzunehmen, da die Standorte über *Transmission* Technologien miteinander verbunden werden, für welche *Calliope* ebenfalls Ergebnisse, wie eine optimale Kapazität, ermittelt und abspeichert. Jedoch zeigt die Untersuchung in Abschnitt 4.6, dass die Optimierung des „Expansion Examples“ in einer Darstellung aller Technologien in einem Standort zu einer erhöhten Optimierungsdauer führt, während die Optimierung des „Commitment Examples“ eine geringere Optimierungsdauer zur Folge hat, wenn nur ein Standort verwendet wird.

Mithilfe von einer Zeitreihenaggregation kann zudem die Optimierungszeit deutlich verbessert werden. Dies erfolgt zwar auf Kosten eines abweichenden Optimums, welches voraussichtlich von der Komplexität des Gesamtsystems abhängt, kann jedoch einen ersten Eindruck bezüglich der Größenordnung der Kosten und Emissionen liefern.

Auch wenn eine Vielzahl von Modellen mit verschiedensten Komponenten und Parametern im Zuge dieser Arbeit getestet wurden, ist nicht auszuschließen dass Parameterkombinationen existieren, welche Unstimmigkeiten in der *Calliope* Integration hervorrufen.

Inwieweit die Wahl eines anderen Tools zur Integration in *Tessif* besser oder schlechter ausgefallen wäre, lässt sich nicht feststellen. Insgesamt ist jedoch festzuhalten, dass *Calliope* eine für die Integration in *Tessif* geeignete Wahl darstellt. So handelt es sich bei *Calliope* um ein Free and Open Source Tool, welches seit 2013 existiert, stetig um zusätzliche Funktionen erweitert wurde und noch immer aktiv entwickelt wird. Die wesentlichen Anforderungen an zukünftige Aufgaben der Modellierung von Energiesystemen, wie die Kopplung von Sektoren und die Berücksichtigung von Emissionen, sind ebenfalls mit *Calliope* erfüllt. Mit der Integration *Calliope*'s in *Tessif* lässt sich eine Vielzahl von in *Tessif* erstellten Modellen mit *Calliope* untersuchen und mit anderen Tools vergleichen. Hierbei sind die meisten Parameter in *Calliope* verhältnismäßig ähnlich zu jenen in *Tessif*. Bei Bedarf ist es zudem möglich ein umgewandeltes Modell im Nachhinein um *Calliope* spezifische Funktionen zu erweitern.

5.2. Vergleich mit Hilfe von *Tessif*

Die in Kapitel 4 durchgeführten Optimierungen zeigen eine Vielzahl von Gemeinsamkeiten zwischen den in *Tessif* integrierten Tools *Calliope*, *FINE*, *Oemof* und *PyPSA*, jedoch ebenso einige bedeutende Unterschiede.

Im Vergleich zu den anderen in *Tessif* integrierten Tools zeigt sich dass *Calliope* das einzige dieser Tools ist, welches bei Energiespeichern nicht den Wirkungsgrad beim Laden und Entladen separat voneinander vorschreibt, sondern einen einzelnen Wert verwendet, welcher in beide Richtungen gilt. Außerdem ist die Beschränkung der Leistung, welche in *Tessif* einen Wert vorsieht, welcher das Maximum des Ladens und Entladens beschränkt, in *Calliope* grundlegend anders. *Calliope* beschränkt das Maximum der Differenz zwischen dem Lade- und Entladevorgang zweier aufeinanderfolgender Zeitschritte. Dies ist der einzige Unterschied, der in dem in Abschnitt 4.2 untersuchten Modell Einfluss auf das Gesamtergebnis nimmt.

Nichtsdestotrotz existieren weitere Unterschiede, welche ebenso identifiziert werden konnten. So ist in *Calliope* keine Möglichkeit gegeben einen finalen Speicherstand anzugeben, es sei denn dieser entspricht dem initialen Speicherstand. In einem Gesamtsystem mit mehreren Speichern kann jedoch nicht explizit jeder Speicher als zyklisch oder nicht zyklisch festgelegt werden. Diese Angabe ist einmalig zu machen und gilt für alle Speicher. In *FINE* kann zwischen zyklischen und nicht zyklischen Speichern in einem System unterschieden werden, jedoch ist die Angabe eines finalen sowie initialen Speicherstandes nicht möglich. Da diese Parameter deutlich individueller in *Oemof* und *PyPSA* anzugeben sind, ist der Detailgrad der Parametrisierung von Speichern insgesamt in *Oemof* und *PyPSA* höher als in *Calliope* und *FINE*.

Jedoch haben auch diese ihre Nachteile. So bezieht *Oemof*, ebenso wie *Calliope*, einen initialen Speicherstand auf die optimale installierte Kapazität und nicht auf die initial installierte Kapazität. Für *Calliope* ist dies sinnig aufgrund dessen, dass

keine initialen Kapazitäten vorliegen. Bei *Oemof* wäre jedoch grundsätzlich zu erwarten, dass ein initialer Speicherstand auf die initiale Kapazität bezogen ist. Ein solcher Bezug existiert somit ausschließlich in *PyPSA*. Jedoch wird zur Ermittlung der Emissionen in *PyPSA* lediglich der initiale mit dem finalen Speicherstand verglichen und nicht, wie in den anderen Tools, jeder aus dem Speicher fließende Energiestrom mit Emissionen behaftet.

PyPSA berücksichtigt generell lediglich die Emissionen von Energiequellen so wie *Tessif* es vorsieht. Emissionen von Speichern werden grundlegend verschieden ermittelt und die Emissionen anderer Komponenten nicht berücksichtigt, weshalb in der Transformation des *Tessif*-Modells zu einem *PyPSA*-Modell einfach Energiewandler automatisch zu Quellen umformuliert werden. Daher ist bei der Optimierung eines Modells mit einem gegebenen Emissionslimit in der Regel *PyPSA* nicht die beste Wahl. Es sei denn, dem System sind ausschließlich emissionsbehaftete Erzeuger gegeben oder es wird die *PyPSA* spezifische Definition der Berechnung der Speicheremissionen erwünscht.

Dieser Unterschied ist zudem erst dann zu erkennen, wenn dem System ein Emissionslimit vorgegeben ist. Dies ist darin begründet, dass die Emissionen jeder Komponente zugeordnet sind und aus den *PyPSA* Ergebnissen von *Tessif* ausgelesen werden. Dies erfolgt unabhängig davon, ob *PyPSA* in der Optimierung diese Emissionen berücksichtigt oder nicht.

Darüber hinaus werden in *PyPSA* die *Tessif* Konnektoren nicht exakt abgebildet, da die entsprechende *PyPSA* Komponente bei der Verwendung eines Wirkungsgrades in eine Richtung, für die entgegengesetzte Richtung den Kehrwert annimmt. In der *Tessif* Integration wird daher auf den Wirkungsgrad verzichtet, da ein Verlust in die eine Richtung andernfalls eine unerwünschte Energieerzeugung bei einem Energiefluss in die andere Richtung zur Folge hätte. *Calliope*, *FINE* und *Oemof* hingegen bilden die Konnektoren wie von *Tessif* erwartet ab.

Bezüglich der Optimierungszeiten der Tools lässt sich kein finales Urteil, bezüglich eines schnellsten oder langsamsten Tools, aufstellen. Während *Oemof* das „Commitment Example“ am schnellsten optimiert, ist es das langsamste Tool bei dem „Expansion Example“. In der angepassten Variante des „Expansion Examples“, welche durch Entfernen der Emissionen der Batterie sowie der Power-to-Heat Anlage eine bessere Vergleichbarkeit zu *PyPSA* herstellt, ist *Oemof* jedoch erneut das schnellste Tool. *PyPSA* liefert überwiegend solide Optimierungszeiten, jedoch ist vor allem beim „Expansion Example“ zu bedenken, dass die schnelle Optimierung unter anderem von der abweichenden Berücksichtigung der Emissionen profitiert.

Calliope, welches das „Commitment Example“ am langsamsten und das „Expansion Example“ am zweit langsamsten optimiert, hat im Vergleich dieser Optimierungszeiten die geringste relative Zunahme. Da *Calliope* jedoch im angepassten „Expansion Example“ sowie im „Source Example“ ebenfalls das langsamste Tool darstellt, sind die Optimierungszeiten von *Calliope* überwiegend als höher festzuhalten. Zudem konnte gezeigt werden, dass die Darstellung jeder Technologie in

einem separaten Standort nicht zwangsläufig schlechtere Optimierungszeiten zur Folge hat, als die Darstellung aller Komponenten in einem Standort.

Aufgrund der hohen Optimierungszeiten *Calliope*'s kann die integrierte Option der Aggregation von Zeitreihen helfen, um dennoch in kürzerer Zeit einen Eindruck von der Größenordnung der Ergebnisse sowie der korrekten Berücksichtigung der Komponenten zu erlangen. Jedoch ist im „Commitment Example“ mit 6 Stunden Zeitschritten eine relativ große Aggregation notwendig, um vergleichbare Optimierungszeiten wie die anderen Tools zu erreichen. Die Aggregation von 6 Zeitschritten führt dabei zu einer Unterschätzung des Kostenoptimums von 0,3 % im Vergleich zur Lösung ohne Aggregation.

Beim „Expansion Example“ führt bereits eine Aggregation von 2 Zeitschritten zu einer schnelleren Optimierung als die anderen Tools. Das Kostenoptimum weicht hierbei mit 0,8 % ebenfalls nur gering von dem Ergebnis ohne Aggregation ab. Zum Vergleich liegt eine Abweichung von 6,3 % vor, wenn eine Aggregation von 6 Zeitschritten gewählt wird. Dies zeigt, dass neben der Schrittweite auch das Modell selbst großen Einfluss auf die Abweichungen nimmt.

Neben *Calliope* verfügt *FINE* ebenfalls über die Möglichkeit der Zeitreihenaggregation. Während *FINE* in der *Tessif* Integration eine feste Anzahl repräsentativer Tage identifiziert und aggregiert, werden in der *Calliope* Integration lediglich eine vorgegebene Anzahl an Zeitschritten zu einem Zeitschritt zusammengefasst. In beiden Ansätzen geht mit der reduzierten Optimierungsdauer eine Unterschätzung der entstehenden Kosten einher. Im direkten Vergleich zeigt sich, dass *FINE* hierbei die bessere Kombination aus schneller Optimierung und geringen Abweichungen zum Ergebnis ohne Aggregation liefert.

Calliope ist das einzige Tool bei welchem, zumindest ein Stück weit, die Möglichkeit einer Priorisierung einer von zwei kostengleichen Komponenten identifiziert werden konnte. Der Einfluss der Bezeichnung der Komponenten ist jedoch nicht der einzige Einfluss, weshalb keine konsequente Wahl der Leistungsbereitstellung in alphabetischer Reihenfolge erfolgt. Welche weiteren Faktoren die Priorisierung beeinflussen, konnte anhand des untersuchten Modells nicht abschließend geklärt werden.

Die Vielzahl von Modellen und deren abweichenden Komponenten und Parameter, welche untersucht wurden, zeigen somit einige Gemeinsamkeiten und Unterschiede der betrachteten Tools. Nicht auszuschließen ist jedoch, dass weitere Verschiedenheiten bei anderen Parametervariationen zu beobachten sind, welche in den gewählten Modellen nicht zum Vorschein kommen.

5.3. Grenzen des Vergleichs mit *Tessif*

Aufgrund der geforderten Einheitlichkeit des Modells sowie der Darstellung der Ergebnisse in *Tessif* sind Optionen der einzelnen Tools nicht in *Tessif* abzubilden. Zu diesen gehört beispielsweise die „Spores“ Funktion von *Calliope*, mit welcher

nahezu optimale Ergebnisse ermittelt werden. *Calliope* liefert hierbei Ergebnisse, welche in einem vorgegebenen Rahmen von dem Optimum abweichen und eine möglichst hohe Variation der verwendeten Technologien aufweisen. Darüber hinaus bietet *Calliope* die Option, ein System mit fest installierten Leistungen im „Operate“ Modus, mit einem rollierenden Horizont Ansatz zu optimieren. Der Ansatz eines rollierenden Horizontes ist bisher in *Tessif* nicht explizit implementiert.

Calliope gibt die einfache Modellformulierung über `YAML` und `CSV` Dateien als einen Vorteil an [28]. Bei der Nutzung von *Tessif* ist jedoch ohnehin das Modell in Python in der von *Tessif* geforderten Art und Weise zu erstellen. Demnach wird von dem Vorteil *Calliope*'s erst Gebrauch gemacht, sobald ein Modell entweder direkt für *Calliope* erstellt oder ein mit *Tessif* übersetztes Modell für *Calliope* angepasst wird. Zu diesem Zeitpunkt ist jedoch schon eine gewisse Einarbeitung in das Erstellen von Modellen in *Tessif* vonnöten.

Zudem sind in allen Tools weitere Komponenten und Parameter abzubilden, als die in *Tessif* eingebundenen [22, 25, 26, 28]. So können in *Calliope* und *FINE* ebenfalls Wirkungsgrade der *Transmissions* abhängig von ihrer Länge angegeben werden. Während diese Länge in *FINE* direkt anzugeben ist [22], kann *Calliope* diese, wenn keine direkte Angabe vorliegt, anhand der Koordinaten der Standorte selbst berechnen [28]. *PyPSA* verfügt über Optionen zur spezifischen Untersuchung des Stromsektors. So können beispielsweise Wechselstrom- und Gleichstromnetzwerke modelliert und miteinander verknüpft werden [26]. *Oemof* hat mit *Tessif* von allen Tools die meisten Gemeinsamkeiten. Dies ist darauf zurückzuführen dass *Tessif* in Anlehnung an das *Oemof*-Modell entwickelt wurde.

Tessif deckt folglich zwar nicht alle Möglichkeiten der Modellierung von Energiesystemen ab, jedoch den Großteil, welcher aller Voraussicht nach den meisten Anwendungsfällen entspricht. So sind bis auf die *Transmission* und *Supply_Plus* Technologie alle *Calliope* Technologien in *Tessif* enthalten [28]. Wobei die Fähigkeiten einer *Supply_Plus* Technologie grundsätzlich über eine Quelle und einen separaten Speicher dargestellt werden können. Ebenso sind nahezu alle *Oemof* und *FINE* Basiskomponenten in *Tessif* vorhanden. Die weiteren Komponenten welche in *Oemof* und *FINE* existieren, abgesehen von der *Transmission* in *FINE*, sind Erweiterungen dieser Basiskomponenten [22, 25]. Die meisten *PyPSA* Komponenten, welche nicht in *Tessif* abgebildet sind, sind jene welche der Analyse des Stromsektors dienen, auf welche sich *PyPSA* spezialisiert hat [26]. Somit ist davon auszugehen, dass *Tessif* den wesentlichen Bereich der Modellierung von Energiesystemen abdeckt.

Tessif bietet folglich mit vier integrierten Tools, einer Bibliothek mit Beispielmodellen, sowohl mit *Tessif*-Modellen als auch Tool spezifischen Modellen, und einer entsprechenden Dokumentation dieser Inhalte [21] einen guten Einstieg in die Modellierung von Energiesystemen. Nicht auszuschließen ist jedoch, dass bei einer speziellen Problemstellung eine Recherche bezüglich der Tools über *Tessif* hinaus erforderlich bzw. hilfreich ist.

6. Zusammenfassung

Zum Abschluss folgt ein Fazit, in welchem ebenso die wesentlichen Inhalte dieser Arbeit zusammengefasst werden. Zudem soll ein Ausblick mögliches weiteres Verwendungspotenzial der Integration von *Calliope* in *Tessif* sowie der im Rahmen dieser Arbeit gewonnen Erkenntnisse aufführen.

6.1. Fazit

Modellierungstools gewinnen aufgrund des voranschreitenden Klimawandels und der Notwendigkeit einer Anpassung der Energieversorgungssysteme immer mehr an Bedeutung. Jedoch existiert eine Vielzahl von Softwaretools, welche sich allesamt, je nach zu untersuchendem Energiesystem und der dazugehörigen Problemstellung, stark unterscheiden und demnach mehr oder weniger gut für eine Untersuchung eignen.

Ziel dieser Arbeit ist es durch die Integration eines weiteren Free and Open Source Software Tools in das als Vergleichsplattform agierende Framework *Tessif* einen Vergleich zu den drei bereits integrierten Tools *FINE*, *Oemof* und *PyPSA* herzustellen und aus diesem Stärken und Schwächen der einzelnen Tools zu identifizieren. Angesichts des ebenfalls auf Python basierten Codes, der Aktualität, der Möglichkeit mehrere Sektoren abzubilden, sowie einiger weiterer Auswahlkriterien wurde *Calliope* als ausgezeichneter Kandidat zur Integration in *Tessif* identifiziert.

In der Integration, deren Hauptbestandteile die Übersetzung des *Tessif*-Modells in ein *Calliope*-Modell, sowie das Umformen der Ergebnisse von *Calliope* in den einheitlichen Datenoutput von *Tessif* darstellen, werden infolge der unterschiedlichen Parametrisierungsmöglichkeiten einige Kompromisse eingegangen. So bleiben einige *Tessif* Parameter ohne Effekt im *Calliope*-Modell.

Nichtsdestotrotz zeigt die Optimierung verschiedener Beispielmodelle, dass die Menge an Gemeinsamkeiten überwiegt und ein solider Vergleich zu den anderen in *Tessif* integrierten Tools möglich ist. Neben vielen Gemeinsamkeiten zwischen *Calliope* und den weiteren drei integrierten Tools, zeigt diese Arbeit ebenso einige Unterschiede auf. Die Unterschiede halten sich jedoch, voraussichtlich auch aufgrund der einheitlichen Modellformulierung über *Tessif*, in Grenzen. Derweil verfügen die Tools alle über modellierbare Komponenten und Optimierungsmethoden, welche über den Rahmen von *Tessif* hinaus gehen und sich dem durchgeführten Vergleich folglich entziehen.

Die wesentlichen Ergebnisse des Vergleichs, bezüglich eines am Besten geeigneten Tools hinsichtlich verschiedener Aspekte einer zu untersuchenden Problemstellung, lassen sich wie folgt zusammenfassen:

- Ist eine einfache Modellformulierung erwünscht so ist *Calliope*, aufgrund der Formulierung des Modells in `YAML` und `CSV` Dateien, zu empfehlen.
- Ist eine schnelle Optimierung von hoher Priorität, welche auf Kosten der Genauigkeit der Ergebnisse erzielt werden kann, so sind *Calliope* und *FINE* aufgrund der Möglichkeiten der Zeitaggregation zu empfehlen.
- Ist eine schnelle Optimierung von hoher Priorität, jedoch ebenso die Genauigkeit der Ergebnisse, so ist keines der Tools klar zu empfehlen, da die Vor- und Nachteile stark vom zu untersuchenden Modell abhängen. Jedoch gehört *Calliope* in den untersuchten Modellen überwiegend zu den langsamen Tools und *PyPSA* überwiegend zu den schnelleren.
- Ist neben dem optimalen Ergebnis auch die Ausgabe von nahezu optimaler alternativer Lösungen erwünscht, so ist *Calliope* mit dessen „Spores“ Funktion zu empfehlen.
- Ist eine detaillierte Darstellung von Energiespeichern erforderlich, so ist die Wahl eines Tools stark abhängig vom zu untersuchenden Modell und dessen Parametrisierung, da sich die Darstellung von Energiespeichern in allen Tools innerhalb verschiedener Parameter bedeutend unterscheiden.
- Ist eine detaillierte Darstellung des Stromnetzes mit Unterscheidung von Wechselstrom- und Gleichstromnetzen erforderlich, so ist *PyPSA* zu empfehlen.
- Ist die Berücksichtigung von Emissionen bei vielen verschiedenen Klassen von Komponenten erforderlich und zudem ein Emissionslimit gegeben, so sind alle Tools außer *PyPSA* zu empfehlen.

Die aufgeführten Punkte beziehen sich auf die im Rahmen dieser Arbeit gewonnen Erkenntnisse. Diese beinhalten, aufgrund der durchgeführten Integration *Calliope*'s in *Tessif*, eine tiefere Kenntnis der Fähigkeiten von *Calliope* als jene der anderen Tools. Die anderen Tools sind überwiegend anhand ihrer Integration in *Tessif* zu bewerten und verfügen über Möglichkeiten, welche über diese Untersuchung hinaus gehen.

Zudem ist festzuhalten, dass es sich bei allen Tools um Free and Open Source Softwaretools handelt, welche sich durch ihre Aktualität, der Abbildung mehrerer Sektoren, der Berücksichtigung von Emissionen, ihrer Programmierung in Python sowie der Formulierung von LP und MILP Problemen auszeichnen. Ist einer oder sind mehrere dieser genannten Faktoren von erhöhter Bedeutung, so so ist jedes einzelne, der in *Tessif* integrierten Tools einsetzbar und zu empfehlen.

Insgesamt eignet sich *Tessif* für einen Überblick und einen Eindruck zu den implementierten Tools sehr gut. Die meisten Anwendungsfälle sind aller Voraussicht nach mit einem *Tessif*-Modell abzubilden und ohne großen Aufwand mit den integrierten Tools zu analysieren. Jedoch verfügen die Tools zudem über spezifische Modellierungs- und Optimierungsmöglichkeiten, welche über ihre Integration in *Tessif* hinaus gehen. Demnach ist bei einer realen Problemstellung durchaus denkbar, dass ein Vergleich eben dieser spezifischen Möglichkeiten erwünscht ist. Dennoch kann *Tessif* auch in solch einem Fall einen Vergleich unterstützen, indem es Unterschiede aufzeigt, welche zum Ausschluss einzelner Tools führen und somit den Umfang des nachfolgenden Vergleiches reduzieren kann.

Der wesentliche Vorteil liegt hierbei darin, dass ein *Tessif*-Modell einmalig zu erstellen ist, um die Ergebnisse von vier Tools zu erzeugen und miteinander zu vergleichen. Somit reduziert sich ebenfalls der Aufwand und die Fehleranfälligkeit bei jeder weiteren Anpassung an dem Modell, da diese Anpassung lediglich einmalig am *Tessif*-Modell durchzuführen ist und nicht explizit für jedes einzelne Tool.

6.2. Ausblick

Die im Rahmen dieser Arbeit durchgeführte Integration des Modellierungstools *Calliope* in *Tessif* stellt eine sowohl quantitative als auch qualitative Erweiterung *Tessif*'s dar. Die Integration weiterer Tools kann den Umfang von *Tessif* erneut erhöhen und unter Umständen, bei einer ähnlichen Struktur der Modellformulierung oder des Datenoutputs, von den für *Calliope* geschrieben Funktionen profitieren, da diese eine Integration jener Tools beschleunigen.

Die aus dem Vergleich der Optimierungsergebnisse gewonnenen Erkenntnisse können dazu dienen die Auswahl eines geeigneten Tools entsprechend einer Problemstellung zu identifizieren oder die Kandidaten einzuschränken. Hierfür werden die Optimierungsergebnisse sowie die verwendeten Modelle zusätzlich in der *Tessif* Dokumentation festgehalten.

Darüber hinaus existieren, trotz der Vielzahl untersuchter Modelle, Parameterkombinationen welche in dieser Arbeit nicht abgebildet sind. Das Erstellen und Untersuchen weiterer Modelle, ebenso wie die Modellierung von möglichst realitätsnahen Energiesystemen über längere Zeiträume, können in zukünftigen Arbeiten untersucht werden und neue Erkenntnisse liefern.

Zudem gilt es die Integration *Calliope*'s, sofern die Kompatibilität zu den weiteren in *Tessif* integrierten Tools dies zulässt, auf einem möglichst aktuellen Stand zu halten. Aktuellere Versionen agieren möglicherweise weniger zeit- und rechenaufwendig. Nicht auszuschließen ist darüber hinaus beispielsweise, dass zukünftige Versionen mit Änderungen der Standardparameter oder abweichenden Bezeichnungen im Modell oder im Datenoutput einhergehen. Jene Änderungen könnten Anpassungen an den in *Tessif* geschriebenen Funktionen zur Übersetzung des *Tessif*-

Modells in ein *Calliope*-Modell sowie zum Umformen der *Calliope* Ergebnisse in den *Tessif* Datenoutput erfordern.

Letztlich zeigt die Formulierung eines *Tessif*-Modells, welches in Anlehnung an *Oemof* entwickelt wurde, zwar die größte Ähnlichkeit zu *Oemof*, bildet jedoch ebenso die wesentlichen Komponenten und Parameter der anderen Tools ab. Eine Ausweitung der in *Tessif* definierten Parameter eines Modells ist dennoch durchaus denkbar. Hierbei ist vor allem die Parametrisierung der Energienetze zu nennen. Mithilfe eines Konnektors ist es möglich in *Tessif* die Verluste darzustellen, jedoch ist es nicht möglich Netzkapazitäten darzustellen. In *Calliope* und *FINE* können jedoch die Kapazitäten sowie damit einhergehende Kosten abgebildet werden. Diese ließen sich beispielsweise in einem *Bus* in *Tessif* integrieren. Die einzelnen Funktionen zum Übersetzen des *Tessif*-Modells in ein Modell eines der integrierten Tools, sowie zur Darstellung der Ergebnisse gilt es nach einer solchen Anpassung des *Tessif*-Modells ebenso zu überarbeiten.

Als Alleinstellungsmerkmal von *Calliope* ist der „Spores“ Modus hervorzuheben. Dieser wird im folgenden auf das „Commitment Example“ aus Abschnitt 4.5.2 angewendet, um somit einen Ausblick auf dessen Möglichkeiten zu bieten, sowie die Kombinationsmöglichkeiten von *Tessif* und Tool-spezifischen Funktionalitäten aufzuzeigen.

Die mit Hilfe des `es2es` aus dem *Tessif*-Modell erstellten `YAML` Dateien werden hierfür genutzt und, in Anlehnung an die Beispielmodelle der *Calliope* Dokumentation [28], angepasst. Den *Transmission* Technologien für den Strom- und Wärmetransport werden mit einem sogenannten „spores_score“ versehen. Aufgrund dessen, dass *Calliope* die Netzkapazitäten zwischen allen Komponenten getrennt mit der Bezeichnung „Standort_1::Transmission:Standort_2“ abspeichert, wird jede Strom- und Wärmeverbindung zweier Standorte von *Calliope* separat betrachtet. Nach einer Kostenoptimierung werden zwei weitere Optimierungen mit alternativen Lösungen bei maximal 10 % höheren Kosten ermittelt. Die entsprechenden Anpassungen an den Dateien `techs.yaml` und `model.yaml` sind dem Quellcode A.1 sowie A.2 im Anhang zu entnehmen.

Wie aus Abbildung 6.1 hervorgeht, sind die Kosten der alternativen Lösungen, wie gefordert, 10 % höher als das Kostenoptimum. Die Emissionen liegen ebenso in beiden alternativen Lösungen höher. Dies ist vor allem der Tatsache geschuldet, dass die emissionsarmen onshore Windanlagen und PV-Anlagen in der ursprünglichen Optimierung, aufgrund der geringen Kosten, bereits Anwendung finden. Daher finden diese Komponenten in den alternativen Lösungen weniger Anwendung.

Dies zeigt sich vor allem bei der Betrachtung der Stromerzeugung in Abbildung 6.2. Diese zeigt, dass bis auf dem Braunkohle Kraftwerk alle Komponenten in den alternativen Lösungen relativ stark variieren. Die geringen Abweichungen in der Nutzung des Braunkohle Kraftwerks sind voraussichtlich darauf zurückzuführen, dass dieses die kostengünstigste grundlastfähige Stromerzeugung des Modells darstellt und somit eine erhöhte Braunkohle Nutzung mehr Möglichkeiten in der Nutzung der weiteren Komponenten ermöglicht.

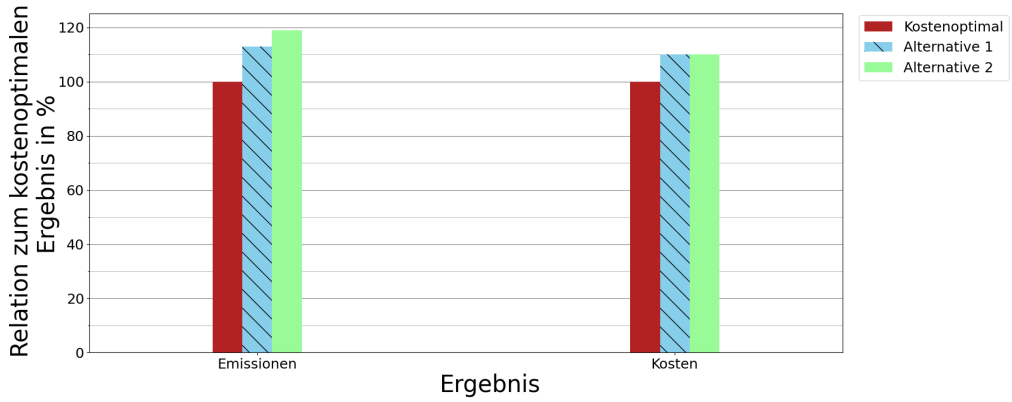

Abbildung 6.1.: Kosten und Emissionen des „Commitment Example“ bei Verwendung des Spores Modus in Relation zum kostenoptimalen Ergebnis

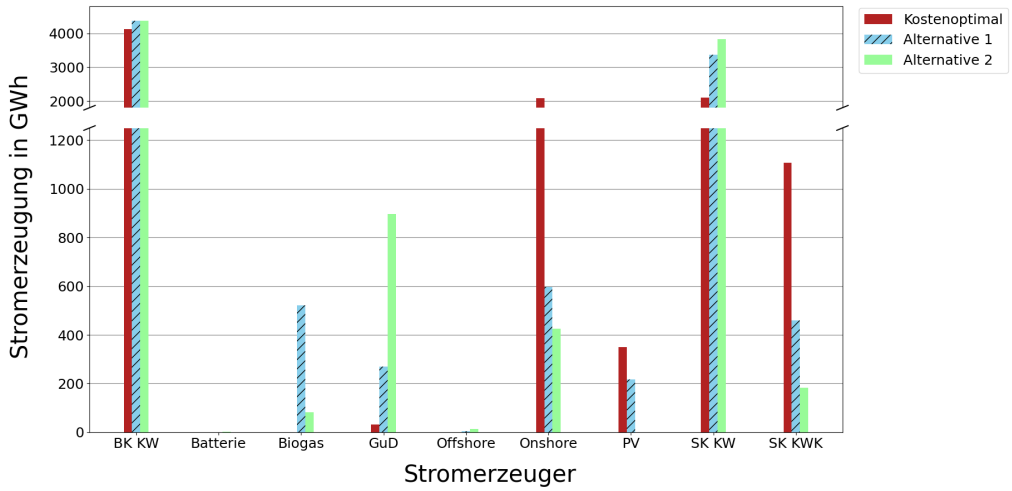

Abbildung 6.2.: Stromerzeugung des „Commitment Examples“ bei Verwendung des Spores Modus

Die Wärmeerzeugung in Abbildung 6.3 zeigt ebenfalls eine große Variabilität in der Nutzung der Komponenten zwischen den verschiedenen Lösungen. Dennoch sind einige Komponenten, wie zum Beispiel die Batterie, die offshore Windanlagen oder die Power-to-Heat Anlage kaum genutzt. Dies ist womöglich dem Modell an sich geschuldet, sodass eine Nutzung dieser Komponenten mit der Einhaltung des Kostenlimits nur schwierig zu vereinbaren ist.

Außerdem ist die gewählte Formulierung des „spores_score“ im Strom- und Wärmenetz ein Nachteil für die Power-to-Heat Anlage. Diese ist an beide Netze angebunden und wird demnach im Vergleich zu den anderen Komponenten doppelt gewichtet. Ein „spores_score“ in den Komponenten selbst stellt eine weitere Option der Parametrierung dar. Jedoch können die Komponenten im „Commitment Example“ nicht ausgebaut werden und das „Expansion Example“ wäre zu wählen.

Dieses gestattet jedoch aufgrund des geringen Emissionslimits keine große Variabilität in der Nutzung der Komponenten.

Nichtsdestotrotz zeigen die Ergebnisse, dass der Spores Modus in der Tat alternative Lösungen ermittelt. Aller Voraussicht nach bietet sich dieser Modus besonders bei noch komplexeren Modellen an, welche beispielsweise gleiche Technologien an verschiedenen Standorten abbilden. So könnten beispielsweise mehrere nahezu optimale Lösungen hinsichtlich ihrer Unterschiede in der lokalen Netzauslastung verglichen werden.

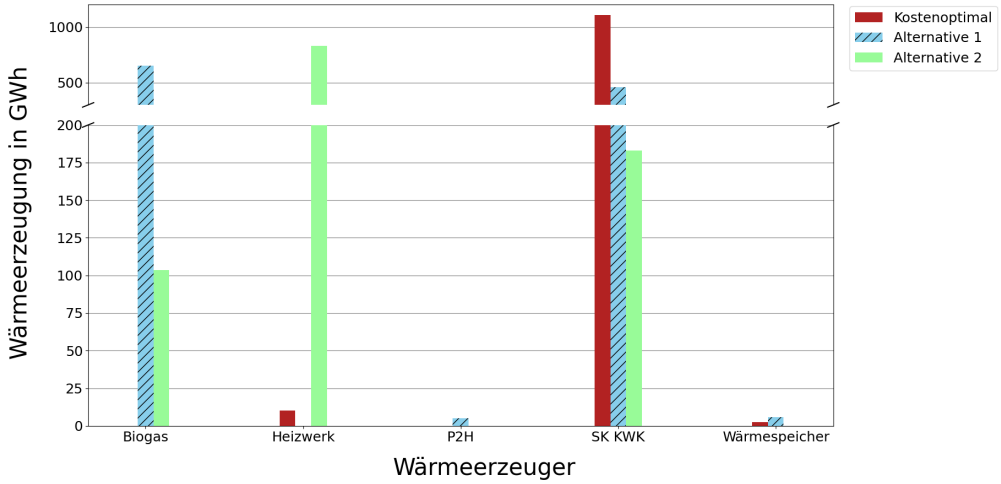

Abbildung 6.3.: Wärmeerzeugung des „Commitment Examples“ bei Verwendung des Spores Modus

Die in dieser Arbeit untersuchten Modelle, inklusive einer Auswahl von Optimierungsergebnissen, sind in einem extra hierfür erstellten Abschnitt der *Tessif* Dokumentation [21] aufgeführt. Dies vereinfacht die Transparenz und Reproduzierbarkeit der Ergebnisse und erlaubt es zudem weitere Untersuchungen, ausgehend von diesen Modellen als Grundlage, durchzuführen.

Literatur

[1] O. D. Doleski, T. Kaiser, M. Metzger, S. Niessen und S. Thiem: *Digitale Dekarbonisierung: Technologieoffen die Klimaziele erreichen*. Wiesbaden: Springer Fachmedien Wiesbaden, 2021.

[2] P. Lopion, P. Markewitz, M. Robinius und D. Stolten: „A review of current challenges and trends in energy systems modeling“. In: *Renewable and Sustainable Energy Reviews* 96 (2018), S. 156–166.

[3] D. Connolly, H. Lund, B. V. Mathiesen und M. Leahy: „A review of computer tools for analysing the integration of renewable energy into various energy systems“. In: *Applied Energy* 87.4 (2010), S. 1059–1082.

[4] T. Horschig und D. Thrän: „Are decisions well supported for the energy transition? A review on modeling approaches for renewable energy policy evaluation“. In: *Energy, Sustainability and Society* 7 (2017).

[5] H.-K. Ringkjøb, P. M. Haugan und I. M. Solbrekke: „A review of modelling tools for energy and electricity systems with large shares of variable renewables“. In: *Renewable and Sustainable Energy Reviews* 96 (2018), S. 440–459.

[6] United Nations Framework Convention on Climate Change: *The Paris Agreement*. URL: `https://unfccc.int/process-and-meetings/the-paris-agreement/the-paris-agreement` (besucht am 17. 05. 2022).

[7] C. Redl, F. Hein, M. Buck, P. Graichen und D. Jones: *The European Power Sector in 2020: Up-to-Date Analysis on the Electricity Transition*. Hrsg. von Agora Energiewende und Ember. 2021.

[8] D. F. Dominković, J. M. Weinand, F. Scheller, M. D'Andrea und R. McKenna: „Reviewing two decades of energy system analysis with bibliometrics“. In: *Renewable and Sustainable Energy Reviews* 153 (2022), S. 111749.

[9] H. Lund, F. Arler, P. Østergaard, F. Hvelplund, D. Connolly, B. Mathiesen und P. Karnøe: „Simulation versus Optimisation: Theoretical Positions in Energy System Modelling“. In: *Energies* 10.7 (2017), S. 840.

[10] J. DeCarolis, H. Daly, P. Dodds, I. Keppo, F. Li, W. McDowall, S. Pye, N. Strachan, E. Trutnevyte, W. Usher, M. Winning, S. Yeh und M. Zeyringer: „Formalizing best practice for energy system optimization modelling“. In: *Applied Energy* 194 (2017), S. 184–198.

[11] M. Chang, J. Z. Thellufsen, B. Zakeri, B. Pickering, S. Pfenninger, H. Lund und P. A. Østergaard: „Trends in tools and approaches for modelling the energy transition“. In: *Applied Energy* 290 (2021), S. 116731.

[12] S. Pfenninger, J. DeCarolis, L. Hirth, S. Quoilin und I. Staffell: „The importance of open data and software: Is energy research lagging behind?“ In: *Energy Policy* 101 (2017), S. 211–215.

[13] S. Oberle und R. Elsland: „Are open access models able to assess today's energy scenarios?“ In: *Energy Strategy Reviews* 26 (2019), S. 100396.

[14] S. Pfenninger, L. Hirth, I. Schlecht, E. Schmid, F. Wiese, T. Brown, C. Davis, M. Gidden, H. Heinrichs, C. Heuberger, S. Hilpert, U. Krien, C. Matke, A. Nebel, R. Morrison, B. Müller, G. Plessmann, M. Reeg, J. C. Richstein, A. Shivakumar, I. Staffell, T. Tröndle und C. Wingenbach: „Opening the black box of energy modelling: Strategies and lessons learned“. In: *Energy Strategy Reviews* 19 (2018), S. 63–71.

[15] H. Farzaneh: *Energy Systems Modeling*. Singapore: Springer Singapore, 2019.

[16] T. Ommen, W. B. Markussen und B. Elmegaard: „Comparison of linear, mixed integer and non-linear programming methods in energy system dispatch modelling“. In: *Energy* 74.2 (2014), S. 109–118.

[17] F. Lombardi, B. Pickering, E. Colombo und S. Pfenninger: „Policy Decision Support for Renewables Deployment through Spatially Explicit Practically Optimal Alternatives“. In: *Joule* 4 (2020), S. 2185–2207.

[18] U. Krien, P. Schönfeldt, J. Launer, S. Hilpert, C. Kaldemeyer und G. Plessmann: „oemof.solph—A model generator for linear and mixed-integer linear optimisation of energy systems“. In: *Software Impacts* 6.1 (2020), S. 100028.

[19] T. Brown, J. Hörsch und D. Schlachtberger: „PyPSA: Python for Power System Analysis“. In: *Journal of Open Research Software* 6.3 (2018), S. 12.

[20] Open Knowledge Foundation: *The Open Definition - Open Definition - Defining Open in Open Data, Open Content and Open Knowledge*. URL: `https://opendefinition.org/` (besucht am 17. 05. 2022).

[21] M. Ammon: *Tessif - Transforming Energy Supply System (Modelling) Frameworks*. Dokumentation (unveröffentlicht), Institut für Energietechnik, Technische Universität Hamburg.

[22] FINE Developer Team: *Welcome to FINE's documentation! — FINE 2.2.2 documentation*. URL: `https://vsa-fine.readthedocs.io/en/latest/` (besucht am 16. 06. 2022).

[23] D. Schnute: „Vergleichende Analyse von Software zur Modellierung von Energiesystemen mittels Integration in ein Framework zur Transformation“. Masterarbeit. Hamburg: Technische Universität Hamburg, 2022.

[24] Oemof Developer Group: *Open Energy Modelling Framework (oemof) – oemof documentation*. URL: https://oemof.readthedocs.io/en/latest/ (besucht am 11. 05. 2022).

[25] Oemof Developer Group: *Welcome to oemof's documentation! — oemof.solph 0.4.5.dev0 documentation*. URL: https://oemof-solph.readthedocs.io/en/latest/ (besucht am 16. 06. 2022).

[26] PyPSA Developers: *PyPSA: Python for Power System Analysis – PyPSA 0.19.3 documentation*. URL: https://pypsa.readthedocs.io/en/latest/ (besucht am 16. 06. 2022).

[27] J. Allegrini, K. Orehounig, G. Mavromatidis, F. Ruesch, V. Dorer und R. Evins: „A review of modelling approaches and tools for the simulation of district-scale energy systems". In: *Renewable and Sustainable Energy Reviews* 52.4 (2015), S. 1391–1404.

[28] Calliope contributors: *Calliope: a multi-scale energy systems modelling framework — Calliope 0.6.8 documentation*. URL: https://calliope.readthedocs.io/en/stable/ (besucht am 16. 06. 2022).

[29] D. Atabay: *ficus: A (mixed integer) linear optimisation model for local energy systems — ficus 0.1 documentation*. URL: https://ficus.readthedocs.io/en/latest/ (besucht am 11. 05. 2022).

[30] KTH-dESA: *OSeMOSYS Documentation: Release 0.0.1*. URL: https://osemosys.readthedocs.io/en/latest/ (besucht am 11. 05. 2022).

[31] J. Dorfner: *urbs: A linear optimisation model for distributed energy systems — urbs 1.0.0 documentation*. URL: https://urbs.readthedocs.io/en/latest/ (besucht am 30. 05. 2022).

[32] *GitHub - FZJ-IEK3-VSA/FINE: The FINE python package provides a framework for modeling, optimizing and assessing energy systems*. URL: https://github.com/FZJ-IEK3-VSA/FINE (besucht am 26. 04. 2022).

[33] *GitHub - oemof/oemof-solph: A model generator for energy system modelling and optimisation (LP/MILP)*. URL: https://github.com/oemof/oemof-solph (besucht am 26. 04. 2022).

[34] *GitHub - PyPSA/PyPSA: PyPSA: Python for Power System Analysis*. URL: https://github.com/PyPSA/PyPSA (besucht am 26. 04. 2022).

[35] *GitHub - calliope-project/calliope: A multi-scale energy systems modelling framework*. URL: https://github.com/calliope-project/calliope (besucht am 26. 04. 2022).

[36] *GitHub - tum-ewk/ficus: A (mixed integer) linear optimisation model for local energy systems*. URL: https://github.com/tum-ewk/ficus (besucht am 26. 04. 2022).

[37] *GitHub - bje-/NEMO: National Electricity Market Optimiser*. URL: https://github.com/bje-/NEMO (besucht am 26. 04. 2022).

[38] *GitHub - OSeMOSYS/OSeMOSYS: OSeMOSYS - the Open Source Energy Modelling System*. URL: `https://github.com/OSeMOSYS/OSeMOSYS` (besucht am 26. 04. 2022).

[39] *GitHub - switch-model/switch: A Modern Platform for Planning High-Renewable Power Systems*. URL: `https://github.com/switch-model/switch` (besucht am 26. 04. 2022).

[40] *GitHub - tum-ens/urbs: A linear optimisation model for distributed energy systems*. URL: `https://github.com/tum-ens/urbs` (besucht am 26. 04. 2022).

[41] T. Brown, D. Schlachtberger, A. Kies, S. Schramm und M. Greiner: „Synergies of sector coupling and transmission reinforcement in a cost-optimised, highly renewable European energy system“. In: *Energy* 160 (2018), S. 720–739. URL: `https://arxiv.org/pdf/1801.05290v2`.

[42] *nemo API documentation*. URL: `https://nemo.ozlabs.org/pdoc/index.html` (besucht am 11. 05. 2022).

[43] S. Pfenninger und B. Pickering: „Calliope: a multi-scale energy systems modelling framework“. In: *Journal of Open Source Software* 3.29 (2018), S. 825.

[44] B. Pickering und R. Choudhary: „Quantifying resilience in energy systems with out-of-sample testing“. In: *Applied Energy* 285 (2021), S. 116465.

[45] L. Kotzur, P. Markewitz, M. Robinius und D. Stolten: „Impact of different time series aggregation methods on optimal energy system design“. In: *Renewable Energy* 117 (2018), S. 474–487.

[46] P. Díaz Redondo und O. van Vliet: „Modelling the energy future of switzerland after the phase out of nuclear power plants“. In: *Energy Procedia* 76 (2015), S. 49–58.

[47] P. Díaz, O. van Vliet und A. Patt: „Do We Need Gas as a Bridging Fuel? A Case Study of the Electricity System of Switzerland“. In: *Energies* 10 (2017), S. 861.

[48] M. Labordena und J. Lilliestam: „Cost and Transmission Requirements for Reliable Solar Electricity from Deserts in China and the United States“. In: *Energy Procedia* 76 (2015), S. 77–86.

[49] S. Pfenninger: „Dealing with multiple decades of hourly wind and PV time series in energy models: A comparison of methods to reduce time resolution and the planning implications of inter-annual variability“. In: *Applied Energy* 197.4 (2017), S. 1–13.

[50] T. Tröndle, J. Lilliestam, S. Marelli und S. Pfenninger: „Trade-Offs between Geographic Scale, Cost, and Infrastructure Requirements for Fully Renewable Electricity in Europe“. In: *Joule* 4 (2020), S. 1929–1948.

[51] T. Tröndle: „Supply-side options to reduce land requirements of fully renewable electricity in Europe“. In: *PloS one* (2020).

[52] G. Pontes Luz und R. Amaro e Silva: „Modeling Energy Communities with Collective Photovoltaic Self-Consumption: Synergies between a Small City and a Winery in Portugal“. In: *Energies* 14 (2021), S. 323.

[53] S. Pfenninger und J. Keirstead: „Comparing concentrating solar and nuclear power as baseload providers using the example of South Africa“. In: *Energy* 87.12 (2015), S. 303–314.

[54] B. Pickering und R. Choudhary: „District energy system optimisation under uncertain demand: Handling data-driven stochastic profiles“. In: *Applied Energy* 236 (2019), S. 1138–1157.

[55] M. Reimer: „Entwicklung eines Komponenten basierten Szenarios zum Vergleich von Free and Open Source Energiesystemmodellierungssoftware in Python“. Projektarbeit. Hamburg: Technische Universität Hamburg, 2021.

[56] T. Hanke: „Entwickeln eines netzbasierten Energiesystemmodells zum Vergleich von Free und Open Source Energiesystemmodellierungssoftware in Python“. Projektarbeit. Hamburg: Technische Universität Hamburg, 2021.

A. Anhang

Tabelle A.1.: Parameter des Refernzmodells [55]
Abkürzungen: CHP - Combined Heat and Power, PP - Power Plant

Komponente	Betriebs-kosten	Anlage-kosten	installierte Leistung bzw. Kapazität	Emissionen	Wirkungsgrad
	€/MWh	€/kW		t/MWh	
Solar Panel	80	1000	1100 MW	0,05	-
Onshore Wind	60	1750	1100 MW	0,02	-
Offshore Wind	105	3900	150 MW	0,02	-
Biogas CHP_{el}	150	3500	200 MW	0,25	0,4
Biogas CHP_{th}	11,25	262,5	250 MW	0,01875	0,5
Lignite PP	65	1900	500 MW	1	0,4
Hard Coal CHP_{el}	80	1750	300 MW	0,8	0,4
Hard Coal CHP_{th}	6	131,25	300 MW	0,06	0,4
Hard Coal PP	80	1650	500 MW	0,8	0,43
Combined Cycle PP	90	950	600 MW	0,35	0,6
Heat Plant	35	390	450 MW	0,23	0,9
Power To Heat	20	100	100 MW	0,0007	0,99
Battery	400	1630	100 MWh	0,06	0,9
Heat Storage	20	4,5	50 MWh	0	0,85

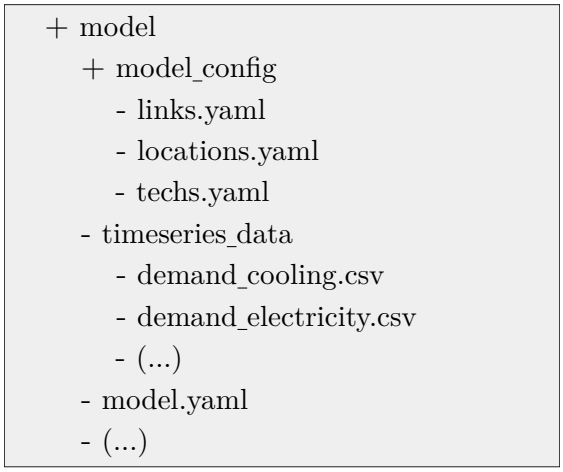

Abbildung A.1.: Darstellung der Ordnerstruktur des *Calliope*-Modells Bangalore[1]. Ein „+“ steht für einen Ordner, während ein „-“ eine Datei darstellt. (in Anlehnung an [28])

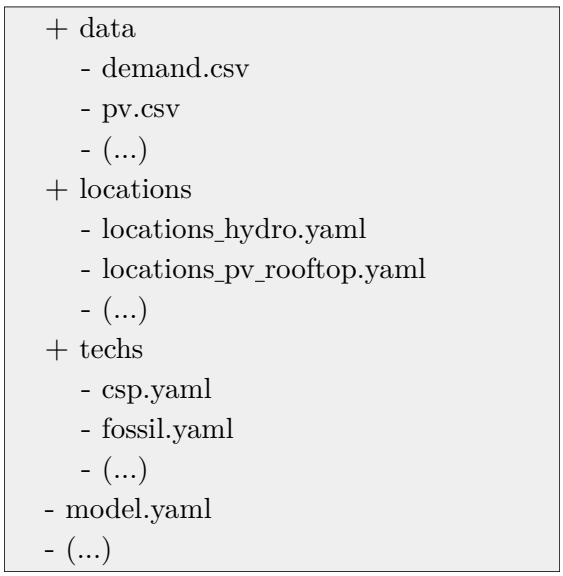

Abbildung A.2.: Darstellung der Ordnerstruktur des *Calliope*-Modells Großbritannien[2]. Ein „+“ steht für einen Ordner, während ein „-“ eine Datei darstellt. (in Anlehnung an [28])

[1] `https://github.com/brynpickering/bangalore-calliope`
[2] `https://github.com/calliope-project/uk-calliope`

Tabelle A.2.: Gesamtergebnis des „Commitment Examples“ (gerundet auf ganze Zahlen)

	Calliope	*FINE*	*Oemof*	*PyPSA*
Emissionen [t]	6.982.535	6.815.288	6.813.556	6.833.502
Kosten [€]	688.509.348	688.509.319	688.509.325	688.509.325
Opex [€]	688.509.352	688.509.319	688.509.325	688.509.325
Capex [€]	0	0	0	0
Zeit [s]	410	70	44	48

Tabelle A.3.: Gesamtergebnis des „Expansion Examples“ (gerundet auf ganze Zahlen)

	Calliope	*FINE*	*Oemof*	*PyPSA*
Emissionen [t]	250.000	250.000	250.000	265.508
Kosten [10^3 €]	42.289.121	42.289.118	42.289.118	37.727.777
Opex [10^3 €]	734.204	734.140	734.140	823.008
Capex [10^3 €]	41.554.918	41.554.976	41.554.978	36.904.768
Zeit [s]	983	720	1464	505

Tabelle A.4.: Gesamtergebnis des „Expansion Examples“ mit angepassten Emissionen (gerundet auf ganze Zahlen)

	Calliope	*FINE*	*Oemof*	*PyPSA*
Emissionen [t]	250.000	250.000	250.000	250.000
Kosten [10^3 €]	37.793.513	37.793.513	37.793.513	37.793.513
Opex [10^3 €]	820.206	820.206	820.206	820.206
Capex [10^3 €]	36.973.307	36.973.309	36.973.307	36.973.307
Zeit [s]	1.383	820	456	491

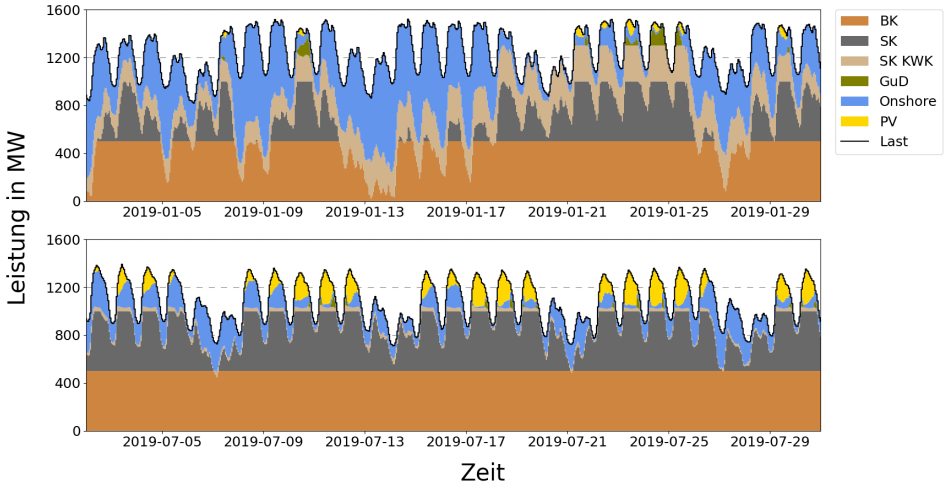

Abbildung A.3.: Leistungsverlauf des „Commitment Examples“ für den Stromsektor mit einer Stunde pro Zeitschritt

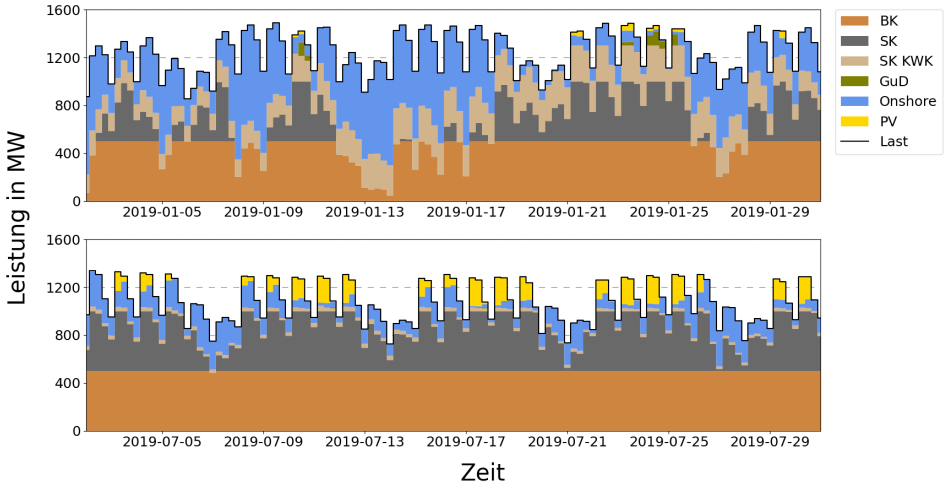

Abbildung A.4.: Leistungsverlauf des „Commitment Examples“ für den Stromsektor mit sechs Stunden pro Zeitschritt

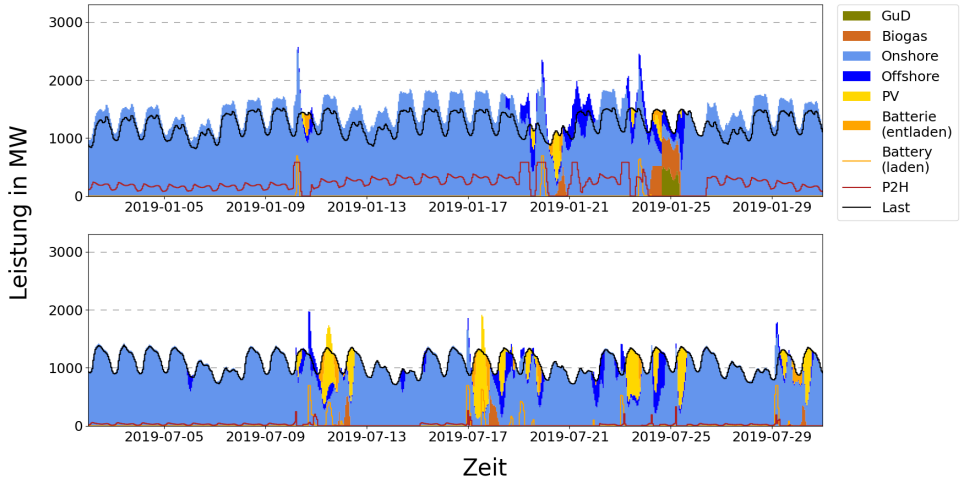

Abbildung A.5.: Leistungsverlauf des „Expansion Examples“ für den Stromsektor mit einer Stunde pro Zeitschritt

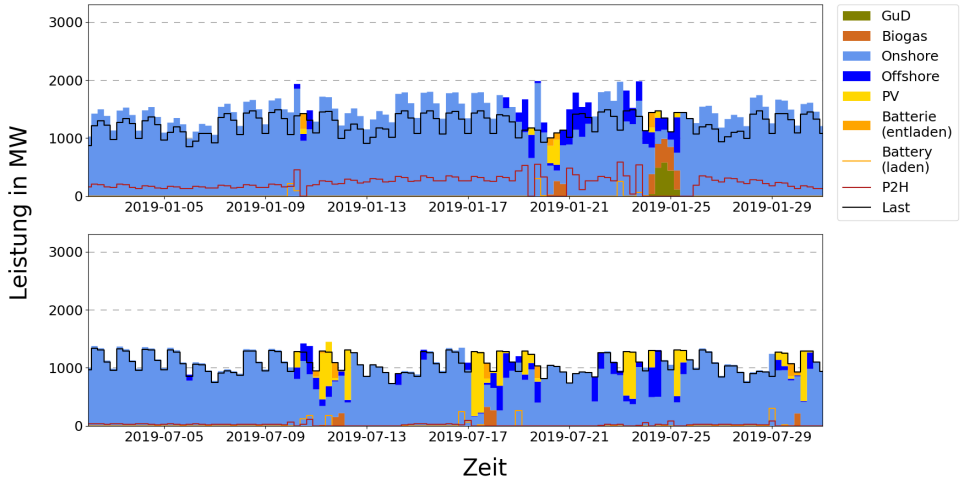

Abbildung A.6.: Leistungsverlauf des „Expansion Examples“ für den Stromsektor mit sechs Stunden pro Zeitschritt

Quellcode A.1: Anpassungen der `techs.yaml` Datei zur Untersuchung des Spores Modus (hot_water transmission analog)

```
...
  electricity transmission:
    essentials:
      name: electricity transmission
      parent: transmission
      carrier: electricity
    constraints:
      energy_eff: 1
      one_way: true
      lifetime: 1  # added
    costs:  # added
      spores_score:  # added
        energy_cap: 0  # added
        interest_rate: 1  # added
...
```

Quellcode A.2: Anpassungen der `model.yaml` Datei zur Untersuchung des Spores Modus

```
...
group_constraints:
  systemwide_emission_cap:
    cost_max:
      emissions:

  systemwide_cost_max:  # added
    cost_max:  # added
      monetary: 1e10  # added

run:
  solver: cbc

  cyclic_storage: false

  mode: spores  # adjusted

  spores_options:  # added
    score_cost_class: 'spores_score'  # added
    slack_cost_group: 'systemwide_cost_max'  # added
    slack: 0.1  # added
    spores_number: 2  # added

  objective_options.cost_class: {'monetary': 1, 'spores_score': 0}
       # adjusted
```