IET
Institut für Energietechnik

TUHH
Technische Universität Hamburg

Masterarbeit

Integration von Free and Open Source Software zur Optimierung von Energiesystemen in ein bestehendes Framework von Energiesystem Modellen

Tim Jonas Hanke

November 2022

IET
Institut für Energietechnik

TUHH
Technische Universität Hamburg

Masterarbeit

Integration von Free and Open Source Software zur Optimierung von Energiesystemen in ein bestehendes Framework von Energiesystem Modellen

Tim Jonas Hanke

Matr.-Nr.: 21599137

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

DOI: https://doi.org/10.15480/882.4877

Hamburg, 28. November 2022

TUHH
Technische Universität Hamburg

Institut für Energietechnik

DR.-ING. K. ABEL-GÜNTHER

Masterarbeit für Herrn Tim Jonas Hanke

Matr.-Nr. 21599137

Integration von Free and Open Source Software zur Optimierung von Energiesystemen in ein bestehendes Framework von Energiesystem Modellen

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, 28. November 2022

(Tim Jonas Hanke)

Abstract

The Transforming Energy Supply System (Modelling) Framework, or Tessif for short, is a framework developed with Python at the Institute of Energy Technology of the Technical University of Hamburg that enables the comparison of different software tools for modeling and optimizing energy systems. For the comparison, Tessif transforms a previously created energy system model into the model of the software tools integrated in Tessif, which then optimizes the energy system. The results of the optimizations are finally unified by Tessif and thus become comparable. Currently, the programs Oemof, PyPSA, FINE and Calliope are integrated in Tessif and can be used within the framework of Tessif for the modeling and optimization of energy systems.

In the context of this work, Urbs, another program for modeling and optimization, is integrated into Tessif. The integration consists to a large extent of the transformation of the energy system model from Tessif into Urbs model formulation (*es2es*) as well as the conversion of the optimization results into Tessifs uniform representation mode (*es2mapping*). Herein, the various modeling approaches and their differences as well as the implementation of the transformation are discussed in detail.

The correct integration of Urbs is ensured by comparing various example energy system models. In addition, the differences between the various software tools are elaborated. Finally, the results of the comparison are used to classify Urbs in the context of Tessif and to identify the advantages and disadvantages of using Urbs.

Kurzfassung

Das Transforming Energy Supply System (Modelling) Framework oder auch kurz Tessif ist ein am Energietechnik Institut der Technischen Universität Hamburg in Python entwickeltes Framework, das den Vergleich verschiedener Softwaretools zum Modellieren und Optimieren von Energiesystemen ermöglicht. Für den Vergleich transformiert Tessif ein zuvor erstelltes Energiesystemmodell in die verschiedenen Modelle der in Tessif integrierten Softwaretools, die das Energiesystem anschließend optimieren. Die Ergebnisse der Optimierungen werden abschließend von Tessif vereinheitlicht und somit vergleichbar. Zum jetzigen Zeitpunkt sind die Programme Oemof, PyPSA, FINE und Calliope in Tessif integriert und können in Tessif für die Modellierung und Optimierung von Energiesystemen verwendet werden.

Im Rahmen dieser Arbeit wird mit Urbs ein weiteres Programm zur Modellierung und Optimierung in Tessif integriert. Die Integration besteht zu einem großen Teil aus der Transformation des Energiesystemmodells von Tessifs in Urbs Modellformulierung (*es2es*) sowie der Umwandlung der Optimierungsergebnisse in Tessifs einheitliche Darstellungsweise (*es2mapping*). In dieser Arbeit wird dabei ausführlich auf die verschiedenen Modellierungsansätze und deren Unterschiede sowie die Umsetzung der Transformation eingegangen.

Die korrekte Integration von Urbs wird anhand eines exemplarischen Vergleichs diverser Beispielenergiesystemmodelle gewährleistet. Zusätzlich werden die Unterschiede zwischen den verschiedenen Softwaretools ausgearbeitet. Abschließend werden die Ergebnisse des exemplarischen Vergleichs genutzt, um Urbs im Rahmen von Tessif einzuordnen und die Stärken sowie Schwächen von Urbs zu identifizieren.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

Nomenklatur

Symbol	Definition	Einheit
Abkürzungen		
BIP	Bruttoinlandsprodukt	
CapEx	Investitionskosten; engl.: capital expenditure	
CHP	Combined Heat and Power	
cllp	Calliope	
DF	Dataframe	
DSM	Demand Side Management	
EE	Energy Economics	
ES	Energiesystem	
es2es	energysystem to energysystem	
es2mapping	energysystem to mapping	
FINE	A Framework for Integrated Energy System Assessment	
FOSS	Free and Open Source Software	
GuD-Kraftwerk	Gas- und Dampfkraftwerk	
IET	Institut für Energietechnik	
KW	Kraftwerk	
LP	Lineare Programmierung; engl.: linear programming	
MILP	Gemischt-ganzzahlige lineare Programmierung; engl.: mixed-integer linear programming	
NLP	Nichtlineare Programmierung; engl.: nonlinear programming	
Oemof, omf	Open Energy Modelling Framework	
OpEx	Betriebskosten; engl.: operational expenditures	
P2H	Power to Heat	
PP	Power Plant	
PSE	Process Systems Engineering	
PyPSA, ppsa	Python for Power System Analysis	
rbs	Urbs	
SoC	State of Charge	
Tessif	Transforming Energy Supply System (Modelling) Framework	
tsam	Time Series Aggregation Module	
TUHH	Technische Universität Hamburg	
Uid	Eindeutiger Identifikator; engl.: unique identifier	
WT	Wind Turbine	

Symbol	Definition	Einheit
Indizes		
el	elektrisch	
ges	gesamt	
spez.	spezifisch	
th	thermisch	

1. Einleitung

Die UN-Klimarahmenkonvention hat sich als Ziel gesetzt, eine gefährliche, anthropogene Störung des Klimasystems zu verhindern, indem eine Stabilisierung der Treibhausgaskonzentration in der Atmosphäre erzielt wird [1]. Im Integrierten Energie- und Klimaprogramm der Bundesregierung (IEKP) hat sich die deutsche Klimaschutzpolitik eine Treibhausgasreduktion um 80% bis 95% bis zum Jahr 2050 als Ziel gesetzt. Der Energiesektor übernimmt dabei eine Schlüsselfunktion bei der Minderung der Treibhausgaskonzentration, da mehr als 80% der Emissionen in Deutschland auf diesen zurückzuführen sind [2].

Energiesystemmodelle können Erkenntnisse und Strategien für die Planung einer kohlenstoffneutralen Zukunft bringen, sodass Modellierungen von Energiesystemen mehr und mehr an Bedeutung gewinnen [3]. Übersichten über verschiedene Modellierungsprogramme können angesichts der Komplexität und Vielfalt an Energiesystemmodellen mit unterschiedlichen Zielsetzungen ein Hilfsmittel für die Auswahl eines Modellierungstools sein. Sie können einen Überblick über bestehende Modelle geben und somit helfen, für eine bestimmte Forschungsfrage geeignete Modelle zu finden [4].
Eine Reihe von Arbeiten befasst sich mit dem Vergleich und der Einordnung verschiedener Modellierungstools [4–7]. Diese Vergleiche fokussieren sich allerdings meistens auf die Einordnung anhand der charakteristischen Eigenschaften des Modells. Zu diesen zählen beispielweise das Ziel, die Methodik oder die temporale und räumliche Auflösung der Modellierung. Dadurch besteht ein Mangel an detaillierten, tiefergehenden Vergleichen von Modellierungstools, die sowohl auf der Nutzung als auch einem Vergleich der Ergebnisse basieren.

Das Transforming Energy Supply System (Modelling) Framework (Tessif) ermöglicht solche Vergleiche durch die Verwendung einer vereinheitlichten Modellformulierung und einer einheitlichen Darstellungsweise der Optimierungsergebnisse. Tessif ist ein am Institut für Energietechnik (IET) der Technischen Universität Hamburg (TUHH) entwickeltes Framework, das im Rahmen dieser Arbeit durch die Integration eines weiteren Modellierungsprogrammes erweitert wird [8].

1.1. Aufbau der Arbeit

Zunächst wird in Kapitel 2 auf die Theorie der Modellierung und Optimierung von Energiesystemen eingegangen. Der Fokus liegt dabei auf der Klassifizierung und Einordnung verschiedener Energiesystemmodelle. Darüber hinaus wird der Begriff Energiesystemmodellierung, wie er im Rahmen dieser Arbeit verwendet wird und zu verstehen ist, erläutert. Abschließend wird in diesem Kapitel ein Überblick über verschiedene Arbeiten gegeben, die sich mit dem Vergleich von Modellierungstools befassen, um Tessif im Rahmen der Literatur einordnen zu können. In Kapitel 3 werden die im Rahmen dieser

Arbeit verwendeten Programme vorgestellt, wobei insbesondere die unterschiedlichen Modellansätze von Tessif und Urbs im Detail dargestellt werden. Kapitel 4 befasst sich mit der Auswahl des zu integrierenden Modellierungstools und der eigentlichen Integration. Dabei werden die verschiedenen zur Integration nötigen, programmierten Funktionen näher erläutert. Durch einen exemplarischen Vergleich in Kapitel 5 der verschiedenen in Tessif implementierten Modellierungsprogramme wird überprüft, ob die Integration von Urbs erfolgreich war und gleichzeitig wird anhand der Ergebnisse eine erste Einordung beziehungsweise Bewertung von Urbs durchgeführt. Abschließend wird in Kapitel 6 der Inhalt der Arbeit kurz zusammengefasst, ein Fazit gezogen und ein Ausblick gegeben.

1.2. Ziel der Arbeit

Ziel dieser Arbeit ist die Auswahl und Integration eines weiteren mit Tessif kompatiblen Modellierungstools in das Framework. Auf diese Weise wird Tessif sowohl qualitativ als auch quantitativ erweitert. Zusätzlich wird im Rahmen dieser Arbeit ein exemplarischer Vergleich zwischen dem neu hinzugefügten Modellierungsprogramm und den bereits in Tessif integrierten Tools Oemof, PyPSA, FINE und Calliope durchgeführt. Mithilfe des Vergleichs wird überprüft, ob die Integration erfolgreich war und gleichzeitig wird eine erste Einordnung des neu hinzugefügten Porgammes ermöglicht.

2. Theorie der Modellierung und Optimierung von Energiesystemen

Dieses Kapitel beschäftigt sich mit den Grundlagen der Modellierung und der Optimierung von Energiesystemen. Zunächst wird der Modellbegriff erläutert und im Weiteren werden verschiedene Möglichkeiten der Modellierung sowie der Klassifizierung von Energiesystemmodellen dargelegt. Abschließend wird die Optimierung von Energiesystemmodellen erläutert, wobei der Fokus auf den mathematischen Grundlagen der Optimierung liegt.

2.1. Definition des Modellbegriffes

Im Allgemeinen können Modelle (lat. Maß, Maßstab) nach Kastens und Büning [9] als Abbild eines vorhandenen Originals definiert werden. Dabei kann sowohl das Modell als auch das Modellierte konkret sowie abstrakt sein. Ein Modell ist absichtlich nicht originalgetreu, sondern vereinfacht das Original, sodass nur die für den intendierten Verwendungszweck relevanten Eigenschaften des Originals abgebildet werden. Im wissenschaftlichen Bereich soll meist die Untersuchung eines Forschungsthemas durchgeführt werden, die sich am Original als zu kosten- oder zeitintensiv erweisen würde [9]. Im Verlauf dieser Arbeit werden verschiedene Modelle von Energiesystemen sowie die abstrakten Modellierungsmethoden betrachtet. Die Modelle erfüllen dabei den Zweck, eine Optimierung des jeweiligen Energiesystems bezüglich der Kosten durchführen zu können.

2.2. Klassifizierung von Energiesystemmodellen anhand des Modellierungsansatzes

Grundsätzlich können Energiesystemmodelle nach Subramanian, Gundersen und Adams [10] gemäß Abbildung 2.1 nach der Art des Modellierungsansatzes kategorisiert werden. Dabei lassen sich die Energiesystemmodelle zunächst in drei verschiedenen Klassen einteilen:

- rechnerische Modelle (computational)
- mathematische Modelle (mathematical)
- physikalische Modelle (physical)

Nach Fisher und Henzinger [11] sind Rechenmodelle formale Modelle, die sich durch eine operative primäre Semantik auszeichnen. Das bedeutet, das Modell schreibt eine Abfolge von Schritten beziehungsweise Anweisungen vor, die von einem Computer ausgeführt werden können.
Im Gegensatz dazu sind mathematische Modelle eine Reihe von Gleichungen, die Beziehungen zwischen verschiedenen bedeutsamen Variablen und Parametern beschreiben. Mithilfe numerischer Verfahren können Annäherungen an mathematische Modelle entwickelt werden, die dann von einem Computer gelöst werden können. Sofern mathematische Modelle auf einem Computer numerisch implementiert werden, schlagen Fisher und Henzinger [11] zudem vor, den Begriff „Ausführung“ (execution) für Rechenmodelle und „Simualtion“ (simulation) für mathematische Modelle zu verwenden [11].
Physikalische Modelle sind Modelle, bei denen die Phänomene des realen Systems tatsächlich auftreten. Allerdings sind die Modelle in kleinerem Maßstab oder mit geringerer Komplexität als das originale System ausgeführt [10].

Da die in dieser Arbeit verwendeten Programme den mathematischen Modellierungsansatz verwenden, wird sowohl der rechnerische als auch der physikalische Ansatz nicht näher erläutert. Die mathematischen Modelle können nach Subramanian, Gundersen und Adams [10] weiter in statistische (statistical) und mechanistische (mechanistical) Modelle unterteilt werden. Die statistischen Modelle nutzen dabei Techniken wie Regression und Optimierung oder selbstoptimierte Steuerung, um aus Eingabedaten mithilfe einer Reihe mathematischer Beziehungen Ausgabedaten abzuleiten. Die mechanistischen Modelle verwenden stattdessen grundlegende Theorien wie Strömungsmechanik, Thermodynamik, Ökonomie oder Massen- und Energiebilanzen. Diese liefern die Modellstruktur und generieren die Gleichungen, um die Phänomene des realen Systems zu beschreiben.
Alternativ kann eine Einteilung mathematischer Modelle, wie zum Beispiel in diskrete und kontinuierliche Modelle, stationäre und dynamische Modelle sowie deterministische und stochastische Modelle erfolgen [10].

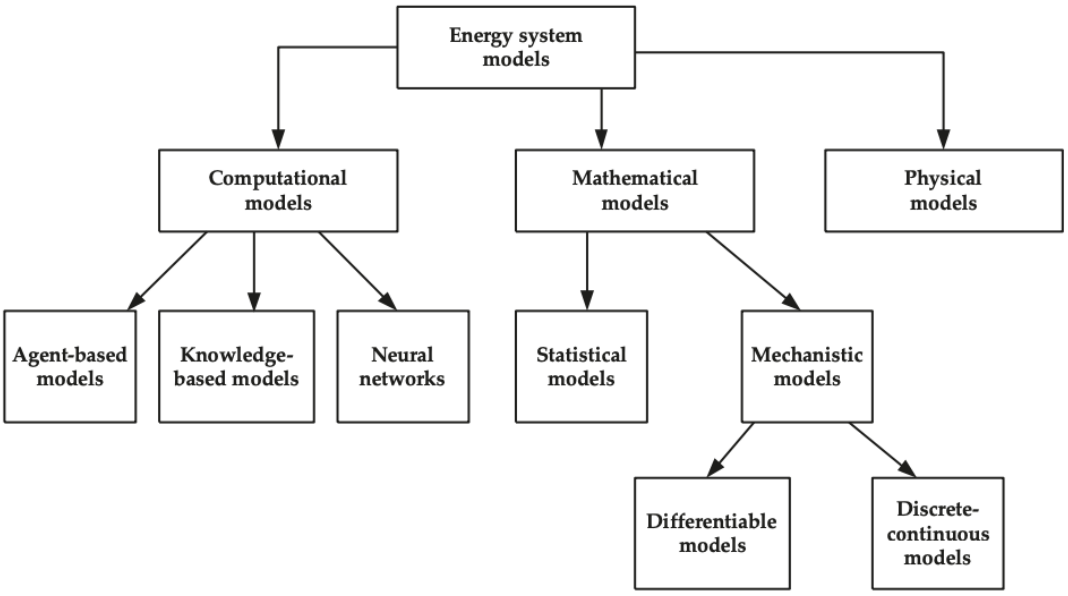


Abbildung 2.1.: Klassifikation von Energiesystemmodellen nach dem Modellierungsansatz nach [10]

2.3. Klassifizierung von Energiesystemmodellen anhand des Forschungsgebietes

Neben der Klassifizierung über den Modellierungsansatz, können Energiesystemmodelle anhand ihrer Verwendungszwecke auch verschiedenen Forschungsgebieten zugeordnet werden. Dabei werden nach Subramanian, Gundersen und Adams [10] im Wesentlichen Energiesysteme in der Forschung in zwei verschiedenen Bereichen untersucht, dem *Process Systems Engineering* (PSE) und der *Energy Economics* (EE) [10].
Die Tabelle 2.1 veranschaulicht die grundlegenden Unterschiede zwischen der PSE- und der EE-Modellierung anhand verschiedener Kriterien. In den nachfolgenden Abschnitten 2.3.1 und 2.3.2 werden zudem die wesentlichen Ziele beziehungsweise Merkmale der beiden Forschungsbereiche näher erläutert und mit Modellbeispielen verdeutlicht.

Bereich	**Process Systems Engineering (PSE)**		
Art der Variablen			
endogen	Technologische Variablen (z.B. Temperatur, Druck, Enthalpie, freie Gibbs-Energie, Prozessgröße)		
exogen	Ökonomische Variablen (z.B. Rohstoffpreise, Ausrüstungspreise, Produktnachfrage, Zinssätze) Umwelt-Variablen (z.B. Treibhauspotenzial, Ökotoxizität, Ressourcenverknappung)		
Theoretische Grundlagen	Thermodynamik, Strömungsmechanik, Kinetik		
Stufe der Aggregation von Technologien	Einheitsoperation, Verarbeitungsanlage, Lieferkette		
Räumliche Skalierung	Lokal, Regional, National, Global		
Entscheidungsfindungs Hierachie	Strategisch	Taktisch	Operativ
Zeitliche Sklaierung	mehrere Jahre	Tage bis Wochen	Sekunden bis Stunden
Klassiche Verwendung	- Prozessdesign und Integration - Lieferkettendesign/-infrastruktur	- Produktionsplanung und Terminplanung	- Prozesssteuerung - Flexibler Betrieb
Aktuelle Trends	- Mehrskalige Systemtechnik - Nachhaltige Prozessanalyse und -gestaltung		

Bereich	**Energy Economics (EE)**
Art der Variablen	
endogen	Ökonomische Variablen
exogen	Technologische Variablen, Umwelt-Variablen
Theoretische Grundlagen	Ökonomie (Produzententheorie, Konsumententheorie und Marktgleichgewicht)
Stufe der Aggregation von Technologien	Gesamter Energiesektor, Alle Wirtschaftssektoren
Räumliche Skalierung	Regional, National, Global
Entscheidungsfindungs Hierachie	Strategisch
Zeitliche Sklaierung	mehrere Jahre
Klassiche Verwendung	- Nachhaltige Energiepolitikplanung - Langfristige Energieprognosen
Aktuelle Trends	- nicht im Rahmen der Arbeit untersucht

Tabelle 2.1.: Klassifikation von Energiesystemmodellen nach Bereichen nach [10]

2.3.1. Process Systems Engineering (PSE)

Die Modellierung von Energiesystemen in der PSE wird genutzt, um einen Einblick in die technologische Leistungsfähigkeit des Energiesystems zu gewinnen und um optimale Entscheidungen in Bezug auf Planung, Betrieb und Steuerung des Energiesystems und einzelner Komponenten treffen zu können.
Ursprünglich hat sich der PSE-Ansatz auf die Modellierung chemischer Umwandlungssysteme konzentriert. Diese Systeme entsprachen dabei dem Maßstab einzelner Verarbeitungsanlagen. Die Systeme können auch als Sammlung mehrerer miteinander verbundener Subsysteme ein komplexes Netzwerk ergeben.
Wie bereits zuvor erwähnt, werden Anlagenbetriebsmodelle in der Regel in Zusammenspiel mit Simulation und Optimierung für eine Vielzahl an Zwecken verwendet. So können sie beispielweise für die Synthese von Reaktionswegen, das Designen eines Prozessfließschemas oder aber für die Betriebs-/Steuerungsmodellierung eingesetzt werden.
Da PSE-Modelle auf ingenieurwissenschaftlichen Grundlagen wie Thermodynamik, Strömungsmechanik oder Kinetik basieren, konzentrieren sich die Energiesystemmodelle üblicherweise auf die Beschreibung der technologischen Eigenschaften der Komponenten. Wirtschaftliche, ökologische oder soziale Merkmale werden dabei als exogene Variablen an das Modell weitergegeben. So werden zum Beispiel auf Prozessebene Betriebsbedingungen wie Temperaturen, Drücke oder Durchflussraten explizit modelliert, während Parameter wie Kraftstoffpreise oder Umwelteinflüsse exogen berücksichtigt werden [10].
Nach Grossmann und Westerberg [12] lässt sich der Anwendungsbereich von PSE erweitern, sodass alle Ansätze für eine verbesserte Entscheidungsfindung beim Erstellen und Betrieb chemischer Lieferketten berücksichtigt werden [12].
Allgemein können dem PSE-Forschungsansatz zwei Anwendungsgebiete untergeordnet werden. Zum einen das *Multi-Scale Systems Engineering* und das *Modellieren von Nachhaltigkeitskriterien*. Beim *Multi-Scale Systems Engineering* handelt es sich um die Entwicklung von Modellen mit hoher Genauigkeit (High-fidelity model). Um das Gesamtsystemverhalten in einem großen Maßstab möglichst genau zu erfassen, werden Informationen genutzt, die durch Modelle mit höherer Auflösung in kleinerem Maßstab zur Verfügung gestellt werden. Das *Modellieren von Nachhaltigkeitskriterien* befasst sich primär mit der Suche beziehungsweise Entwicklung von neuartigen Prozessen mit verbesserter wirtschaftlicher und ökologischer Leistung [10].
Da der PSE-Ansatz im Rahmen dieser Arbeit nicht weiter relevant ist, wird er nicht ausführlicher erläutert.

2.3.2. Energy Economics (EE)

Der Themenbereich *Energy Economics* beschäftigt sich mit den wirtschaftlichen Aspekten der Modellierung und Simulation beziehungsweise Optimierung von Energiesystemen. Im Fokus steht dabei insbesondere die optimale Allokation begrenzter Energieressourcen zur Befriedigung der Verbrauchernachfrage. Die EE-Modelle werden üblicherweise zur Unterstützung in der strategischen Entscheidungsfindung auf regionaler, nationaler oder globaler Ebene genutzt.
EE-Modelle verwenden dabei die Prinzipien der Ökonomie als theoretische Grundlage. Demzufolge werden die wirtschaftlichen Aspekte des Energiesystems endogen und die technischen und ökologischen Merkmale exogen modelliert [10]. Nachfolgend sind die drei wichtigsten Prinzipien aufgelistet und deren Bedeutung kurz erläutert.

- **Verbrauchertheorie:** Die Verbrauchertheorie besagt, dass rationale Verbraucher ihr Einkommen für Waren ausgeben, welche die größte Befriedigung bieten. (Dient zur Bestimmung der Menge bestimmter nachgefragter Güter wie Strom oder Brennstoffe)
- **Produzententheorie:** Die Produzententheorie erklärt, dass Produzenten die ideale Menge an Produkten herstellen, um die Rentabilität zu maximieren.
- **Marktgleichgewicht:** Das Marktgleichgewicht löst den Konflikt zwischen Verbraucher- und Produzententheorie, indem der Begriff des „Gleichgewichtspreises“ eingeführt wird. Dadurch entspricht die gelieferte Warenmenge der nachgefragten Warenmenge.

Der EE-Ansatz für Energiesysteme umfasst insbesondere Energienachfrage- und Energieangebotsprognosemodelle, Top-down- und Bottom-up-Energiesystemmodelle [10].

Modelle zur Prognose von Angebot und Nachfrage

Modelle, die für die Bestimmung eines zukünftigen Energiebedarfs verwendet werden, sogenannte **Bedarfsprognosemodelle,** nutzen verbrauchertheoretische und statistische Daten, um eine Energienachfrage zu prognostizieren. Bedarfsprognosemodelle verwenden dabei entweder ökonometrische Ansätze oder den Ansatz der Endverwendungsbuchhaltung. Der ökonometrische Ansatz basiert auf der Beziehung zwischen der nachgefragten Energiemenge und mehreren treibenden Variablen wie BIP, Durchschnittseinkommen, Energiepreis, Technologiemerkmalen und verschiedenen weiteren Werten. Die Abhängigkeiten von Energienachfrage und treibenden Variablen werden dabei durch die statistische Analyse von historischen Daten gewonnen.
Bei der Endverbrauchsbuchhaltung (*end-use accounting*) wird im Unterschied zum ökonometrischen Ansatz zunächst der Energiebedarf in eine Reihe homogener Kategorien (z.B. Verkehr, Wohnen, Industrie, Gewerbe) unterteilt. Mithilfe des historischen Energiebedarfs und den Hauptreibern wird für jeden dieser Sektoren der Energiebedarf prognostiziert. Dadurch kann die Nachfrage mit einem höheren Granularitätsgrad bestimmt werden, allerdings wird auch eine umfangreichere Datengrundlage benötigt und es werden möglicherweise nicht alle sektorübergreifenden Interaktionen berücksichtigt. In der Praxis wird daher häufig eine Kombination aus beiden Ansätzen verwendet [10].

Die **Versorgungsprognosemodelle**, zur Vorhersage der Energieversorgung, variieren in Abhängigkeit der betrachteten Primärenergiequelle. Erneuerbare Energien wie Solar- oder Windenergie können meteorologische Modelle erfordern, um die fluktuierenden Eigenschaften der erneuerbaren Energiequellen abzubilden. Des Weiteren können detaillierte Beschaffungsmodelle genutzt werden, um zum Beispiel die Verfügbarkeit von Biomasse zu modellieren. Zum Bestimmen der Verfügbarkeit fossiler Reserven können exponentielle Zerfallsproduktionsmodelle in Kombination mit Daten zur Exploration fossiler Brennstoffe oder die Analyse von Investitionstrends diverser Unternehmen verwendet werden. Alternativ kann eine Prognose der Versorgung ähnlich zur Bedarfsprognose mithilfe des ökonometrischen Ansatzes erfolgen [10].

Top-Down-Modelle

Beim Top-Down Modellansatz werden alle Sektoren einer gesamten Wirtschaft modelliert, wobei die Energie als Input für jeden dieser Sektoren betrachtet wird. Der Top-Down-Ansatz fokussiert sich primär auf die Interaktionen beziehungsweise Auswirkungen zwischen dem Energiesektor und den Wirtschaftssektoren. So kann zum Beispiel ein Wachstum im Industriesektor zu erhöhten Energiekonsum und somit zu höheren Energiepreisen führen. Alternativ kann aber auch eine Produktivitätssteigerung Einfluss auf die Energieerzeugungskosten und den Energiepreis haben [10].
Nach Farzaneh [13] ist der Begriff „Top-Down“ im Zusammenhang von Energiesystemmodellen gleichbedeutend mit dem Begriff „aggregiert“. Dies ist darauf zurückzuführen, dass Top-Down-Modelle meist sehr stark aggregierte Datengrundlagen nutzen. Durch diese Aggregation ist dieser Ansatz häufig simpler und weniger datenintensiv. [13] Ein weiterer Vorteil des Top-Down-Modells ist nach Subramanian, Gundersen und Adams [10] die konsequente Berücksichtigung der gesamten Wirtschaft. Zu den Nachteilen zählen verallgemeinerte Ergebnisse mit meist geringer Erklärungskraft, eine unzureichende Betrachtung der technologischen Aspekte sowie unzureichende Details in geringeren Maßstäben infolge der Aggregation der Daten [10].
Demzufolge eignen sich Top-Down-Modelle insbesondere als Unterstützung zur politischen Gestaltung von Steuern, Energiesubventionen oder Klimaschutzgesetzen auf regionaler, nationaler oder globaler Ebene [10].

Bottom-Up-Modelle

Bottom-Up-Modelle zeichnen sich durch einen hohen technologischen Detailierungsgrad aus und werden insbesondere bei der Planung der Wirtschaftlichkeit, der Bewertung von Technologieoptionen, der Simulation nachfrageseitiger Energiesparmaßnahmen sowie der Effizienzbetrachtung der Angebotsseite eingesetzt. Durch die detaillierte Betrachtung verschiedener Technologien werden disaggregierte Daten verwendet, sodass Bottom-Up-Modelle datenintensiv ausfallen [13].
Bottom-Up-Modelle benötigen Module, die den Energiebedarf sowie die Primärenergieversorgung aus sowohl Importen als auch Energieressourcen prognostizieren können. Dafür kann beispielsweise der Endverbrauchsbuchungsansatz aus 2.3.2 genutzt werden. Darüber hinaus werden noch die wirtschaftlichen Merkmale der einzelnen im Modell verwendeten Technologien benötigt. Zu diesen zählen zum Beispiel die Investitions- und Betriebskosten, aber auch Wartungskosten und Kapazitäten von Verarbeitungsanlagen. Mit dem Ziel, die strategische Planung und Politikgestaltung zu unterstützen, werden Bottom-Up-Modelle in der Regel mit mehrperiodischen Optimierungsschemata verwendet. Das Optimierungsproblem kann dabei nach Subramanian, Gundersen und Adams [10] folgendermaßen beschrieben werden:

> Ausgehend von einer Reihe von Endverbrauchern und Prognosen für deren Nachfrage über einen bestimmten (in der Regel langfristigen) Zeithorizont, einer Reihe von in Frage kommenden Primärenergiequellen und einer Reihe von entsprechenden Umwandlungs- und Verteilungstechnologien ist die optimale Energiesystemkonfiguration zu bestimmen, die die Gesamtkosten minimiert (oder die Gesamteffizienz maximiert), so dass die Energienachfrage in jedem Zeitraum durch das Angebot gedeckt wird [10].

Das Schlüsselprinzip des Bottom-Up-Ansatzes ist die Markträumungsfunktion (*market-clearing-condition*) in Zusammenspiel mit der von jeder Komponente erfüllten Energiebilanz. Die Lösung des zuvor definierten, mehrperiodischen Optimierungsproblems präsentiert die Konfiguration des Energiesystems mit dem optimalen Primärenergiemix und der optimalen Auswahl der gegenwärtigen und zukünftigen Umwandlungs- und Verteilungstechnologien.

Abbildung 2.2 zeigt ein Schema eines Bottom-Up-Modells auf nationaler Ebene. Die über die Modellgrenzen nach innen gerichteten Pfeile repräsentieren die Eingaben für das Modell, während die nach außen zeigenden Pfeile die Modellausgaben darstellen. Im Modell sind Umwandlungsprozesse (Kohleverarbeitung, Raffinerien, Kraftwerke, KWK-Anlagen) und Verteilprozesse (Gasnetze, Transportnetze, Fernwärmenetze) in Rot dargestellt. Die Primärenergie aus nationalen Quellen und Importen ist blau und die Endverbraucherkategorien (Industrie, Gewerbe, Haushalte und Verkehr) sind orange abgebildet. In Grün sind die von der Energie erbrachten Dienstleistungen wie beispielweise Prozessenergie oder Strom dargestellt.

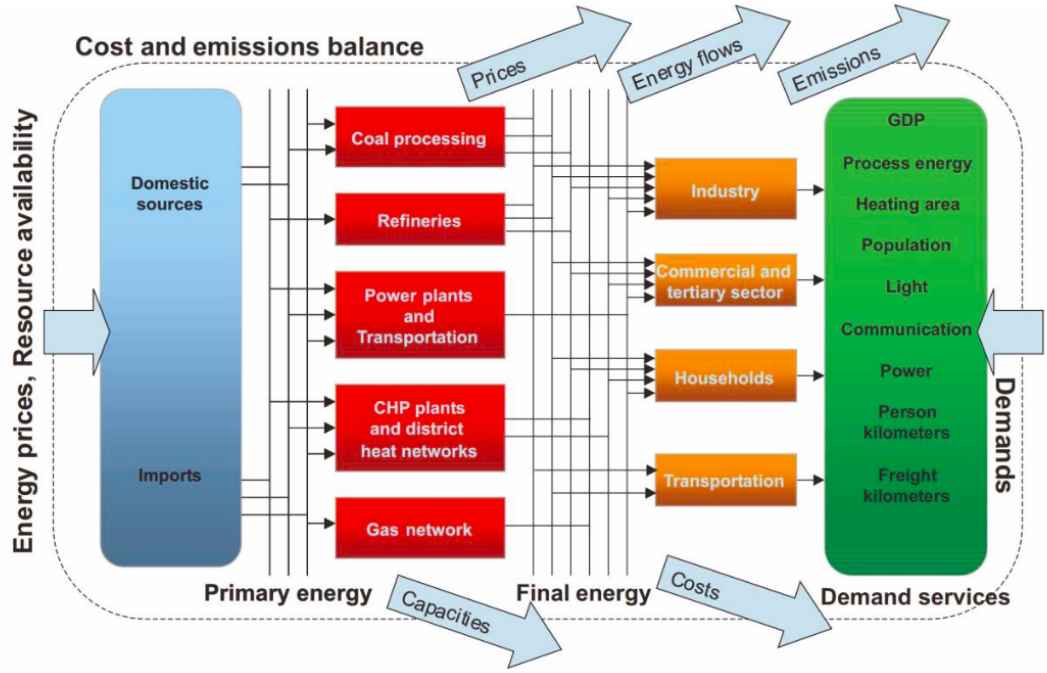


Abbildung 2.2.: Schema eines Bottom-Up-Modells auf nationaler Ebene nach [10]

Bottom-Up-Modelle können insbesondere als Grundlage für politische Entscheidungen in Bezug auf die Einführung neuer Energietechnologien verwendet werden. Durch den hohen technologischen Detailierungsgrad verfügen Bottom-Up-Modelle zudem über viel Erklärungskraft. Allerdings erfordert dieser hohe Detailierungsgrad auch eine große Menge an Daten sowie einen hohen Rechenaufwand [10].

2.4. Energiesystemmodellierung im Rahmen dieser Arbeit

Im Rahmen dieser Arbeit kann grundsätzlich zwischen zwei Arten von Modellen unterschieden werden. Einerseits werden die mathematischen Modelle der einzelnen Optimierungsprogramme (Oemof, PyPSA, FINE, Calliope, Urbs) aus Abschnitt 3 genutzt, andererseits wird der Modellbegriff für die Energiesystemmodelle, welche durch den User erstellt werden, genutzt. Diese werden anschließend von der jeweiligen Optimierungssoftware in das entsprechende mathematische Modell gewandelt. Somit werden im Grunde keine zwei verschiedenen Modelle, sondern nur ein Modell eines Energiesystems mit unterschiedlichen Darstellungsweisen verwendet. Zum einen gibt es die mathematische Darstellung der Optimierungssoftware und zum anderen die für den Nutzer vereinfachte Darstellung zum Beispiel mit einzelnen Komponenten oder in tabellarischer Form.
Die in dieser Arbeit verwendeten Softwares können dem Bereich der *Energy Economics* (siehe Abschnitt 2.3.2) zugeordnet werden und basieren demzufolge auf den Prinzipien der Ökonomie (Verbrauchertheorie, Produzententheorie, Marktgleichgewicht). Darüber hinaus nutzen die Tools den zuvor beschriebenen Bottom-Up-Ansatz aus Abschnitt 2.3.2. Die Abbildung 2.3 verdeutlicht die Nutzung von Energiesystemmodellierung und -optimierung, wie sie im Rahmen dieser Arbeit durchgeführt wird.

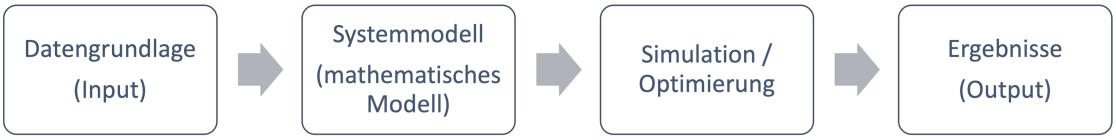


Abbildung 2.3.: Schema der Modellierung und Optimierung eines Energiesystems im Rahmen dieser Arbeit

Die verschiedenen Optimierungsprogramme benötigen, wie in Abbildung 2.3 gezeigt, zunächst Input-Daten. Diese Datengrundlage besteht einerseits aus wetterbasierten Einspeisedaten erneuerbarer Energien, den Energieverbräuchen sowie dem eigentlichen Energiesystemmodell (Struktur und technologische Parameter). Die Vorgabe der Daten erfolgt dabei spezifisch für jedes Optimierungsproramm wie zum Beispiel Oemof, PyPSA oder Urbs. So kann das Energiesystem in tabellarischer Form oder als komponentenbasiertes Modell angegeben werden.
Aus den Input-Daten können die einzelnen Programme anschließend das eigentliche mathematische Modell generieren und dieses in einem nächsten Schritt simulieren beziehungsweise optimieren. Abschließend können die Ergebnisse der Simulation/Optimierung ausgegeben werden. Beim Bottom-Up-Ansatz bestehen die Ergebnisse der Optimierung in der Regel, wie in Abbildung 2.2 gezeigt, aus den Kosten, Emissionen, Kapazitäten der Technologien sowie Energieflüssen des Energiesystems.

2.5. Grundlagen der Optimierung von Energiesystemmodellen

Dieser Abschnitt befasst sich mit den Grundlagen der Optimierung von Energiesystemmodellen. Zunächst wird der Begriff Optimierung näher erläutert, wobei insbesondere die Differenzen von Optimierung und Simulation im Fokus stehen. Darüber hinaus werden

mit der linearen Programmierung und der gemischt-ganzzahligen linearen Programmierung zwei der wesentlichen mathematischen Grundlagen der Optimierung präsentiert.

2.5.1. Simulation und Optimierung

Bei der Simulation wird ein Modell eines Systems verwendet, um Informationen über mögliche Verhaltensweisen des tatsächlichen Systems zu erhalten. Ein System kann dabei als Reihe von interagierenden Komponenten oder Entitäten definiert werden. Diese arbeiten innerhalb des Systems zusammen, um ein gemeinsames Ziel zu erreichen. So kann beispielweise eine Fertigungsanlage mit verschiedenen Maschinen, Förderbändern, Produktionsplänen und hergestellten Produkten als ein System betrachtet werden.
Simulationsmodelle werden dabei primär verwendet, um die Leistung eines Systems zu messen beziehungsweise um diese abzuschätzen. Das Ergebnis der Simulation kann genutzt werden, um Informationen über mögliche Fehler innerhalb des Systems zu erhalten und somit den Betrieb der Anlage verbessern [13].

Bei Optimierungen wird hingegen eine Zielfunktion wie beispielweise eine Kostenfunktion definiert. Mithilfe eines mathematischen Algorithmus kann der Minimal- beziehungsweise Maximalwert dieser Funktion bestimmt werden. Optimierungen werden verwendet, um Systeme zu verbessern, indem zum Beispiel Kosten reduziert oder aber die Effektivität gesteigert wird [14].
Optimierungsmodellen müssen neben der Zielfunktion noch zusätzlich Entscheidungsvariablen vorgegeben werden, welche die Freiheitsgrade festlegen. Darüber hinaus können noch Restriktionen, also zu berücksichtigende Einschränkungen des Modells, definiert werden. Sofern die Zielfunktion, die Entscheidungsvariablen sowie die Restriktionen in Form eines expliziten Gleichungssystems aus Gleichungen und Ungleichungen ausgedrückt werden können, kann eine optimale Lösung durch verschiedene Optimierungsmethoden generiert werden [15].
Die im Rahmen dieser Arbeit verwendeten Programme nutzen die Methode der Optimierung. In Kapitel 5 werden durch das Softwaretool Tessif Kostenoptimierungen von Energiesystemen mit Oemof, PyPSA, FINE, Calliope und Urbs durchgeführt.

2.5.2. Mathematische Grundlagen der Optimierung

Dieser Abschnitt befasst sich mit den mathematischen Grundlagen der Optimierung. Modellierte Optimierungsprobleme können zum Lösen an einen Solver übergeben werden. Es gibt viele verschiedene Solver, darunter auch frei zugängliche Solver wie den *CBC (Coin or branch and cut) Solver* [16] oder das *GLPK (Gnu Linear Programming Kit)* [17]. In Abhängigkeit des verwendeten Solvers kann es bei Optimierungen zu unterschiedlichen Rechenzeiten kommen. Die Solver arbeiten mit den mathematischen Grundlagen der linearen Programmierung (LP) sowie der gemischt-ganzzahligen linearen Programmierung (MILP). Diese mathematischen Methoden werden nachfolgend näher erläutert.

Linear Programming (LP)

Die Lösungstechnologien der linearen Optimierung (*Linear Programming*, LP) können verwendet werden, sofern sowohl die Zielfunktion als auch die Restriktionen des Optimierungsmodells Linearkombinationen der Entscheidungsvariablen sind [15].

Lineare Probleme gehören, in Bezug auf theoretische Aussagen sowie die effektiven Lösungsmethoden, zu den einfachsten Optimierungsaufgaben. Bei der linearen Optimierung wird ein System $A \in \mathbb{R}^{n,m}$ durch Anpassen der Entscheidungsvariablen oder auch Aktivitäten $x = (1, \ldots, n)^\top$ und unter Berücksichtigung der Nebenbedingungen $b = (1, \ldots, m)^\top$ minimiert beziehungsweise maximiert.
Im Allgemeinen kann jede Aufgabe der Form

$$\begin{aligned} &\max \,/\, \min\, c^\top \cdot x \\ &A \cdot x \begin{matrix} \leq \\ \equiv \\ \geq \end{matrix} b \end{aligned} \tag{2.1}$$

als lineare Optimierungsaufgabe bezeichnet werden. Dabei ist das Ziel die Maximierung oder Minimierung der Zielfunktion $c^\top \cdot x$ unter Einhaltung aller Nebenbedingungen. Die Nebenbedingungen beziehungsweise Restriktionen verwenden dabei eines der drei Relationszeichen $\leq$, $=$ oder $\geq$.
Bei der Energiesystemoptimierung implizieren die Ziele der Minimierung von Kosten oder Emissionen zusätzlich eine Nichtnegativität der Entscheidungsvariablen x. Eine solche Kostenminimierung eines Energiesystems kann in Form von Gleichung 2.2 beschrieben werden.

$$\begin{aligned} \min\, & c^\top \cdot x \\ & A \cdot x \leq b \\ & x \geq 0 \end{aligned} \tag{2.2}$$

Der optimale Zielfunktionswert der Minimierung aus Gleichung 2.2 und somit der minimal erreichbare Funktionswert von $c^\top \cdot x$ unter Berücksichtigung der Restriktionen kann in Normalform mit

$$\min\{c^\top x : Ax \leq b, x \geq 0\} \tag{2.3}$$

beschrieben werden [18]. Lineare Probleme sind im Grunde ein System aus Gleichungen und Ungleichungen. Eine Wertekombination der Entscheidungsvariablen, die alle Restriktionen erfüllt, kann als Lösung des Systems bezeichnet werden. Dabei ist zu beachten, dass lineare Optimierungsprobleme meist eine unendliche Anzahl an zulässigen Lösungsmöglichkeiten innehaben, da die Restriktionen Freiräume bei der Lösungsauswahl lassen. Das Ziel ist allerdings das Finden der optimalen Lösung aus der zulässigen Lösungsmenge anhand der Zielfunktion.

Kleine lineare Optimierungsmodelle mit nur zwei Variablen können grafisch ohne Computer gelöst werden. Beinhaltet das Problem nur zwei Entscheidungsvariablen, so entsprechen die Einschränkungen Geraden in der Ebene und definieren einen zulässigen Bereich. Wird der Zielfunktionswert konstant gesetzt ($c = c_0$), kann eine so definierte Isogewinngerade eingezeichnet werden. Diese Gerade kann dann in Abhängigkeit der Optimierungsart (Maximierung oder Minimierung) parallel bis zu einer optimalen Ecke des zulässigen Bereichs verschoben werden, um so den optimalen Zielfunktionswert zu erhalten [15]. Die nachfolgende Abbildung 2.4 zeigt die grafische Lösung von 2-dimensionalen LP-Modellen am Beispiel einer Maximierung.

Bei komplexeren linearen Problemen ist ein Bestimmen der Lösung mit der grafischen Methode nicht mehr möglich. Stattdessen kann das **Simplex-Verfahren** zum Lösen verwendet werden. Das Simplex-Verfahren besteht im Grunde aus zwei Phasen, wobei in der ersten Phase zunächst eine zulässige Lösung des linearen Problems bestimmt wird. In

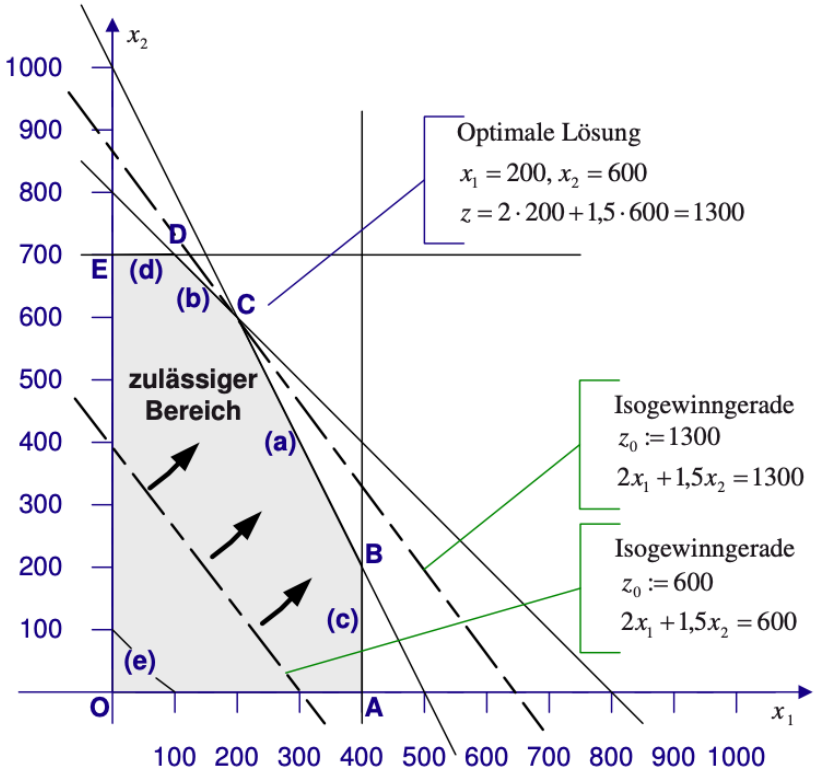


Abbildung 2.4.: Grafische Lösung von 2-dimensionalen LP-Modellen nach [15]

der zweiten Phase wird ausgehend von dieser zulässigen Lösung durch iterative Verbesserung die optimale Lösung gesucht. Unter allgemeinen Bedingungen konvergiert diese Methode immer gegen die optimale Lösung [15].
Das Laufzeitverhalten der Simplex-Methode fällt für viele Praxismodelle gut aus. Allerdings kann dieses theoretisch auch sehr schlecht ausfallen, da die Laufzeit des Simplex-Verfahren exponentiell im Hinblick auf die Problemgröße steigt. Daher kann es von Vorteil sein, die Ellipsoid-Methode, welche auch **Innere-Punkte-Verfahren** genannt wird, zu nutzen, da bei dieser die Laufzeit nur polynominell in Relation zur Problemgröße wächst. Im Gegensatz zum Simplex-Verfahren bewegt sich die Methode nicht an der Oberfläche des zulässigen Bereichs, sondern es werden, wie der Name bereits impliziert, auch die inneren Punkte des zulässigen Bereichs berücksichtigt. Die Abbildung 2.5 verdeutlicht diesen Unterschied zwischen der Simplex-Methode und dem Innere-Punkte-Verfahren [15].

Abschließend ist zu erwähnen, dass keine ideale Methode zum Lösen von linearen Problemen existiert und für viele Modelle Varianten des Simplex-Verfahrens am schnellsten sind. Demzufolge beinhalten effiziente Solver im Allgemeinen beide Methodenklassen (Simplex- und Innere-Punkte-Verfahren) [15].

Mixed-Integer linear Programming (MILP)

Viele praktische Probleme lassen sich nur sinnvoll mit ganzzahligen Variablen darstellen. Insbesondere wenn es um die Teilbarkeit von Ressourcen geht. So ergibt es wenig

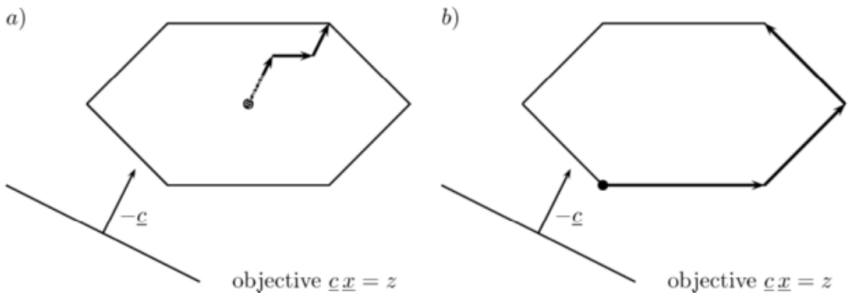


Abbildung 2.5.: Grafischer Vergleich von a) Innere-Punkte-Verfahren und b) Simplex-Verfahren nach Horst W. Hamacher [19]

Sinn, Ressourcen wie Arbeitskräfte beziehungsweise Menschen, Fahrzeuge oder Maschinen mit einer dezimalen Anzahl abzubilden. Gemischt-ganzzahlige lineare Optimierungsprobleme oder auch kurz MILP (Mixed-Integer linear Programming) berücksichtigen eben solche ganzzahligen Variablen. Der Unterschied zwischen MILP und der linearen Optimierung entsteht dabei nur durch die bei einigen oder allen Variablen zusätzlich auftretende Ganzzahligkeitsbedingung. Die Zielfunktion und die Restriktionen bleiben weiterhin linear, nur die Variablen sind diskret und nicht mehr kontinuierlich. Gemischtganzzahlige Optimierungsmodelle können dabei sowohl unteilbare Ressourcen als auch binäre ja/nein-Entscheidungen über 1/0-Variablen abbilden [15].
Die Gleichung 2.4 zeigt die allgemeine Form eines gemischt-ganzzahligen Optimierungsproblems nach Bauer et al. [20].

$$f(x,y) \rightarrow min, \begin{cases} g(x,y) = 0 \\ h(x,y) \leq 0 \end{cases} \quad \text{für } x \in \mathbb{R}^n,\ y \in \{0,1\}^m \tag{2.4}$$

Dabei ist $f(x,y)$ die Zielfunktion, welche minimiert werden soll. Des Weiteren bilden $g(x,y)$ und $h(x,y)$ die Restriktionen in Form von Gleichungen und Ungleichungen ab. Die Variablen des Problems können in kontinuierliche Variablen $x \in \mathbb{R}$ und in diskrete beziehungsweise Binärvariablen $y \in \{0,1\}$ unterteilt werden [20].
Für gemischt-ganzzahlige Optimierungsprobleme existieren oft sehr große Anzahlen von Lösungsmöglichkeiten. Daher ist eine vollständige Enumeration der möglichen Lösungen bei steigender Größe der Probleminstanz zu rechenintensiv. Aus diesem Grund werden bei MILP-Problemen nicht alle möglichen Lösungen untersucht, sondern viele Lösungsmöglichkeiten aufgrund logischer Aspekte ausgeschlossen, da auf Basis des Problems manche Lösungen nicht optimal sein können [15].
Eine der Lösungsmethoden ist nach [21] das Branch & Bound-Verfahren. Beim Branch & Bound-Verfahren wird zunächst in einem ersten Schritt durch eine LP-Relaxation das Problem in ein lineares Problem (LP_0) gewandelt und ohne Berücksichtigen der Ganzzahligkeitsbedingung gelöst. Auf diese Weise erhält man einen maximalen Zielfunktionswert, der nur im Idealfall erreicht werden könnte. Sofern die diskreten Variablen des Problems nicht die Ganzzahligkeitsbedingung erfüllen, werden in einem zweiten Schritt Unterprobleme (LP_1 und LP_2) durch Schrankenbedingungen der ganzzahligen Variablen definiert. Ein Unterproblem bildet dabei einen Ast beziehungsweise einen *Branch* des Problems. Die Schritte werden so lange wiederholt, bis eine zulässige Lösung vorliegt. Liefert ein *Branch* eine Lösung mit schlechterem Zielfunktionswert als eine bereits erhaltene zulässige Lösung, so muss der Zweig nicht weiter betrachtet werden [21].

Die Abbildung 2.6 zeigt eine solche LP-Relaxierung LP_0 mit den beiden ersten Unterproblemen LP_1 und LP_2. Dabei sind $\kappa, \sigma \in \{0, 1, 2\}$ die ganzzahligen Variablen des Problems und $\sigma \leq 1$ und $\sigma \geq 2$ die Schrankenbedingungen für die beiden ersten Unterprobleme. Für die nächsten Unterprobleme, sofern diese betrachtet werden müssen, würden die Schrankenbedingungen für κ aufgestellt werden.

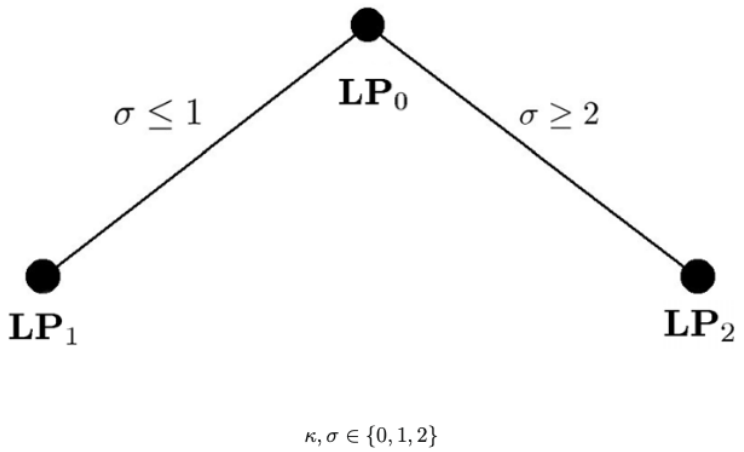


Abbildung 2.6.: LP-Relaxierung und die beiden ersten Unterprobleme und deren Knoten im B&B-Baum nach Kallrath [21]

2.6. Free and Open Source Software (FOSS)

Unter dem Begriff „Free and Open Source Software“ (FOSS) versteht man sowohl „Free Software“ als auch „Open Source Software“. Als FOSS werden grundsätzlich Software-Produkte bezeichnet, die durch den Benutzer genutzt, verändert beziehungsweise modifiziert und weitergegeben werden dürfen.
Beim Verwenden solcher Software muss dem Entwickler weder eine Lizenzgebühr noch eine andere Gebühr entrichtet werden. Die Vertriebsbedingungen einer solcher Software implizieren stattdessen im Rahmen des sogenannten „moralischen Rechts“ die Erwähnung des Entwicklers. Beispiele für diese Art der Software sind das Betriebssystem GNU/Linux, die Programmiersprache PHP oder der Webbrowser Mozilla [22]. Die Nutzung von Open Source Software und Open Source Daten zeichnet sich durch eine hohe Transparenz aus und ermöglicht reproduzierbare Ergebnisse. Darüber hinaus wird Forschern ermöglicht zusammenzuarbeiten und große Programmierprojekte oder Datensätze gemeinsam zu erstellen und zu erweitern [23].

2.7. Vergleich von Software zur Modellierung und Optimierung

Es existiert eine Vielzahl von Software zur Modellierung und Optimierung von Energiesystemen. Daher besteht die Notwendigkeit der Einordnung und Differenzierung von verschiedenen Energiesystemmodellen. Ein solcher Vergleich ermöglicht sowohl Modellierern von Energiesystemen als auch politischen Entscheidungsträgern einen besseren

Überblick über die verfügbaren Modellierungsprogramme und unterstützt zudem die Auswahl eines geeigneten Modells in Bezug auf ein konkretes Forschungsprojekt.
Häufig führt die Verwendung unterschiedlicher Modelle bei der Szenario-Analyse von Energiesystemen zu abweichenden Ergebnissen und Schlussfolgerungen. Dies kann auf inhomogene Eingangsdaten und Unterschiede in den Modellformulierungen zurückgeführt werden. Deshalb kann es sinnvoll sein, verschiedene Modellierungen für ein Problem zu verwenden, um die Verlässlichkeit der Ergebnisse zu steigern [24].

Nach Gils et al.[24] kann die Vielzahl an Vergleichen von Energiesystemmodellen, die in der Literatur zu finden sind, in drei Kategorien unterteilt werden.

- Kategorie I: Theoretische Modellvergleiche, mit Fokus auf der Bewertung und Kategorisierung von Modellfunktionalitäten und -eigenschaften
- Kategorie II: Untersuchung spezifischer technischer Aspekte oder detaillierter Modellierungsunterschiede
- Kategorie III: Szenariobasierte Vergleiche von Modellierungsansätzen mit harmonisierter Datengrundlage

Als Beispiele für Literaturwerke der Kategorie I können die Untersuchungen von Ringkjøb, Haugan und Solbrekke [5] und Connolly et al. [6] genannt werden.

Ringkjøb, Haugan und Solbrekke [5] vergleichen 75 verschiedene Modellierungstools für die Analyse von Energiesystemen mit dem Ziel, einen Überblick über derzeit verfügbare Modellierungswerkzeuge und deren Fähigkeiten zu geben. Die Modelle werden dabei anhand ihrer allgemeinen Logik, der räumlichen und zeitlichen Auflösung sowie technologischer und wirtschaftlicher Parameter kategorisiert. Unter der allgemeinen Logik des Modells ist beispielweise der Verwendungszweck, der Modellansatz aber auch die angewendete Methode (Simulation, Optimierung, etc.) zu verstehen. Des Weiteren werden die jeweiligen Softwaregrundlagen der Modelle wie zum Beispiel Python oder GAMS berücksichtigt [5].
Unter ähnlichen Aspekten kategorisiert Connolly et al. [6] 37 verschiedene Programme zur Analyse der Integration von erneuerbaren Energien. Sie vergleichen zunächst die Modelle anhand der Nutzerzahlen sowie der Verfügbarkeit (kommerzielle oder öffentliche Software). Darüber hinaus erfolgt eine Einteilung nach der Art beziehungsweise dem Nutzungszweck des jeweiligen Tools. Tiefergehend erfolgt eine Kategorisierung nach der zeitlichen und geographischen Auflösung sowie dem zeitlichen Rahmen der Modelle. Abschließend betrachten sie noch in welchem Maße verschiedene Energiesektoren und die Integration von erneuerbaren Energiequellen berücksichtigt werden. Dabei ist das Ziel von Connolly et al. [6] einen Überblick zu generieren, der es Forschern und Entwicklern ermöglicht, die passende Software zu finden [6].
Neben verschiedensten Literaturwerken, die sich mit dem Vergleich und der Kategorisierung von Modellierungstools von Energiesystemen befassen, existiert seit 2014 die Plattform *Open Energy Modelling Initiative (Open Mod)*[7]. Diese Open Mod-Plattform führt eine Liste mit öffentlich zugänglicher Software zur Modellierung von Energiesystemen. Auch eine Kategorisierung der Modelle erfolgt anhand der grundlegenden Charakteristika wie des Verwendungszweckes, der mathematischen Methode oder auch der zeitlichen und räumlichen Auflösung sowie der betrachteten Sektoren. Die Plattform stellt dabei auch weitere für die Modellierung wichtige Informationen zur Verfügung und

präsentiert zudem wissenschaftliche Veröffentlichungen, die auf der Nutzung von Open-Source-Programmen zur Analyse von Energiesystemen basieren. Der Gedanke hinter der *Open Energy Modelling Initiative* ist dabei die Qualität, Transparenz und Glaubwürdigkeit und damit auch die Forschung sowie Politikberatung im Allgemeinen zu verbessern, indem sowohl offene Energiemodelle als auch offene Daten verwendet werden [7].

Der Kategorie II lässt sich die Untersuchung von Wilkerson et al. [25] zuordnen, die verschiedene Bewertungsmodelle zur CO_2-Politik vergleichen. Der Fokus liegt dabei auf dem Identifizieren der Hauptunterschiede im Modellverhalten durch einen Vergleich der unterschiedlichen Simulations-/Optimierungsergebnisse. Wilkerson et al. [25] kommen zu dem Schluss, dass die Unterschiede auf unterschiedlicher Parametrisierung und insbesondere auf der verwendeten Modellart basieren. So sind Differenzen der Ergebnisse primär auf die Unterschiede zwischen Simulation und intertemporaler Optimierung zurückzuführen. Darüber hinaus beeinflusst auch die Voraussicht der jeweiligen Modelle die Variation der Ergebnisse [25].
Ommen, Markussen und Elmegaard [26] untersuchen den Einfluss von verschiedenen mathematischen Methoden (LP, MILP, NLP) auf die Optimierung. Unter Verwendung eines einheitlichen Modells bemerken sie eine höhere Kohärenz der Ergebnisse von MILP und NLP und kommen letztlich zu dem Ergebnis, dass der MILP-Ansatz für die Optimierung unter dem Aspekt der Genauigkeit und der Laufzeit am besten geeignet ist [26].
Einen anderen Ansatz verfolgen Priesmann, Nolting und Praktiknjo [27], die den Zusammenhang zwischen der Modellkomplexität und der Genauigkeit der Ergebnisse untersuchen. Dafür implementieren sie 160 verschiedene Netzoptimierungsmodelle unterschiedlicher Komplexität. Sie kommen zu der Erkenntnis, dass ein gewisses Maß an Komplexität für ausreichend genaue Ergebnisse erforderlich ist, allerdings auch eine Abwägung in Bezug auf die Rechenressourcen notwendig ist. Des Weiteren kommen sie zu dem Ergebnis, dass mit zunehmender Komplexität des Modells die zunehmende Genauigkeit der Ergebnisse abnimmt und somit kein lineares Verhalten auftritt [27].
Steinbrink et al. [28] wiederum beschäftigen sich mit der simulationsbasierten Smart-Grid-Systemvalidierung und kommen zu dem Schluss, dass für komplexere Systeme standardisierte Schnittstellen und ein verbessertes Datenmanagement notwendig sind. Darüber hinaus kann durch eine Standardisierung und Visualisierung der Simulationsergebnisse die Plausabilitätsprüfung der Ergebnisse vereinfacht werden [28].

Nach Misconel et al. [29] sind viele modellbasierte Szenarioanalysen aufgrund von unterschiedlicher Datenstrukturen und mathematischer Ansätze kaum vergleichbar. Daher untersuchen sie in ihrer Veröffentlichung vier elektrische Systemmodelle mit detaillierter, harmonisierter Datengrundlage. Dies hat den Vorteil, dass Ergebnisunterschiede direkt mit den Modelleigenschaften verknüpft werden können. Misconel et al. kommen zu dem Ergebnis, dass neben den Unterschieden in den mathematischen Ansätzen auch die kurzsichtige Vorausschau entscheidend für die Streuung der erzielten Ergebnisse ist [29].
Ebenfalls unter Verwendung einer vollständig harmonisierten Datengrundlage und einer zusätzlich stark vereinfachten Systemkonfiguration, um modellspezifische Effekte zu isolieren, vergleicht Gils et al. [24] systematisch 9 Stromsektormodelle. Dabei legen sie den Fokus auf die Analyse der Auswirkungen von Unterschieden in den Darstellungen von Technologien, den Optimierungsansätzen und weiteren Modellmerkmalen auf die Optimierungsergebnisse. Aufgrund der harmonischen Datengrundlage können die Veröffentlichungen von Gils et al. [24] und Misconel et al. [29] der Kategorie III zugeordnet werden.

Zusammenfassend lässt sich sagen, dass für die Wahl des richtigen Modellierungstools

zunächst eine Kategogrisierung anhand der Charakteristika der Modelle eine mögliche Auswahl von passenden Programmen liefert. Durch einen Vergleich der Simulations- beziehungsweise Optimierungsergebnisse kann letztlich das ideale Modell bestimmt werden oder durch Betrachten mehrerer Ergebnisse die Verlässlichkeit gesteigert werden.
Des Weiteren kann für einige Untersuchungen eine harmonisierte Datengrundlage sinnvoll sein. Durch eine standardisierte Darstellung sowie Visualisierung der Ergebnisse kann die Qualität und Effizienz eines Vergleichs zusätzlich erhöht werden. Zurzeit gibt es noch keine Möglichkeit, verschiedene Modellierungsprogramme automatisiert miteinander zu vergleichen.
Aus diesem Grund wurde am Institut für Energietechnik der TUHH das Transforming Energy Supply System (Modelling) Framework (Tessif) entwickelt.

3. Verwendete Software

3.1. Transforming Energy Supply System (Modelling) Framework (Tessif)

Das Transforming Energy Supply System (Modelling) Framework oder auch kurz Tessif [8] ist ein Projekt, das für die Vereinheitlichung und Analyse verschiedener Open-Source-Softwaretools zur Modellierung sowie Optimierung von Energieversorgungssystemen entwickelt wurde. Tessif bietet dabei eine präzise und einheitliche Schnittstelle zu verschiedenen Open-Source-Programmen. Derzeit unterstützt Tessif die Modellierungs- und Optimierungstools *Oemof* [30], *PyPSA* [31], *FINE* [32] sowie *Calliope* [33].
Insbesondere ermöglicht Tessif die Nutzung dieser verschiedenen Modellierungstools über eine Schnittstelle, ohne für jedes Tool ein spezifisches Modell erstellen zu müssen. Durch die standardisierte Präsentation der Optimierungsergebnisse bietet Tessif zudem eine Vergleichsgrundlage zwischen verschiedenen Modellierungstools. Tessif bietet außerdem eine Bibliothek von diversen, für den Vergleich von Modellierungssoftware ausgelegten Beispielmodellen von Energiesystemen. Auf diese Weise können für verschiedene Anwendungen die jeweils optimalen Modellierungs-/Optimierungsprogramme gewählt werden.

Die Abbildung 3.1 zeigt schematisch die Funktionsweise von Tessif. Aus der Darstellung wird ersichtlich, dass Tessif mithilfe der es2es (*energy-system to energy-system*) Funktion das Tessif-Energiesystem in ein Modell des jeweiligen Open-Source Programmes transformiert. Danach erfolgt eine Optimierung beziehungsweise Simulation des Energiesystems durch das Modellierungsprogramm. Abschließend werden die Optimierungsergebnisse durch die Funktion es2mapping (*energy-system to mapping*) in die einheitliche Ergebnisausgabe von Tessif gewandelt.
Für jede Open-Source-Software, die in Tessif integriert ist oder zukünftig integriert werden soll, muss dabei eine individuelle es2es sowie es2mapping Funktion programmiert werden.

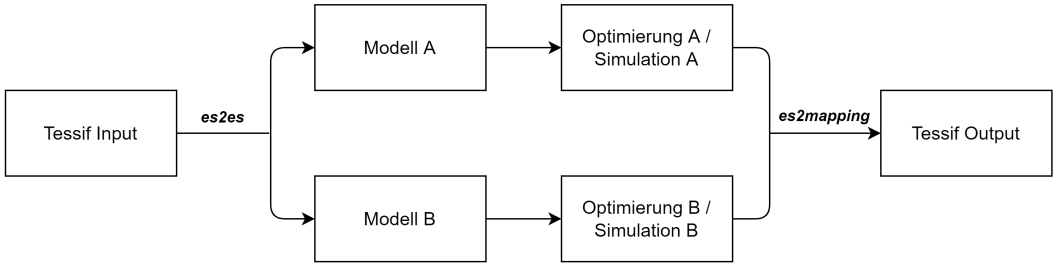


Abbildung 3.1.: Schematische Darstellung der Funktionsweise von Tessif

3.1.1. Modell

Tessif verfügt über ein eigenes Energiesystemmodell. Tessifs ES-Modell ist als arbiträre Ansammlung von Energiesystemkomponenten definiert. Diese Komponenten können miteinander verbunden sowie parametrisiert sein. In Tessif können Energiesysteme durch Kombination der nachfolgenden sechs grundlegenden Komponenten abgebildet werden.

1. Bus
2. Source
3. Sink
4. Transformer
5. Connector
6. Storage

Jeder Komponente wird in Tessif ein eindeutiges, gleichbleibendes Uid-Object zugeordnet, welches für die Identifizierung genutzt wird. Das Uid-Object setzt sich dabei aus den nachfolgenden Identifizierungsparameter zusammen.

- Name (*Hashable*): Der Name der Komponente
- Latitude (*Number*): Die geografische Breite in Grad
- Longitude (*Number*): Die geografische Länge in Grad
- Region (*String*): Die regionale Zuordnung der Komponente
- Carrier (*String*): Der Energieträger der Komponente
- Sector (*String*): Die Zuordnung zu einem Energiesektor
- Component (*String*): Die Art der Komponente im Energiesystem
- node_type (*String*): Kategorisierung des Knotenpunktes (Node Type)

Für eine eindeutige Identifikation ist es allerdings ausreichend nur den Namen der Komponente vorzugeben, sodass alle weiteren Identifizierungsparameter optional sind. Diese werden aber insbesondere für spezifische Auswertungsmethoden von Tessif benötigt.
Neben den Identifizierungsparametern werden den Komponenten noch weitere Parameter vorgegeben, welche die energietechnischen Eigenschaften der Komponenten definieren. Da diese aber in Abhängigkeit der Art der Komponente variieren, werden die energietechnischen Parameter im nachfolgenden Abschnitt 3.1.2 erläutert.

Das vollständige Energiesystemmodell kann im Grunde als diskret mathematischer Graph betrachtet werden. Dieser besteht aus **Nodes** (Knoten) sowie **Edges** (Kanten) und wird deshalb im Weiteren nur noch als Node and Edge Graph bezeichnet. Die Nodes stellen bei Tessif die Komponenten des Energiesystems dar, also beispielweise Kraftwerke, elektrische Verbraucher oder Energiespeicher. Die Edges stellen die Flüsse zwischen den einzelnen Komponenten beziehungsweise Nodes dar und können allgemein als Energieflüsse interpretiert werden. Die Abbildung 3.2 zeigt beispielhaft ein Energiesystem in Tessifs Darstellung als Node and Edge Graph.

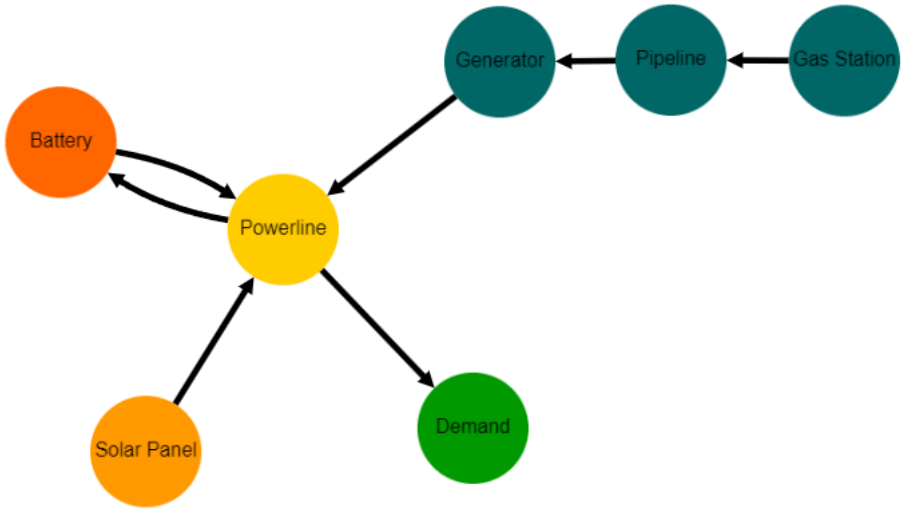


Abbildung 3.2.: Einfaches, visualisiertes Energiesystem als Node and Edge Graph (erstellt mit Tessif [8])

3.1.2. Komponenten

Wie bereits zuvor erwähnt, werden für die einzelnen Komponenten neben den Uid-Parametern aus 3.1.1 auch noch die für die Energietechnik relevanten Parameter vorgegeben. Diese werden in den nachfolgenden Abschnitten mit der Funktion für die jeweiligen Komponenten erläutert.
Da viele Komponenten auf die gleichen Parameter zurückgreifen, werden nachfolgend zunächst einige für mehrere Komponenten relevante Parameter erklärt.

- **Flow-Parameter:**
  - **Flow Rates** geben die minimale/maximale Menge des Output/Input pro diskretem Zeitschritt an. Jeder Fluss muss innerhalb des minimalen/maximalen Bereichs liegen.
  - **Flow Costs** geben die Kosten an. Die Kosteneinheit wird pro Menge pro Zeit berücksichtigt.
  - **Flow Emissions** geben die Emissionen an. Die Emissionseinheit wird ebenfalls pro Menge pro Zeit berücksichtigt.
  - **Flow Gradients** beschreiben die maximale positive beziehungsweise negative Flussänderung zwischen zwei Zeitschritten.
  - **Gradient Costs** geben die Kosten an, die bei einer Änderung der Flussmenge auftreten.

  - **Timeseries** geben der Komponente für jeden Zeitschritt einen konkreten Output/Input vor. Dieser Parameter kann genutzt werden, um fluktuierende Einspeisungen von beispielweise Solaranlagen darzustellen.

- **Expanison-Parameter:**

  - **Expandable** gibt an, ob die Komponente ausgebaut werden darf oder nicht. (Expansion Problem)
  - **Expansion Costs** geben entsprechend die Kosten für den Ausbau an.
  - **Expansion Limits** beschreiben die Grenzen der Expansion im Optimierungszeitraum.

- **Zusätzliche-Parameter:**

  - **MILP** gibt vor, ob ein Input/Output den gemischt ganzahligen linearen Beschränkungen unterliegt. (Dies wird für Inputs, die sich nicht teilen lassen z.B. Arbeitskräfte, benötigt.)
  - **Initial Status** gibt an, ob die Komponente zu Beginn der Optimierung läuft.
  - **Status Inertia** beschreibt die minimalen Betriebs- und Stillstandszeiten einer Komponente.
  - **Status Changing Cost** gibt die Kosten zum Ein- sowie Ausschalten der Komponente an.
  - **Number of Status Changes** limitiert die Anzahl der möglichen Statusänderungen.
  - **Costs for being active** sind die festen Betriebskosten pro Zeitschritt.

Die Flow-Parameter definieren die verschiedenen Energieflüsse zwischen den einzelnen Komponenten, beziehungsweise diskretisieren die Inputs sowie Outputs von einzelnen Komponenten. Die Expansion-Parameter sind nur für Ausbauszenarien von Energiesystemen relevant, um den Ausbau der Kapazität beziehungsweise Leistung von einzelnen Komponenten zu ermöglichen. Die Auflistung und Beschreibung der zuvor erläuterten Parameter wurde der Ausarbeitung [34] von Hanke entnommen.
Alle Parameter und deren Bedeutung für die einzelnen Komponenten können in der Tessif Dokumentation [8] unter dem Abschnitt Components eingesehen werden. Im Folgenden werden die sechs grundlegenden Komponenten näher vorgestellt.

Bus

Die Bus-Komponente kann grundsätzlich als Leitung verstanden werden. Bei Bussen müssen neben den Identifizierungsparametern nur Inputs und Outputs vorgegeben werden, da für die Bus-Komponente nur relevant ist, welche Komponenten Energie einspeisen und welche Energie beziehen. Somit verbindet der Bus die einzelnen Komponenten miteinander und kann daher als Energienetz des Systems gesehen werden.

Source

Source-Komponenten sind Quellen, die durch einen oder mehrere Outputs definiert sind. In Energiesystemen können Sources einerseits Rohstoffquellen beziehungsweise Treibstoffquellen für ein konventionelles Kraftwerk oder aber direkte Energiequellen, beispielweise in Form einer Photovoltaikanlage darstellen. Der Output der Source wird dabei durch die Flow-Parameter aus 3.1.2 definiert. Zusätzlich kann der Quelle noch die *Accumulated Amounts* als weiterer Parameter vorgegeben werden. Dieser gibt an, welche minimale oder maximale Menge der Output über den gesamten Zeitbereich zur Verfügung hat.

Sink

Die Sink-Komponenten verhalten sich im Grunde komplementär zu den Source-Komponenten. Sie bilden Senken, die einen durch die Flow-Parameter (3.1.2) definierten Input aufnehmen können. Damit können sie in Energiesystemmodellen Verbräuche darstellen. Ebenso wie bei den Quellen kann zusätzlich der Parameter *Accumulated Amounts* bei den Sink-Komponenten verwendet werden.

Transformer

Die Transformer-Komponente kann verwendet werden, um einen oder mehrere Inputs in einen oder mehrere Outputs zu transformieren. Der Transformer kann daher sowohl konventionelle als auch CHP-Kraftwerke (Combined Heat and Power) abbilden. Beispielweise kann ein Kraftstoff-Input in einen Strom- und einen Wärme-Output gewandelt werden. Dabei werden die Eigenschaften der In- und Outputs jeweils über die Flow-Parameter (3.1.2) definiert. Zusätzlich wird noch ein *Conversions*-Parameter benötigt, der die Umwandlungswirkungsgrade der Transformationen angibt.

Connector

Durch die Connector Komponente können zwei Bus-Komponenten miteinander verbunden und somit zwei unabhängige Energiesysteme miteinander verknüpft werden. Dabei findet nur der *Interfaces*-Parameter sowie der *Conversions*-Parameter Anwendung. Der *Interface*-Parameter gibt an, welche Bus-Komponenten miteinander verbunden sind. Die Übertragungswirkungsgrade zwischen den verbundenen Bussen werden durch die *Conversions* vorgegeben.

Storage

Die Storage-Komponente ist in der Lage Flüsse auf- und zu späteren Zeitschritten wieder abzugeben und bildet somit Energiespeicher im Energiesystemmodell ab. Neben den Flow-Parametern aus 3.1.2 müssen noch die Kapazität (*Capacity*), der Speicherstand bei Optimierungsbeginn (*Initial SOC*), der Ladestand zum Ende der Optimierung (*Final SOC*), die Änderungen des Speichers zwischen Zeitschritten (*Idle Changes*) sowie die Ein- und Ausspeicherwirkungsgrade des Speichers (*Flow Efficiencies*) angegeben werden.

3.1.3. Umsetzung in Tessif

Der Code-Ausschnitt A.1 im Anhang zeigt beispielhaft, wie die einzelnen Komponenten eines Energiesystems in Tessifs Modell definiert werden. Das Code-Beispiel zeigt ein Biogas CHP-Kraftwerk, dessen Eigenschaften durch die zuvor genannten Parameter beschrieben werden. Durch Kombination mehrerer Komponenten können verschiedenste Energiesysteme mit Tessifs Modell abgebildet werden. Das Code Beispiel A.1 wurde der Tessif py_hard-Datei entnommen.

3.2. Urbs

Urbs [35] ist ein lineares Optimierungsprogramm, das insbesondere für die Erforschung innovativer Wege für die Planung städtischer Energieinfrastruktur, von Dorfner an der Technischen Universität München entwickelt wurde [36]. Das verwendete Modell urbs ist die weiterentwicklung des Modells URBS und wird im Rahmen dieser Arbeit vereinfachend als Urbs bezeichnet.
Mithilfe von Urbs können sowohl Einsatzpläne als auch die Planung der Kapazitätserweiterungen von Energiesystemen optimiert werden. Insbesondere ist Urbs für die Modellierung von Energiesystemen mit verschiedenen Energieträgern geeignet. Der Fokus liegt dabei insbesondere auf einer idealen Dimensionierung der Größe sowie Nutzung von Speichern. Das Modellierungsprogramm ist sehr kompakt programmiert, da es auf der Python Package Pyomo basiert. Zu den Funktionen von Urbs zählen insbesondere auch eine Präsentation sowie eine grafische Visualisierung der Optimierungsergebnisse [37].

Die Funktion von Urbs wird in Abbildung 3.3 ersichtlich. Die Grafik 3.3 stellt dabei unter dem Punkt Input die von Urbs benötigten Eingabedaten und unter dem Punkt Output die von Urbs nach der Optimierung ausgegebenen Ergebnisse übersichtlich dar.

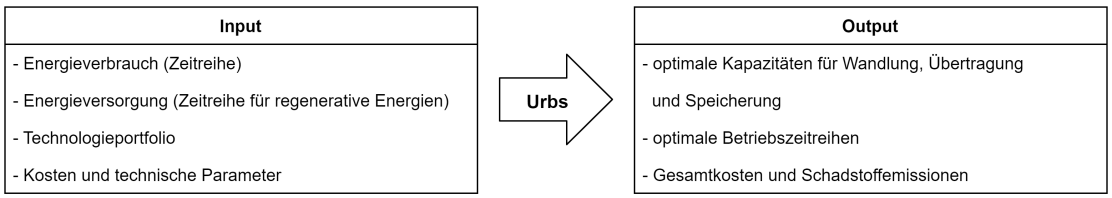


Abbildung 3.3.: Input/Output Flussdiagramm von Urbs nach [36]

Die nachfolgende Abbildung 3.4 gibt einen kleinen Überblick über die Architektur von Urbs sowie den Datenfluss innerhalb eines Urbs-Modell.

Wie in der Abbildung 3.4 ersichtlich, werden zunächst mithilfe der input.py-Datei die Eingabe-Parameter aus der Excel Tabelle eingelesen und dann als *data* abgespeichert. Mit diesen Daten wird anschließend durch die model.py-Datei ein Pyomo-Modell erstellt, welches durch den gewählten Solver optimiert werden kann. Die Ergebnisse der

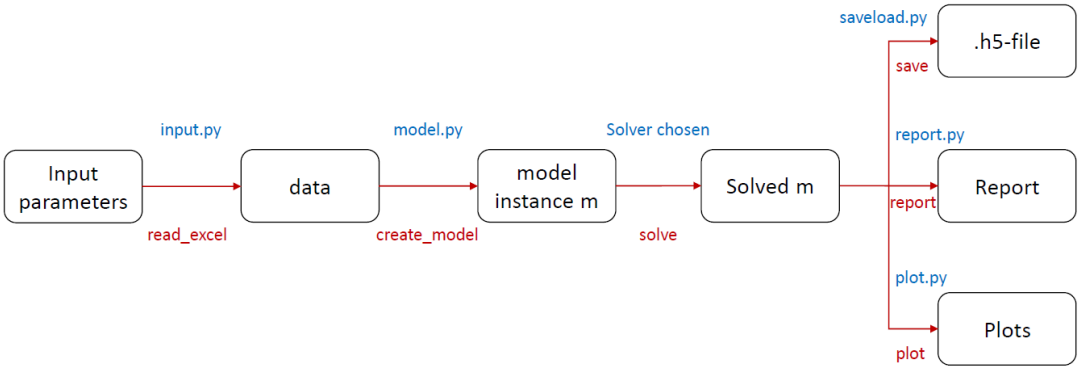


Abbildung 3.4.: Datenfluss in einem Urbs-Modell nach [37]

Optimierung können anschließend gespeichert (saveload.py), in Form einer xls-Tabelle ausgegeben (report.py) oder grafisch dargestellt werden (plot.py).

3.2.1. Modell

Anders als Tessif nutzt Urbs keine Klassen-Objekte, um die Komponenten der Energiesysteme zu modellieren. Alle Modelle von Energiesystemen, die in Urbs generiert werden können, setzen sich aus den nachfolgenden Einheiten zusammen [37]:

- **Commodities:** Rohstoffe beziehungsweise Energieträger, welche die verschiedenen Formen von Energie- sowie Stoffflüssen innerhalb des Energiesystems repräsentieren.
- **Processes:** Prozesse, die Rohstoffe oder Energieträger in andere Energieträger umwandeln. Prozesse sind immer nach dem Multiple-Input/Multiple-Output (MiMo) Prinzip definiert. Das bedeutet, es sind für jeden Prozess Eingangsstoffe definiert, welche in Ausgangstoffe umgewandelt werden.
- **Transmission lines:** Übertragungsleitungen, die den Austausch von Energieträgern zwischen zwei räumlich getrennten Gebieten ermöglichen.
- **Storages:** Speicher, welche das Speichern eines einzelnen Energieträgers innerhalb eines Gebietes ermöglichen.
- **DSM potentials:** DSM-Potentiale ermöglichen es dem Energiesystem Lastmanagement zu betreiben und Lasten zeitlich zu verschieben.

Die Vorgabe eines Energiesystems erfolgt dabei nicht durch Vorgabe definierter Komponenten und deren Verknüpfungen untereinander, sondern in tabellarischer Form. Die Tabelle ist dabei in die folgenden zwölf Arbeitsblätter unterteilt [37]:

1. Global
2. Site
3. Commodity
4. Process
5. Process-Commodity
6. Transmission
7. Storage
8. DSM
9. Demand
10. SupIm
11. Buy-Sell-Price
12. TimeVarEff

Im weiteren Verlauf dieses Abschnittes wird noch genauer auf die einzelnen Arbeitsblätter eingegangen, um den Modellansatz von Urbs zu verdeutlichen.
Die Abbildung A.1 im Anhang verdeutlicht beispielhaft die tabellarische Darstellungsweise der Energiesystemmodelle von Urbs.

Global

Im Bereich Global werden die generellen Informationen des Energiesystems vorgegeben. Diese bestehen aus dem *CO2 limit* und dem *Cost limit*. Sie geben die Grenze für die maximalen CO_2 Emissionen beziehungsweise die maximalen Kosten für den Optimierungszeitraum an. Die Grenzen werden allerdings nur berücksichtigt, wenn es sich dabei nicht um die Optimierungsvariable handelt.

Site

Im Abschnitt Site können verschiedene Gebiete (Sites) definiert werden. Jedes dieser Gebiete bildet im Grunde ein eigenständiges Energiesystem, wobei verschiedene im Arbeitsblatt Transmission 3.2.1 festgelegte Energieträger zwischen den einzelnen Sites übertragen werden können. Zudem kann unter der Variable *area* noch die Größe des Gebiets in Quadratmetern angegeben werden. Dies ist allerdings nur relevant, wenn beispielweise für den Ausbau von Photovoltaik-Anlagen eine bestimmte Fläche benötigt wird, um die Leistung zu erhöhen ($\frac{m^2}{MW}$).

Commodity

In der Commodity-Tabelle werden den einzelnen Sites beziehungsweise Gebieten die verschiedenen Energieträger (Commodities) zugeordnet. Des Weiteren wird in dem Unterpunkt *Type* die Art des Energieträgers angegeben. Dabei kann zwischen den nachfolgenden Arten unterschieden werden:

- **Stock:** Als Stock werden üblicherweise die Ressourcen bezeichnet, auf welche das Energiesystem im Weiteren zurückgreifen kann. So ist beispielweise Kohle eine solche Ressource, die dann in Kraftwerken zur Energieerzeugung genutzt werden kann.
- **SupIm:** Bei SupIm handelt es sich im Grunde ebenfalls um solche Ressourcen. Allerdings können diese Ressourcen nicht wie beim Typ Stock beliebig genutzt werden, sondern unterliegen Zeitreihen, welche im Teil SupIm (3.2.1) vorgeben werden.
- **Demand:** Der Typ Demand bezeichnet die Energieträger, welche in der Site den Verbrauch darstellen. Hierbei handelt es sich meist um Strom oder Wärme. Allerdings kann die Art des Verbrauchs frei gewählt werden. Die Verbrauchswerte werden im Abschnitt Demand (3.2.1) in Form von Zeitreihen vorgegeben.
- **Env:** Außerdem können Commodities auch dem Typ Env zugeordnet werden. Dabei handelt es sich um Umweltvariablen. Häufig wird CO_2 für die Kohlenstoffemissionen als Umweltvariable verwendet. Es können aber auch allgemeine Schadstoffemissionen angegeben werden. Für die angegebenen Umweltvariablen (Env-Variablen) muss entsprechend im Abschnitt Global (3.2.1) ein Limit vorgegeben werden.
- **Buy:** Commodities, die zusätzlich eingekauft werden können, werden dem Typ Buy zugeordnet (z.B. Strom, der importiert wird).
- **Sell:** Als Sell werden Commodities bezeichnet, die verkauft werden können, wie zum Beispiel beim Stromexport in andere Länder.

Zusätzlich zur Art (*Type*) der Commodity kann auch noch der Preis (*price*) des Energieträgers ($\frac{€}{MWh}$) sowie die maximal jährlich (*max* in MWh) und stündlich (*maxperhour* in MW) nutzbaren Beträge der Ressourcen angegeben werden.

Abhängig von der Art der Commodity haben die drei zuvor genannten Variablen unterschiedliche Bedeutungen. Für Stock-Commodities gibt der Preis die Kosten für eine Einheit der Ressource an. Die jährlichen und stündlichen Limits begrenzen dabei die maximal nutzbaren Mengen. Bei SupIm-Ressourcen sowie Verbräuchen entfallen die zusätzlichen Angaben.
Für Umweltvariablen können durch den Preis Kosten vorgeben werden, die beim Erzeugen einer Einheit dieser Commodity entstehen. Diese Kosten könnten beispielweise eine CO_2-Steuer oder Vergleichbares repräsentieren. Zudem kann zusätzlich zum globalen Emissionslimit über die Variable *max* ein lokales Limit für die jeweilige Site definiert werden.
Bei den Buy- beziehungsweise Sell-Commodities gibt die *price*-Variable entsprechend den Kauf- oder Verkaufspreis der Ressource an.

Process

Im Abschnitt Process werden die verschiedenen Prozesse definiert, die verschiedene Ressourcen (Commodities) in Energie oder weitere Ressourcen umwandeln. Zu diesen Prozessen zählen beispielweise konventionelle Kohlekraftwerke, aber auch Solar- oder Windkraftanlagen. Unter dem Punkt *Site* wird der Prozess einem Gebiet zugeordnet und zusätzlich über die Variable *Process* benannt, sodass die einzelnen Kraftwerke später unterschieden werden können.
Des Weiteren müssen noch einige Variablen vorgegeben werden, welche insbesondere die Kapazität, aber auch die Kosten des Prozesses definieren. So muss sowohl die installierte Leistung (*inst-cap*) als auch eine minimale sowie maximale Leistung (*cap-lo*, *cap-up*) vorgegeben werden. Sofern die maximale Leistung größer als die installierte Leistung ist, kann die Komponente ausgebaut werden. Wird die minimale Leistung größer als die installierte gewählt, so wird ein Ausbau der Komponente erzwungen. Zusätzlich kann noch der maximale Gradient (*max-grad*) sowie ein Mindestlastanteil (*min-fract*) definiert werden. Über den Gradienten kann eingestellt werden, welche maximalen Leistungsänderungen pro Zeitschritt erlaubt sind. Der Mindestlastanteil gibt den minimal möglichen Anteil der Prozesskapazität vor, bei welchem der Prozess noch ausgeführt werden kann. Außerdem können noch Investitionskosten (*inv-cost*), jährliche Fixkosten (*fix-cost*) und variable Kosten (*var-cost*), die in Abhängigkeit der produzierten Einheit (meistens Leistung) entstehen, vorgegeben werden. Als weitere Variablen können noch der Prozentsatz der Kapitalkosten nach Steuern (*wacc* [weighted average cost of capital]) sowie der Abschreibungszeitraum (*depreciation*) angegeben werden, um damit den Annuitätenfaktor der Investitionskosten und somit letztlich die Wirtschaftlichkeit zu bestimmen.
Abschließend kann noch angegeben werden, welche Fläche pro Kapazität (*area-per-cap*) der Prozess benötigt. Diese Variable findet üblicherweise nur Gebrauch bei stark flächenabhängigen Prozessen, wie beispielweise Photovoltaikanlagen.

Process-Commodity

In der Process-Commodity Tabelle werden den einzelnen Prozessen beziehungsweise den Kraftwerken die Umwandlungen der Energieträger zugeordnet. Dafür wird zunächst unter dem Punkt *Process* der Prozess und unter dem Aspekt *Commodity* ein zugehöriger Energieträger vorgegeben. Durch Auswählen der Flussrichtung (*Direction*, entweder „In“ oder „Out“) kann bestimmt werden, ob der Energieträger in dem Prozess verbraucht oder erzeugt wird.
Da die Prozesse nicht mit einer Umwandlungsrate von 100% arbeiten, kann über die Variable *ratio* der Wirkungsgrad der Umwandlung vorgegeben werden. Zusätzlich besteht die Möglichkeit, einen Faktor für den Prozess vorzugeben, sollte dieser mit Mindestlast (*min-fract*) betrieben werden. Mithilfe der Process-Commodity Tabelle werden ebenfalls die Schadstoffemissionen der Prozesse definiert.

Transmission

Urbs ermöglicht auch einen Austausch von Energie beziehungsweise Energieträgern zwischen verschiedenen Gebieten (Sites). Dafür müssen im Bereich Transmission die beiden Sites (*Site In* und *Site Out*), die Art der Übertragung (*Transmission*, z.B. hvac) sowie der Energieträger (*Commodity*) angegeben werden. Die Übertragung wird durch dieselben

Variablen beschrieben, die auch im Abschnitt Process 3.2.1 genutzt werden. Nachfolgend werden die Variablen kurz aufgelistet.

- Effizienz (*eff*)
- Investitionskosten (*inv-cost*)
- jährliche Fixkosten (*fix-cost*)
- Variable Kosten (*var-cost*)
- Installierte Kapazität (*inst-cap*)
- Minimale Kapazität (*cap-lo*)
- Maximale Kapazität (*cap-up*)
- Prozentsatz der Kapitalkosten nach Steuern (*wacc*)
- Abschreibungszeitraum (*depreciation*)

Für elektrische Übertragungen können zusätzlich die Reaktanz der Übertragungsleitung (*reactance*), das Limit der Spannungswinkeldifferenz zwischen den Gebieten (*difflimit*) und die Grundspannung der Übertragungsleitung (*base_voltage*) definiert werden.

Storage

Speicher können im Bereich Storages angegeben werden. Zum Definieren des Speichers muss das Gebiet (*Site*), in welchem der Speicher sich befindet, der Name beziehungsweise die Art des Speichers (*Storage*) sowie der Energieträger (*Commodity*) vorgegeben werden. Mithilfe weiterer Variablen werden die physikalischen Eigenschaften der Speicher abgebildet. Die nachfolgenden Variablen sind zum einen auf die Speicherkapazität (c) und zum anderen auf die Speicherleistung (p) bezogen.

- Installierte Kapazität (*inst-cap-c, inst-cap-p*)
- Minimale Kapazität (*cap-lo-c, cap-lo-p*)
- Maximale Kapazität (*cap-up-c, cap-up-p*)
- Investitionskosten (*inv-cost-c, inv-cost-p*)
- jährliche Fixkosten (*fix-cost-c, fix-cost-p*)
- Variable Kosten (*var-cost-c, var-cost-p*)

Des Weiteren kann auch hier wieder der Prozentsatz der Kapitalkosten nach Steuern (*wacc*) sowie der Abschreibungszeitraum angegeben werden.
Zudem werden noch speicherspezifische Variablen wie die Ein- und Ausspeicherwirkungsgrade (*eff-in* und *eff-out*), der Speicherstand zu Beginn der Optimierung und die Selbstentladerate des Speichers (*discharge*) benötigt. Über den Punkt *ep-ratio* (Energy to power ratio) kann die Leistungskapazität vom Ladestand der Batterie abhängig gemacht werden.

DSM

Im Abschnitt DSM können Gebiete angegeben werden, in denen ein Lastmanagement möglich ist. Dafür muss zum einen die *Site* und zum anderen die Art der Last (*Commodity* definiert werden. Zusätzlich muss vorgegeben werden, über wie viele Zeitschritte die Last verschoben werden darf (*delay*) und in welchen zeitlichen Abständen die nächste Verschiebung möglich ist (*recov*). Zudem wird der Wirkungsgrad der Lastverschiebung und die maximal vor- beziehungsweise zurückgeschobene Energiemenge (*cap-max-do, cap-max-up*) benötigt.

Demand

Im Abschnitt Demand werden die Zeitreihen für die Verbräuche vorgegeben. Dafür muss für jeden Energieträger, der im Bereich Commodity als Demand bezeichnet wurde, entsprechend eine Zeitreihe für die Verbrauchswerte angegeben werden. Dabei wird in der ersten Spalte der Demand-Tabelle der jeweilige Zeitschritt t und in den weiteren Spalten der Tabelle die absoluten Verbrauchswerte aufgeführt. Über das Gebiet (Site) sowie den Energieträger (Commodity) in der Kopfzeile können die Zeitreihen den einzelnen Verbräuchen zugeordnet werden.

SupIm

Ebenso wie die Verbräuche im Abschnitt Demand werden auch die Zeitreihen für erneuerbare Energien angegeben. Allerdings werden dafür keine absoluten Zahlen, sondern nur normierte Faktoren definiert. Mit der jeweilig installierten Kapazität des Prozesses ergeben sich die Leistungswerte für die einzelnen erneuerbaren Erzeuger.

Buy-Sell-Price

In der Buy-Sell-Price Tabelle werden die Kosten beziehungsweise Erträge für Buy- und Sell-Commodities definiert. Da diese Preise zeitlich nicht immer konstant sind, werden auch hier wieder Zeitreihen benötigt. Die Struktur der Zeitreihen ist dabei wie im Abschnitt Demand beschrieben. Für jeden Energieträger, der im Bereich Commodity als Buy oder Sell Variable definiert ist, muss eine Zeitreihe vorgegeben werden.

TimeVarEff

Alternativ zu erneuerbaren Energien, die durch die SupIm-Darstellung abgebildet werden, können für alle weiteren Prozesse Wirkungsgradschwankungen festgelegt werden. Diese werden ebenfalls in Form einer Zeitreihe vorgeschrieben. Durch die variablen Effizienzen können beispielweise Umwelteinflüsse auf bestimmte Prozesse beziehungsweise Kraftwerke abgebildet werden.

3.2.2. Runfile

Um mit Urbs die Optimierung eines Energiesystems durchführen zu können, wird neben dem Modell des Energiesystems noch ein Runfile benötigt. In diesem Runfile wird neben dem Speicherort, an welchem sich das zu optimierende Energiesystemmodell in Form einer xsl-Tabelle befindet, auch der Speicherort vorgegeben, an welchem die Ergebnisse der Optimierung gespeichert werden sollen.
Zusätzlich werden noch weitere für den Optimierungsvorgang relevante Informationen definiert. Zu diesen zählen die Anzahl der zu optimierenden Zeitschritte, der von der Software zu verwendende Solver sowie die Variable, nach der das Energiesystemmodell optimiert werden soll. Bei der Optimierungsvariable kann zwischen einer Kostenoptimierung und der Optimierung der Schadstoffemissionen differenziert werden.
In Bezug auf die Ausgabe der Ergebnisse muss vorgegeben werden, welche Sites und Commodities in der Auswertung dargestellt werden sollen. Üblicherweise erfolgt eine Auswertung der Verbrauchskomponenten aller Gebiete (meistens Strom). Da Urbs in der Lage ist eine grafische Auswertung der Ergebnisse zu erstellen, können auch die grafisch auszuwertenden Gebiete und Energieträger definiert werden. Über die Runfile-Datei kann zudem die Benennung sowie die farbliche Gestaltung der einzelnen Kurven angepasst werden.
Urbs ist zudem in der Lage ein Energiesystemmodell mit mehreren Szenarien zu optimieren. Diese Szenarien müssen allerdings zuvor in der Datei urbs.scenarios.py definiert werden. Dabei können verschiedenste Szenarien umgesetzt werden. Es können zum Beispiel Szenarien mit niedrigeren Grenzen für Schadstoffemissionen oder Szenarien mit höheren Rohstoffpreisen (Stock-Preise) erstellt werden. Im Runfile können dann die Szenarien benannt werden, mit denen das Energiesystemmodell optimiert werden soll.

3.2.3. Fähigkeiten und Grenzen

Urbs bietet durch die Modellerstellung in tabellarischer Form insbesondere für unerfahrenere Nutzer eine Möglichkeit Energiesysteme zu optimieren und die Ergebnisse grafisch zu präsentieren. Durch eindeutige Formulierungen der Parameter und diverse, ausführlich erläuterte Beispiele benötigt die Einarbeitung wenig Zeit. Obwohl Urbs ursprünglich für den Stromsektor entwickelt wurde, können beliebige vom Nutzer gewählte Energiesektoren sowie Energieträger in dem Modell verwendet werden. Urbs kann sowohl eine Einsatzplanung (Dispatchproblem) als auch eine Ausbauplanung (Investmentproblem) durchführen.
Die zeitliche Skalierung erfolgt stündlich, kann aber bei einheitlicher Vorgabe für alle Daten variiert werden. Anders als der Name vermuten lässt, kann Urbs (lat. Stadt) nicht nur für die Optimierung von Städten genutzt werden, sondern Energiesysteme von einzelnen Nachbarschaften bis hin zur Energieversorgung von ganzen Kontinenten abbilden. Des Weiteren ist Urbs in der Lage verschiedene Szenarien für ein Energiesystemmodell zu optimieren. Auf diese Weise lässt sich insbesondere auch eine Monte-Carlo-Analyse des zu untersuchenden Energiesystems durchführen.
Der Detailgrad der Prozesse beziehungsweise der Darstellung der einzelnen Komponenten des Energiesystems weist sich durch eine Vielzahl an technischen sowie ökonomischen Parametern aus. Dies ermöglicht die Integration von sowohl erneuerbaren Energieerzeugern mit vorgegebenen Erzeugungsprofilen als auch konventionellen Kraftwerken sowie CHP-Kraftwerken (Combined Heat and Power).

Der Komplexität der zu untersuchenden Energiesysteme sind im Grunde keine Grenzen gesetzt, allerdings kann insbesondere bei großen Energiesystemen mit vielen Komponenten und geographischen Unterteilungen die Excel Tabelle schnell unübersichtlich werden. Zudem sind einige spezielle Komponenten, wie zum Beispiel variable CHP-Kraftwerke, nicht direkt in das Modell implementiert und können nur durch unintuitive Methoden genutzt werden. Abschließend ist zu erwähnen, dass Urbs durch die lineare Programmierung nicht in der Lage ist, Probleme mit gemischt-ganzzahligen linearen Variablen zu lösen.

3.3. Oemof (Open Energy Modelling Framework)

Oemof steht für Open Energy Modelling Framework und kann für die Energiemodellierung sowie insbesondere die Energiesystemmodellierung genutzt werden. In dieser Arbeit wird nur das Oemof.solph Paket als Basis für den exemplarischen Vergleich in Kapitel 5 genutzt. Oemof.solph ist ein Modellgenerator, der sowohl für die Modellierung als auch die Optimierung von Energiesystemen genutzt werden kann [30]. Die Modellierung von Energieversorgungssystemen erfolgt dabei über Kombination verschiedener in der Oemof Modellbibliothek definierter Komponenten. Die Optimierungsprobleme können dabei sowohl linear als auch gemischt-ganzzahlige lineare (mixed-integer-linear-model) Modelle sein [14]. Im weiteren Verlauf dieser Arbeit wird der Begriff Oemof vereinfachend für das Oemof.solph Paket verwendet.

3.4. PyPSA (Python for Power System Analysis)

Python for Power System Analysis oder auch kurz PyPSA ist eine Open-Source Software und kann für die Modellierung sowie Optimierung von Energieversorgungssystemen genutzt werden. Den Fokus legt PyPSA dabei insbesondere auf den elektrischen Energiefluss. Insbesondere ist PyPSA in der Lage sowohl Wechsel- als auch Gleichstromnetze abzubilden und in Modelle zu integrieren. Dabei ist PyPSA vor allem für die Modellierung und Optimierung großer Energienetze über lange Zeiträume ausgelegt. Der statische Leistungsfluss kann dabei unter Verwendung der vollständigen nichtlinearen Netzgleichungen sowie über linearisierte Netzgleichungen bestimmt werden. Mithilfe der linearen Netzgleichungen kann eine Kostenoptimierung durchgeführt werden, sodass ein linearer, optimaler Leistungsfluss bestimmt wird [31].

3.5. FINE (A Framework for Integrated Energy System Assessment)

Das Framework for Integrated Energy System Assessment beziehungsweise FINE ist eine am Forschungszentrum Jülich entwickelte Python Package. Das Framework wurde entwickelt, um Fragen zukünftiger Energiesysteme zu beantworten. FINE kann Energiesysteme modellieren, optimieren und bietet einen Rahmen für die Bewertung. Das Ziel von FINE ist, bei der Modellierung und Optimierung die jährlichen Gesamtkosten zu minimieren und zeitgleich technische sowie ökologische Randbedingungen einzuhalten. Insbesondere zeichnet sich FINE durch die Integration des *Time Series Aggregation*

Module (tsam) [38] aus, das die zeitliche Auflösung durch Zusammenfassen periodischer Zeitbereiche anpasst. Auf diese Weise kann die Komplexität des Energiesystemmodells und somit auch die Rechenzeit der Optimierung erheblich reduziert werden [32].

3.6. Calliope (A multi-scale energy systems modelling Framework)

Calliope ist ein Framework, das primär für die Energiewende und den Übergang zu erneuerbaren Energien entwickelt wurde. Das Open-Source-Tool ermöglicht das Erstelllen von Energiesystemmodellen verschiedenster Größenordnungen. Den Fokus legt Calliope dabei auf Flexibilität und eine hohe zeitliche und räumliche Auflösung. Darüber hinaus kann Calliope in einem optionalem Betriebsmodus auch ein vordefiniertes System unter verschiedenen Betriebsbedingungen testen. Zusätzlich zeichnet sich Calliopes Modellbeschreibung durch ein leicht lesbares und maschinell verarbeitbares YAML-Format aus [33].

Calliope wird im weiteren Verlauf dieser Arbeit ebenso wie Oemof, PyPSA und FINE als Basis des exemplarischen Vergleichs in Kapitel 5 verwendet. Insbesondere werden Oemof und PyPSA beim detaillierten Vergleich in Abschnitt 5.3.2 genutzt.

4. Methode

4.1. Auswahl eines Softwaremodells

Um ein weiteres Modellierungsprogramm zu Tessif hinzuzufügen, muss dieses einige grundlegende Anforderungen erfüllen, damit eine Integration in Tessif erfolgreich durchgeführt werden kann. So muss die Software zunächst frei verwendbar sein. Darüber hinaus ist es von Vorteil, eine Software zu wählen, die ebenso wie Tessif selbst auf der Programmiersprache Python basiert.
Zur Auswahl einer passenden Software kann die bereits in Abschnitt 2.7 erwähnte Datenbank von Openmod [7] verwendet werden. Die Tabelle 4.1 zeigt die Softwareeigenschaften der bereits in Tessif integrierten Modelle Oemof [30], PyPSA [31] und Calliope [33]. Des Weiteren werden die Charakteristika der Software Urbs, die im Rahmen dieser Arbeit in Tessif integriert werden soll, aufgelistet. Das Modellierungsprogramm FINE [32] ist aktuell nicht in der Openmod-Datenbank hinterlegt und wird dementsprechend hier nicht weiter berücksichtigt.
Aus der Tabelle 4.1 wird ersichtlich, dass bei Urbs ebenso wie bei den anderen Modellen der Quellcode öffentlich ist und sowohl die Modellierungssoftware als auch die ausführende Software auf der Programmiersprache Python basiert. Demzufolge sollte eine Integration hinsichtlich dieser Aspekte möglich sein.

Model	Modelling software	Processing software	Model source public	Data availability	Open future
Oemof	Python, Pyomo, Coin-OR	Python, PostgreSQL, PostGIS	true	some	true
PyPSA	Python, Pyomo	Pandas	true	all	false
Calliope	Python (Pyomo)	Python (pandas et al)	true	some	true
Urbs	Python (Pyomo)	Python (pandas et al)	true	some	true

Tabelle 4.1.: Kategorisierung anhand Charakteristika der Software nach Openmod [7]

Neben den grundlegenden Softwareeigenschaften müssen zusätzlich Anforderungen an spezifische Charakteristika des Modells erfüllt sein. Da Tessif sich auf den Vergleich von Software zur Optimierung von Energiesystemen fokussiert, sollte das zu integrierende Modell ebenfalls in der Lage sein, ein Optimierungsmodell zu erstellen. Zudem müssen beispielweise diverse Sektoren abgebildet werden können, damit alle Energiesysteme von Tessif ebenso in Urbs dargestellt werden können. Unter diesem Aspekt ist auch eine variable räumliche Auflösung relevant, da Tessif Energiesysteme von regionaler bis zu nationaler und globaler Größe darstellen kann. Eine einheitlich zeitliche Auflösung (Timeresolution) wird insbesondere für die automatisierte, standardisierte Ergebnisdarstellung benötigt.
Die nachfolgende Tabelle 4.2 präsentiert erneut die wichtigsten Charakteristika der verschiedenen Modelle. Auch hier zeigt sich eine Kompatibilität zwischen Urbs und Tessif,

da Urbs nicht von den bereits in Tessif integrierten Modellen abweicht.

Model	Sectors	Model class	Math modeltype	Timeresolution	Georesolution	Is suited for many scenarios
Oemof	Electricity, Heat, Mobility	Energy Modelling Framework	Optimization, Simulation	Hour	User-dependent	false
PyPSA	Electricity, Heat, Transport, User-defined	Energy System Modell	Optimization, Simulation	Hour	User-dependent	true
Calliope	User-dependent	Framework	Optimization	Hour	User-dependent	true
Urbs	User-dependent, Electricity	Energy Modelling Framework	Optimization	Hour	User-dependent	true

Tabelle 4.2.: Kategorisierung anhand Charakteristika der Modelle nach Openmod [7]

Weitergehend müssen noch zusätzliche Aspekte geprüft werden. So muss eine Kompatibilität der Python-Package Versionen von Tessif und Urbs bestehen und das Optimierungsproblem muss durch einen der kompatiblen Solver (CBC, Glpk) gelöst werden können. Außerdem sollten im Idealfall alle Tessif-Komponenten auch in der hinzuzufügenden Software umsetzbar sein.
In Tabelle 4.3 sind die relevantesten Kriterien übersichtlich aufgezählt. Es wird ersichtlich, dass Urbs die Anforderungen erfüllt und sich somit für die Tessif-Integration eignet. Einzig die Tatsache, dass Urbs nicht in der Lage ist, Optimierungen mit gemischtganzzahligen Variablen zu lösen, beschränkt die Nutzung von Urbs innerhalb von Tessif auf lineare Probleme. Allerdings bietet Urbs durch das Demand-Side-Management (DSM) auch neue Funktionen, die zum aktuellen Stand noch nicht in Tessif vorhanden sind.
Da sich Urbs zudem durch eine gut zugängliche und übersichtliche Softwaremodellarchitektur auszeichnet, wird Urbs im Rahmen dieser Arbeit für die Integration in Tessif ausgewählt.

Bedingung	Urbs
Python basiert	Ja
Package Versionen	Mehrheitlich kompatibel
Kompabilität der Tessif-Komponenten	Ja
LP/MILP	LP
Solver	Ja
Bottom-up Ansatz	Ja
Szenarioanalyse	Ja
Sektorenkopplung	Ja
Zeitliche Auflösung	Ja
Räumliche Auflösung	Ja

Tabelle 4.3.: Bedingungen der Tessif-Integration

4.2. Integration des Softwaremodells in Tessif

Die Integration von Urbs in Tessif erfolgt in drei wesentlichen Schritten. Zunächst wird in Tessif die *transform.es2es.rbs.py*-Datei erstellt, welche die Transformation von Tessifs Energiesystemmodellen zu Urbs Eingangsdaten ermöglicht. Des Weiteren wird die *transform.es2mapping.rbs.py*-Datei benötigt, um die Optimierungsergebnisse von Urbs standardisiert darzustellen und um eine anschließende Visualisierung sowie diverse Vergleiche durchführen zu können. Damit die Simulation automatisch durchgeführt werden kann, muss außerdem die *simulate.py*-Datei um eine Funktion für die Optimierung von Urbs erweitert werden. Abschließend wird in diesem Abschnitt noch auf die in Urbs durchgeführten Anpassungen eingegangen.

4.2.1. Es2Es

Die *transform()*-Funktion der *es2es.rbs.py*-Datei ermöglicht die Transformation von Tessifs Energiesystemmodellen, sodass diese von Urbs verwendet werden können. Urbs nutzt anders als die bereits in Tessif integrierten Modellierungsprogramme kein komponentenbasiertes Modell, sondern wie bereits zuvor beschrieben ein tabellenbasiertes Modell. Demnach wird die Transformation zwischen den Modellen komplexer, da die Eigenschaften der einzelnen Komponenten nicht einfach von der Tessif-Komponente zur entsprechenden Modell-Komponente weitergegeben werden können. Vielmehr werden die Eigenschaften einer Tessif-Komponente verschiedenen Tabellen des Urbs-Modells zugeordnet.
Darüber hinaus ist zu beachten, dass Urbs und Tessif ein unterschiedliches Verständnis der räumlichen Struktur eines Energiesystems haben. Bei Tessif sind die Strukturen durch Verbindungen zwischen den einzelnen Komponenten klar definiert. Bei Urbs hingegen werden die einzelnen Prozesse, Verbräuche und Energiequellen einem Gebiet (Site) zugeordnet. Innerhalb dieser Site kann jeder Prozess auf alle in der Site vorhandenen Commodities zurückgreifen. Dieser Aspekt muss bei der Transformation des Energiesystems berücksichtigt werden, damit das Urbs Energiesystem dem ursprünglichen Tessif Energiesystem entspricht.
Die Abbildung 4.1 verdeutlicht die verschiedenen Modellgrundlagen.

Grundsätzlich könnte die Transformation ein Tessif-Energiesystem als Urbs-Energiesystem in Form einer Excel-Tabelle (.xls) ausgeben, die anschließend von Urbs eingelesen und optimiert werden kann. Da dieser Weg allerdings recht umständlich ist, erfolgt die Transformation direkt in die Form, in welcher die *urbs.input.py*-Datei das Energiesystem an Urbs weitergibt. Damit wird die Einlese-Funktion von Urbs umgangen und die *transform()*-Funktion erstellt direkt Pandas-Dataframes mit entsprechenden Multiindexen anstelle der einzelnen Tabellen.
Wie die Transformation im Detail umgesetzt ist, wird in den nachfolgenden Abschnitten tiefergehend anhand einzelner Funktionen erläutert.

transform()

Die *transform()*-Funktion ist die Hauptfunktion die vom Nutzer zur Transformation des Energiesystems aufgerufen wird. Beim Aufrufen der transform()-Funktion muss das zu transformierende Energiesystem angegeben werden. Des Weiteren kann vorgegeben

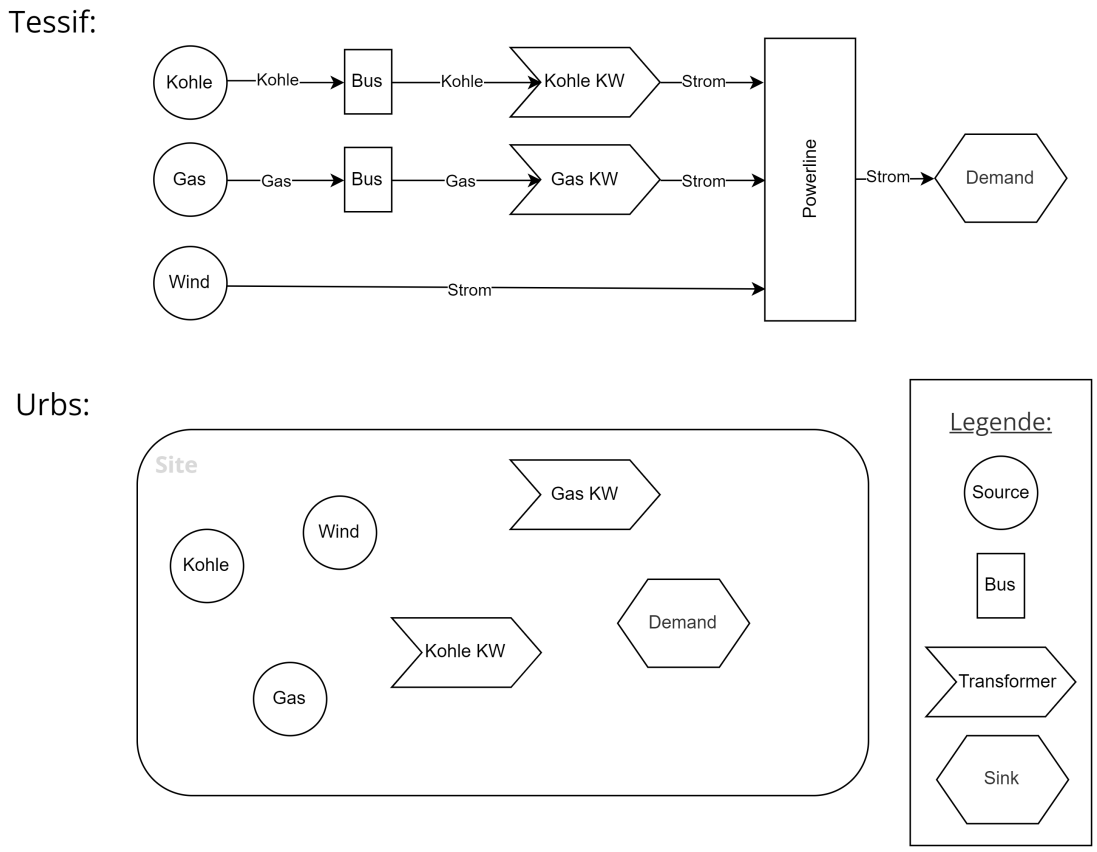


Abbildung 4.1.: Grafische Darstellung der verscheidenen Modellgrundlagen von Urbs und Tessif

werden, ob das Energiesystem in der Lage ist, Lastverschiebungen (DSM) zu nutzen. Die genaue Umsetzung der DSM-Funktion wird im Abschnitt 4.1 näher erläutert. Die *transform()*-Funktion ruft dann weitere Funktionen, die nachfolgend konkreter erläutert werden, auf und erstellt die benötigten Dataframes, um das Urbs-Energiesystem abzubilden. Die *transform()*-Funktion gibt abschließend das Urbs-Energiesystem zurück, das anschließend mit Urbs durch Tessif optimiert werden kann. Das nachfolgende Beispiel 4.1 zeigt die Verwendung der *transform()*-Funktion zum Erstellen eines Urbs-Energiesystems ohne und mit Verwendung von DSM.

```
1 import tessif.examples.data.tsf.py_hard as tsf_py
2 import tessif.transform.es2es.rbs as es2urbs
3
4 es = tsf_py.create_fpwe()   # Laden eines ES aus der Tessif-Bibliothek
5
6 urbs_es = es2urbs.transform(es)   # Urbs-ES ohne DSM
7 urbs_es_dsm = es2urbs.transform(es, True)   # Urbs-ES mit DSM
```

Quellcode 4.1: Beispiel der Verwendung der *transform()*-Funktion mit und ohne Verwendung von DSM

generate_urbs_global()

Die *generate_urbs_global()*-Funktion entnimmt die globalen Informationen aus den global_constraints des Tessif-Energiesystems und generiert damit das urbs_global-Dataframe. Dieses Dataframe beinhaltet zum einen das CO_2-Limit und zum anderen das Cost-Limit des Energiesystems. Das CO_2-Limit wird dabei gemäß Gleichung 4.1 bestimmt.

$$CO_2\text{-Limit} = \text{Tessif-Emissionen} \cdot \frac{8760}{\text{Number of Timesteps}} \tag{4.1}$$

Die Umrechnung wird benötigt, da sich die Emissionen in Tessif auf den Zeitbereich der Optimierung (auf die Anzahl der Zeitschritte) und in Urbs auf den Zeitbereich eines Jahres beziehen. Der Faktor 8760 ergibt sich aus der Anzahl der Stunden eines Jahres. Das Kostenlimit erfordert keine weitere Umrechnung und kann direkt aus den Global-Constraints des Tessif-ES (ressources) übernommen werden.

generate_urbs_site()

Die *generate_urbs_site()*-Funktion wird genutzt, um die Struktur des Energiesystems für Urbs kompatibel zu transformieren. Das Modell von Urbs basiert, wie bereits zuvor erwähnt, nicht auf Komponenten und deren Verbindungen untereinander. Energiesysteme in Urbs sind in Gebiete (Sites) unterteilt und jedes dieser Gebiete bildet im Grunde ein eigenes kleineres Energiesystem. Den einzelnen Gebieten können Verbräuche, Quellen und Transformatoren zugeordnet werden. Innerhalb eines Gebietes gibt es allerdings keine Struktur, sodass zum Beispiel jede Quelle, die Kohle bereitstellt, auch von jedem Kraftwerk, das mit Kohle betrieben wird, genutzt werden kann. Eine konkrete Zuordnung von Quelle zu Transformator ist nur indirekt durch Anpassen des Energieträgers (Commodity) möglich.
Theoretisch könnte die Struktur des Tessif-Energiesystems in Urbs abgebildet werden, indem jede Bus-Komponente durch ein Gebiet (Site) dargestellt wird. Um das Urbs-Energiesystem nicht unnötig komplex zu gestalten, werden nur Busse, die mit Connector-Komponenten verbunden sind oder mehr als einen Input/Output haben, als eigene Gebiete dargestellt. Bus-Komponenten, die nur den Zweck der direkten Versorgung eines Transformators erfüllen, werden innerhalb der Site dargestellt.
So würde zum Beispiel das Energiesystem aus Abbildung 3.2 mit den Bus-Komponenten *Powerline* und *Pipeline* durch eine Site (*Powerline*) in Urbs abgebildet werden.

Alternativ besteht zudem die Möglichkeit, primäre und sekundäre Sites manuell bei der Transformation beziehungsweise dem Ausführen der *transform()*-Funktion anzugeben. Dies ist allerdings nur sinnvoll, sofern die Optimierung aufgrund der automatischen Strukturierung nicht durchführbar sein sollte. Bei den innerhalb dieser Arbeit durchgeführten Untersuchungen wurde dies jedoch nicht benötigt.

generate_urbs_es_dfs()

In der *generate_urbs_es_dfs()*-Funktion werden die Eigenschaften der ursprünglichen Komponenten des Tessif Energiesystems in die verschiedenen Dataframes von Urbs eingetragen. Die *generate_urbs_es_dfs()*-Funktion wird dabei für jede Bus-Komponente, die im Site-Dataframe (siehe generate_urbs_site()) registriert ist, ausgeführt. Die Abbildung 4.2 zeigt den Ablauf innerhalb der *transform()*-Funktion. Zunächst wird eine Base_Node mit den Node-Informationen der Bus-Komponenten aus dem Site-Dataframe bestimmt. Diese Base_Node wird dann mit den leeren, zuvor indizierten

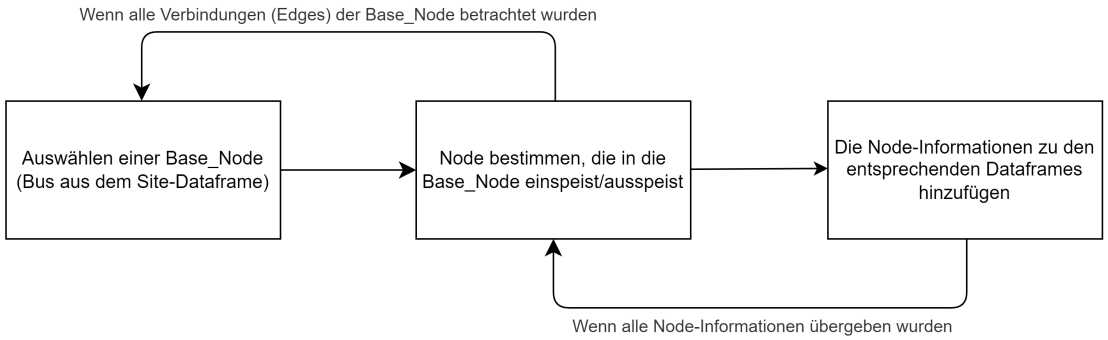


Abbildung 4.2.: Schema der Erstellung der Urbs-Dataframes

Dataframes sowie den Node- und Edge-Informationen des Tessif Energiesystems an die *generate_urbs_es_dfs()*-Funktion übergeben. Durch die Edge-Informationen können die Nodes und somit die Komponenten bestimmt werden, die in die Base_Node ein- oder ausspeisen. Abschließend können die Node-Informationen für die jeweilige Komponente in die entsprechenden Dataframes eingetragen werden. Aufgrund der grundlegenden Modellidee von Urbs ist die Zuordnung zur Base_Node von besonderer Relevanz, um die Struktur des ursprünglichen Energiesystems abzubilden. Dieser Prozess wiederholt sich so lange, bis sowohl jede Komponente, die einen Input liefert oder einen Output aus der Base_Node bezieht, als auch jede Base_Node aus dem Site-Dataframe berücksichtigt wurde.
In Abhängigkeit der jeweiligen Komponente werden die Node-Informationen dabei an unterschiedliche Urbs-Dataframes weitergegeben. Nachfolgend werden die einzelnen Komponenten und ihre Zuordnung näher erläutert. Insbesondere werden unterschiedliche Zuordnungen aufgrund verschiedener Parametrierung einzelner Komponenten erläutert.

Source-Komponente: In Urbs muss zunächst zwischen Source-Komponenten mit und ohne Timeseries unterschieden werden. In Tessif werden Timeseries in der Regel genutzt, um Verläufe von erneuerbaren Energien abzubilden. Source-Komponenten ohne Timeseries können in Urbs einfach als Stock im Commodity-Dataframe angegeben werden. Dabei ist allerdings zu beachten, dass diese in einem Expansionsszenario nicht in ihrer Kapazität erweitert werden können.
Die Integration von Source-Komponenten mit Timeseries ist in Urbs komplexer und insbesondere im Bezug auf variable Zeitreihen nur indirekt realisierbar. Eine solche zum Teil unintuitive Verwendung von Urbs wird allerdings in der Urbs Dokumentation [37] präsentiert und dementsprechend bei der Integration in Tessif berücksichtigt. Die nachfolgende Abbildung 4.3 zeigt die unterschiedlichen Umsetzungen in Urbs für verschieden parametrierte Source-Komponenten mit Timeseries.
Wie bereits zuvor erläutert, wird eine Source ohne Timeseries als Stock im Commodity-Dataframe angegeben. Betrachtet man nun eine Tessif-Source mit Timeseries, bei der in jedem Zeitschritt der maximale und der minimale Wert identisch sind, wird die Source als SupIm im Commodity Dataframe definiert. Darüber hinaus muss die Zeitreihe dem SupIm-Dataframe und ein Prozess im Process- und Process-Commodity-Dataframe hinzugefügt werden. Der Prozess wird benötigt, um den SupIm-Energieträger (z.B. Wind) in den eigentlichen Energieträger Strom zu wandeln. Die Zeitreihe des SupIm ist dabei als Faktor der im Prozess angegebenen Kapazität definiert. Die Werte der SupIm-Reihe

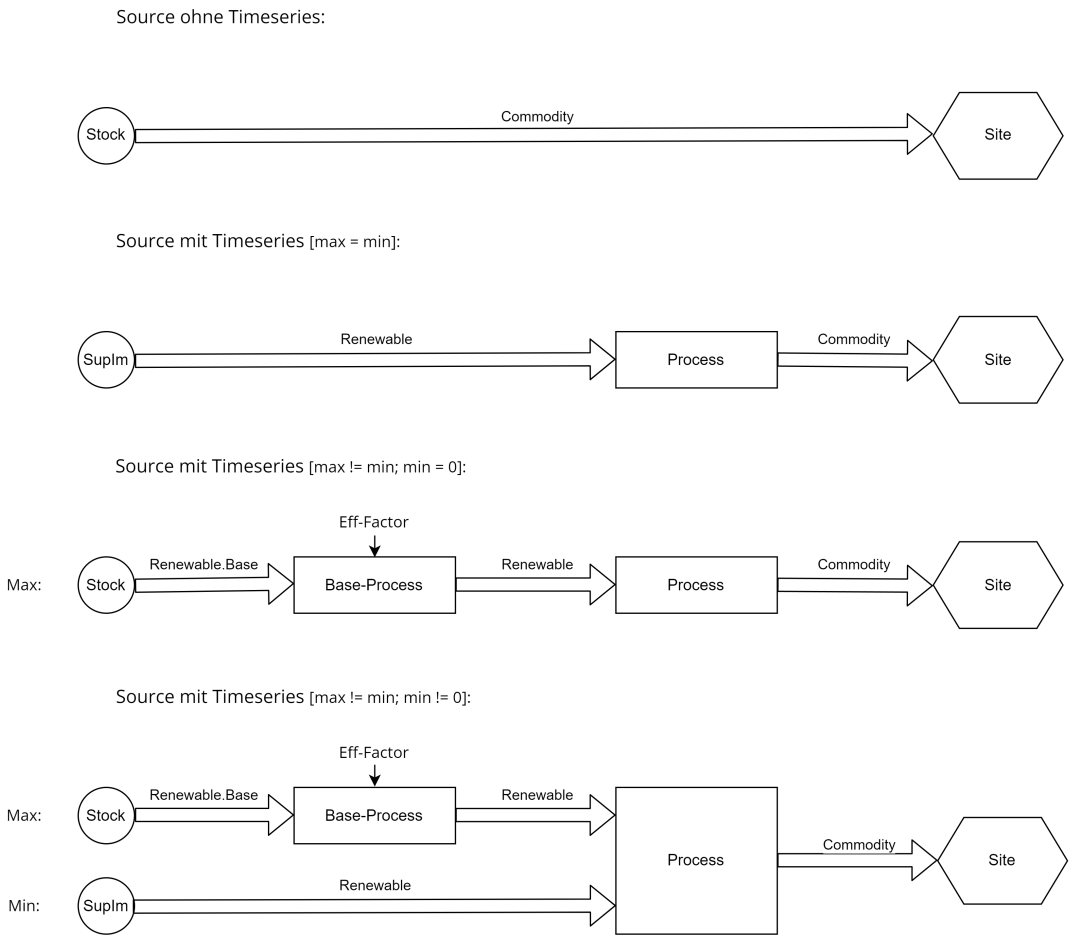


Abbildung 4.3.: Integration von Tessif-Source-Komponenten in Urbs

können gemäß Formel 4.2 bestimmt werden.

$$\text{Wert der SupIm-Reihe} = \frac{\text{Wert der Timeseries}}{\text{Max. Wert der Timeseries}} \tag{4.2}$$

Soll die Source zusätzlich zur Timeseries variabel sein (max!=0; min=0), wird die Timeseries über einen Prozess dargestellt, der von einer uneingeschränkten Stock-Commodity versorgt wird. Dieser Prozess wird innerhalb der Urbs Implementierung als *Base-Process* bezeichnet. Dem *Base-Process* kann zu jedem Zeitschritt ein Effizienzfaktor (Eff-Factor, TimeVarEff) vorgegeben werden, der die Umwandlung beeinflusst. Dementsprechend verändert sich auch die maximale Kapazität zu jedem Zeitschritt. Da durch den Effizienzfaktor der Prozess ineffektiver wird, können die Kosten und Emissionen nicht dem *Base-Process* zugeordnet werden. Deshalb wird nach dem Base-Process ein normaler Process benötigt, der die Kosten und Emissionen berücksichtigt und eine direkte Umwandlung der Energieträger (Wirkungsgrad 100%) realisiert.
Wird zudem eine minimale Zeitreihe bei der Source (max!=min; min!=0) berücksichtigt, muss, wie in Abbildung 4.3 gezeigt, eine zusätzliche SupIm-Commodity mit der minimalen Zeitreihe hinzugefügt werden. Diese erzeugt ebenfalls die *Renewable*-Commodity, die

der Process in die Ziel-Commodity wandeln muss. Es muss allerdings darauf geachtet werden, dass die minimale Zeitreihe zuvor bei der maximalen Zeitreihe abgezogen werden muss, da sonst die maximale Kapazität/Flussrate der Source-Komponente erhöht werden würde.

Transformer-Komponente: Bei den Transformer Komponenten können drei verschiedene Fälle unterschieden werden:

1. Transformer als Connectoren (Der Input ist eine andere Site als der Output)
2. Single Output Transformer
3. Multiple Output Transformer

Beim Hinzufügen der Transformer-Komponenten zu den Urbs-Dataframes wird zunächst geprüft, ob der Transformer nur einen oder mehrere Outputs hat. Handelt es sich um einen Transformer mit mehreren Outputs, muss zudem geprüft werden, ob bereits ein Prozess für den Transformer im Process-Dataframe existiert. Dies kann der Fall sein, wenn der Transformer bereits bei einer anderen Site hinzugefügt wurde.
Daraufhin werden mittels der Node-Informationen die Kosten, Kapazitäten sowie die Expansionsvariablen zum Process-Dataframe hinzugefügt.
Im nächsten Schritt wird geprüft, ob der Transformer ein Connector ist, also der Transformer-Input aus einer anderen Site und somit nicht aus der Base_Node kommt. Ist dies der Fall, muss das Transmission-Dataframe ergänzt werden, sodass die Input-Commodity von der entsprechenden Site an die Base_Node-Site übertragen werden kann. Andernfalls wird der Transformer von einer reinen „Supply Source“ (Bus mit einer Source) versorgt. Diese wird dann, wie bereits zuvor erläutert, als Stock dem Commodity-Dataframe hinzugefügt. Abschließend werden die Conversions der Transformer-Komponente im Process-Commodity-Dataframe angegeben.
Wenn es sich bei dem betrachteten Transformer um einen Transformer mit mehreren Outputs handelt, wird zusätzlich das Transmission-Dataframe erweitert, um eine Übertragung des sekundären Outputs an die andere Site zu ermöglichen. Die nachfolgende Abbildung 4.4 zeigt die drei wesentlichen Integrationsmöglichkeiten von Transformern.

Storage-Komponente: Die Speicher-Komponente kann vollständig über das Storage-Dataframe zum Urbs Energiesystem hinzugefügt werden. Nur Emissionen können Speichern in Urbs nicht zugeordnet werden, sodass diese dementsprechend vernachlässigt werden.

Connector-Komponente: Connectoren können ebenfalls vollständig durch das Transmission-Dataframe in Urbs abgebildet werden.

Sink-Komponente: Sinks werden im Grunde bei der Erstellung der Dataframes des Energiesystems nicht beachtet, da die Verbräuche des ursprünglichen Energiesystems in der separaten Funktion *generate_urbs_demand()* transformiert werden. Allerdings werden unlimitierte Senken (min=0; max=inf) in der *generate_urbs_es_dfs()*-Funktion berücksichtigt. Da Verbräuche in Urbs immer fest definiert sein müssen, erfolgt die Integration eines solchen variablen Verbrauchs über das Storage-Dataframe. Dem Speicher wird dabei ein Ausspeicherwirkungsgrad von 0% und eine Selbstentladung von 100% vorgegeben, damit der Speicher keine aufgenommene Energie zu einem späteren Zeitpunkt abgeben kann. Die Funktion einer solchen variablen unlimitierten

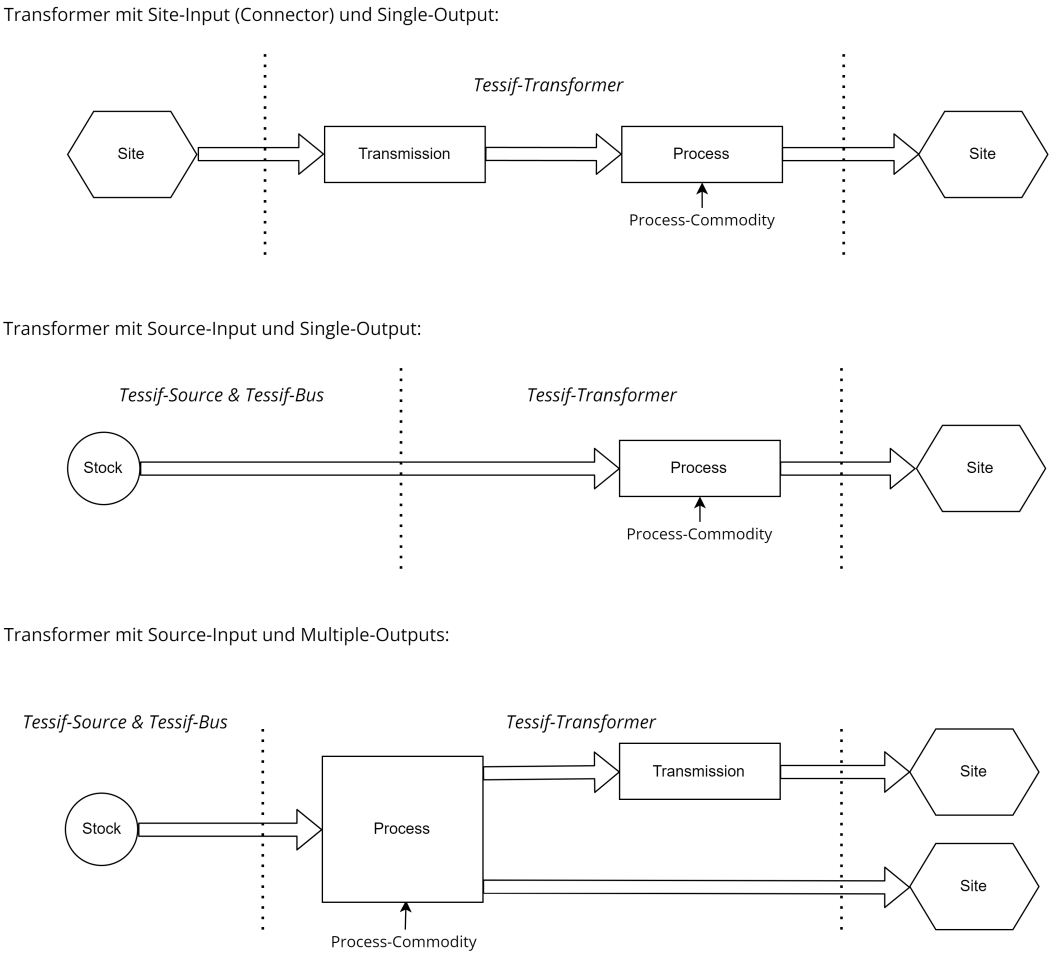


Abbildung 4.4.: Integration von Tessif-Transformer-Komponenten in Urbs

Senke wird somit auf eine alternative Art und Weise umgesetzt, da diese insbesondere beim Erstellen eines neuen Energiesystemmodells oder bei einer Unlösbarkeit von Energiesystemen sehr hilfreich sein kann.

Für einen genaueren Überblick über die Transformation der einzelnen Attribute der Komponenten, kann die Tabelle A.1 im Anhang oder der Code der *tessif.transform.es2es.rbs*-Datei betrachtet werden.

generate_urbs_demand()

Die *generate_urbs_demand()*-Funktion wird genutzt, um die Verbräuche des ursprünglichen Energiesystems in die für Urbs kompatible Form zu konvertieren. In Urbs kann anders als in Tessifs Modell in einem Gebiet (Site) jeweils nur ein Verbrauch für einen Energieträger (Commodity) vorgegeben werden. Darüber hinaus müssen Verbräuche in Urbs eindeutig definiert sein, sodass keine variablen Verbräuche verwendet werden können.
Die *generate_urbs_demand()*-Funktion erstellt zunächst für jede Bus-Komponente, die ein Gebiet (Site) bildet, ein Dataframe in dem alle Verbräuche (Sink-Komponenten)

des jeweiligen Busses aufgelistet werden. Dabei werden üblicherweise die Zeitreihen (Timeseries) der Sink-Komponenten angegeben. Wenn keine Zeitreihe definiert ist, wird der Verbrauch über die maximale beziehungsweise minimale Flussrate (Flow-Rate) der Senke bestimmt, indem ein über den gesamten Zeitbereich konstanter Verbrauch angenommen wird. Die minimale Flow-Rate wird dabei nur genutzt, wenn eine unendliche maximale Flussrate vorgegeben ist. Versorgt ein Bus mehrere Senken, beinhaltet die Tabelle dieser Bus-Komponente so viele Spalten, wie Senken vorhanden sind. Die Werte der einzelnen Spalten werden summiert, sodass in Urbs dem Gebiet ein einzelner Verbrauch zugeordnet werden kann. Abschließend werden die einzelnen Dataframes der Sites durch den Namen und den Energieträger der Bus-Komponente indiziert und zum Demand-Dataframe hinzugefügt.

Zudem benötigt Urbs für jeden Energieträger, der durch Transmission in ein anderes Gebiet übertragen werden kann, eine Angabe des Verbrauchs in dem jeweiligen Gebiet. Daher wird nach Erstellen des eigentlichen Demand-Dataframes überprüft, ob bereits ein Eintrag für die verschiedenen Energieträger existiert und sofern noch kein Verbrauch angegeben ist, wird der Verbrauch mit Null für jeden Zeitschritt angenommen.

generate_urbs_dsm()

Wie bereits im Abschnitt 4.2.1 erläutert, ist Urbs in der Lage Demand Side Management (DSM) beziehungsweise Lastverschiebungen in Energiesystemmodelle zu integrieren. Aktuell sind die Möglichkeiten zur Modellierung von Lastmanagement in Urbs weitaus umfangreicher als in Tessif, sodass in der Standard-Parametrierung eines Tessif Energiesystems nicht ausreichend Informationen vorhanden sind. Damit Urbs trotzdem in der Lage ist, DSM zu nutzen, werden zusätzliche Informationen benötigt. Diese werden, sofern beim Aufrufen der *transform()*-Funktion die DSM-Nutzung als „True“ deklariert ist, durch eine Nutzereingabe bereitgestellt.
Beim Ausführen der *transform()*-Funktion muss der Nutzer zunächst eine Bus-Komponente des Energiesystems angeben, die in der Lage ist, DSM zu betreiben. Darüber hinaus werden zusätzlich die nachfolgenden fünf Variablen benötigt.

1. delay time (h): Die Zeit, um welche die Last nach vorne beziehungsweise nach hinten verschoben werden darf.
2. efficiency: Die Effizienz oder der Wirkungsgrad, mit dem die Lastverschiebung durchgeführt wird.
3. recovery time (h): Die Dauer zwischen zwei Lastverschiebungen.
4. downshift capacity (MW): Die maximal senkbare Energiekapazität in einem Zeitschritt.
5. upshift capacity (MW): Die maximal erhöhbare Energiemenge in einem Zeitschritt.

Nach der Abfrage und Eingabe der Werte für die einzelnen Variablen in die Kommandozeile, kann der Nutzer noch weitere Bus-Komponenten, die DSM verwenden, angeben oder die DSM-Eingabe beenden, sodass automatisch die Transformation des Energiesystems fortgesetzt wird.
Sollte die DSM-Nutzung beim Ausführen der *transform()*-Funktion nicht aktiviert sein, wird die Nutzereingabe direkt übersprungen und die *generate_urbs_dsm()*-Funktion gibt ein leeres Dataframe aus.

indexing_of_dataframes()

Die Funktion *indexing_of_dataframes()* wandelt die zuvor durch die *generate_urbs_es_dfs()*-Funktion generierten Dataframes (Commodity, Process, Process-Commodity, Transmission, Storage-Urbs, SupIm und Eff-Factor) durch die nachfolgenden drei Schritte in die von Urbs geforderte Form.

1. Entfernen von Duplikaten [drop_duplicates()]
2. Setzen von Indexen [set_index()]
3. Bearbeiten von Indexen [split_columns(); Nur bei SupIm und Eff-Factor]

Im ersten Schritt werden doppelte Einträge aus den Dataframes entfernt. Diese können beispielweise beim Generieren durch wiederholtes Abfragen oder aus anderen notwendigen Bedingungen auftreten. Sollten beim Transmission-Dataframe Variablen doppelt aufgelistet sein, wird der Datensatz mit der größeren installierten Kapazität entfernt, um beispielweise Limitierung des Netzes abbilden zu können und diese nicht durch eine unendlich große Übertragungskapazität zu ersetzen.
Im zweiten Schritt werden die Dataframes so indiziert, dass Urbs im weiteren Verlauf auf die einzelnen Daten zugreifen und aus diesen ein lineares Optimierungsmodell erstellen kann. Die verschiedenen Indexe sind entweder bereits in den zuvor generierten Dataframes enthalten oder werden zum Beispiel in Form eines zeitlichen Indexes innerhalb der Funktion erstellt.
Im dritten und letzten Schritt werden zuvor definierte Indexe von Datenreihen in Multiindexe gewandelt. Dies ist allerdings nur bei den Daten von SupIm und Eff-Factor nötig, da die Spaltenindexe sich aus den Site- und Commodity-Informationen zusammensetzen und diese in der weiteren Nutzung von Urbs als separate Multiindexe benötigt werden.

4.2.2. Es2Mapping

Da jedes Softwaremodell eine andere Darstellungsweise der Optimierungsergebnisse verwendet, können erzielte Ergebnisse nicht direkt für Vergleichszwecke genutzt werden. Um einen solchen Vergleich zu ermöglichen, muss zunächst eine standardisierte Darstellung als Vergleichsbasis erzielt werden. Zu diesem Zweck überträgt die Es2Mapping-Funktion die Ergebnisse der Optimierung in die einheitliche Darstellungsweise von Tessif. Dafür werden die Urbs-Ergebnisse den verschiedenen Resultiers von Tessif zugeordnet.
Zunächst wird in diesem Abschnitt die von Urbs verwendete Darstellungsweise der Ergebnisse präsentiert und im weiteren Verlauf wird auf die verschiedenen relevanten Tessif-Resultiers weiter eingegangen. Der Fokus liegt dabei insbesondere auf der Übertragung und eventuell benötigten Umrechnungen der Ergebnisse von Urbs zu Tessifs Darstellungsweise. Zu den relevanten Resultiers gehören der *Urbs Resultier*, der *Load Resultier*, der *Capacity Resultier*, der *Storage Resultier*, der *Flow Resultier* und der *Integrated Global Resultier*.

Urbs Darstellung der Ergebnisse

Auf die Ergebnisse der Optimierung von Urbs kann durch die Befehle im Code 4.2 zurückgegriffen werden.

```
# Laden und Optimieren des Beispielenergiesystems (fpwe)
import tessif. examples.data.tsf.py_hard as coded_examples
es = coded_examples.create_fpwe()
import tessif.transform.es2es.rbs as es2urbs
urbs_es = es2urbs.transform(es)
import tessif.simulate as simulate
optimized_urbs_es = simulate.rbs_form_es(urbs_es)

# Auswerten der Bus Komponente (Powerline)
from datetime import date
year = year.date.today().year
from urbs.output import get_timeseries
(created, consumed, stored, imported, exported, dsm, voltage_angle)
    \
    = get_timeseries(optimized_urbs_es['model'], year, 'electricity',
    'Powerline')

# Auswerten eines Teils des Energiesystems (Ausgabe der Outputs der
    Prozesse)
from urbs.pyomoio import get_entity, get entities
process_out = get_entity(optimized_urbs_es['model'], 'e_pro_out')
```

Quellcode 4.2: Funktionen, um auf Urbs Optimierungsergebnisse zuzugreifen

Dabei kann die get_timeseries()-Funktion nur für Bus-Komponenten, die ein eigenes Gebiet (Site) darstellen, verwendet werden. Die Funktion erstellt automatisch verschiedene Dataframes für die Erzeugung, den Import, den Export, den Verbrauch und die Speicherung des angegeben Energieträgers. Tabelle 4.4 zeigt das Dataframe „created“ aus dem Beispiel 4.2.

In der Tabelle werden die Einspeisungen des ausgewählten Energieträgers für alle Prozesse innerhalb der Site dargestellt. Bei diesem Beispiel speisen nur der Generator und das Solar Panel Strom ein. Darüber hinaus wird noch angegeben, ob der Energie-

t	Generator	Solar Panel	Stock
1	0.0	12.0	0
2	0.0	3.0	0
3	13.1	7.0	0

Tabelle 4.4.: Dataframe für die created-Informationen der Powerline des optimierten ES

träger direkt durch eine Stock-Commodity erzeugt wird. Die Werte werden dabei für jeden Zeitschritt separat angegeben. Alle weiteren Dataframes, die beim Aufrufen der get_timeseries-Funktion erstellt werden, sind auf die gleiche Weise strukturiert und können für die Transformation in Tessifs standardisierte Ergebnisdarstellung verwendet werden.
Die get_entity()-Funktion kann zum Erstellen verschiedener Dataframes genutzt werden. Die für die weitere Auswertung wichtigsten, von Urbs generierten Dataframes sind nachfolgend aufgelistet.

- **e_pro_in**: Inputs der einzelnen Prozesse
- **e_pro_out**: Outputs der einzelnen Prozesse
- **e_co_stock**: Outputs von Stock-Commodities (Sources)
- **e_sto_in, e_sto_out, e_sto_con**: Speicher Inputs, Outputs und SoC
- **e_tra_in**: Inputs der Transmissionen
- **e_tra_out**: Outputs der Transmissionen
- **cap_pro**: Kapazitäten der Komponenten (Nur Prozesse und Transmissionen)
- **cap_pro_new**: Neue Kapazitäten nach Ausbau (Nur bei Expansionsmodellen)

Durch die get_entity()-Funktion können somit die wichtigsten Ergebnisse der Optimierung in Form von Dataframes zugänglich gemacht werden. Nachfolgend zeigt die Tabelle 4.5 das in Beispiel 4.2 erstellte Dataframe „process_out“.

Die Tabelle 4.5 zeigt alle Outputs von allen Prozessen des Energiesystems. In diesem Fall sind es nur zwei Prozesse (Generator und Solar Panel), die jeweils nur Elektrizität und CO_2 ausgeben. Neben den Werten wird der aktuelle Zeitschritt, die Site, in der sich der Prozess befindet, sowie der Name des Prozesses mitangegeben. Dies ermöglicht im Postprocessing das Zuordnen der Werte zu den entsprechenden Komponenten des Tessif-Energiesystems. Die weiteren aufgelisteten Dataframes, die durch die get_entity()-Funktion erstellt werden können, sind auf eine ähnliche Weise strukturiert. In Abhängigkeit des Dataframes können allerdings unterschiedliche Indizes auftreten.

Urbs Resultier

Der Urbs Resultier bildet die Basis aller weiteren Resultiers. Im Urbs Resultier werden die Komponenten und die Struktur des optimierten Energiesystems in Tessifs Darstellungsweise mit Nodes und Edges gebracht. Dabei erstellt der Resultier eine Liste der

t	stf	sit	pro	com	e_pro_out
1	2022	Powerline	Generator	CO2	0.0
1	2022	Powerline	Generator	electricity	0.0
1	2022	Powerline	Solar Panel	CO2	0.0
1	2022	Powerline	Solar Panel	electricity	12.0
2	2022	Powerline	Generator	CO2	0.0
2	2022	Powerline	Generator	electricity	0.0
2	2022	Powerline	Solar Panel	CO2	0.0
2	2022	Powerline	Solar Panel	electricity	3.0
3	2022	Powerline	Generator	CO2	224.57143
3	2022	Powerline	Generator	electricity	13.1
3	2022	Powerline	Solar Panel	CO2	0.0
3	2022	Powerline	Solar Panel	electricity	7.0

Tabelle 4.5.: Dataframe der e_pro_out-Informationen des optimierten ES

Nodes und eine Liste der Edges des Energiesystems. Darüber hinaus werden die Uid-Informationen der einzelnen Nodes beziehungsweise Komponenten in einem Dictionary (uid_nodes) hinterlegt.
Das optimierte Urbs Energiesystem besitzt allerdings keine eindeutige Struktur, die genutzt werden kann, um die benötigten Node- und Edge-Informationen abzuleiten. Die Abbildung 4.1 verdeutlicht die Strukturlosigkeit von Urbs Energiesystemen. Da diese Informationen für Tessifs Auswertung grundlegend sind, werden die Node- und Edge-Informationen des ursprünglichen Tessif Energiesystems beim Transformieren und Optimieren jeweils weitergegeben. Somit enthält das optimierte Urbs Energiesystem die Optimierungsergebnisse und die Node- und Edge-Informationen des ursprünglichen Energiesystems.

Load Resultier

Beim Optimieren des Energiesystems wird der ideale zeitliche Verlauf der Energieerzeugung bestimmt. Die Verbräuche sind in Urbs grundsätzlich durch ein festes Verbrauchsprofil definiert, sodass diese bei der Optimierung möglichst effektiv durch die Profile der Energieerzeuger abgedeckt werden müssen. Die Darstellungsweisen der Erzeugungs- sowie der Lastprofile können dabei sehr unterschiedlich ausfallen. Um die Profile vergleichbar zu machen, werden die Optimierungsergebnisse im Load Resultier in Tessifs einheitliche Darstellungsweise umgewandelt.
Tessifs Darstellung zeichnet sich durch eine tabellarische Darstellung der Energieflüsse (Inflows und Outflows) der einzelnen Komponenten aus. Dabei sind Inflows mit einem negativen und Outflows mit einem positiven Vorzeichen definiert. Darüber hinaus wird über die Beschriftung der Spalten angegeben, welche andere Komponente jeweils den Energiefluss abgibt oder aufnimmt.
Für die Auswertung wird zunächst über die einzelnen Nodes beziehungsweise Komponenten iteriert und für jede Node in Abhängigkeit der Art der Komponente die entsprechenden Tessif Tabellen erstellt. Der Load Resultier nutzt dabei die Node- und Edge Informationen des Energiesystems, um den jeweiligen Komponenten die entsprechenden ein- und ausgespeisten Leistungen über die Multiindexe der von Urbs ausgegebenen Dataframes zuzuordnen. Je nach Art der Komponente wird dabei auf andere Urbs Ergebnistabellen (siehe 4.2) zurückgegriffen. Die genaue Zuordnung der Ergebnisdataframes

zu den Komponenten kann der Tabelle A.2 im Anhang entnommen werden.
Da in Urbs die Verbräuche innerhalb eines Gebietes und damit die Verbräuche einer Bus-Komponente zusammengefasst sind, müssen die Zeitreihen der Verbräuche ebenfalls, wie die Node- und Edge-Informationen direkt aus dem ursprünglichen Tessif Energiesystem bezogen werden.
Die nachfolgende Tabelle 4.6 verdeutlicht die Darstellungsweise von Tessif anhand der Powerline (Bus-Komponente) des in Tessif integrierten Beispielenergiesystems (fpwe()) nach der Optimierung mit Urbs und der Transformation der Ergebnisse durch den Load Resultier.

	Battery	Generator	Solar Panel	Battery	Demand
1990-07-13 00:00:00	-0.0	-0.0	-12.0	1.0	11.0
1990-07-13 01:00:00	-8.0	-0.0	-3.0	0.0	11.0
1990-07-13 02:00:00	-0.0	-13.1	-7.0	9.1	11.0

Tabelle 4.6.: Lasten und Einspeisungen der Powerline in Tessifs Darstellungsweise (fpwe())

Die in die Powerline eingespeisten Leistungen (Battery, Generator und Solar Panel) werden nach Tessifs Konvention mit negativem Vorzeichen aufgelistet. Die ausgespeisten Leistungen (Battery und Demand) werden entsprechend mit positiven Vorzeichen dargestellt. Komponenten, die Leistungen sowohl ein- als auch ausspeisen (Battery), werden in Tessifs Darstellungsweise durch zwei separate Einträge angegeben.

Storage Resultier

Der Storage Resultier gibt den aktuellen Speicherstand (SoC) der im Energiesystem verwendeten Speicherkomponenten an. Dieser kann nach der Optimierung mit Urbs aus dem Dataframe *e_sto_con* für jeden Speicher des Energiesystems entnommen werden. Sowohl in Urbs als auch in Tessif ist der Speicherstand in absoluten Werten angegeben, sodass keine weitere Umrechnung nötig ist. Allerdings ist im Urbs Dataframe zusätzlich der initiale Speicherstand in einem nullten Zeitschritt angegeben und muss daher beim Übergeben der Werte an Tessif ignoriert werden.

Capacity Resultier

Der Capacity Resultier gibt die relevanten Informationen bezüglich der Kapazität der Komponenten an. Der Capacity Resultier kann dabei in die nachfolgenden vier Funktionen unterteilt werden.

- _map_installed_capacities
- _map_original_capacities
- _map_expansion_costs
- _map_characteristic_values

Die *_map_installed_capacities*-Funktion gibt die installierte Leistung der einzelnen Komponenten an. Die Werte der Kapazitäten der Erzeuger werden dabei aus dem Urbs Dataframe *cap_pro* entnommen. Allerdings ordnet Urbs nur Prozessen und somit nur

Transformern und gegebenenfalls Quellen Werte zu. Sofern für eine Komponente keine Kapazität im Urbs Dataframe vorhanden ist, wird der maximale Wert der Zeitreihe beziehungsweise die maximale Flussrate als installierte Kapazität definiert. Bei einer unbegrenzten Flussrate wird stattdessen der maximale Wert der abgegebenen oder aufgenommenen Leistung verwendet.
Da sich die Kapazität bei Urbs auf den Input und bei Tessif auf den Output bezieht, muss bei Transformer-Komponenten zusätzlich der Umwandlungsfaktor gemäß Gleichung 4.3 berücksichtigt werden.

$$\text{inst\_cap} = \text{cap\_pro [node]} \cdot \text{conversion\_factor} \tag{4.3}$$

Die Kapazität von Speichern kann aus dem Dataframe *cap_sto_c* entnommen werden. Bei Ausbauszenarien geben die *Installed Capacities* die Werte nach dem Ausbau der Komponenten wieder.

Als *Original Capacities* werden die ursprünglichen Kapazitäten der Komponenten bezeichnet. Diese werden im Grunde identisch zu den *Installed Capacities* bestimmt, mit dem Unterschied, dass die neu hinzugefügten Kapazitäten von den installierten Kapazitäten abgezogen werden. Die im Zuge des Ausbaus hinzugefügten Kapazitäten können dabei den Dataframes *cap_pro_new* und *cap_sto_c_new* entnommen werden. Somit ergeben sich die originalen Kapazitäten durch die nachfolgende Gleichung.

$$\text{inst\_cap}_{\text{original}} = \text{cap\_pro [node]} - \text{cap\_pro\_new [node]} \tag{4.4}$$

Die Funktion *_map_expansion_costs* listet die Expansionskosten der einzelnen Nodes beziehungsweise Komponenten auf. Dabei wird zunächst geprüft, ob die Komponente ausgebaut werden kann und abhängig davon wird der Wert für die Expansionskosten in die Liste übertragen.

Darüber hinaus werden die *Characteristic Values* der Komponenten bestimmt. Diese sind insbesondere für die grafische Auswertung von Tessif relevant. Der charakteristische Wert ist in diesem Kontext als

$$cv = \frac{\text{sum(characteristic flow)}}{\text{installed capacity}} \tag{4.5}$$

für Quellen, Senken sowie Transformer und als

$$cv = \frac{\text{sum(SoC)}}{\text{capacity}} \tag{4.6}$$

für Speicher definiert. Hierbei ist die Summe der *Characteristic Flows* für Source- und Sink-Komponenten als Summe der auf- oder abgegebenen Leistungen definiert. Bei Transformern bildet die Summe über die erste ausgegebene Leistung den *Characteristic Flow*.

Flow Resultier

Der Flow Resultier bezieht die Ergebnisse der Optimierung auf die verschiedenen Edges des Energiesystems. Als Edge werden alle Verbindungen zwischen den einzelnen Komponenten bezeichnet. Die Edges werden durch die jeweiligen Ausgangs- und Zielkomponenten eindeutig definiert.

$$\text{Edge(source='Komponente A', target='Komponente B')} \tag{4.7}$$

Die Informationen der Edges werden direkt aus dem ursprünglichen Energiesystem an das Es2Mapping übergeben. Anschließend ordnet der Flow Resultier den einzelnen Edges spezifische Emissionen und Kosten des jeweiligen Energieflusses zu. Die Kosten und Emissionen werden dabei den entsprechenden Node-Informationen entnommen. Darüber hinaus werden die insgesamt umgesetzten Energieflüsse der Edges bestimmt. Diese können aus den zuvor erstellten Erzeugungs- und Lastprofilen des Load Resultiers durch Aufsummieren der Energieflüsse ermittelt werden.
Der Flow Resultier ermöglicht die eindeutige Beschreibung jedes Energieflusses und somit die Darstellung flussbezogener Ergebnisse. Dies bildet insbesondere für die grafische Auswertung und damit den automatisierten Softwarevergleich eine wichtige Grundlage.

Integrated Global Resultier

Der Integrated Global Resultier fasst die wesentlichen Ergebnisse der Optimierung zusammen. Dabei liegt der Fokus von Tessifs Darstellungsweise der Gesamtergebnisse auf den Kosten und Emissionen. Tessif unterteilt die Kosten weitergehend in operative Kosten (OpEx) und Investitionskosten (CapEx). Die operativen Kosten werden in Tessifs Kontext als Kosten, die beim Betrieb des Energiesystems anfallen, und die Investitionskosten als Kosten, die durch den Ausbau von Komponenten in Expansionsszenarien anfallen, definiert.
Die Emissionen des Energiesystems über den Optimierungszeitraum können nach Gleichung 4.8 aus dem *e_pro_out*-Dataframe bestimmt werden.

$$Emissionen_{ges} = \sum_{K=1}^{n} \sum_{t=1}^{T} Emissionen_{K,t} \tag{4.8}$$

Dabei wird zunächst die Summe der Emissionen einer Komponente K über alle Zeitschritte T gebildet. Die Gesamtemissionen können anschließend durch Aufsummieren der Emissionen der einzelnen Komponenten berechnet werden.
Im Integrated Global Resultier werden die Emissionen durch die Summe aller Einträge des *e_pro_out*-Dataframes bestimmt, deren Commodity-Index (com) mit CO2 angegeben ist (siehe 4.5). Dadurch wird sowohl die Summe über alle Komponenten n als auch die Summe über alle Zeitschritte T gebildet.

Über die Urbs Funktion *get_constants()* kann auf die Kosten des optimierten Energiesystems zurückgegriffen werden. Allerdings unterteilt Urbs die Kosten in die vier folgenden Kategorien.

- Invest
- Fixed
- Variable
- Fuel

Die festen und variablen Kosten bilden gemeinsam mit den Kosten für den Kraftstoff die operativen Kosten des Energiesystems, die durch den Betrieb entstehen. Da in Urbs mit Ausnahme der Investitionskosten alle weiteren Kosten auf den Zeitraum eines Jahres skaliert sind, müssen die Kosten gemäß Gleichung 4.9 umgerechnet werden.

$$\mathrm{OpEx}_{\mathrm{Tessif}} = (\mathrm{Fixed} + \mathrm{Variable} + \mathrm{Fuel}) \cdot \frac{\text{Anzahl der Zeitschritte}}{8760} \tag{4.9}$$

Auf diese Weise können die in Tessif angegebenen operativen Kosten für den optimierten Zeitbereich bestimmt werden.
Die Investitionskosten des Global Resultiers werden durch die nachfolgende Gleichung 4.10 berechnet.

$$\text{CapEx}_{\text{Tessif}} = \text{Invest } (\cdot\text{Abschreibungszeitraum}) \tag{4.10}$$

Dabei muss im Grunde nur ein Faktor für den Abschreibungszeitraum berücksichtigt werden. Dieser ist darauf zurückzuführen, dass in Urbs jedem expandierbaren Prozess eine wirtschaftliche Lebensdauer zugeordnet wird. Diese Lebensdauer bildet den Abschreibungszeitraum, auf den die Investitionskosten aufgeteilt werden können. Die von Urbs ausgegebenen Investitionskosten (Invest) sind somit die jährlichen Investitionskosten, die sich aus den jeweiligen auf den Abschreibungszeitraum aufgeteilten Investitionskosten ergeben.
In der Tessif Implementation von Urbs ist die wirtschaftliche Lebensdauer für alle Prozesse als ein Jahr definiert, sodass der Abschreibungszeitraum keinen Einfluss auf die Investitionskosten hat. Demzufolge sind die Investitionskosten von Tessif und Urbs identisch.
Letztlich lassen sich die Gesamtkosten des Energiesystems durch Summieren der operativen Kosten und der Investitionskosten bestimmen.

$$\text{Gesamtkosten}_{\text{Tessif}} = \text{OpEx}_{\text{Tessif}} + \text{CapEx}_{\text{Tessif}} \tag{4.11}$$

4.2.3. Tessif-Simulate

Mit der *rbs_from_es()*-Funktion der Simulate Datei wird die Optimierung des transformierten Energiesystems in Tessif durch Urbs ausgeführt. Beim Aufrufen der Funktion muss im Grunde nur das zu optimierende Energiesystem angegeben werden. Allerdings besteht die Möglichkeit, einen individuellen Zeitbereich für die Optimierung zu wählen, sodass eine geringere Anzahl an Zeitschritten optimiert wird. Dabei ist zu beachten, dass kein Zeitbereich angegeben wird, der größer als die angegebenen Zeitschritte des Energiesystems ist, da sonst nicht ausreichend Daten für Verbräuche oder Zeitreihen von erneuerbaren Erzeugern existieren. Sofern keine manuelle Vorgabe erfolgt, wird die Anzahl der Zeitschritte des zu optimierenden Energiesystems übernommen.
Des Weiteren kann noch der zu verwendende Solver gewählt werden. Standardmäßig wird der Solver CBC [16] verwendet. Alternativ kann aber auch der Solver GLPK [17] oder ein anderer mit Pyomo [39] kompatibler Solver für die Optimierung verwendet werden.
Urbs ist, wie bereits zuvor erwähnt, in der Lage das Optimierungsziel zu wechseln. So kann zwischen einer Optimierung bezüglich der Kosten und der Emissionen gewählt werden.
Im Grunde ersetzt die *rbs_from_es()*-Funktion den Urbs Runfile (3.2.2). Da allerdings nicht alle Funktionen von Urbs genutzt werden und insbesondere die Auswertungstools von Urbs nicht weiter relevant sind, entfallen einige Aspekte des Runfiles. Innerhalb der *rbs_from_es()*-Funktion wird letztlich auch die *run_scenario()*-Funktion und somit die Optimierung in Urbs ausgeführt.
Abschließend fügt die *rbs_from_es()*-Funktion die für die Auswertung zusätzlich benötigten Informationen zu den Optimierungsergebnissen hinzu. Das sind zum einen die Node- und Edge-Informationen sowie der Zeitrahmen der Optimierung und zum anderen die Verbrauchsprofile der Senken sowie die Site-Informationen des Urbs Energiesystems.

4.2.4. Anpassungen in Urbs

Um eine Integration von Urbs in Tessif zu ermöglichen, müssen auch einige Änderungen in Urbs erfolgen. Die Änderungen können in zwei Arten unterteilt werden. So gibt es Änderungen, die interne Funktionen von Urbs beeinflussen und nur zur Nutzung von Urbs benötigt werden. Zu den letzteren gehören hauptsächlich kleine Änderungen, die durch unterschiedliche Versionen von Python Packages benötigt werden. So werden insbesondere aufgrund unterschiedlicher Pyomo und Pandas Versionen von Tessif und Urbs Anpassungen notwendig. Dies ist der Fall, da neuere oder ältere Versionen zum Teil andere Funktionen beziehungsweise andere Notationen verwenden. Die durchgeführten Anpassungen haben allerdings keinen Einfluss auf die von Urbs erzeugten Optimierungsergebnisse.
Neben solchen versionsbedingten Anpassungen gibt es noch Änderungen, die einige der Urbs Funktionen beeinflussen. Die Änderungen beziehen sich im Wesentlichen auf die Auswertungsfunktionen von Urbs, die durch die einheitliche Auswertung von Tessif nicht benötigt werden. Dazu zählen zum einen die Ausgabe der Ergebnisse in Form einer Tabelle oder als Graph. Diese können einfach unterdrückt werden, indem die jeweiligen Befehle *report()* und *result_figures()* in der *urbs.runfunctions.py*-Datei auskommentiert werden.
Des Weiteren muss der Input so geändert werden, dass das zu optimierende Energiesystem nicht durch die *urbs.input.py*-Datei und die Funktion *read_input()* aus einer xls-Datei eingelesen, sondern direkt vorgegeben wird. Dies ist möglich, da beim Transformieren mit der *Es2Es*-Datei das Energiesystem direkt in die von Urbs benötigte Form gewandelt wird.

Zu erwähnen ist dabei, dass keine der durchgeführten Anpassungen das Modell oder die Optimierung von Urbs und somit auch die Ergebnisse der Optimierung beeinflussen. Durch die Änderungen wird nur eine Versionskompatibilität mit den Python Packages von Tessif sowie eine Unterdrückung der Auswertungstools von Urbs durchgeführt.

4.2.5. Integration von Urbs in Tessif

Die nachfolgendende Abbildung 4.5 verdeutlicht übersichtlich, wie die Integration von Urbs in Tessif erfolgt. Insbesondere wird ersichtlich, welche Funktionen die zuvor erläuterten Python Files Es2Es, Es2Mapping und Tessif-Simulate übernehmen. Wie aus der Grafik 4.5 ersichtlich wird, wandelt die Es2Es-Datei das Tessif-Energiesystem in ein Urbs-Energiesystem um. Zusätzlich ermöglicht die Es2Es-Datei noch die manuelle Eingabe der Demand-Side-Management (DSM) Informationen, wodurch Urbs in der Lage ist eine Laststeuerung zu berücksichtigen. Des Weiteren entnimmt die Simulate-Datei dem Tessif Energiesystem für die Optimierung relevante Informationen, wie beispielweise die Anzahl der zu optimierenden Zeitschritte. Die Simulate-Datei ersetzt den ursprünglichen Runfile von Urbs.
Die Es2Mapping-Datei wandelt die Optimierungsergebnisse von Urbs in das einheitliche Ergebnisformat von Tessif um. So können die Vergleichswerkzeuge von Tessif genutzt werden, um die Optimierungsergebnisse verschiedener in Tessif implementierter Modellierungs-/Optimierungsprogramme wie Oemof, PyPSA, FINE, Caliope oder Urbs zu vergleichen. Wie in Abschnitt 4.2.2 beschrieben, werden insbesondere für eine grafische Auswertung der Ergebnisse die Node-Informationen des Energiesystemmodells benötigt. Da im Urbs-Energiesystem diese Informationen nicht vorhanden sind, werden diese direkt aus dem Tessif-Energiesystem an die Es2Mapping-Datei und das Postprocessing

weitergegeben.

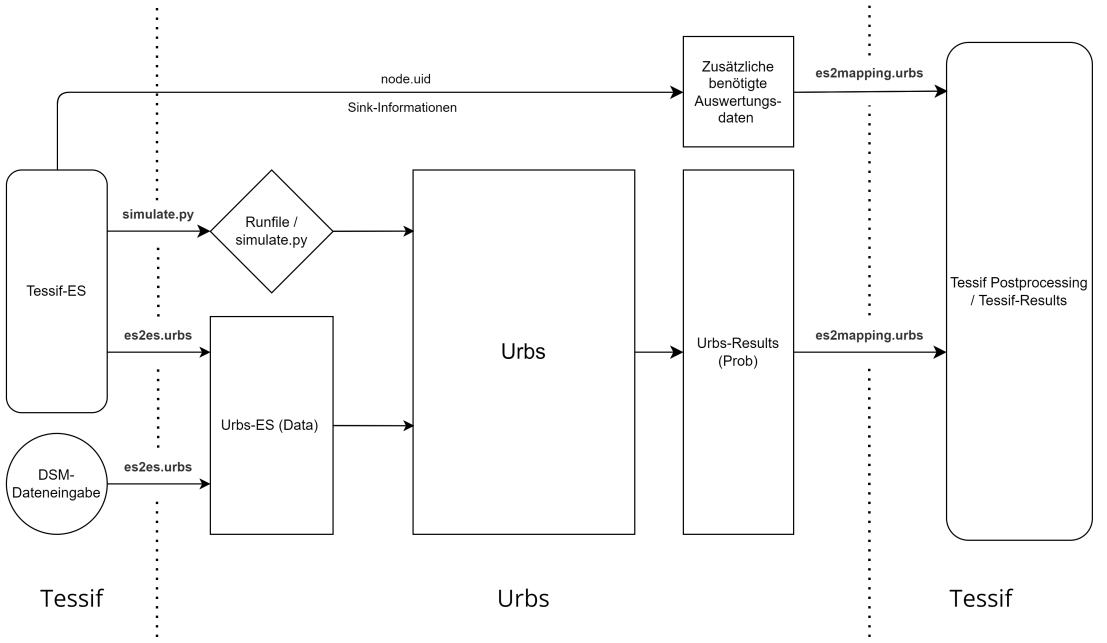


Abbildung 4.5.: Grafische Darstellung der Integration von Urbs in Tessif

5. Exemplarischer Vergleich und Auswertung

Der exemplarische Vergleich wird durchgeführt, um zu prüfen, ob die Implementierung von Urbs in Tessif erfolgreich war. Der im Rahmen dieser Arbeit durchgeführte Vergleich setzt sich aus zwei verschiedenen Vergleichen zusammen. In einem ersten Schritt wird der automatische Vergleich von Tessif (Auto-Comparatier) verwendet, um verschiedene simple, in Tessif hinterlegte Energiesysteme mit Oemof, PyPSA, FINE, Calliope und Urbs zu optimieren. Die Ergebnisse der Optimierungen werden durch den Auto-Comparatier in Form von Kosten und Emissionen abgebildet und anschließend verglichen. Die für diesen Vergleich verwendeten Energiesysteme werden im Abschnitt 5.1 genauer beschrieben.
Im zweiten Teil des exemplarischen Vergleichs wird anhand des komponentenbasierten Energiesystems (component_es) eine detaillierte Auswertung für Oemof, PyPSA und Urbs durchgeführt. Dabei wird insbesondere auch die grafische Auswertung durch Tessif betrachtet und zum Vergleichen der Ergebnisse genutzt. Das komponentenbasierte Energiesystem bietet sich aufgrund des langen Optimierungszeitraums und der Möglichkeit zur Expansion besonders für einen tiefergehenden Vergleich an.
Neben der Validierung der Ergebnisse kann der exemplarische Vergleich genutzt werden, um mögliche Einschränkungen von Urbs beziehungsweise Unterschiede zwischen den verschiedenen Modellierungsprogrammen aufzuzeigen.

5.1. Auswahl und Beschreibung der verwendeten Energiesysteme

Im Rahmen des exemplarischen Vergleichs werden nur Energiesystemmodelle verwendet, die in Tessifs Bibliothek (*tessif.examples.data.tsf.py_hard.py*) hinterlegt sind. Welche Modelle im Detail genutzt werden, wird im Abschnitt 5.1.1 und 5.3.2 beschrieben.
Durch den nachfolgenden Befehl 5.1 können Energiesysteme aus Tessifs Modellbibliothek geladen werden.

```
# Laden des Tessifmodells
import tessif.examples.data.tsf.py_hard as tsf_py
esys = tsf_py.create_fpwe()
```

Quellcode 5.1: Aufrufen des voll parametrierten Beispielmodells (create_fpwe()) über *tessif.examples.data.tsf.py_hard*

Zum Prüfen des Modells können entweder die Komponenten des Energiesystems aufgelistet oder der Node Graph des Energiesystems gezeichnet werden. Die Befehle dafür sind in 5.2 gezeigt. Bei der Visualisierung wurde zusätzlich die empfohlene Farbcodierung aus der *tessif.examples.data.tsf.py_hard*-Datei angewendet, um die einzelnen Komponenten farblich darzustellen. Aus Gründen der Übersicht wird diese nicht im Code 5.2 gezeigt. Der auf diese Weise erstellte Node Graph ist in Abbildung 5.1 zu sehen.

```
# Auflisten der Komponenten
for node in esys.nodes:
    print(node.uid.name)

# Grafisches Darstellen des Energiesystems
import tessif.visualize.dcgrph as dcv
app = dcv.draw_generic_graph(
  energy_system=esys,
  color_group={},  # Standard Farbkodierung
  )
app.run_server(debug=False)
```

Quellcode 5.2: Anzeigen der Komponenten sowie grafische Darstellung des Energiesystems

Einigen Modellen können beim Aufrufen noch zusätzliche Modellparameter vorgegeben werden. Insbesondere zur Auswahl der zu optimierenden Zeitschritte oder um die Expansion von Modellen zu ermöglichen. In der Tessif-Dokumentation sind weitere Informationen dazu gegeben. Sind keine weiteren Werte vorgegeben, werden beim Erstellen des Tessif-Modells die hinterlegten Standardwerte genutzt.

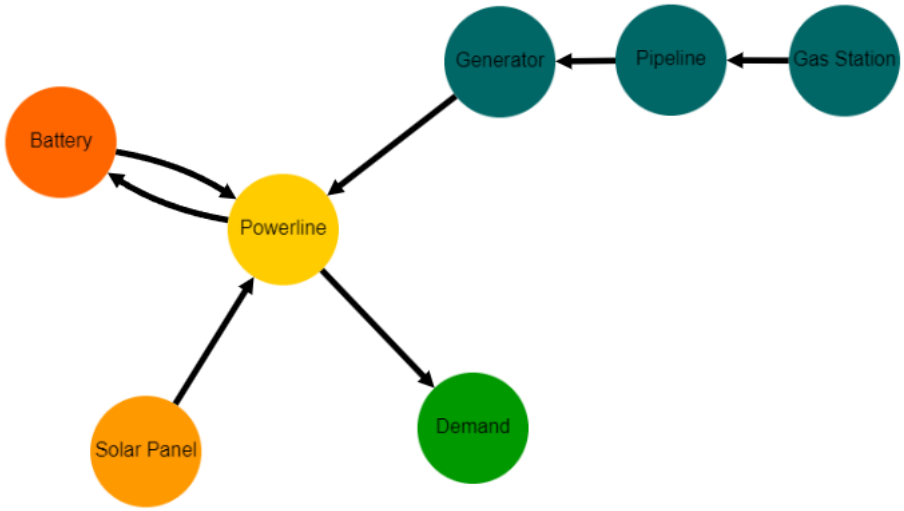


Abbildung 5.1.: Node Graph des voll parametrierten Beispielmodells

5.1.1. Automatischer Vergleich

Für den automatischen Vergleich von Urbs mit Oemof, PyPSA, FINE und Calliope werden die nachfolgenden aufgelisteten Energiesystemmodelle aus Tessifs Bibliothek verwendet. Bei den meisten dieser Modelle handelt es sich um sehr simple Energiesysteme, die sich auf einen speziellen Aspekt wie beispielweise die Emissionen oder einzelne Komponenten fokussieren. Dies hat den Vorteil, dass die Ursache unterschiedlicher Ergebnisse einfacher zu identifizieren beziehungsweise diese besser auf die verschiedenen Modellierungsprogramme zurückzuführen sind. Die verwendeten Modelle sind:

- **Minimal Working Example** (create_mwe()):
  Ein minimal parametriertes Beispielenergiesystem bestehend aus Powerline, Verbrauch, Batterie und Generator mit Gasquelle und Pipeline.

- **Fully Parametrized Working Example** (create_fpwe()):
  Ein vollständig parametriertes Beispielenergiesystem, das im Grunde wie das *Minimal Working Example* aufgebaut ist. Allerdings ist das Energiesystem zusätzlich um ein Solarpanel erweitert. Darüber hinaus sind alle Parameter wie Kosten und Emissionen für jede Komponente definiert.

- **Emission Objective** (emission_objective()):
  Ein Energiesystem mit Fokus auf die CO_2-Emissionen und mit Berücksichtigung eines Emissionslimits. Der Strom kann durch einen Generator, ein Gaskraftwerk oder durch Windkraft erzeugt werden. Das Windkraftwerk ist dabei emissionsfrei und die beiden Transformer-Komponenten sowie die dazugehörigen Quellen sind jeweils mit Emissionen behaftet.

- **Connected Energy System** (create_connected_es()):
  Zwei Energiesysteme bestehend aus Source, Sink und Bus-Komponente, die durch einen Connector miteinander verbunden sind. Der zeitliche Verlauf des Verbrauchs und der Erzeugung provoziert einen Energietransfer von Energiesystem zu Energiesystem über die Connector-Komponente.

- **Combined Heat and Power** (create_chp()):
  Ein Energiesystem mit einem CHP-Gaskraftwerk, das einen Wärme- und Strombedarf deckt. Neben der CHP-Komponente gibt es jeweils eine Notfallquelle für Wärme und Strom. Diese sind allerdings mit hohen Kosten verbunden, sodass das Gaskraftwerk wenn möglich verwendet wird.

- **Storage Example** (create_storage_example()):
  Ein Energiesystem bestehend aus Powerline, Generator, Verbrauch und Speicher. Der Generator ist dabei nicht in der Lage den Verbrauch in jedem Zeitschritt direkt abzudecken, sodass eine Speichernutzung forciert wird.

- **Expansion Plan Example** (create_expansion_plan_example()):
  Ein Energiesystem mit zwei emissionsfreien und erweiterbaren Quellen sowie einer emittierenden Quelle. Die Einschränkung der Emissionen bedingt den Ausbau einer oder beider emissionsfreier Quellen.

- **Energy System Grid „Kupferplatte“** (create_grid_kp_es()):
  Ein komplexeres Energiesystem mit verschiedenen Spannungsebenen und diversen Stromerzeugern und Verbrauchern. Zusätzlich findet eine Kopplung des Wärmesektors mit dem Stromsektor innerhalb des Energiesystems statt. Das Energiesystem orientiert sich grob an einer möglichen, zukünftigen Energieversorgung der Bundesrepublik Deutschland. Der Optimierungszeitraum des Energiesystems beträgt 24 Stunden.
  Weitere Informationen bezüglich der Entwicklung und des Aufbaus des Energiesystemmodells können Hanke [34] entnommen werden.

- **Component Energy System** (create_component_es()):
  Ein komplexes Energiesystem, das aus diversen Komponenten zur Strom- und Wärmeversorgung besteht. Der Fokus diese Energiesystems liegt dabei auf dem

optimalen Einsatz der verschiedenen Komponenten, insbesondere auch über einen längeren Optimierungszeitraum. Darüber hinaus kann das Energiesystem zu einem Expansionsszenario gewandelt werden, das sich mit dem idealen Einsatz und Ausbau der Komponenten beschäftigt. Im Kontext dieses Vergleichs wird kein Ausbauszenario beachtet und der Optimierungszeitraum beschränkt sich auf die standardmäßig angegebenen drei Zeitschritte.
Weitere Informationen zu diesem Energiesystemmodell sind in der Veröffentlichung „Entwicklung eines Komponenten basierten Szenarios zum Vergleich von Free and Open Source Energiesystemmodellierungssoftware in Python“ [40] von Reimer zu finden.

Alle zuvor aufgezählten Energiesystemmodelle sind im Anhang A.4 durch Tessif visualisiert abgebildet.

5.1.2. Detailierter Vergleich

Der detaillierte Vergleich basiert auf dem zuvor bereits erwähnten komponentenbasierten Energiesystem (*component_es*), das im Rahmen einer Projektarbeit [40] von Reimer entwickelt wurde.
Das in Abbildung 5.2 dargestellte Energiesystemmodell orientiert sich an keinem existierenden Energieversorgungssystem, sondern verfolgt einen allgemeinen Ansatz, um verschiedene Anwendungsfälle abzubilden. Aus diesem Grund berücksichtigt das Energiesystem sowohl fossile als auch erneuerbare Energieträger und zusätzlich zum Stromsektor einen Wärmesektor. Die beiden Sektoren sind dabei über Kraft-Wärme-Kopplung sowie direkt über eine Power-to-Heat-Komponente miteinander verbunden.
Wie aus Abbildung 5.2 ersichtlich wird, besteht das Energiesystem aus drei fossilen Energieträgern (Braunkohle, Steinkohle und Gas), vier erneuerbaren Energieträgern (on- und offshore Windkraft, Solarenergie und Biogas), einem Wärme- und Stromspeicher sowie einer Power-to-Heat-Komponente. Die Braunkohlequelle speist ein Braunkohlekraftwerk, mit dem Strom erzeugt wird. Die Steinkohle betreibt ein gewöhnliches Steinkohlekraftwerk sowie ein Blockheizkraftwerk zur Strom- und Wärmegewinnung. Die Gasquelle wird von einem GuD-Kraftwerk zur Stromproduktion und von einem Heizwerk verwendet. Die Biogasanlage verfügt über eine Kraft-Wärme-Kopplung und versorgt sowohl den Strom- als auch den Wärmesektor mit Energie. Darüber hinaus sind alle Bus-Komponenten des Energiesystems als ideal und somit verlustfrei angenommen.
Die relevanten technischen und ökonomischen Parameter der einzelnen Komponenten sind in der Tabelle 5.1 aufgelistet. Für das Energiesystemmodell wurden zwei verschiedene Szenarien entwickelt. So gibt es zum einen das Dispatch-Szenario, dass sich mit dem optimalen Einsatz der einzelnen Komponenten beschäftigt und zum anderen das Expansions-Szenario, welches neben dem optimalen Einsatz der bestehenden Komponenten einen idealen Ausbau des Energiesystems bestimmt. Beim Expansionsproblem wird zusätzlich eine Emissionsbegrenzung definiert, um einen Ausbau emissionsarmer Komponenten zu provozieren.

Szenario	Ausbau	Restriktionen
Dispatch	nicht möglich	keine
Expansion	möglich	$250.000 \frac{t\ CO_2}{Jahr}$

Tabelle 5.2.: Spezifikationen des *component_es* Modells

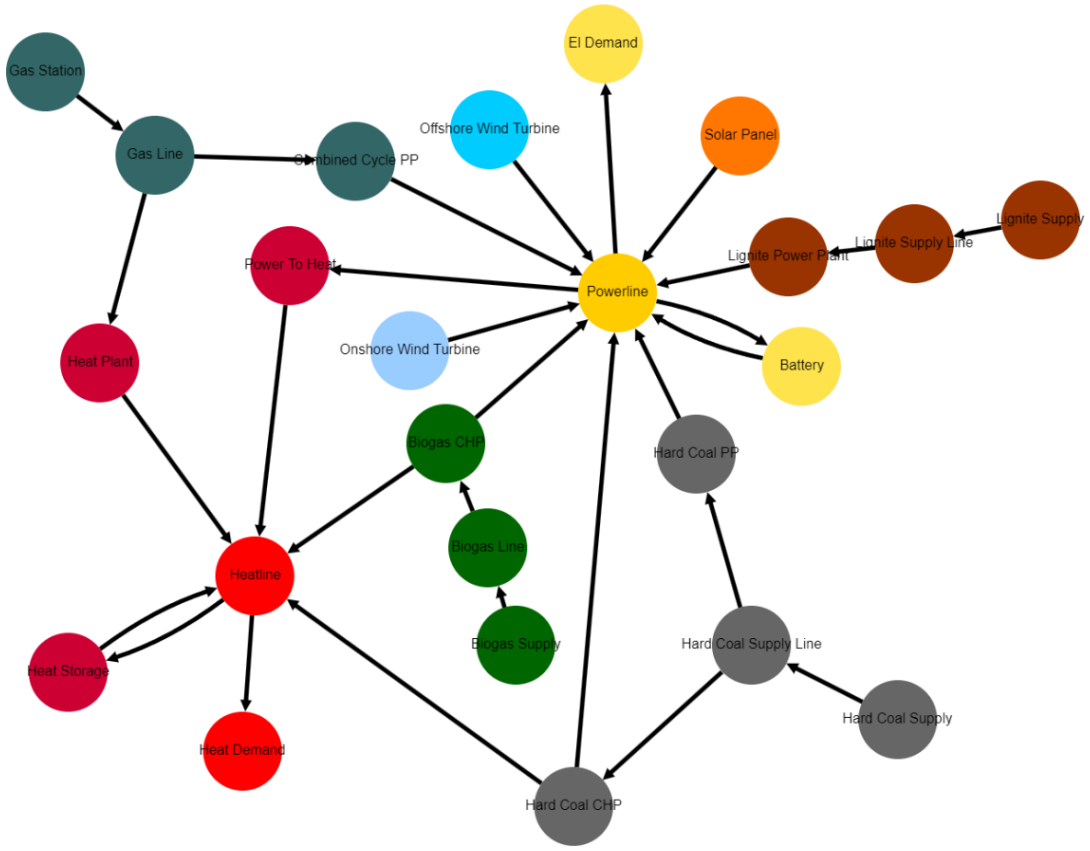


Abbildung 5.2.: Component Energy System Modell

Komponente	Inst. Leistung [MW]/[MWh]*	Spez. Emissionen $[\frac{t}{MWh}]$	Wirkungsgrad -	Betriebskosten $[\frac{€}{MWh}]$	Investitionskosten $[\frac{€}{kW}]$
Solar Panel	1100	0,05	-	80	1000
Onshore WT	1100	0,02	-	60	1750
Offshore WT	150	0,02	-	105	3900
Biogas CHP_{el}	200	0,25	0,40	150	3500
Biogas CHP_{th}	250	0,01875	0,50	11,25	262,5
Lignite PP	500	1	0,40	65	1900
Hard Coal CHP_{el}	300	0,80	0,40	80	1750
Hard Coal CHP_{th}	300	0,06	0,40	6	131,25
Hard Coal PP	500	0,80	0,43	80	1650
Combined Cycle PP	600	0,35	0,60	90	950
Heat Plant	450	0,23	0,90	35	390
Power-to-Heat	100	0,0007	0,99	20	100
Battery	100*	0,06	0,90	400	1630
Heat Storage	50*	0	0,85	20	4,50

Tabelle 5.1.: Technische und ökonomische Parameter der Komponenten des *component_es*

Im Rahmen des detaillierten Vergleichs werden beide Szenarien mit Urbs, Oemof und PyPSA optimiert und die Ergebnisse verglichen. Darüber hinaus wird das Dispatch-Szenario für einen Vergleich von Urbs mit und ohne Demand Side Management verwendet. In allen drei Fällen wird der Optimierungszeitraum auf einen Monat, also 720 Zeitschritte, festgelegt. Auf diese Weise sind ausreichend Zeitschritte für eine detaillierte Betrachtung vorhanden und zeitgleich wird die Auswertung nicht zu komplex.

Aufgrund der geringeren Anzahl von Zeitschritten müssen bei der Optimierung ebenfalls die CO_2-Restriktionen verringert werden. Deshalb werden innerhalb des Expansionsvergleichs die Emissionen auf 50.000 t CO_2 beschränkt, um einen Ausbau erneuerbarer Energien zu forcieren.
Für die Untersuchung der DSM-Funktion von Urbs müssen bei der Transformation des Energiesystems noch die in Abschnitt 4.1 beschriebenen, zusätzlichen Informationen angegeben werden. Nach Ladwig [41] sind die Parameter der DSM-Anwendungen und DSM-Prozesse komponentenabhängig und somit variieren zum Beispiel die Verschiebezeiten der betrachteten Komponenten zwischen einer und 24 Stunden sowie die Eingriffshäufigkeit zwischen 12 mal am Tag und 24 mal im Jahr. Daraus ergibt sich für die DSM-Untersuchung im Rahmen dieser Arbeit eine maximale Verschiebedauer von 12 Stunden und eine Erholungszeit von 2 Stunden. Darüber hinaus wird der Wirkungsgrad der DSM-Verschiebungen mit 0,99 angenommen. Das DSM-Potential für die deutsche Energieversorgung des Jahres 2013 wurde von Ladwig als 25 GW bestimmt. Zusammen mit der Lastspitze der deutschen Energieversorgung von 75,6 GW im Jahr 2013 kann über das Verhältnis von DSM-Potenzial und Lastspitze gemäß Formel 5.1 das Potenzial für das Energiesystemmodell berechnet werden [41].

$$\text{Potenzial}_{\text{Modell}} = \frac{\text{Potenzial}_{2013}}{\text{Lastspitze}_{2013}} \cdot \text{Lastspitze}_{\text{Modell}} = \frac{25\ GW}{75,6\ GW} \cdot 1520\ MW = 502,65\ MW \tag{5.1}$$

Nach Gleichung 5.1 ergibt sich die maximal senk- oder erhöhbare Leistung zu ungefähr 500 MW. Diese DSM-Parameter sollen im Rahmen dieser Arbeit keine reale Untersuchung des Potenzials von DSM ermöglichen, sondern nur eine realistische Möglichkeit sein, die DSM-Funktion von Urbs zu untersuchen.
Weitere Informationen bezüglich des Energiesystemmodells oder der Szenarien sind in der Ausarbeitung von Reimer zu finden.

5.2. Auswertung durch Tessif

Tessif bietet grundsätzlich zwei verschiedene Möglichkeiten der Auswertung und des Vergleichs. Einerseits besteht die Möglichkeit, Modelle manuell mit den einzelnen Modellierungsprogrammen in Tessif zu optimieren und andererseits kann ein automatischer Vergleich mehrerer Modellierungsprogramme durch den Autocomparatier von Tessif durchgeführt werden.

5.2.1. Auto Comparison verschiedener Modellierungsprogramme

Zur Optimierung und Auswertung kann Tessifs automatischer Vergleich verschiedener Modelle genutzt werden (Auto Comparison). Dafür muss zunächst das zu optimierende Energiesystem aus Tessifs *py_hard*-Datei geladen und anschließend in einem der unterstützten Dateiformate (in diesem Fall im hdf5-Format) gespeichert werden. Der Tessif-Comparatier kann die hdf5-Datei einlesen und mithilfe der angegebenen Modelle (in diesem Fall Urbs, Oemof, PyPSA, FINE und Calliope) das Energiesystem modellieren und optimieren. Zum Vergleich der Ergebnisse können die in die Powerline ein- und ausgespeisten Leistungen sowie die Integrated Global Results der Optimierungen ausgegeben werden. Hierbei gibt der Comparatier in den Tabellen die entsprechenden Werte aller verwendeten Modellierungsprogramme an. Nachfolgend ist der Code angegeben, um das *Fully Parametrized Working Example* mithilfe des Tessif-Comparatiers zu optimieren und auszuwerten.

```
import tessif.frused.configurations as configurations
import os
from tessif.frused.paths import write_dir
import tessif.analyze
import tessif.parse
import tessif.examples.data.tsf.py_hard as tsf_py
import pandas as pd
# Einstellungen zum Anzeigen der gesamten Ergebnisse
configurations.spellings_logging_level = 'debug'
desired_width = 620
pd.set_option('display.width', desired_width)
pd.set_option('display.max_columns', 20)

# Aufrufen des Energiesystems
esys = tsf_py.create_fpwe()

# Formatieren des Energiesystems
output_msg = esys.to_hdf5(
    directory=os.path.join(write_dir, 'tsf'),
    filename='grid_comparison.hdf5',
)

# Modellierung und Optimierung
comparatier = tessif.analyze.Comparatier(
    path=os.path.join(write_dir, 'tsf', 'grid_comparison.hdf5'),
    parser=tessif.parse.hdf5,
    models=('urbs', 'oemof', 'pypsa', 'FINE', 'calliope'),
)

# Kontrollausgabe der benutzten Modelle
for model in sorted(comparatier.models):
    print(model)

# Ausgeben der Global Results
print(comparatier.integrated_global_results)

# Ausgeben der Node Loads
print(comparatier.comparative_results.loads['Powerline'])
```

Quellcode 5.3: Optimieren und Auswerten mittels Tessif-Comparatier

5.2.2. Manuelle Auswertung und Vergleich verschiedener Modellierungsprogramme

Alternativ zum automatischen Vergleich kann auch eine manuelle Optimierung sowie Auswertung durchgeführt werden. Zunächst muss bei der manuellen Auswertung das gemäß Abschnitt 5.1 aus der Tessif Bibliothek geladene Energiesystem durch den Code 5.4 optimiert werden. Der nachfolgend gezeigte Code zum Optimieren und Auswerten der Ergebnisse bezieht sich auf die Verwendung von Urbs, ist aber für Oemof und PyPSA äquivalent durchzuführen.

```
# Laden des Tessifmodells
import tessif.examples.data.tsf.py_hard as tsf_py
esys = tsf_py.create_fpwe()

# Transformieren des Tessifmodells
from tessif.transform.es2es.rbs import transform
urbs_es = transform(esys)
```

```
8
9  # Optimieren des transformierten Modells
10 import tessif.simulate as simulate
11 optimized_urbs_es = simulate.rbs_from_es(urbs_es)
```

Quellcode 5.4: Aufrufen und Optimieren des voll parametrierten Beispielmodells (create_fpwe()) über *tessif.examples.data.tsf.py_hard*

Sofern das Energiesystem optimiert wurde, können über Tessifs Auswertungstools und den in 5.5 gezeigten Code die von den unterschiedlichen Komponenten aufgenommenen sowie abgegebenen Leistungen eingesehen werden. Des Weiteren können die Global Results des Energiesystems, also die Emissionen und Kosten, dargestellt werden. Abschließend kann durch den Code 5.5 noch ein Node Graph des Energiesystems erzeugt werden, der die zwischen den einzelnen Komponenten übertragenen Energiemengen sowie die verursachten Kosten und Emissionen abbildet.

```
1  # Ausgeben der Node Loads
2  import tessif.transform.es2mapping.rbs as urbs_results
3  resultier_urbs = urbs_results.LoadResultier(optimized_urbs_es)
4  print(resultier_urbs.node_load['Powerline'])
5
6  # Ausgeben der Global Results
7  igr_urbs = urbs_results.IntegratedGlobalResultier(optimized_urbs_es)
8  print(igr_urbs.global_results)
9
10 # Grafische Darstellung des Erzeugungsprofiles der Powerline
11 import tessif.visualize.loads as visualize_loads
12 import matplotlib.pyplot as plt
13 all_resultier_urbs = urbs_results.AllResultier(optimized_urbs_es)
14 pl_loads = all_resultier_urbs.node_load['Powerline']
15 pl_loads = pl_loads.loc[:, (pl_loads != 0).any(axis=0)]
16 visualize_loads.component(
17     pl_loads,
18     component_type='bus')
19 pl_plot = plt.gcf()
20 pl_plot.show()
```

Quellcode 5.5: Auswertung des mittels Urbs optimierten Energiesystems

Nachfolgend sind die mittels Urbs bestimmten Global Results des *Fully Parametrized Working Example* in der Tabelle 5.3 dargestellt. Die Tabelle 5.4 und die Abbildung 5.3 zeigen die ein- und ausgespeisten Leistungen der *Powerline* sowie das nach 5.5 erstellte Stromerzeugungsprofil des Energiesystems. Weitere Auswertungsmöglichkeiten, insbesondere Auswertungen grafischer Art, sind ausführlich in der Tessif-Dokumentation [8] beschrieben.

emissions (sim)	costs (sim)	opex (ppcd)	capex (ppcd)
225.0	443.0	443.0	0.0

Tabelle 5.3.: Global Results des *fpwe()*

	Battery	Generator	Solar Panel	Battery	Demand
1990-07-13 00:00:00	-0.0	-0.0	-12.0	1.0	11.0
1990-07-13 01:00:00	-8.0	-0.0	-3.0	0.0	11.0
1990-07-13 02:00:00	-0.0	-13.1	-7.0	9.1	11.0

Tabelle 5.4.: Global Results des *fpwe()*

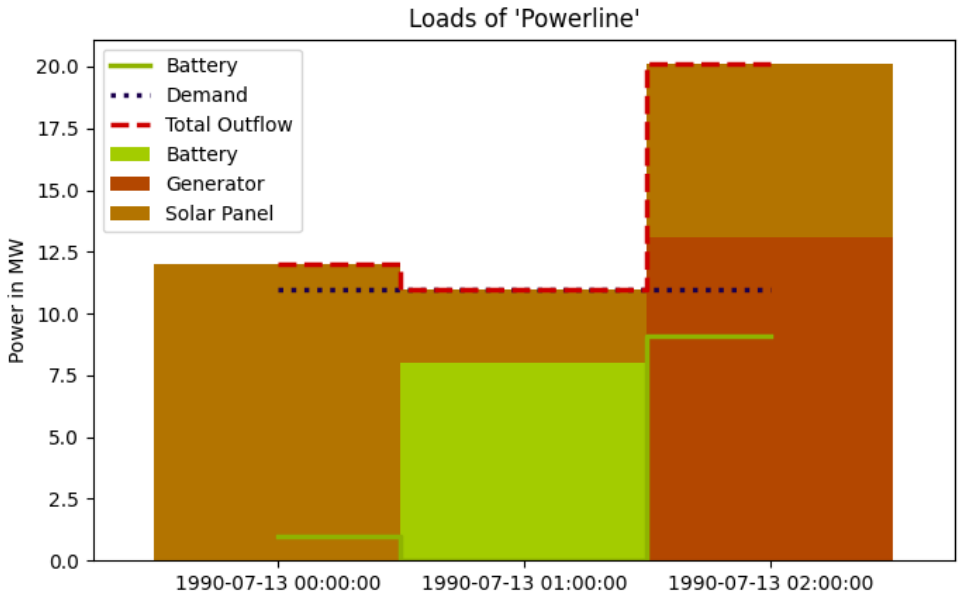


Abbildung 5.3.: Durch Tessif visualisiertes, Erzeugungsprofil der Powerline des optimierten Modells *fpwe()*, erstellt mit Urbs

5.3. Ergebnisse der Optimierung mit Tessif

Nachfolgend werden die Ergebnisse des allgemeinen und detaillierten Vergleichs dargestellt und ausgewertet. Die Optimierungsergebnisse wurden, wie zuvor in Abschnitt 5.2 erläutert, erstellt und durch Tessifs Auswertungstool dargestellt. Weitere Möglichkeiten sowie detailliertere Erklärungen der Funktionen von Tessif sind in der Tessif-Dokumentation [8] beschrieben.

5.3.1. Vergleich von Urbs mit Oemof, PyPSA, FINE und Calliope über Tessifs Autocomparatier

Dieser Abschnitt befasst sich mit den Ergebnissen sowie der Auswertung der mit Urbs, Oemof, PyPSA, FINE und Calliope modellierten und optimierten Energiesysteme. Insbesondere werden zu Vergleichszwecken die Integrated Global Results (Kosten und Emissionen) und die zwischen den einzelnen Komponenten übertragenen Energiemengen betrachtet. Die Global Results werden dafür nachfolgend tabellarisch abgebildet und setzen sich aus den Emissionen sowie Kosten zusammen. Zusätzlich wird die Zeit und der Arbeitsspeicher, die für die Datenverarbeitung und Optimierung benötigt werden, angegeben.

Die Tabellen mit den übertragenen Energiemengen der verschiedenen Energiesysteme befinden sich im Anhang im Abschnitt A.5.1. Die Tabellen zeigen dabei die über alle Zeitschritte summierten Energiemengen, die jeweils zwischen zwei Komponenten übertragen werden. Die zuerst genannte Komponente gibt dabei im gesamten Zeitbereich die jeweilige Energiemenge an die zweite Komponente ab.
Die Emissionen und Kosten können als Indikator für Unterschiede in der Optimierung genutzt werden, da identische Emissionen und Kosten meistens auf eine gleiche Nutzung der Komponenten zurückzuführen sind. Bei abweichenden Global Results kann eine Analyse der übertragenen Energiemengen hilfreich sein, um Unterschiede der verschiedenen Modellierungsprogramme zu identifizieren. Die Hauptintention dieser Auswertung ist es, zu prüfen, ob die Integration von Urbs in Tessifs Framework erfolgreich ist und ob Unterschiede durch eine fehlerhafte Integration oder durch unterschiedliche Modellierungsansätze entstehen. Aus diesem Grund liegt der Fokus der Untersuchung auf den Ergebnissen von Urbs. Die Ergebnisdifferenzen der anderen Modellierungsprogramme werden nicht im Detail betrachtet. Für weitere Untersuchungen von FINE und Calliope im Rahmen von Tessif können die Untersuchungen von Schnute [42] und Reimer [43] betrachtet werden.

Minimal Working Example

Betrachtet man die Ergebnisse der Optimierung in Tabelle 5.5, so fällt zunächst auf, dass die operativen Kosten und somit auch die Gesamtkosten bei Urbs und FINE höher als bei Oemof, PyPSA und Calliope ausfallen. Aus Tabelle A.3 im Anhang wird ersichtlich, dass die Mehrkosten auf eine erhöhte Nutzung des Generators und damit auch der Gas Station zurückzuführen sind. Gleichzeitig wird keine Energie von der Battery an die Powerline abgeben. Dies resultiert aus der Tatsache, dass Speicherkomponenten innerhalb von Urbs mit einem identischen Start- und Endspeicherstand definiert sind. In Urbs muss demnach ein Speicher am letzten Zeitschritt immer denselben Speicherstand (Final-SoC) wie zum Beginn der Optimierung (Initial-SoC) haben. Demzufolge wird die Energie nicht aus dem Speicher genommen, sondern zusätzlich durch den Generator bereitgestellt.
Investitionskosten und Emissionen treten innerhalb dieses Energiesystems nicht auf, da es sich um ein minimal parametriertes Beispielenergiesystem handelt und somit weder Emissionswerte noch ein Ausbauszenario definiert sind. Des Weiteren ist in der Tabelle A.3 auffällig, dass für PyPSA nicht alle Werte angegeben sind, weil in PyPSA reine Versorgungsstränge zusammengefasst werden und somit nicht zusätzlich abgebildet werden.
Beim Betrachten der Zeit und des Arbeitsspeichers, welche für die Modellierung sowie Optimierung des Energiesystems benötigt werden, wird ersichtlich, dass die Werte bei Urbs, PyPSA und FINE recht ähnlich ausfallen. Oemof tendiert zu kürzeren Zeiten und geringerem Speicherbedarf und Calliope zu längeren Zeiten bei gleichzeitig mehr benötigtem Speicherplatz. Allerdings ist sowohl bei der Optimierungsdauer als auch dem Speicherbedarf zu berücksichtigen, dass insbesondere bei der Untersuchung wenig komplexer Energiesysteme mit geringer Anzahl von Zeitschritten die Werte stark variieren können und daher nur bedingt aussagekräftig sind.

	cllp	FINE	omf	ppsa	rbs
emissions (sim)	0.0	0.0	0.0	0.0	0.0
costs (sim)	61.0	80.0	61.0	61.0	80.0
opex (ppcd)	61.0	80.0	61.0	61.0	80.0
capex (ppcd)	0.0	0.0	0.0	0.0	0.0
time (s)	2.7	0.8	0.4	1.1	1.1
memory (MB)	1.7	1.1	0.6	1.4	1.1

Tabelle 5.5.: Integrated Global Results des Minimal Working Examples optimiert mit Calliope, FINE, Oemof, PyPSA und Urbs

Fully Parametrized Working Example

Auch beim vollständig parametrierten Beispielenergiesystem fallen Urbs und FINE mit höheren Emissionen und Kosten auf. Aufgrund der Ähnlichkeit dieses Energiesystems mit dem minimal parametrierten Energiesystem aus 5.3.1 sind die Unterschiede ebenfalls wieder auf den Speicherstand am Ende der Optimierung und auf die daraus resultierende erhöhte Verwendung des Generators und der Gas Station zurückzuführen. Bei FINE wird der initiale Speicherstand nicht berücksichtigt, sodass sich daraus höhere Kosten als bei Calliope, Oemof und PyPSA ergeben. Die Kosten und Emissionen fallen allerdings geringer als bei Urbs aus, da bei Urbs durch die Nutzung der Speicherkomponente zusätzlich noch die Speicherverluste abgedeckt werden müssen.
Auch bei diesen Modellierungen sowie Optimierungen verhalten sich die Optimierungszeit und der benötigte Speicherbedarf ähnlich zum Energiesystem aus 5.3.1. So benötigt Oemof wieder eine geringere Zeit und weniger Speicher, während die Optimierung bei Calliope zeit- und speicherintensiver ist.

	cllp	FINE	omf	ppsa	rbs
emissions (sim)	53.0	190.0	53.0	53.0	225.0
costs (sim)	105.0	375.0	105.0	105.0	443.0
opex (ppcd)	105.0	375.0	105.0	105.0	443.0
capex (ppcd)	0.0	0.0	0.0	0.0	0.0
time (s)	3.1	1.1	0.5	1.3	1.2
memory (MB)	2.0	1.4	0.6	1.4	1.2

Tabelle 5.6.: Integrated Global Results des Fully Parametrized Working Examples optimiert mit Calliope, FINE, Oemof, PyPSA und Urbs

Emission Objective Example

Das Emission Objective Example fokussiert sich auf die Emissionen des Energiesystems. Insbesondere ist ein Limit der Emissionen (60 Einheiten) definiert. Betrachtet man die Tabellen 5.7 und A.5 wird ersichtlich, dass alle fünf untersuchten Modellierungsprogramme identische Optimierungsergebnisse erzeugen. Zusätzlich ist zu erwähnen, dass alle Programme das Emissionslimit einhalten und somit auch in der Lage sind, komplexere Energiesysteme unter Berücksichtigung von Einschränkungen der Emissionen zu optimieren. Auch bei diesem Energiesystem verhalten sich die Optimierungsdauer und der benötigte Speicherplatz äquivalent zu den zuvor untersuchten Energiesystemmodellen.

	cllp	FINE	omf	ppsa	rbs
emissions (sim)	60.0	60.0	60.0	60.0	60.0
costs (sim)	252.0	252.0	252.0	252.0	252.0
opex (ppcd)	252.0	252.0	252.0	252.0	252.0
capex (ppcd)	0.0	0.0	0.0	0.0	0.0
time (s)	2.5	1.1	0.5	1.1	1.2
memory (MB)	2.1	1.2	0.6	1.4	1.1

Tabelle 5.7.: Integrated Global Results des Emission Objective Examples optimiert mit Calliope, FINE, Oemof, PyPSA und Urbs

Connected Energy System Example

Das Connected Energy System Example wird, wie die Tabellen 5.8 und A.6 zeigen, ebenfalls von Urbs, Oemof, PyPSA, FINE und Calliope identisch optimiert. Dies wird zum einen aus den identischen Kosten und Emissionen der Global Results und zum anderen aus den zwischen den Komponenten übertragenen Energiemengen ersichtlich. Das Energiesystem basiert auf der Funktionalität der Tessif Connector-Komponente, die somit in jedem untersuchten Modellierungsprogramm gegeben ist. Auch hier setzt sich der Trend bezüglich der abweichenden Optimierungsdauer sowie des Speicherbedarfs von Oemof und Calliope fort.

	cllp	FINE	omf	ppsa	rbs
emissions (sim)	52.0	52.0	52.0	52.0	52.0
costs (sim)	52.0	52.0	52.0	50.0	52.0
opex (ppcd)	52.0	52.0	52.0	50.0	52.0
capex (ppcd)	0.0	0.0	0.0	0.0	0.0
time (s)	1.9	0.7	0.4	1.1	1.1
memory (MB)	1.6	1.1	0.5	1.4	1.1

Tabelle 5.8.: Integrated Global Results des Connected Energy System Examples optimiert mit Calliope, FINE, Oemof, PyPSA und Urbs

CHP Example

Dieses Energiesystem besteht aus einem Stromnetz und einem Wärmenetz, die durch ein Combined-Heat-and-Power Kraftwerk miteinander verbunden sind. Der Fokus liegt dabei auf der Sektorenkopplung von Strom und Wärme. Wie die Ergebnisse der Optimierung in den Tabellen 5.9 und A.7 zeigen, sind alle untersuchten Programme in der Lage, eine solche Kopplung mehrerer Energiesektoren abzubilden. Auch hier erfolgen die Optimierungen identisch. Darüber hinaus verhält sich die Optimierungsdauer und der benötigte Speicherplatz äquivalent zu den vorherigen Ergebnissen.

	cllp	FINE	omf	ppsa	rbs
emissions (sim)	160.0	160.0	160.0	160.0	160.0
costs (sim)	307.0	307.0	307.0	307.0	307.0
opex (ppcd)	307.0	307.0	307.0	307.0	307.0
capex (ppcd)	0.0	0.0	0.0	0.0	0.0
time (s)	2.8	0.8	0.5	1.3	1.4
memory (MB)	2.0	1.2	0.7	1.6	1.2

Tabelle 5.9.: Integrated Global Results des CHP Examples optimiert mit Calliope, FINE, Oemof, PyPSA und Urbs

Storage Energy System Example

Der Fokus des Storage Energy System Examples liegt auf der Tessif Storage-Komponente. Betrachtet man die zwischen den einzelnen Komponenten übertragenen Energiemengen in Tabelle A.8, so fällt auf, dass die Optimierungen identisch sind. Allerdings weichen die Global Results 5.10 von Urbs von den Werten der anderen Modellierungsprogramme ab. So sind die Kosten minimal höher und es treten keine Emissionen auf. Dass in Urbs keine Emissionen registriert werden, liegt an der Speicherkomponente, da, anders als in Tessif, in Urbs Speichern keine Emissionen zugeordnet werden können. Da der Speicher in diesem Energiesystem die einzige Komponente ist, für die Emissionen definiert sind, werden diese nicht in Urbs abgebildet. Die leicht höheren Kosten von Urbs lassen sich auf die variablen Kosten (*flow_costs*) des Speichers zurückführen. In Tessif beziehen sich die Flusskosten nur auf die aus dem Speicher abgeführte Leistung, während in Urbs die variablen Kosten der Speicherkomponente immer sowohl auf den Input als auch den Output des Speichers angewendet werden. Aus diesem Grund wurden die Kosten bei der Transformation des Tessif-Energiesystems in ein Urbs-Energiesystem halbiert, sodass die Kosten sich über den Speicher In- und Output ähnlich wie die Kostendefinition von Tessif und den anderen Modellierungstools verhalten. Die Gleichungen 5.2 und 5.3 verdeutlichen den Unterschied zwischen Urbs und den anderen Programmen am Beispiel von Oemof.

$$\text{Kosten}_{\text{Urbs}} = (\text{Input}+\text{Output})\cdot 0.5 \cdot \text{Kosten}_{\text{spez.}} = (30+20)MW \cdot 0.5 \cdot 1\frac{€}{MW} = 25€ \quad (5.2)$$

$$\text{Kosten}_{\text{Oemof}} = \text{Output} \cdot \text{Kosten}_{\text{spez.}} = 20MW \cdot 1\frac{€}{MW} = 20€ \quad (5.3)$$

Die 5 € Unterschied aus dem Beispiel entsprechen dabei der Kostendifferenz der Global Results aus Tabelle 5.10. Auch bei diesen Modellierungen mit anschließenden Optimierungen benötigt Calliope am meisten Speicher und Zeit, während Oemof am wenigsten Speicher und die geringste Zeit zum Optimieren benötigt.

	cllp	FINE	omf	ppsa	rbs
emissions (sim)	10.0	10.0	10.0	10.0	0.0
costs (sim)	134.0	134.0	134.0	134.0	139.0
opex (ppcd)	134.0	134.0	134.0	134.0	139.0
capex (ppcd)	0.0	0.0	0.0	0.0	0.0
time (s)	2.1	0.7	0.3	0.9	0.7
memory (MB)	1.6	1.1	0.5	1.4	1.0

Tabelle 5.10.: Integrated Global Results des Storage Energy System Examples optimiert mit Calliope, FINE, Oemof, PyPSA und Urbs

Expansion Plan Example

Beim Expansion Plan Example forciert ein Emissionslimit gezielt den Ausbau von erneuerbaren Energien, um die Emissionsgrenzen einhalten zu können. Aus den Global Results 5.11 und Tabelle A.9 wird ersichtlich, dass sowohl Oemof, PyPSA, FINE und Calliope als auch Urbs die *Uncapped Renewable*-Komponente ausbaut. Die Gesamtkosten setzten sich aus den operativen und den Investitionskosten zusammen und sind bei allen Tools identisch. Somit sind alle der untersuchten Programme in der Lage, Komponenten auszubauen und damit auch Expansionsszenarien zu optimieren. Dabei verhalten sich die Optimierungszeiten und der benötigte Speicherbedarf wie bei den zuvor untersuchten Energiesystemen, sodass nur Calliope und Oemof minimal von den Werten der anderen Modellierungsprogramme abweichen.

	cllp	FINE	omf	ppsa	rbs
emissions (sim)	20.0	20.0	20.0	20.0	20.0
costs (sim)	41.0	41.0	41.0	41.0	41.0
opex (ppcd)	40.0	40.0	40.0	40.0	40.0
capex (ppcd)	1.0	1.0	1.0	1.0	1.0
time (s)	1.8	0.6	0.3	0.9	0.7
memory (MB)	1.6	0.9	0.5	1.4	0.9

Tabelle 5.11.: Integrated Global Results des Expansion Plan Examples optimiert mit Calliope, FINE, Oemof, PyPSA und Urbs

Component Energy System

Das auf Komponenten basierende Energiesystem ist im Vergleich zu den zuvor untersuchten Energiesystemen ein komplexeres Energiesystem, das aus einer Vielzahl von Komponenten besteht. Die Optimierungsergebnisse für einen Zeitraum von drei Zeitschritten sind in den Tabellen 5.12 und A.10 angegeben. Die Global Results lassen darauf schließen, dass die Optimierung bei allen Programmen gleich erfolgt, da sowohl die Kosten als auch die Emissionen identisch sind. Einzig auffällig ist, dass die operativen Kosten bei allen Modellierungsprogrammen außer Urbs größer als die Gesamtkosten sind. Dies ist auf die Tatsache zurückzuführen, dass bei Oemof, PyPSA, FINE und Calliope die Gesamtkosten den Optimierungsergebnissen entnommen werden und die operativen Kosten im Postprocessing aus den übertragenen Energiemengen und den spezifischen Kostenwerten berechnet werden. Durch unterschiedliche Rundungsmethoden können so minimale Abweichungen entstehen. Bei Urbs hingegen werden die einzelnen Kosten in

den Optimierungsergebnissen ausgegeben und mithilfe der einzelnen Kosten werden gemäß Abschnitt 4.2 die Kosten für den Global Resultier bestimmt. Dadurch haben unterschiedliche Rundungsmethoden bei Urbs keinen Einfluss auf die im Integrated Global Resultier angegebenen Kosten. Auch die in Tabelle A.10 gezeigten, zwischen den Komponenten übertragenen Energiemengen weisen auf eine identische Optimierung durch die verschiedenen verwendeten Programme hin.
Die für die Optimierung benötigte Zeit ist bei Calliope mit 9.4 Sekunden mehr als dreimal so hoch wie bei Oemof, PyPSA und FINE und fast doppelt so lang wie die Optimierungsdauer von Urbs (4.9 s). Die erhöhte Optimierungsdauer von Urbs ist auf die komplexere Struktur des Energiesystems zurückzuführen, da diese in Urbs nur durch mehrere Gebiete (Sites) abgebildet werden kann. Dadurch werden die zu lösenden Gleichungssysteme entsprechend komplexer und es resultiert ein höherer Zeitaufwand für die Optimierung. Der bei der Optimierung benötigte Speicherbedarf liegt bei Urbs, Oemof, PyPSA und FINE in einem ähnlichen Bereich, nur Calliope benötigt mit 9.2 MB mehr als den dreifachen Speicherplatz.

	cllp	FINE	omf	ppsa	rbs
emissions (sim)	1115.0	1115.0	1115.0	1115.0	1115.0
costs (sim)	161551.0	161551.0	161551.0	161551.0	161551.0
opex (ppcd)	161552.0	161552.0	161552.0	161552.0	161551.0
capex (ppcd)	0.0	0.0	0.0	0.0	0.0
time (s)	9.4	2.3	1.7	2.4	4.9
memory (MB)	9.2	2.7	1.5	2.1	2.5

Tabelle 5.12.: Integrated Global Results des Component Energy Systems optimiert mit Calliope, FINE, Oemof, PyPSA und Urbs

Energy System Grid „Kupferplatte“

Das „Kupferplatten“-Modell stellt die Netzstruktur eines modernen Versorgungsnetzes mit mehreren Spannungsebenen dar und wird über einen Zeitraum von 24 Stunden optimiert. Die Global Results aus Tabelle 5.13 und die übertragenen Energiemengen aus Tabelle A.11 zeigen die Ergebnisse der Optimierungen. Dabei wird ersichtlich, dass die Ergebnisse von Urbs, Oemof, PyPSA und FINE identisch sind und sich nur Calliope durch höhere Kosten und geringere Emissionen von den anderen Modellierungsprogrammen unterscheidet. Die abweichenden Kosten und Emissionen resultieren dabei, wie aus Tabelle A.11 ersichtlich wird, aus unterschiedlicher Verwendung einzelner Komponenten. Da sich diese Untersuchung auf die Funktion von Urbs fokussiert, werden die Ursachen für die abweichende Nutzung der Komponenten nicht weiter betrachtet. Die minimale Abweichung der Gesamtkosten zwischen Urbs und den anderen Programmen ist vermutlich auf Rundungsdifferenzen innerhalb des Optimierungsprozesses zurückzuführen. Die Abweichungen der operativen Kosten von den Gesamtkosten bei Oemof, PyPSA und FINE entstehen durch das Bestimmen der operativen Kosten im Postprocessing. Diese Ursache ist bereits zuvor beim *Component Energy System* in Abschnitt 5.3.1 näher erläutert worden.
Sowohl der Speicherbedarf als auch die für die Modellierung und Optimierung benötigte Zeit fällt bei Urbs mit 6.7 MB und 4.7 s etwas höher als bei Oemof, PyPSA und FINE (∅: 3.23 MB, 2.5 s) aus. Calliope benötigt mit 15.5 MB und 9.4 s allerdings noch deutlich mehr Zeit und Speicherplatz als Urbs.

Abschließend lässt sich sagen, dass Urbs auch in der Lage ist, komplex strukturierte Energiesysteme, die beispielweise aus mehreren Spannungsebenen bestehen, zu modellieren sowie zu optimieren.

	cllp	FINE	omf	ppsa	rbs
emissions (sim)	441567.0	443159.0	443159.0	443159.0	443159.0
costs (sim)	202437193.0	202259102.0	202259102.0	202259102.0	202259101.0
opex (ppcd)	202437194.0	202259103.0	202259103.0	202259103.0	202259101.0
capex (ppcd)	0.0	0.0	0.0	0.0	0.0
time (s)	9.4	2.6	2.2	2.7	4.7
memory (MB)	15.5	4.1	2.4	3.2	6.7

Tabelle 5.13.: Integrated Global Results des Energy System Grids optimiert mit Calliope, FINE, Oemof, PyPSA und Urbs

5.3.2. Detaillierter Vergleich von Urbs mit Oemof und PyPSA

Dieser Abschnitt befasst sich mit dem detaillierten Vergleich des in Abschnitt 5.3.2 vorgestellten Energiesystemmodells. Aus Gründen des Umfanges werden nur die Ergebnisse von Urbs, Oemof und PyPSA miteinander verglichen. Der detaillierte Vergleich setzt sich dabei aus drei verschiedenen Untersuchungen zusammen. Zum einen wird ein Dispatch Problem betrachtet, das den Einsatz der vorhandenen Energieerzeugungskomponenten optimiert und zum anderen wird ein Expansionsszenario untersucht, welches sich mit dem optimalen Ausbau und Einsatz der Komponenten des Energiesystems beschäftigt. Darüber hinaus wird die Demand Side Management (DSM) Funktion von Urbs getestet und mit den Ergebnissen von Urbs ohne DSM verglichen.
Als Vergleichsgrundlage werden wie zuvor die Global Results (Kosten und Emissionen) aber auch die in das Strom- und Wärmenetz eingespeisten aufsummierten Energiemengen verwendet. Des Weiteren werden auch die jeweiligen Erzeugungsprofile für den Strom- sowie Wärmesektor betrachtet.

Dispatch Problem

Die Ergebnisse des Dispatch Problems des *Component Energy Systems* sind in der Tabelle 5.15 und im Anhang in den Tabellen A.12 und A.13 dargestellt. Betrachtet man die Global Results in Tabelle 5.15, so fällt zunächst auf, dass die Emissionen der optimierten Energiesysteme voneinander abweichen. Des Weiteren unterscheiden sich auch die Kosten von Urbs mit denen von Oemof beziehungsweise PyPSA. Die Mehrkosten in Urbs von ungefähr 1000 € lassen sich auf die Speicherkomponente (Heat Storage) zurückführen. Wie zuvor in Abschnitt 5.3.1 und durch die Gleichungen 5.2 und 5.3 erläutert, bezieht Urbs die variablen Kosten (*flow_costs*) von Speichern auf die aufgenommene sowie abgegebene Energiemenge. Bei Oemof und PyPSA hingegen beziehen sich die Kosten nur auf die vom Speicher abgegebene Energiemenge. Trotz der Berücksichtigung des Kostenfaktors von 0.5 bei der Transformation in ein Urbs Energiesystem kommt es zu Kostendifferenzen. Dies liegt unter anderem an den Speicherverlusten, da diese bei Urbs durch den Kostenbezug auf die eingespeiste Energiemenge zusätzlich Kosten verursachen. Durch die teurere Speicherkomponente verschiebt sich die Energieerzeugung in Urbs, da die Einspeisekosten des Speichers durch kurze Speicherdauern und somit geringere Speicherverluste gering gehalten werden. Demzufolge stellt das Hard Coal CHP-Kraftwerk ungefähr 35 MW Strom und Wärme weniger und dafür das Gaskraftwerk (Combined

Cycle PP) entsprechend mehr Strom bereit.
Die geringeren Emissionen des mit Urbs optimierten Energiesystems sind weitestgehend auf die vermehrte Verwendung des Solar Panels in Urbs zurückzuführen. Gleichzeitig stellt das Kohlekraftwerk (Hard Coal PP) entsprechend weniger Strom zur Verfügung. Da beide Komponenten die gleichen Betriebskosten (80 $\frac{€}{\text{MWh}}$, siehe Tabelle 5.1) aber unterschiedliche Emissionen aufweisen, werden dadurch nur die Emissionen nicht aber die Kosten des optimierten Energiesystems beeinflusst.
Dabei ist jedoch wichtig zu erwähnen, dass die Abweichungen sowohl der Kosten als auch der Emissionen von Oemof und PyPSA zu Urbs sehr gering sind (unter 1 %) und somit die Energieerzeugung der optimierten Energiesysteme sehr ähnlich ausfällt. Insbesondere wird dies aus der Abbildung 5.4 ersichtlich. Das Diagramm zeigt die Zusammensetzung der Energieerzeugung des Stromsektors für PyPSA, Oemof und Urbs. Daraus wird ersichtlich, dass die Stromversorgung grundsätzlich durch Braunkohle- und Steinkohle-Kraftwerke sowie offshore Windturbinen gewährleistet wird. Des Weiteren zeigt die Grafik auch die erhöhte Einspeisung des Solarstroms bei Urbs und Oemof. Vergleicht man das Erzeugungsprofil von Urbs für Strom (Abbildung 5.5) und Wärme (Abbildung 5.6) mit den Erzeugungsprofilen von Oemof und PyPSA im Anhang A.5.2 fällt auf, dass die Strom- und Wärmeerzeugung weitestgehend identisch erfolgt. Einzig die erhöhte Einspeisung des Solar Panels bei Urbs wird ersichtlich.
Abschließend lässt sich somit also sagen, dass Urbs in Tessif in der Lage ist, komplexe Energiesysteme über längere Zeiträume zu modellieren und optimieren und dass nur geringe Differenzen zu anderen Modellierungsprogrammen bestehen. Dabei tritt die wesentliche Differenz bei den Emissionen der optimierten Energiesysteme auf, wobei die Verwendung emissionsärmerer Energiequellen von Urbs zu bevorzugen ist. Allerdings benötigt Urbs für die Modellierung und anschließende Optimierung mit ungefähr 33 Sekunden auch fast dreimal so viel Zeit wie Oemof.

	Urbs	Oemof	PyPSA
Preprocess Time in s	0.8125	0.004	1.4159
Process Time in s	19.4064	7.197	10.0283
Postprocess Time in s	12.5468	4.4061	8.2134
Time in s	32.7656	11.6071	19.6576

Tabelle 5.14.: Berechnungszeiten des Dispatch Problems optimiert durch Urbs, Oemof und PyPSA

	Urbs	Oemof	PyPSA
Emissionen in t CO_2	569784.0	574301.0	574933.0
Kosten in €	62193285.0	62192306.0	62192306.0
OpEx in €	62193285.0	62192307.0	62192307.0
CapEx in €	0.0	0.0	0.0

Tabelle 5.15.: Global Results des Dispatch Problems optimiert durch Urbs, Oemof und PyPSA

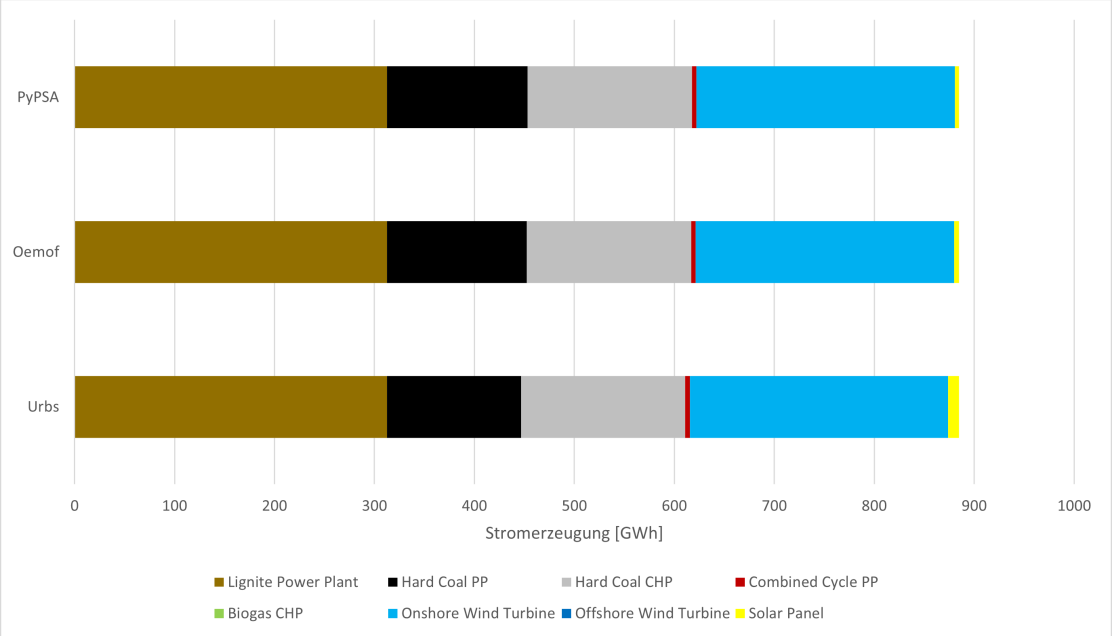


Abbildung 5.4.: Zusammensetzung des Stromsektors für das Dispatch Problem des *component_es* der Tessif-Bibliothek optimiert mit Urbs, Oemof und PyPSA

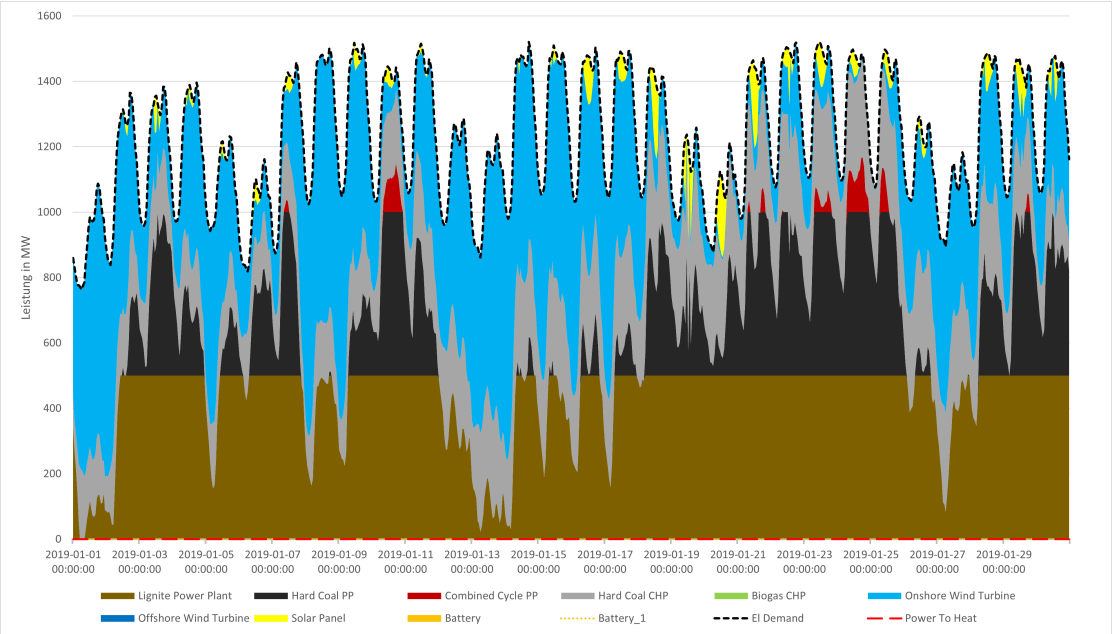


Abbildung 5.5.: Erzeugungsprofil des Stromsektors für das Dispatch Problem des *component_es* der Tessif-Bibliothek optimiert mit Urbs

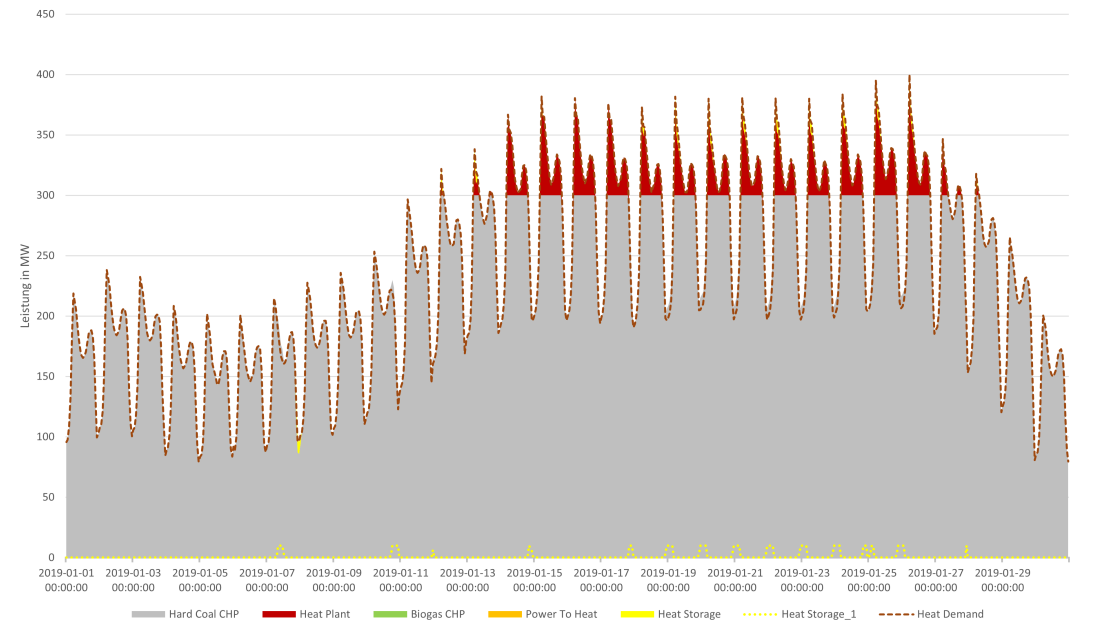


Abbildung 5.6.: Erzeugungsprofil des Wärmesektors für das Dispatch Problem des *component_es* der Tessif-Bibliothek optimiert mit Urbs

Expansion Problem

Das Expansions-Szenario zielt auf den optimalen Einsatz sowie Ausbau der Komponenten eines Energiesystems ab, um mit möglichst geringen Kosten gegebene Restriktionen der Emissionen einzuhalten. Die Abbildung 5.7 zeigt die Zusammensetzung des Stromsektors für die mit PyPSA, Oemof und Urbs optimierten Energiesysteme. Die Zusammensetzung ist bei allen drei Programmen sehr ähnlich und der Großteil des Stroms wird durch onshore Windkraftanlagen bereitgestellt. Im Vergleich zum Dispatch Problem (Abbildung 5.4) verschiebt sich die Stromerzeugung sehr stark von den kostengünstigen fossilen Energien zu den emissionsärmeren erneuerbaren Energien, um die Einschränkungen der Emissionen zu erfüllen. Auch bei den Erzeugungsprofilen von Urbs in den Abbildungen 5.8 und 5.9 sind nur minimale Differenzen zu den Profilen von Oemof und PyPSA erkennbar. Vergleicht man die Erzeugungsprofile mit den Profilen des Dispatch Problems, so fällt auch hier auf, dass sich die Erzeugung so verschoben hat, dass primär auf emissionsarme Komponenten sowohl im Strom- als auch im Wärmesektor zurückgegriffen wird.
Betrachtet man die Global Results in Tabelle 5.17, so fällt zunächst auf, dass Urbs und Oemof die Emissionsbeschränkung von 50.000 t CO_2 einhalten. PyPSA liegt allerdings mit 50.128 t CO_2 ein wenig über der Emissionsgrenze. Dies liegt daran, dass bei PyPSA die Emissionen nicht direkt abgegriffen werden können, sondern Tessif diese selbstständig anhand der von PyPSA ausgegebenen Leistungsflüsse und der spezifischen Emissionswerte des Energiesystems berechnet. Allerdings handhabt PyPSA intern die Emissionswerte anders, sodass diese nicht zwangsweise mit den von Tessif berechneten Werten übereinstimmen. Durch den unterschiedlichen Umgang mit Emissionen von PyPSA kommt es auch zu unterschiedlichen Kosten, da weniger emissionsarme erneuerbare Energien ausgebaut werden müssen und somit die Investitionskosten (CapEx) geringer ausfallen. Dadurch werden zudem die Komponenten unterschiedlich eingesetzt, sodass auch die operativen Kosten beeinflusst werden.

Aus Tabelle 5.17 wird zudem ersichtlich, dass auch Urbs und Oemof sich trotz identischer Emissionen in den Kosten unterscheiden. Diese Kostendifferenzen sind auf mehrere verschiedene Ursachen zurückzuführen. Die höheren operativen Kosten bei Urbs werden wie zuvor beim Dispatch Problem durch die Speicherkomponente und den Bezug der Kosten auf die aufgenommene und abgegebene Energie des Speichers verursacht. Bei Oemof beziehen sich die Kosten nur auf die abgegebene Energie, sodass die Kosten geringer ausfallen. In Abschnitt 5.3.1 wurde dieser Unterschied bereits ausführlicher erläutert. Die Investitionskosten fallen allerdings bei Urbs geringer aus, weil die Speicherkomponenten bei Urbs keine Emissionen berücksichtigen und somit weniger erneuerbare Energien ausgebaut werden müssen, um die Restriktionen zu erfüllen. Zusätzlich können weitere Unterschiede bei den Investitionskosten auftreten, da Urbs beim Ausbau der Speicherkomponenten kein festes Verhältnis zwischen der Kapazität und der Leistung eines Speichers berücksichtigt.
Zusammenfassend lässt sich sagen, dass durch die unterschiedliche Implementierung der Speicher in Urbs und Oemof sowie PyPSA eine abweichende Nutzung der Speicher und damit auch der anderen Komponenten resultiert. Diese unterschiedliche Verwendung der Komponenten beeinflusst wiederum sowohl die Betriebskosten (OpEx) als auch die Investitionskosten (CapEx) des optimierten Energiesystems. Urbs benötigt auch hier für die Optimierung mit ungefähr 35 Sekunden ungefähr dreimal so lange wie Oemof und PyPSA. Dabei ist die abweichende Gesamtzeit von Urbs insbesondere auf die längere Optimierungszeit (Process Time) und die längere Postprocessing-Zeit zurückzuführen (siehe Tabelle 5.16).

	Urbs	Oemof	PyPSA
Preprocess Time in s	0.7343	0.0	2.0625
Process Time in s	22.0313	8.7188	7.8126
Postprocess Time in s	12.5472	3.4375	2.7968
Time in s	35.3127	12.1563	12.6719

Tabelle 5.16.: Berechnungszeiten des Expansion Problems optimiert durch Urbs, Oemof und PyPSA

	Urbs	Oemof	PyPSA
Emissionen in t CO_2	50000.0	50000.0	50128.0
Kosten in €	13921907913.0	13944904323.0	13895530450.0
OpEx in €	73738913.0	73500669.0	73686377.0
CapEx in €	13848169000.0	13871403653.0	13821844156.0

Tabelle 5.17.: Global Results des Expansion Problems optimiert durch Urbs, Oemof und PyPSA

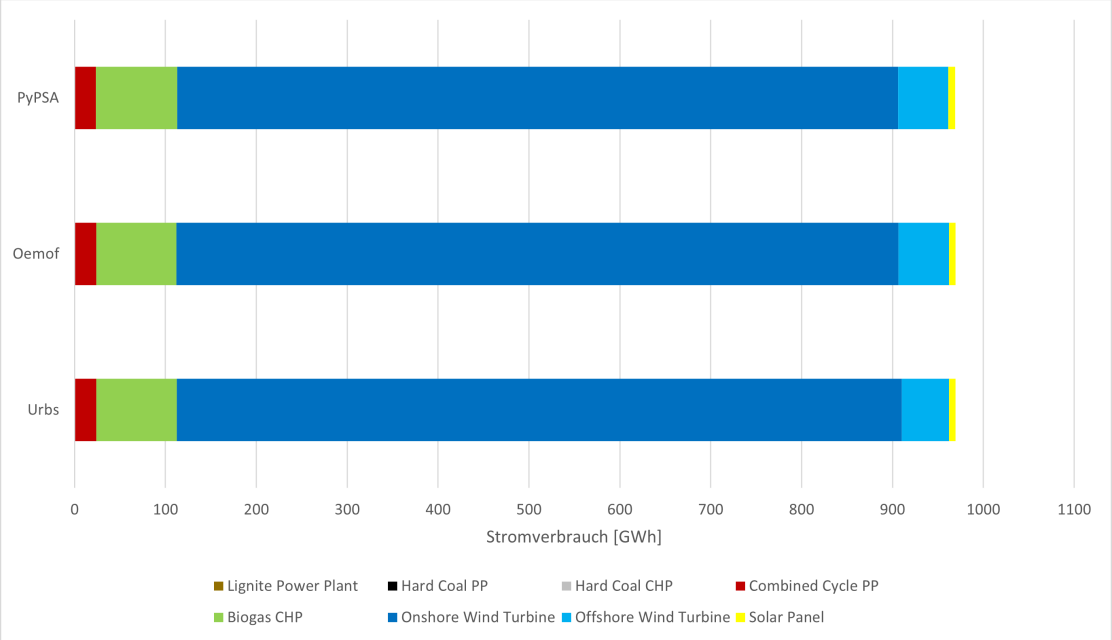


Abbildung 5.7.: Zusammensetzung des Stromsektors für das Expansion Problem des *component_es* der Tessif-Bibliothek optimiert mit Urbs, Oemof und PyPSA

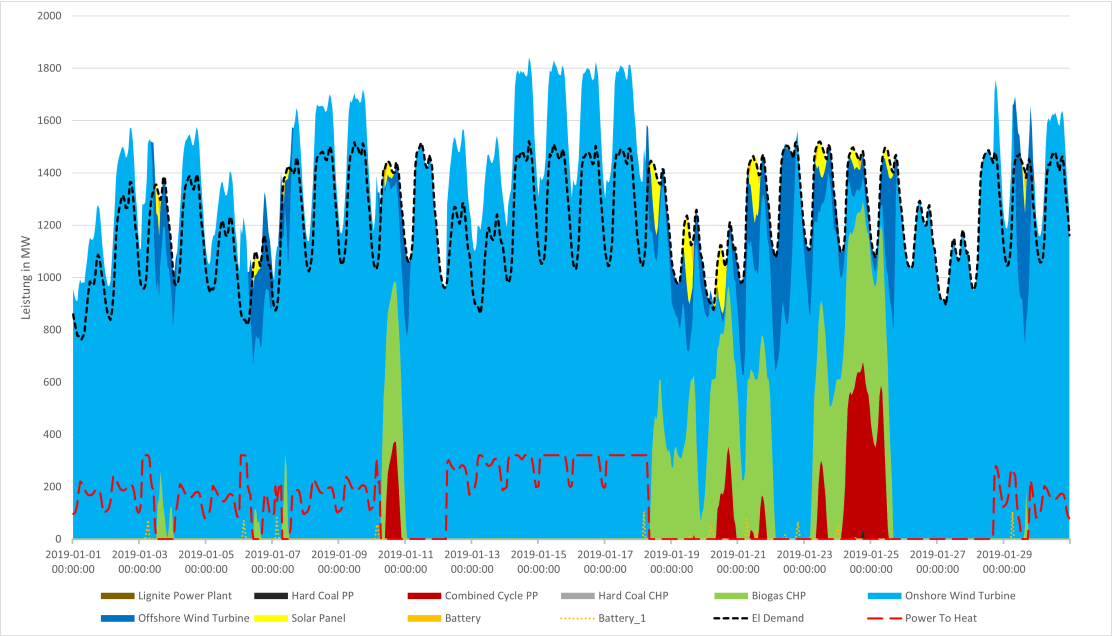


Abbildung 5.8.: Erzeugungsprofil des Stromsektors für das Expansion Problem des *component_es* der Tessif-Bibliothek optimiert mit Urbs

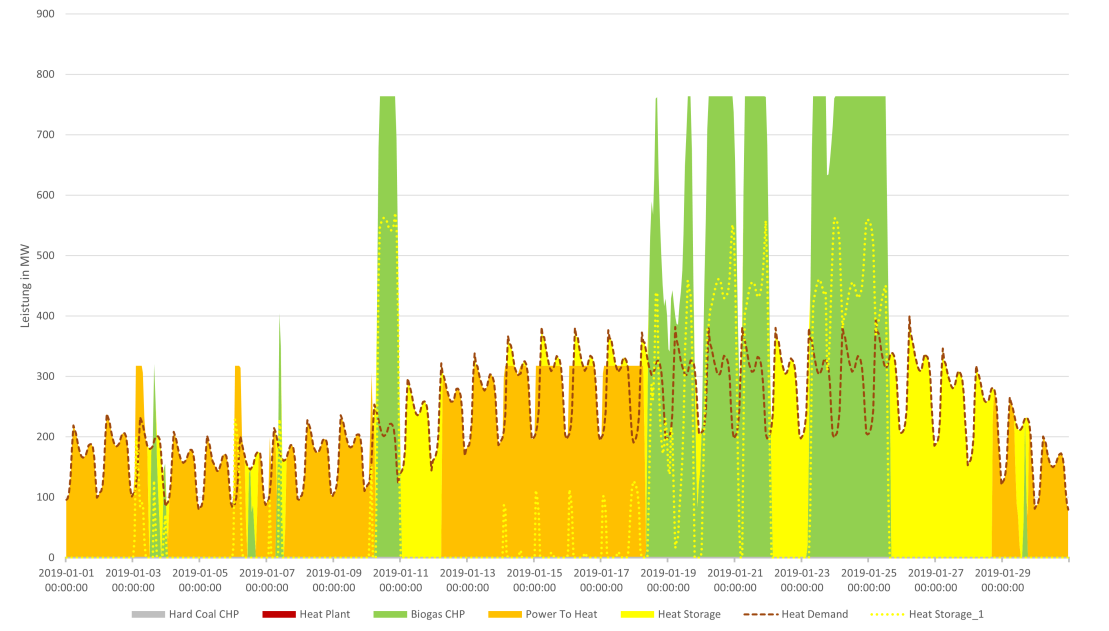


Abbildung 5.9.: Erzeugungsprofil des Wärmesektorssektors für das Expansion Problem des *component_es* der Tessif-Bibliothek optimiert mit Urbs

Demand Side Management

Dieser Abschnitt befasst sich mit dem Vergleich der Optimierung des Energiesystems durch Urbs mit und ohne Berücksichtigung von Lastmanagement (DSM). Für die Optimierung wurde das *component_es* ohne Berücksichtigung des Expansions-Szenario verwendet. Insbesondere ist das Potenzial von DSM in Bezug auf die Reduktion von Emissionen interessant, da Lastmanagement zukünftig verstärkt genutzt werden könnte, um die Lastverläufe den fluktuierenden Einspeisungen von Erneuerbaren Energien anzupassen, sodass weniger Regelleistung durch fossile Kraftwerke bereitgestellt werden muss. Weil bei Optimierungen mit Urbs die Zielvariable angepasst werden kann, erfolgt die Optimierung nicht anhand der Kosten, sondern sowohl für die Untersuchung mit als auch ohne DSM anhand der Emissionen des Energiesystems.
Wie aus der Grafik 5.10 ersichtlich wird, verschiebt sich auch hier die Energieversorgung zu den emissionsarmen erneuerbaren Energien. Dabei zeigt das Erzeugungsprofil aus Abbildung 5.11, dass die Grundversorgung durch Strom aus Solar-, Windkraft- und Biogasanlagen sowie durch das Gaskraftwerk, welches das emissionsärmste fossile Kraftwerk ist, bereitgestellt wird. Weil die erneuerbaren Energien fluktuieren und somit zeitlich nicht konstant sind, werden zusätzlich die Steinkohlekraftwerke verwendet, um Lastspitzen auszugleichen. Vergleicht man die Zusammensetzung der Stromerzeugung mit und ohne DSM im Diagramm 5.10, wird ersichtlich, dass durch das Lastmanagement die Spitzenlasten so verschoben werden, dass mehr Strom durch das emissionsärmere Gaskraftwerk und zeitgleich weniger Strom durch das Kohlekraftwerk bereitgestellt wird. Insbesondere wird dies auch beim Vergleich der Erzeugungsprofile in Abbildung 5.11 ersichtlich. Die DSM-Funktion von Urbs verschiebt die Last so, dass die Lastspitzen weitestgehend durch die fluktuierenden erneuerbaren Energiequellen und das Gaskraftwerk abgedeckt werden können.
Wäre die Erzeugungskapazität der erneuerbaren Energien größer, würde das Lastmanagement versuchen, die Last der fluktuierenden Einspeisung anzupassen. Da die Erzeu-

gungskapazität allerdings begrenzt ist und in diesem Szenario nicht ausgebaut werden kann, wird zusätzlich das Gaskraftwerk verwendet, um die Verbräuche zu decken.
Die Global Results in Tabelle 5.19 zeigen, dass durch das Lastmanagement sowohl die Emissionen als auch die Kosten reduziert werden konnten. Die reduzierten Emissionen sind durch die Verschiebung der Stromerzeugung vom Kohlekraftwerk (0.80 $\frac{\text{t } CO_2}{\text{MWh}}$) zur Erzeugung durch das Gaskraftwerk (0.35 $\frac{\text{t } CO_2}{\text{MWh}}$) zu erklären. Dadurch sind zwar die Kosten minimal gestiegen, allerdings werden durch eine geringere Nutzung der Batterie mehr Kosten eingespart, sodass die Kosten bei der Verwendung von DSM insgesamt etwas geringer ausfallen. Die von den einzelnen Komponenten aufgenommen beziehungsweise an die Powerline abgegebenen Energiemengen sind im Abschnitt A.5.4 im Anhang angegeben.
Abschließend sind die für die Optimierung benötigten Zeiten in der Tabelle 5.18 aufgelistet. Die Optimierung mit DSM braucht mit 58 Sekunden deutlich länger als die Optimierung ohne DSM mit 38 Sekunden. Dabei muss berücksichtigt werden, dass die 20 Sekunden Differenz bei der Transformation des Tessif-Energiesystems zum Urbs-Energiesystem auftreten und auf die manuelle Nutzereingabe der zusätzlich benötigten DSM-Daten zurückzuführen ist (Preprocess Time).

	Urbs ohne DSM	Urbs mit DSM
Preprocess Time in s	0.75	20.0746
Process Time in s	21.8488	25.6693
Postprocess Time in s	15.6538	12.5937
Time in s	38.2526	58.3376

Tabelle 5.18.: Berechnungszeiten der Optimierung des *component_es* durch Urbs mit und ohne DSM

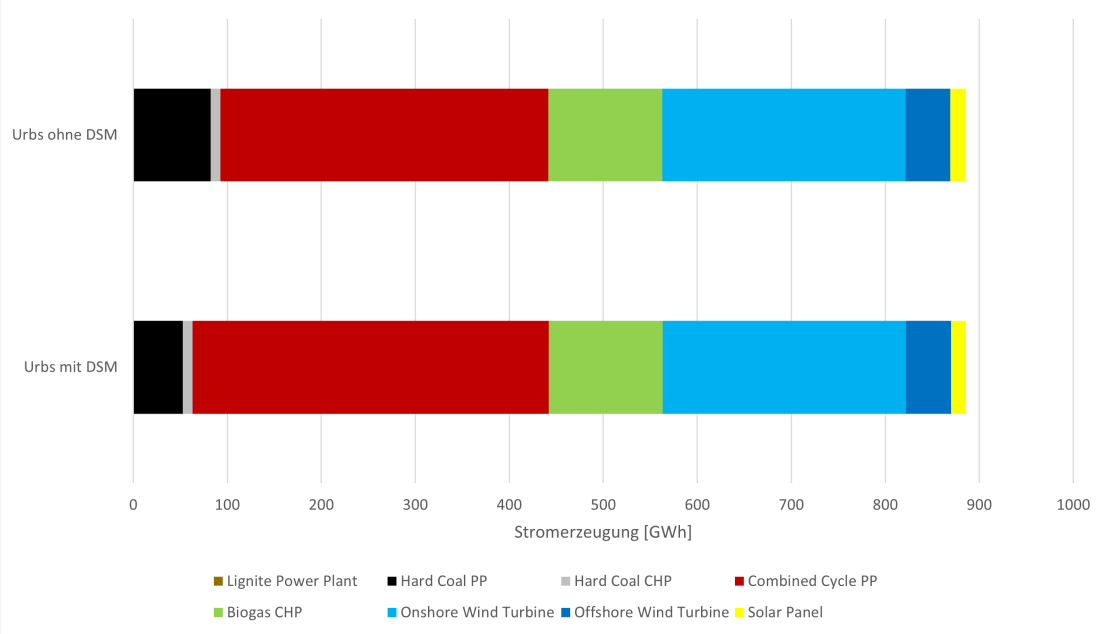


Abbildung 5.10.: Zusammensetzung des Stromsektors für die Optimierung des *component_es* der Tessif-Bibliothek durch Urbs mit und ohne DSM

	Urbs ohne DSM	Urbs mit DSM
Emissionen in t CO_2	239101.0	225726.0
Kosten in €	81672481.0	81448742.0
OpEx in €	81672481.0	81448742.0
CapEx in €	0.0	0.0

Tabelle 5.19.: Global Results der Optimierung des *component_es* durch Urbs mit und ohne DSM

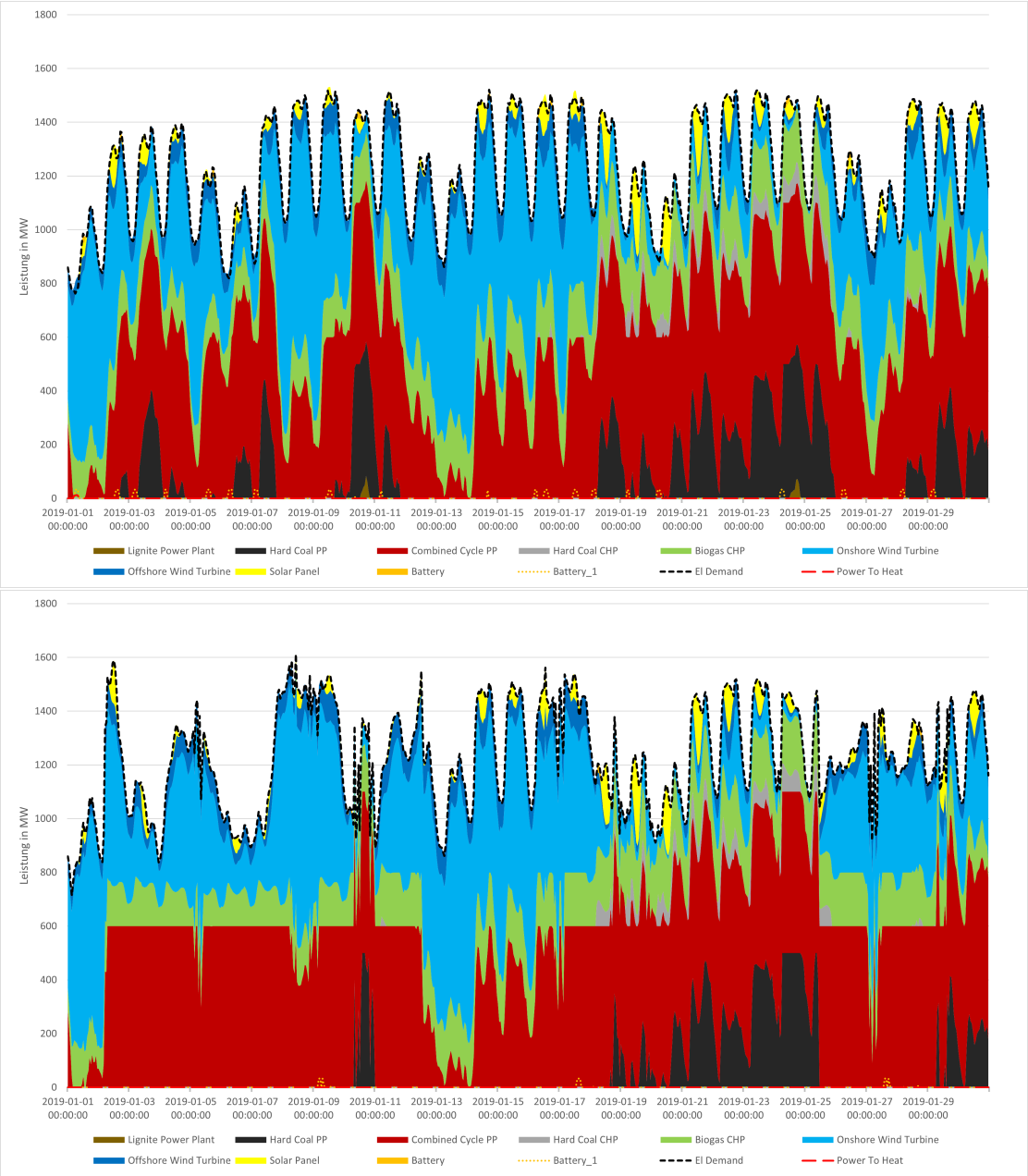


Abbildung 5.11.: Erzeugungsprofil des Stromsektors des durch Urbs ohne (oben) und mit (unten) DSM optimierten *component_es* der Tessif Bibliothek

5.4. Zusammenfassung der Ergebnisse

Sowohl der automatische als auch der detaillierte Vergleich zeigen, dass einige Unterschiede zwischen Urbs und den anderen in Tessif integrierten Optimierungsprogrammen existieren. Die wesentlichen Unterschiede sind dabei in der Implementierung der Speicherkomponente zu finden. So muss, anders als bei Tessif, in Urbs der finale Speicherstand identisch mit dem initialen Speicherstand sein. Zudem werden die variablen Kosten des Speichers bei Tessif nur auf die ausgespeicherte und bei Urbs zusätzlich auf die eingespeicherte Energiemenge bezogen. Des Weiteren können Speicher in Urbs nicht mit Emissionen behaftet werden. Bei komplexeren Energiesystemen mit insbesondere längeren Zeitreihen benötigt Urbs zudem länger, um die Energiesysteme zu modellieren und zu optimieren.
Abschließend lässt sich sagen, dass Urbs, in Tessif integriert, in der Lage ist, die diversen Beispielenergiesysteme und die verschiedenen Szenarien des detaillierten Vergleichs zu optimieren. Zusätzlich können auch weitere Funktionen von Urbs wie das Lastmanagement (DSM) und das Verändern der Optimierungsvariable in Tessif verwendet werden.

6. Zusammenfassung

Abschließend werden die wesentlichen Inhalte dieser Arbeit kurz zusammengefasst sowie ein Fazit gezogen. Der Fokus wird dabei insbesondere auf die Integration und die Verwendung von Urbs in Tessif gelegt. Des Weiteren wird ein kleiner Ausblick über weitere Möglichkeiten der Integration von Urbs in Tessif sowie die Erkenntnisse dieser Arbeit gegeben.

Im Rahmen dieser Arbeit sollte ein weiteres Free and Open Source Software Tool zur Optimierung von Energiesystemmodellen in das als Vergleichsplattform agierende Framework Tessif integriert werden. Bereits in Tessif integriert sind die Modellierungsprogramme Oemof, PyPSA, FINE und Calliope. Aufgrund der Programmierung in Python, diverser Modellierungskriterien sowie der Kompatibilität mit Tessif und der bereits integrierten Optimierungstools wurde Urbs als gut zur Integration geeignetes Programm identifiziert.
Die Integration erfolgt im Wesentlichen durch die Automatisierung der Umwandlung des Tessif-Energiesystems in ein Urbs-Energiesystem und der Umwandlung der Optimierungsergebnisse von Urbs in Tessifs einheitliche Ergebnisdarstellung. Um diese Transformationen zu ermöglichen, wurde im Rahmen dieser Arbeit die Energiesystem-zu-Energiesystem Datei (*es2es*) und die Energiesystem-zu-Mapping Datei (*es2mapping*) für Urbs in Tessif entwickelt. Zusätzlich wurden noch verschiedene kleinere Anpassung in Tessif und Urbs durchgeführt, um die Integration zu vervollständigen.
Abschließend wurde mit den bereits in Tessif integrierten Softwaretools Oemof, PyPSA, FINE und Calliope ein exemplarischer Vergleich durchgeführt, um einerseits die Integration zu überprüfen und andererseits einen ersten Vergleich zwischen den verschiedenen Modellierungsprogrammen zu erhalten.

6.1. Fazit

Der exemplarische Vergleich zeigt, dass die neun untersuchten Energiesysteme aus Tessifs Datenbank transformiert und anschließend durch Urbs in Tessif optimiert werden können. Die Ergebnisse werden dann in Tessifs Darstellungsweise gewandelt und können ausgegeben oder durch weitere Tools von Tessif für eine grafische Auswertung verwendet werden.
Wie die verschiedenen Optimierungen der Beispielenergiesysteme zeigen, können verschiedenene kleinere Differenzen auftreten. Diese Differenzen entstehen nicht durch eine fehlerhafte Integration, sondern sind auf unterschiedliche Definitionen von Komponenten beziehungsweise deren Parameter zurückzuführen. Zusätzlich führen die unterschiedlichen Modellansätze von Tessif und Urbs dazu, dass nicht alle Tessif-Parameter einen Einfluss auf das Urbs-Modell haben. Allerdings weisen die in Tessif integrierten Modellierungsprogramme grundlegend viele Gemeinsamkeiten auf, sodass die Optimierungsergebnisse ebenfalls große Ähnlichkeit aufweisen und somit vergleichbar sind.
Die Integration in Tessif unterbindet aber auch einige Funktionen, Komponenten oder

Optimierungsmethoden der integrierten Tools, die über den Rahmen von Tessif hinaus gehen würden. So wäre Urbs beispielweise in der Lage bei dem Ausbau von Komponenten wie Photovoltaik-Anlagen die pro Kapazität benötigte Fläche oder aber Zinsen der Investitionskosten zu berücksichtigen. Da solche Parameter beim Tessif-Energiesystem nicht definiert sind, werden diese bei der Optimierung mit Urbs in Tessif ebenfalls nicht beachtet. Somit unterdrückt die einheitliche Modellformulierung von Tessif weitere eventuelle Unterschiede und vereinheitlicht die Optimierungsergebnisse.

Neben der Überprüfung der Integration von Urbs in Tessif ermöglicht der exemplarische Vergleich auch einen ersten Vergleich von Urbs mit den anderen bereits in Tessif integrierten Modellierungstools. Die Vorteile von Urbs liegen vor allem in zusätzlichen Funktionen, die mit den anderen Tools in Tessif nicht realisiert werden können.
Nachfolgend werden einige Aspekte aufgezählt, die für und gegen eine Verwendung von Urbs in Tessif sprechen. Die Relevanz der Kriterien zur Auswahl eines geeigneten Modellierungstools variieren in Abhängigkeit der zu untersuchenden Problemstellung.

- Bei Optimierungen, insbesondere komplexerer Energiesystemmodelle über viele Zeitschritte, kann es vorkommen, dass Urbs mehr Zeit zum Modellieren und Optimieren benötigt. Die Ergebnisse des exemplarischen Vergleichs lassen darauf schließen, dass vor allem Oemof oder PyPSA für komplexere Anwendungsfälle besser geeignet sind.

- Urbs ist nur in der Lage lineare Probleme zu lösen (LP). Sollen gemischtganzzahlige Probleme optimiert werden, muss ebenfalls auf ein anderes der in Tessif integrierten Softwaretools zurückgegriffen werden.

+ Urbs hat den Vorteil, dass die Zielvariable der Optimierung verändert werden kann und somit für ein bestehendes Energiesystem neben den minimalen Betriebskosten auch die minimalen Emissionen bestimmt werden können. Diese Funktion von Urbs kann insbesondere genutzt werden, um das Potenzial der Reduktion von Schadstoffemissionen einzuschätzen. Dafür kann auch zusätzlich eine maximale Kostengrenze definiert werden, die nicht überschritten werden darf. Da Urbs als einziges der in Tessif integrierten Tools über eine solche Funktion verfügt, bietet es sich an, Urbs bei Betrachtungen der Emissionen beziehungsweise der Bestimmung des Emissionspotenzials eines Energiesystems zu verwenden.

+ Zudem ist Urbs in der Lage, ein detailliertes Lastmanagement (DSM) zu modelieren, um Spitzenlasten zu anderen Zeitpunkten zu verschieben und somit Kosten oder Emissionen zu reduzieren. Sofern Energiesysteme mit Lastmanagement oder die Potenziale und Einflüsse von Lastmanagement untersucht werden sollen, empfiehlt sich die Nutzung von Urbs.

Die Auswahl eines geeigneten Optimierungsprogrammes hängt von vielen verschiedenen Faktoren ab und kann durch einen direkten Vergleich der Ergebnisse mit Tessif vereinfacht werden. Das Tessif-Modell ist in der Lage, ein breites Spektrum an Anwendungsfällen abzubilden, weshalb Tessif für viele verschiedene Anwendungen einen guten Überblick über die implementierten Softwaretools ermöglicht. Erfordern Problemstellungen Fähigkeiten oder Funktionen der implementierten Modellierungsprogramme, die über die Integration in Tessif hinausgehen, kann durch einen Vergleich mit Tessif zunächst die Menge der potenziellen Softwaretools eingegrenzt werden.
Tessifs wesentlicher Vorteil ist allerdings die Vereinheitlichung der Schnittstellen. So

ermöglicht die einheitliche Darstellung der Ergebnisse einen Vergleich ohne weiteren Aufwand. Zusätzlich wird der Aufwand und die Fehleranfälligkeit reduziert, da nur ein Tessif-Modell erstellt werden muss, das genutzt werden kann, um alle in Tessif implementierten Modellierungsprogramme miteinander zu vergleichen. Ein weiterer großer Vorteil von Tessif ist, die Möglichkeit Optimierungsergebnisse eines Tools durch die Optimierung desselben Modells mit weiteren Softwaretools zu validieren beziehungsweise zu überprüfen.

Abschließend lässt sich zusammenfassen, dass die Integration von Urbs in das Framework Tessif erfolgreich ist. Darüber hinaus können die wesentlichen Funktionen von Urbs, insbesondere auch die Demand Side Management Funktion, im Rahmen von Tessif für die Modellierung und Optimierung verwendet werden. Auf diese Weise erweitert die Implementierung von Urbs Tessifs Funktionalität, da auf zusätzliche Optimierungsergebnisse für Vergleiche zurückgegriffen werden kann und zudem die Auswahl an verwendbaren Modellierungsprogrammen erhöht wurde.

6.2. Ausblick

Um Tessif weiter sowohl qualitativ als auch quantitativ zu erweitern, können noch andere Modellierungsprogramme in das Framework implementiert werden. Dabei kann die Integration und insbesondere die *es2es*-Datei von Urbs als Vorlage beziehungsweise Beispiel für die Implementierung von Softwaretools, deren Modell nicht wie bei Tessif auf Komponenten basiert, genutzt werden.
Zusätzlich kann das Erstellen weiterer Energiesystemmodelle zum einen Tessifs Beispielbibliothek erweitern und zum anderen die Qualität der Untersuchungen und Vergleiche zwischen den Modellierungstools steigern. So können neu entwickelte Energiesystemmodelle helfen, weitere Differenzen aufzudecken und dem Nutzer einen Überblick über die Möglichkeiten und Funktionen von Tessifs Modell zu geben.
Auch kann die Integration, der bereits zu Tessif hinzugefügten Tools, erweitert beziehungsweise verbessert werden, indem die Komponenten von Tessifs Modell um zusätzliche Parameter erweitert werden. So könnten beispielweise für Urbs weitere finanzielle Parameter wie Zinsen oder Abschreibungszeiträume der Kosten bei Expansionsszenarien in Tessifs Modell definiert werden. Dabei ist allerdings zu beachten, dass eine Änderung von Tessifs Modell sich grundlegend auf die *es2es* und *es2mapping* Dateien aller integrierter Tools auswirkt und diese ebenfalls angepasst werden müssen. Durch zusätzliche Parameter könnten weitere Funktionen der Modellierungsprogramme abgebildet werden, sodass die Ergebnisse mit Tessif identischer zu den nativen Ergebnissen ausfallen würden.

Abschließend kann es von Vorteil sein, sowohl Urbs als auch die weiteren in Tessif integrierten Modellierungstools auf dem neuesten Stand zu halten. Die Aktualisierungen können insbesondere die Rechenzeiten und die für die Optimierung benötigten Speicherressourcen verbessern, das jeweilige Tool um zusätzliche Funktionen erweitern oder Fehler der vorherigen Versionen beheben. Aktualisierungen sind nur möglich, wenn die Kompatibilität mit Tessif gegeben ist. Außerdem ist zu beachten, dass Veränderungen der Modellierungsprogramme zu Anpassungen in den Transformationsdateien (*es2es* und *es2mapping*) führen können. Trotz des Aufwandes ist das Aktualisieren der Modellierungsprogramme zu empfehlen, um die Qualität der mit Tessif erzielten Ergebnisse zu erhalten oder sogar zu steigern.

Literatur

[1] United Nations Framework Convention on Climate Change, UNFCCC: *Klimarahmenkonvention*. https://unfccc.int/resource/docs/convkp/conveng.pdf, 1992.

[2] Umweltbundesamt: *Energieziel 2050: 100% Strom aus erneuerbaren Quellen*. https://www.umweltbundesamt.de/sites/default/files/medien/378/publikationen/energieziel_2050.pdf, 2010.

[3] M. Fodstad et al.: „Next frontiers in energy system modelling: A review on challenges and the state of the art“. In: *Renewable and Sustainable Energy Reviews* 160 (Mai 2022), S. 112246.

[4] P. Lopion et al.: „A review of current challenges and trends in energy systems modeling“. In: *Renewable and Sustainable Energy Reviews* 96 (Nov. 2018), S. 156–166.

[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 (Nov. 2018), S. 440–459.

[6] D. Connolly et al.: „A review of computer tools for analysing the integration of renewable energy into various energy systems“. In: *Applied Energy* 87.4 (Apr. 2010), S. 1059–1082.

[7] Open Energy Modelling Initiative: *openmod*. https://openmod-initiative.org/.

[8] M. Ammon: *Tessif Dokumentation*. 2021.

[9] U. Kastens und H. K. Büning: *Modellierung Grundlagen und formale Methoden*. München: Carl Hanser Verlag GmbH & Co. KG, 2018.

[10] A. Subramanian, T. Gundersen und T. Adams: „Modeling and Simulation of Energy Systems: A Review“. In: *Processes* 6.12 (Nov. 2018), S. 238.

[11] J. Fisher und T. A. Henzinger: „Executable cell biology“. In: *Nature Biotechnology* 25.11 (Nov. 2007), S. 1239–1249.

[12] I. E. Grossmann und A. W. Westerberg: „Research challenges in process systems engineering“. In: *AIChE Journal* 46.9 (Sep. 2000), S. 1700–1703.

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

[14] J. Nagel: *Optimization of Energy Supply Systems*. Springer International Publishing, 2019.

[15] L. Suhl und T. Mellouli: *Optimierungssysteme*. Springer Berlin Heidelberg, 2013.

[16] COIN-OR Foundation: *COIN-OR Branch-and-Cut solver: Cbc*. https://github.com/coin-or/Cbc, 2005.

[17] Free Software Foundation: *GLPK (GNU Linear Programming Kit)*. https://pypi.org/project/glpk/, 2012.

[18] S. D. Thomas Unger: *Lineare Optimierung*. Vieweg+Teubner Verlag, März 2010. 156 S.

[19] K. K. Horst W. Hamacher: *Lineare Optimierung und Netzwerkoptimierung*. Vieweg+Teubner Verlag, Apr. 2006. 256 S.

[20] D. Bauer et al.: „Modellbasierte Optimierung von Energiesystemen“. In: Sep. 2015.

[21] J. Kallrath: *Gemischt-ganzzahlige Optimierung: Modellierung in der Praxis*. Springer-Verlag GmbH, Nov. 2012. 381 S.

[22] J. Feller: *Perspectives on free and open source software*. Cambridge, Mass: MIT Press, 2005.

[23] S. Pfenninger et al.: „Opening the black box of energy modelling: Strategies and lessons learned“. In: *Energy Strategy Reviews* 19 (Jan. 2018), S. 63–71.

[24] H. C. Gils et al.: „Modeling flexibility in energy systems — comparison of power sector models based on simplified test cases“. In: *Renewable and Sustainable Energy Reviews* 158 (Apr. 2022), S. 111995.

[25] J. T. Wilkerson et al.: „Comparison of integrated assessment models: Carbon price impacts on U.S. energy“. In: *Energy Policy* 76 (Jan. 2015), S. 18–31.

[26] 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 (Sep. 2014), S. 109–118.

[27] J. Priesmann, L. Nolting und A. Praktiknjo: „Are complex energy system models more accurate? An intra-model comparison of power system optimization models“. In: *Applied Energy* 255 (Dez. 2019), S. 113783.

[28] C. Steinbrink et al.: „Simulation-Based Validation of Smart Grids – Status Quo and Future Research Trends“. In: *Lecture Notes in Computer Science*. Springer International Publishing, 2017, S. 171–185.

[29] S. Misconel et al.: „Systematic comparison of high-resolution electricity system modeling approaches focusing on investment, dispatch and generation adequacy“. In: *Renewable and Sustainable Energy Reviews* 153 (Jan. 2022), S. 111785.

[30] Oemof developer group: *oemof.solph Dokumentation*. `https://oemof-solph.readthedocs.io/en/latest/`, 2021.

[31] PyPsa Developers: *PyPsa Dokumentation*. `https://pypsa.readthedocs.io/en/latest/`, 2021.

[32] FINE Developer Team: *FINE: A Framework for Integrated Energy System Assessment*. Hrsg. von Forschungsteam Jülich. `https://vsa-fine.readthedocs.io/en/latest/`, 2022.

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

[34] T. J. Hanke: „Entwickeln eines netzbasierten Energiesystemmodells zum Vergleich von Free und Open Source Energiesystemmodellierungssoftware in Python“. de. In: (2022).

[35] J. Dorfner et al.: *urbs: v0.5*. Version v0.5. Feb. 2016.

[36] J. Dorfner: „Open Source Modelling and Optimisation of Energy Infrastructure at Urban Scale“. Dissertation. München: Technische Universität München, 2016.

[37] J. Dorfner: *urbs: A linear optimisation model for distributed energy systems*. https://urbs.readthedocs.io/en/latest/, 2022.

[38] tsam Developer Team: *tsam: time series aggregation module*. Hrsg. von Forschungsteam Jülich. https://tsam.readthedocs.io/en/latest/index.html, 2021.

[39] Sandia National Laboratories: *Pyomo*. http://www.pyomo.org/.

[40] M. Reimer: „Entwicklung eines Komponenten basierten Szenarios zum Vergleich von Free and Open Source Energiesystemmodellierungssoftware in Python“. de. In: (2022).

[41] T. Ladwig: „Demand Side Management in Deutschland zur Systemintegration erneuerbarer Energien“. Dissertation. Dresden: Technische Universität Dresden, 10.07.2018.

[42] D. Schnute: „Vergleichende Analyse von Software zur Modellierung von Energiesystemen mittels Integration in ein Framework zur Transformation“. de. In: (2022).

[43] M. Reimer: „Vergleich von Modellierungsprogrammen zur Optimierung von Energiesystemen durch Integration in ein bestehendes Framework zur Transformation von Energiesystem-Modellen“. de. Diss. 2022.

A. Anhang

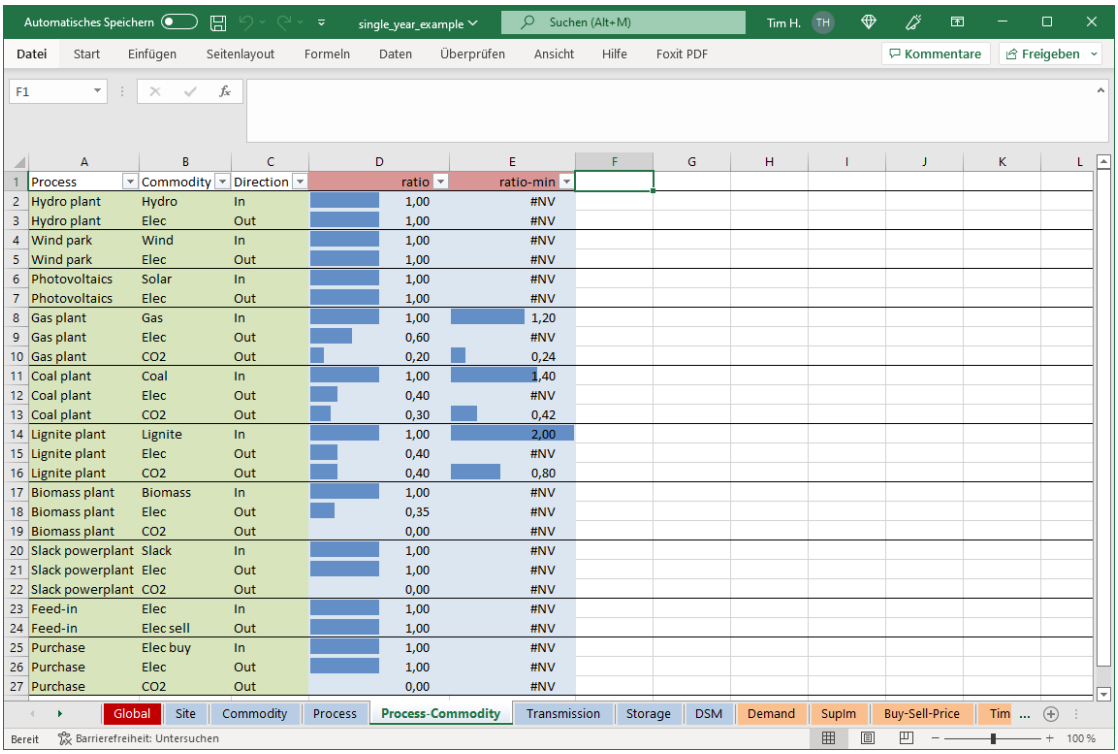
A.1. Modelldarstellungen von Tessif und Urbs

A.1.1. Tessifs Modell-Darstellungsweise

```
biogas_chp = components.Transformer(
        name='Biogas CHP',
        inputs=('biogas',),
        outputs=('electricity', 'hot_water',),
        conversions={('biogas', 'electricity'): 0.4,
                     ('biogas', 'hot_water'): 0.5},
        sector='Coupled',
        carrier='coupled',
        node_type='transformer',
        flow_rates={
            'biogas': nts.MinMax(min=0, max=float('+inf')),
            'electricity': nts.MinMax(min=0, max=200),
            'hot_water': nts.MinMax(min=0, max=250)},
        flow_costs={'biogas': 0, 'electricity': 150, 'hot_water': 11.25},
        flow_emissions={'biogas': 0,
                        'electricity': 0.25, 'hot_water': 0.01875},
        flow_gradients={
            'biogas': nts.PositiveNegative(positive=float('+inf'),
        negative=float('+inf')),
            'electricity': nts.PositiveNegative(positive=float('+inf'),
        negative=float('+inf')),
            'hot_water': nts.PositiveNegative(positive=float('+inf'),
        negative=float('+inf'))},
        gradient_costs={
            'biogas': nts.PositiveNegative(positive=0, negative=0),
            'electricity': nts.PositiveNegative(positive=0, negative=0),
            'hot_water': nts.PositiveNegative(positive=0, negative=0)},
        timeseries=None,
        expandable={'biogas': False,
                    'electricity': expansion_problem,
                    'hot_water': expansion_problem},
        expansion_costs={'biogas': 0,
                         'electricity': 3500000, 'hot_water': 262500},
        expansion_limits={
            'biogas': nts.MinMax(min=0, max=float('+inf')),
            'electricity': nts.MinMax(min=200, max=float('+inf')),
            'hot_water': nts.MinMax(min=250, max=float('+inf'))},
    )
```

Quellcode A.1: Tessifs Darstellung eines Beispiel-Transformers entnommen aus *tessif.examples.data.py_hard*

A.1.2. Urbs Modell-Darstellungsweise



	Process	Commodity	Direction	ratio	ratio-min
2	Hydro plant	Hydro	In	1,00	#NV
3	Hydro plant	Elec	Out	1,00	#NV
4	Wind park	Wind	In	1,00	#NV
5	Wind park	Elec	Out	1,00	#NV
6	Photovoltaics	Solar	In	1,00	#NV
7	Photovoltaics	Elec	Out	1,00	#NV
8	Gas plant	Gas	In	1,00	1,20
9	Gas plant	Elec	Out	0,60	#NV
10	Gas plant	CO2	Out	0,20	0,24
11	Coal plant	Coal	In	1,00	1,40
12	Coal plant	Elec	Out	0,40	#NV
13	Coal plant	CO2	Out	0,30	0,42
14	Lignite plant	Lignite	In	1,00	2,00
15	Lignite plant	Elec	Out	0,40	#NV
16	Lignite plant	CO2	Out	0,40	0,80
17	Biomass plant	Biomass	In	1,00	#NV
18	Biomass plant	Elec	Out	0,35	#NV
19	Biomass plant	CO2	Out	0,00	#NV
20	Slack powerplant	Slack	In	1,00	#NV
21	Slack powerplant	Elec	Out	1,00	#NV
22	Slack powerplant	CO2	Out	0,00	#NV
23	Feed-in	Elec	In	1,00	#NV
24	Feed-in	Elec sell	Out	1,00	#NV
25	Purchase	Elec buy	In	1,00	#NV
26	Purchase	Elec	Out	1,00	#NV
27	Purchase	CO2	Out	0,00	#NV

Abbildung A.1.: Beispielhafte Abbildung der Modell-Darstellungsweise von Urbs

A.2. Zuordnung der Eigenschaften der Komponenten bei der Transformation

Bus-Komponente*

Tessif:	Urbs:
Name	Site.Name
Input	Commodity.Site, Process.Site, Storage.Site
Output	Storage.Site, Demand

**Nur bei Bus Komponenten mit mehr als einem In- oder Output oder in Verbindung mit Connector*

Connector-Komponente

Tessif:	Urbs:
Interface	Transmission.Site In/Out
Conversions	Transmission.eff

Source-Komponente (Renewable/Timeseries)

Tessif:	Urbs:
Name	Process.Process
Outputs	Process_Commodity.Commodity
Accumulated Amounts	/
Flow Rates	Process.Inst-Cap, Process.Min-Frac
Flow Costs	Process.Var-Cost
Flow Emissions	Process_Commodity.Ratio
Flow Gradients	Process.Max-Grad
Timeseries	SupIm
Expandable	/
Expansion Costs	Process.Inv-Cost
Expansion Limits	Process.Cap-Lo/Up
Cost for being active	Process.Fix-Cost

**Source-Komponenten (Renewable) über mehrere SupIm-Sources und Prozesse, um die Funktionen der Erneuerbaren Quellen weitestgehend abzudecken (unterschiedliche min/max Timeseries etc.)*

Source-Komponente*

Tessif:	Urbs:
Name	Process.Process
Outputs	Process_Commodity.Commodity
Accumulated Amounts	Commodity.Max
Flow Rates	Process.Inst-Cap, Process.Min-Frac
Flow Costs	Process.Var-Cost
Flow Emissions	Process_Commodity.Ratio
Flow Gradients	Process.Max-Grad
Timeseries	/
Expandable	/
Expansion Costs	Process.Inv-Cost
Expansion Limits	Process.Cap-Lo/Up
Cost for being active	Process.Fix-Cost

**Source-Komponenten über Stock und Process dargestellt, um Emission, Ausbau, etc. zu ermöglichen*

Sink-Komponente

Tessif:	Urbs:
Name	Demand.Index
Inputs	Demand.index
Accumulated Amounts	/
Flow Rates	Demand
Flow Costs	/
Flow Emissions	/
Flow Gradients	/
Timeseries	Demand
Expandable	/
Expansion Costs	/
Expansion Limits	/
Cost for being active	/

Transformer-Komponente

Tessif:	Urbs:
Name	Process.Process
Inputs	Process_Commodity.Commodity
Outputs	Process_Commodity.Commodity
Conversions	Process_Commodity.Ratio
Flow Rates	Process.Inst-Cap, Process.Min-Frac
Flow Costs	Process.Var-Cost
Flow Emissions	Process_Commodity.Ratio
Flow Gradients	Process.Max-Grad
Timeseries	/
Expandable	/
Expansion Costs	Process.Inv-Cost
Expansion Limits	Process.Cap-Lo/Up
Cost for being active	Process.Fix-Cost

Storage-Komponente

Tessif:	Urbs:
Name	Storage.Storage
Input	Storage.Commodity
Output	Storage.Commodity
Initial SOC	Storage.Init
Final SOC	/
Idle Changes	Storage.Discharge
Flow Rates	Storage.Inst-Cap-P
Flow Efficiencies	Storage.Eff-In/Out
Flow Costs	Storage.Var-Cost-P
Flow Emissions	/
Flow Gradients	/
Gradient Costs	/
Timeseries	/
Expandable	/
Fixed-Expansion-Ratios	/
Expansion Costs	Storage.Inv-Cost-C, Storage.Inv-Cost-P
Expansion Limits	Storage.Cap-Lo/Up-C, Storage.Cap-Lo/Up-P
Costs for being active	Storage.Fix-Cost-P

Tabelle A.1.: Zuordnung der Komponentenparameter bei der Transformation

A.3. Zuordnung der Urbs Ergebnis-Dataframes zu den Tessif Komponenten

Komponente (Tessif):	**Urbs Ergebnis-DF:**
Bus (Input)	e_pro_out
	e_tra_out
	e_sto_out
Bus (Output)*	e_pro_in
	e_tra_in
	e_sto_in
Source (Renewable)	e_pro_in
Source (Supply)	e_co_stock
Sink*	/
Transformer (Input)	e_pro_in
Transformer (Output)	e_pro_out
Storage (Input)	e_sto_in
Storage (Output)	e_sto_out
Connector (Input)	e_tra_in
Connector (Output)	e_tra_out

**Die Verbräuche werden den weitergebenen Node-Informationen entnommen*

Tabelle A.2.: Zuordnung der Urbs Ergebnis-Dataframes bei der Transformation der Ergebnisse

A.4. Verwendete Energiesystemmodelle

Minimal Working Example

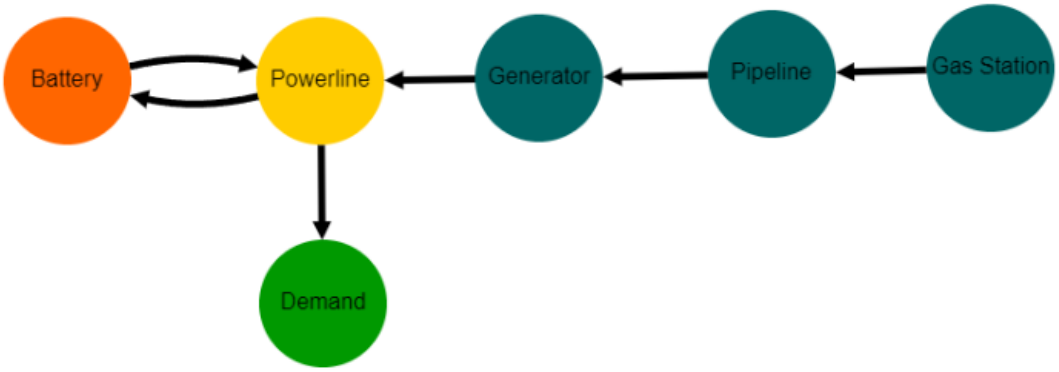


Abbildung A.2.: Minimal Working Example

Fully Parametrized Working Example

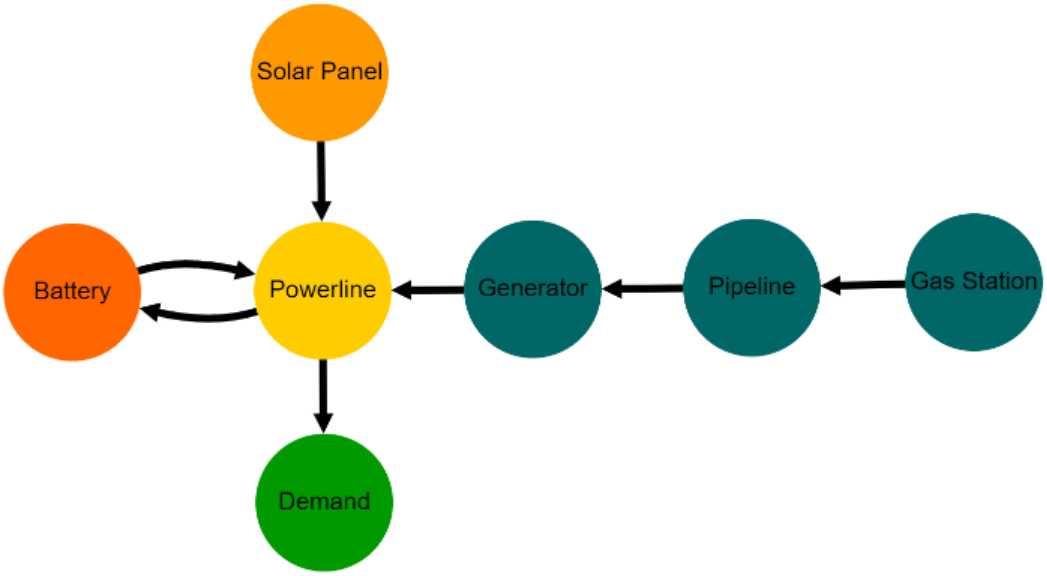


Abbildung A.3.: Fully Parametrized Working Example

Emission Objective

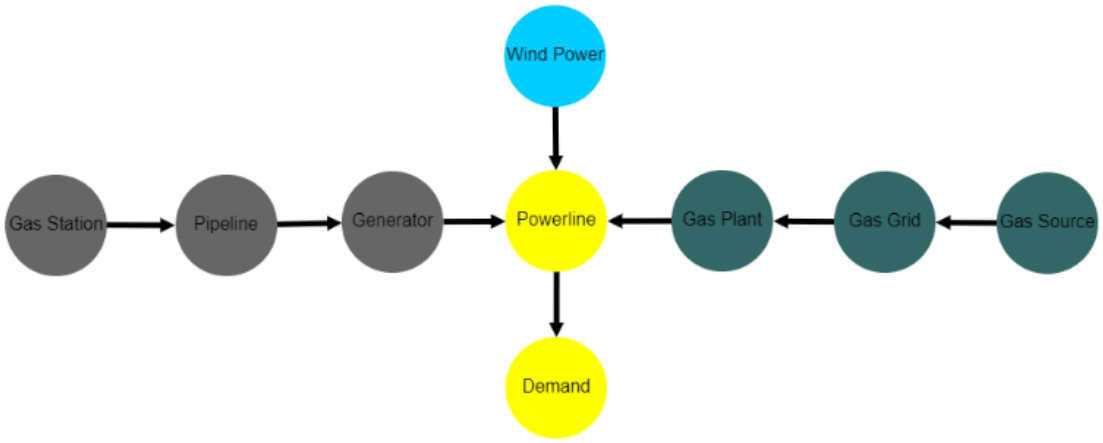


Abbildung A.4.: Emissione Objective

Connected Energy System

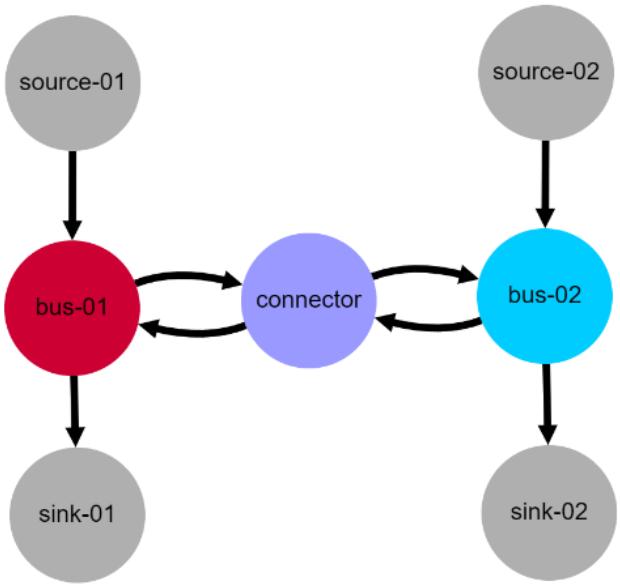


Abbildung A.5.: Connected Energy System

Combined Heat and Power

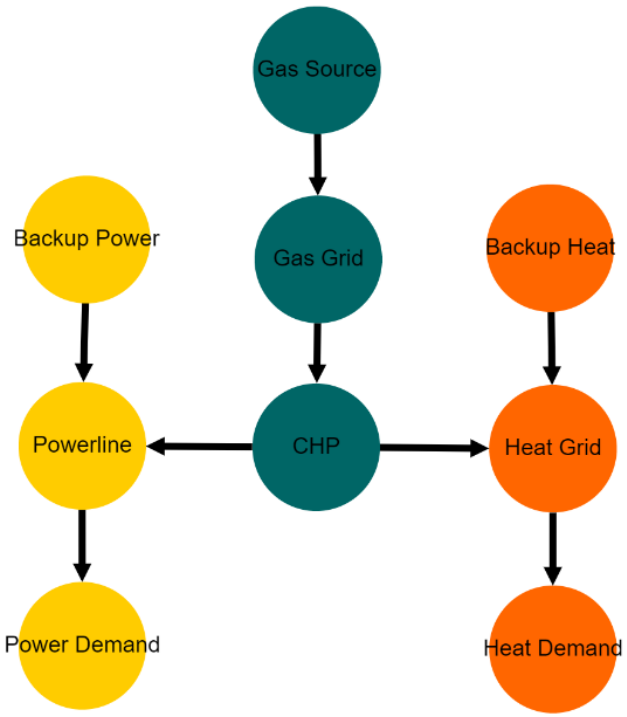


Abbildung A.6.: Combined Heat and Power

Storage Exampe

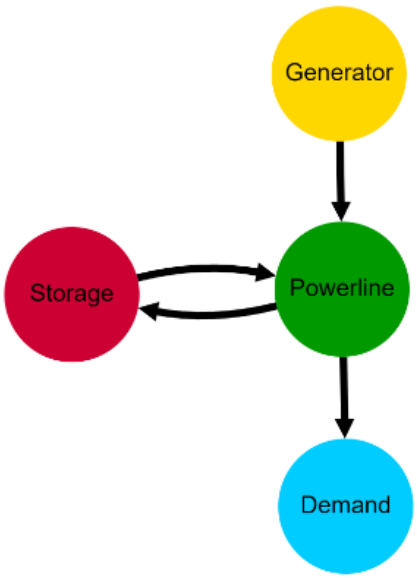


Abbildung A.7.: Storage Example

Expansion Plan Example

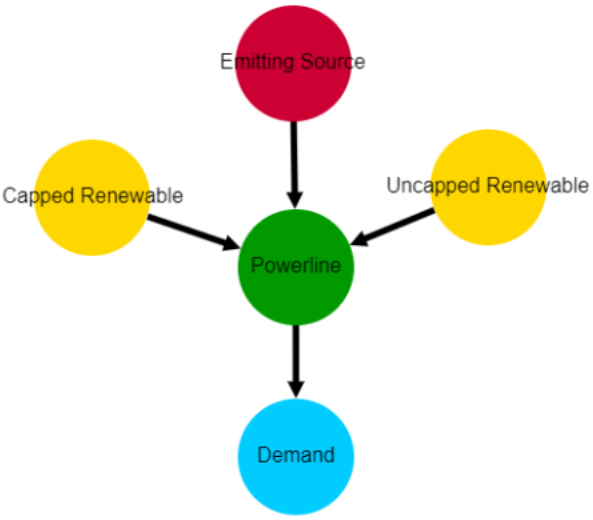


Abbildung A.8.: Expansion Plan Example

Energy System Grid „Kupferplatte“

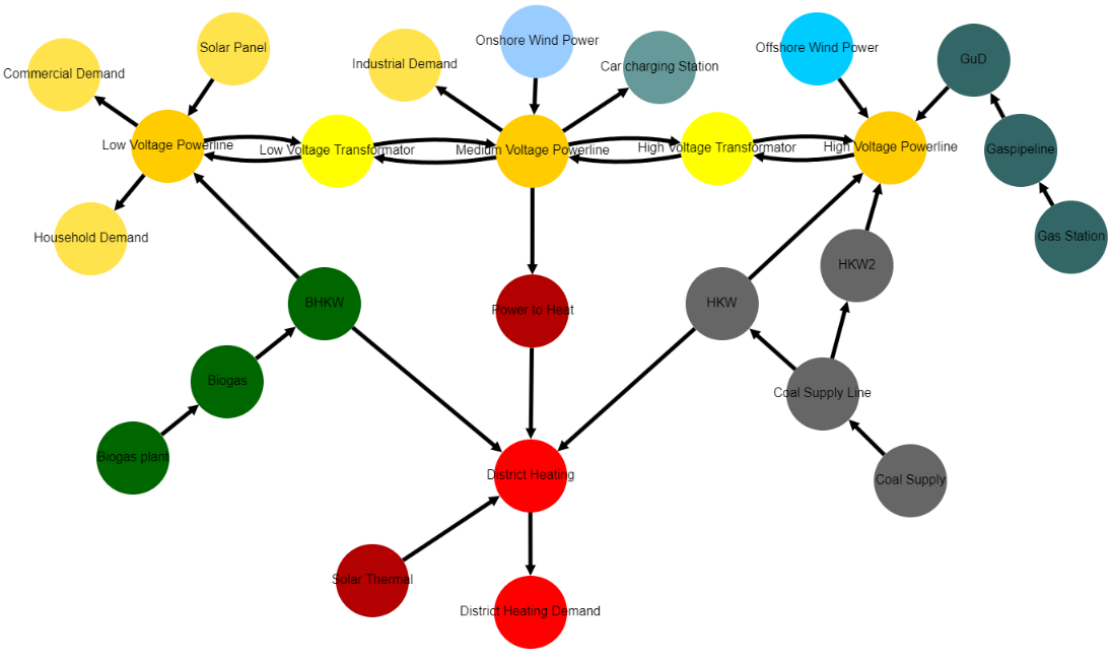


Abbildung A.9.: Energy System Grid „Kupferplatte“

Component Energy System

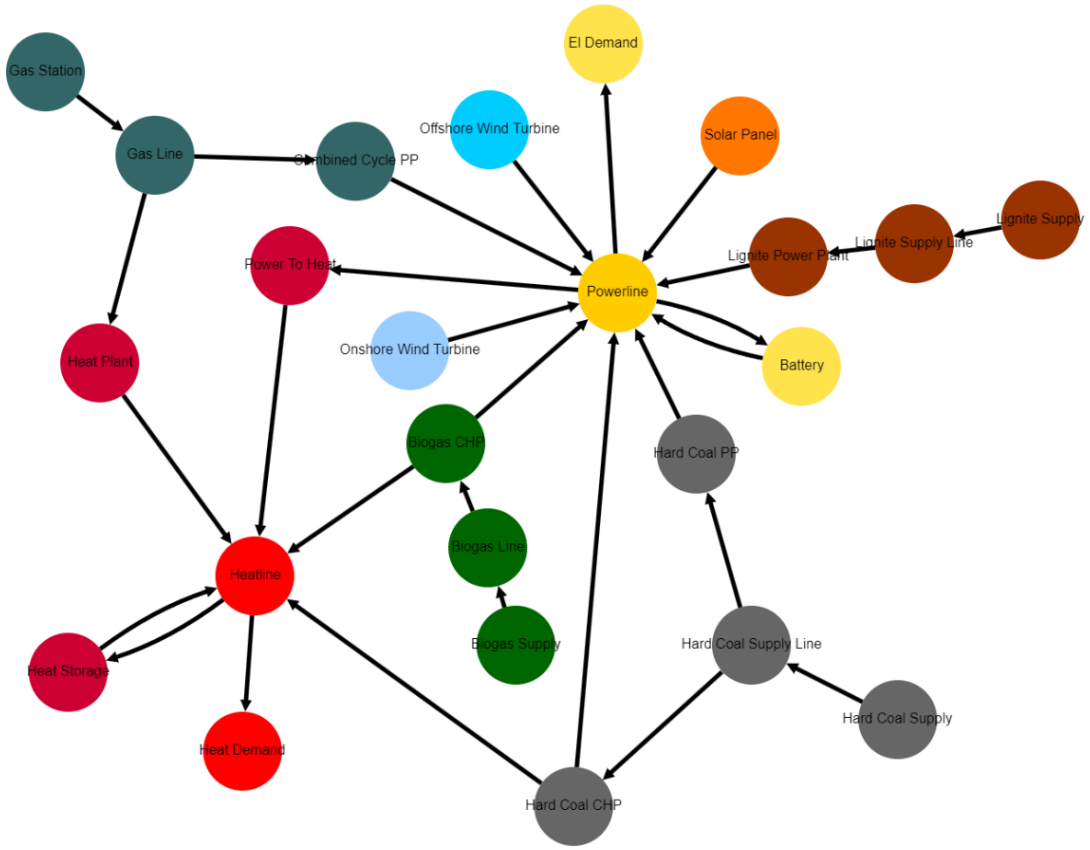


Abbildung A.10.: Component Energy System

A.5. Optimierungsergebnisse

A.5.1. Automatischer Vergleich

Minimal Working Example

		cllp	fine	omf	ppsa	rbs
Battery	Powerline	10.0	0.0	10.0	10.0	0.0
Gas Station	Pipeline	71.43	95.24	71.43		95.24
Generator	Powerline	30.0	40.0	30.0	30.0	40.0
Pipeline	Generator	71.43	95.24	71.43		95.24
Powerline	Battery	0.0	0.0	0.0	0.0	0.0
Powerline	Demand	40.0	40.0	40.0	40.0	40.0

Tabelle A.3.: Optimierungsergebnisse des automatischen Vergleichs des Minimum Working Example

Fully Parametrized Working Example

		cllp	fine	omf	ppsa	rbs
Battery	Powerline	8.9	0.9	8.9	8.9	8.0
Gas Station	Pipeline	7.38	26.43	7.38		31.19
Generator	Powerline	3.1	11.1	3.1	3.1	13.1
Pipeline	Generator	7.38	26.43	7.38		31.19
Powerline	Battery	0.0	1.0	1.0	1.0	10.1
Powerline	Demand	33.0	33.0	33.0	33.0	33.0
Solar Panel	Powerline	21.0	22.0	22.0	22.0	22.0

Tabelle A.4.: Optimierungsergebnisse des automatischen Vergleichs des Fully Parametrized Working Example

Emission Objective

		cllp	fine	omf	ppsa	rbs
Gas Grid	Gas Plant	33.33	33.33	33.33		33.33
Gas Plant	Powerline	20.0	20.0	20.0	20.0	20.0
Gas Source	Gas Grid	33.33	33.33	33.33		33.33
Gas Station	Pipeline	1.21	1.21	1.21		1.21
Generator	Powerline	0.51	0.51	0.51	0.51	0.51
Pipeline	Generator	1.21	1.21	1.21		1.21
Powerline	Demand	40.0	40.0	40.0	40.0	40.0
Wind Power	Powerline	19.49	19.49	19.49	19.49	19.49

Tabelle A.5.: Optimierungsergebnisse des automatischen Vergleichs des Emission Objective Examples

Connected Energy System

		cllp	fine	omf	ppsa	rbs
bus-01	connector	5.56	5.56	5.56	5.0	5.56
bus-01	sink-01	25.0	25.0	25.0	25.0	25.0
bus-02	connector	6.25	6.25	6.25	10.0	6.25
bus-02	sink-02	25.0	25.0	25.0	25.0	25.0
connector	bus-01	5.0	5.0	5.0	10.0	5.0
connector	bus-02	5.0	5.0	5.0	5.0	5.0
source-01	bus-01	25.56	25.56	25.56	20.0	25.56
source-02	bus-02	26.25	26.25	26.25	30.0	26.25

Tabelle A.6.: Optimierungsergebnisse des automatischen Vergleichs des Connected Energy System

Combined Heat and Power

		cllp	fine	omf	ppsa	rbs
Backup Heat	Heat Grid	13.33	13.33	13.33	13.33	13.33
Backup Power	Powerline	0.0	0.0	0.0	0.0	0.0
CHP	Heat Grid	26.67	26.67	26.67	26.67	26.67
CHP	Powerline	40.0	40.0	40.0	40.0	40.0
Gas Grid	CHP	133.33	133.33	133.33	133.33	133.33
Gas Source	Gas Grid	133.33	133.33	133.33	133.33	133.33
Heat Grid	Heat Demand	40.0	40.0	40.0	40.0	40.0
Powerline	Power Demand	40.0	40.0	40.0	40.0	40.0

Tabelle A.7.: Optimierungsergebnisse des automatischen Vergleichs des CHP Examples

Storage Exampe

		cllp	fine	omf	ppsa	rbs
Generator	Powerline	57.0	57.0	57.0	57.0	57.0
Powerline	Demand	47.0	47.0	47.0	47.0	47.0
Powerline	Storage	30.0	30.0	30.0	30.0	30.0
Storage	Powerline	20.0	20.0	20.0	20.0	20.0

Tabelle A.8.: Optimierungsergebnisse des automatischen Vergleichs des Storage Examples

Expansion Plan Example

		cllp	fine	omf	ppsa	rbs
Capped Renewable	Powerline	13.0	13.0	13.0	13.0	13.0
Emitting Source	Powerline	20.0	20.0	20.0	20.0	20.0
Powerline	Demand	40.0	40.0	40.0	40.0	40.0
Uncapped Renewable	Powerline	7.0	7.0	7.0	7.0	7.0

Tabelle A.9.: Optimierungsergebnisse des automatischen Vergleichs des Expansion Plan Examples

Component Energy System

		cllp	fine	omf	ppsa	rbs
Battery	Powerline	0.0	0.0	0.0	0.0	0.0
Biogas CHP	Heatline	0.0	0.0	0.0	0.0	0.0
Biogas CHP	Powerline	0.0	0.0	0.0	0.0	0.0
Biogas Line	Biogas CHP	0.0	0.0	0.0	0.0	0.0
Biogas Supply	Biogas Line	0.0	0.0	0.0	0.0	0.0
Combined Cycle PP	Powerline	0.0	0.0	0.0	0.0	0.0
Gas Line	Combined Cycle PP	0.0	0.0	0.0		0.0
Gas Line	Heat Plant	0.0	0.0	0.0		0.0
Gas Station	Gas Line	0.0	0.0	0.0		0.0
Hard Coal CHP	Heatline	301.68	301.68	301.68	301.68	301.68
Hard Coal CHP	Powerline	301.68	301.68	301.68	301.68	301.68
Hard Coal PP	Powerline	0.0	0.0	0.0	0.0	0.0
Hard Coal Supply	Hard Coal Supply Line	754.19	754.19	754.19	754.19	754.19
Hard Coal Supply Line	Hard Coal CHP	754.19	754.19	754.19	754.19	754.19
Hard Coal Supply Line	Hard Coal PP	0.0	0.0	0.0		0.0
Heat Plant	Heatline	0.0	0.0	0.0	0.0	0.0
Heat Storage	Heatline	0.0	0.0	0.0	0.0	0.0
Heatline	Heat Demand	301.68	301.68	301.68	301.68	301.68
Heatline	Heat Storage	0.0	0.0	0.0	0.0	0.0
Lignite Power Plant	Powerline	828.37	828.37	828.37	828.37	828.37
Lignite Supply	Lignite Supply Line	2070.92	2070.92	2070.92		2070.92
Lignite Supply Line	Lignite Power Plant	2070.92	2070.92	2070.92		2070.92
Offshore Wind Turbine	Powerline	0.0	0.0	0.0	0.0	0.0
Onshore Wind Turbine	Powerline	1362.72	1362.72	1362.72	1362.72	1362.72
Power To Heat	Heatline	0.0	0.0	0.0	0.0	0.0
Powerline	Battery	0.0	0.0	0.0	0.0	0.0
Powerline	El Demand	2492.76	2492.76	2492.76	2492.76	2492.76
Powerline	Power To Heat	0.0	0.0	0.0	0.0	0.0
Solar Panel	Powerline	0.0	0.0	0.0	0.0	0.0

Tabelle A.10.: Optimierungsergebnisse des automatischen Vergleichs des Component Energy Systems

Energy System Grid „Kupferplatte“

		cllp	fine	omf	ppsa	rbs
BHKW	District Heating	19562.57	13825.85	13825.85	13825.85	13825.85
BHKW	LV Powerline	12414.71	8774.1	8774.1	8774.1	8774.1
Biogas	BHKW	37620.32	26588.17	26588.17	26588.17	26588.17
Biogas plant	Biogas	37620.32	26588.17	26588.17	26588.17	26588.17
Coal Supply	Coal Supply Line	1459132.3	1459132.3	1459132.3	1459132.3	1459132.3
Coal Supply Line	HKW	980729.72	990290.92	990290.92	990290.92	990290.92
Coal Supply Line	HKW2	478402.59	468841.39	468841.39	468841.39	468841.39
District Heating	District Heating Demand	865459.85	865459.85	865459.85	865459.85	865459.85
Gas Station	Gaspipeline	197523.52	206773.07	206773.07		206773.07
Gaspipeline	GuD	197523.52	206773.07	206773.07		206773.07
GuD	HV Powerline	116538.88	121996.11	121996.11	121996.11	121996.11
HKW	District Heating	588437.83	594174.55	594174.55	594174.55	594174.55
HKW	HV Powerline	235375.13	237669.82	237669.82	237669.82	237669.82
HKW2	HV Powerline	205713.11	201601.8	201601.8	201601.8	201601.8
HV Powerline	HV Transformator	867448.65	871089.26	871089.26	871089.26	871089.26
HV Transformator	HV Powerline	0.0	0.0	0.0	0.0	0.0
HV Transformator	MV Powerline	867448.65	871089.26	871089.26	871089.26	871089.26
LV Powerline	Commercial Demand	582459.56	582459.56	582459.56	582459.56	582459.56
LV Powerline	Household Demand	552564.84	552564.84	552564.84	552564.84	552564.84
LV Powerline	LV Transformator	111125.15	111125.15	111125.15	111125.15	111125.15
LV Transformator	LV Powerline	640388.02	644028.63	644028.63	644028.63	644028.63
LV Transformator	MV Powerline	111125.15	111125.15	111125.15	111125.15	111125.15
MV Powerline	Car charging Station	37026.34	37026.34	37026.34	37026.34	37026.34
MV Powerline	HV Transformator	0.0	0.0	0.0	0.0	0.0
MV Powerline	Industrial Demand	1229008.36	1229008.36	1229008.36	1229008.36	1229008.36
MV Powerline	LV Transformator	640388.02	644028.63	644028.63	644028.63	644028.63
MV Powerline	Power to Heat	172017.51	172017.51	172017.51	172017.51	172017.51
Offshore Wind Power	HV Powerline	309821.53	309821.53	309821.53	309821.53	309821.53
Onshore Wind Power	MV Powerline	1099866.42	1099866.42	1099866.42	1099866.42	1099866.42
Power to Heat	District Heating	172017.51	172017.51	172017.51	172017.51	172017.51
Solar Panel	LV Powerline	593346.83	593346.83	593346.83	593346.83	593346.83
Solar Thermal	District Heating	85441.94	85441.94	85441.94	85441.94	85441.94

Tabelle A.11.: Optimierungsergebnisse des automatischen Vergleichs des Energy System Grids

A.5.2. Dispatch Problem

Powerline	Urbs	Oemof	PyPSA
Battery	-0.0	-0.0	-0.0
Biogas CHP	-0.0	-0.0	-0.0
Combined Cycle PP	4.642665865000001	4.607566989000001	4.607566989000001
Hard Coal CHP	164.29335712	164.32845598	164.32845588
Hard Coal PP	133.876786868	139.875699604	140.717970898
Lignite Power Plant	312.640205569	312.640205579	312.640205569
Offshore Wind Turbine	-0.0	-0.0	-0.0
Onshore Wind Turbine	258.4738177	258.4738177	258.4738177
Solar Panel	11.015106989	5.016194187	4.173922882000001
Battery	-0.0	-0.0	-0.0
El Demand	-884.9419399999999	-884.9419399999999	-884.9419399999999
Power To Heat	-0.0	-0.0	-0.0

Tabelle A.12.: Optimierungsergebnisse des Stromsektors des Dispatch Problems

Heatline	Urbs	Oemof	PyPSA
Biogas CHP	-0.0	-0.0	-0.0
Hard Coal CHP	164.29335710000004	164.32845598999998	164.32845588
Heat Plant	6.108252678500001	6.108252678500001	6.108252678500001
Heat Storage	0.5097631532	0.5097631532	0.5097631532
Power To Heat	-0.0	-0.0	-0.0
Heat Demand	-170.31774723	-170.31774730800004	-170.31774723
Heat Storage	-0.5936256334	-0.62872451451	-0.62872451451

Tabelle A.13.: Optimierungsergebnisse des Wärmesektors des Dispatch Problems

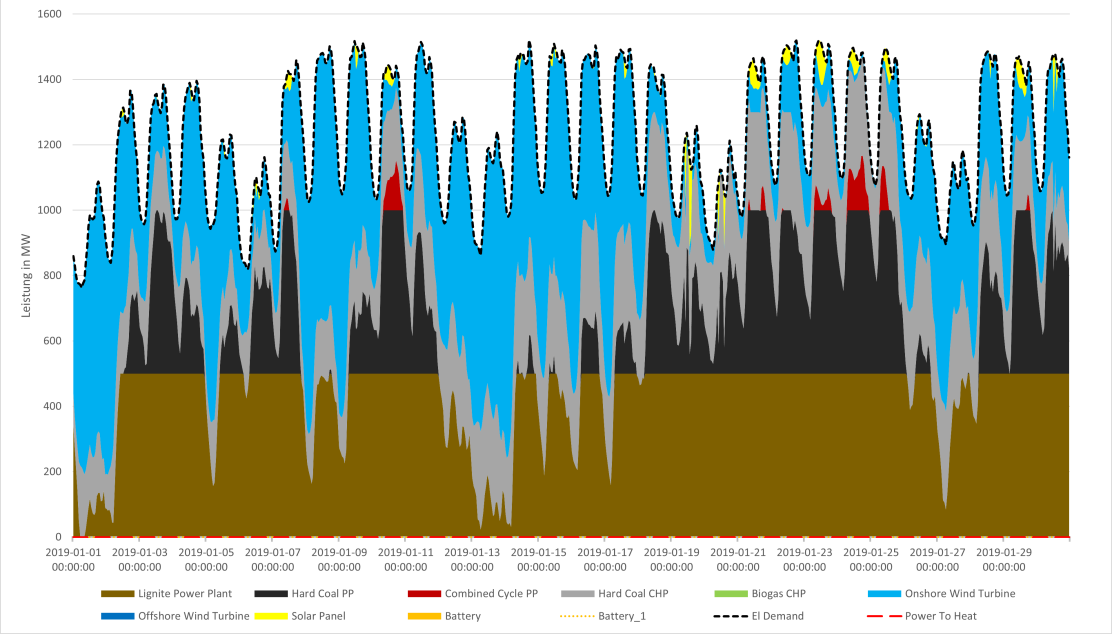


Abbildung A.11.: Optimierungsergebnisse des Stromsektors des Dispatch Problems mit Oemof

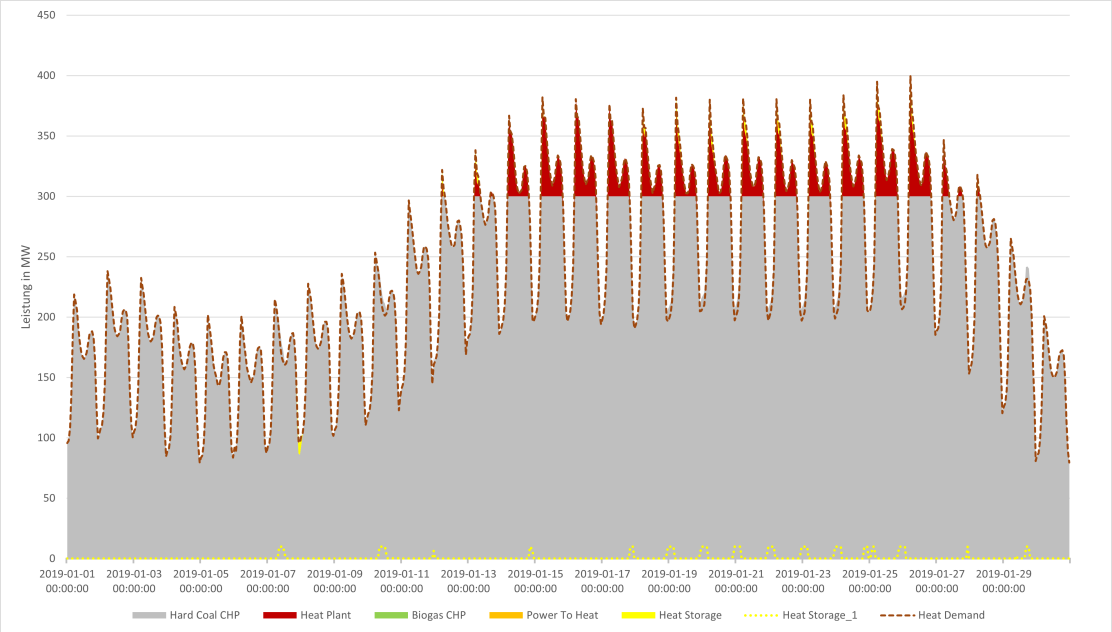


Abbildung A.12.: Optimierungsergebnisse des Wärmesektors des Dispatch Problems mit Oemof

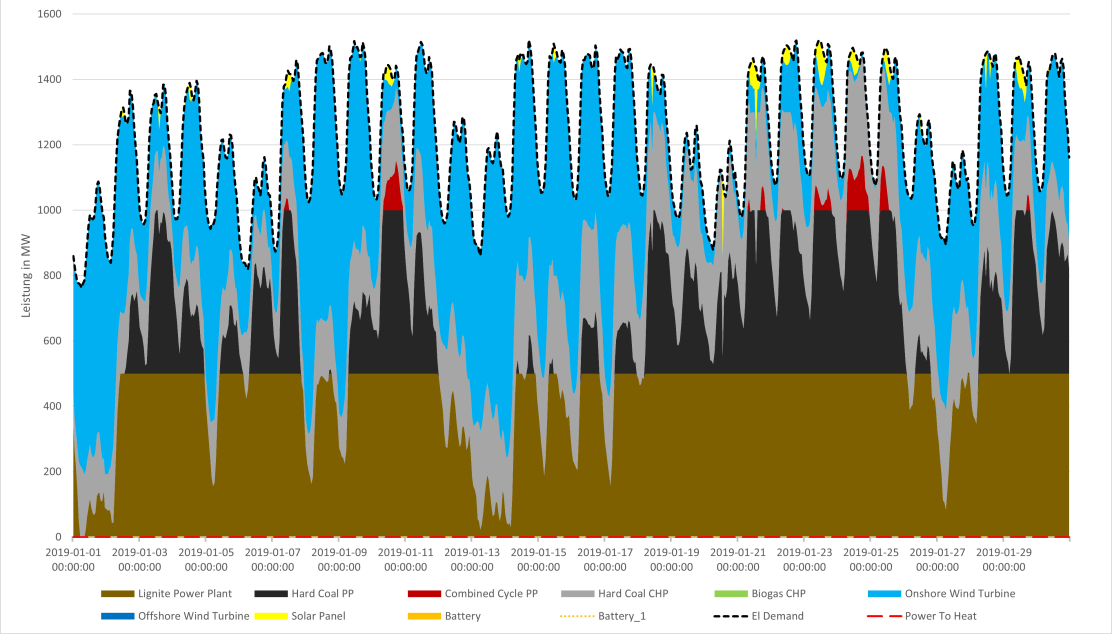


Abbildung A.13.: Optimierungsergebnisse des Stromsektors des Dispatch Problems mit PyPSA

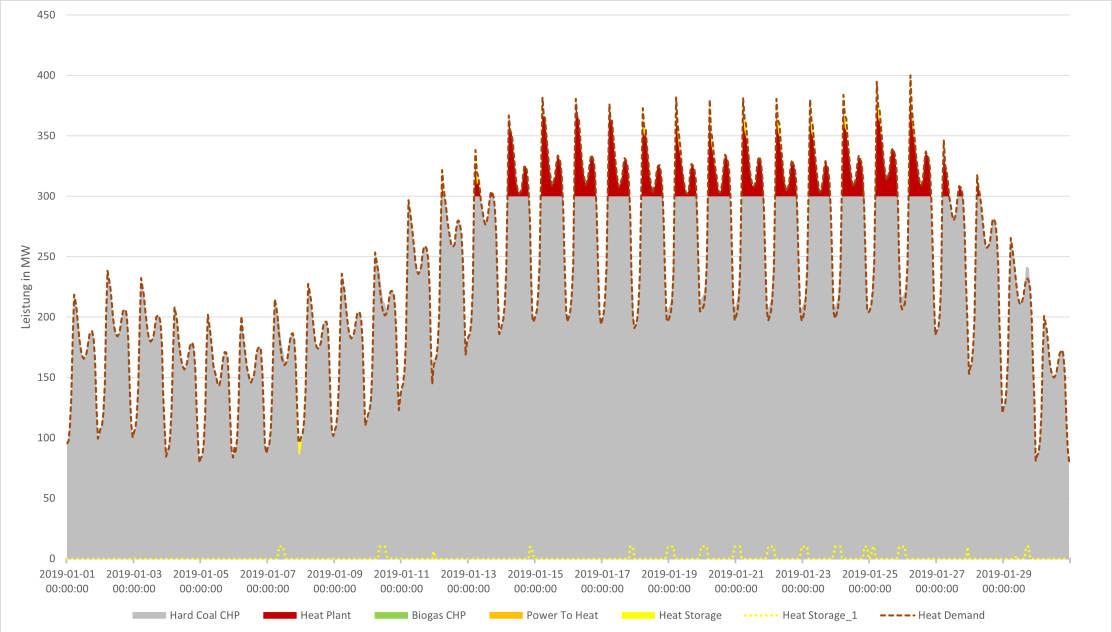


Abbildung A.14.: Optimierungsergebnisse des Wärmesektors des Dispatch Problems mit PyPSA

A.5.3. Expansion Problem

Powerline	Urbs	Oemof	PyPSA
Battery	1.0060240271	0.73514042279	1.0594815902
Biogas CHP	88.3066035947	88.37669606470001	89.37070056920001
Combined Cycle PP	23.917993480300005	23.743500670399996	23.339505179
Hard Coal CHP	-0.0	-0.0	-0.0
Hard Coal PP	0.07233401500000002	0.07077421799999999	0.070909798
Lignite Power Plant	-0.0	-0.0	-0.0
Offshore Wind Turbine	52.16667260638	55.293959964	55.31719385
Onshore Wind Turbine	797.7042136220001	794.581582191	793.44077335
Solar Panel	7.3713601772	7.351590699	7.359699804999999
Battery	-1.16973252904	-0.86259980462	-1.3180242653
El Demand	-884.9419399999999	-884.9419399999999	-884.9419399999999
Power To Heat	-84.433528578	-84.34870646200001	-83.69830219680001

Tabelle A.14.: Optimierungsergebnisse des Stromsektors des Expansion Problems

Heatline	Urbs	Oemof	PyPSA
Biogas CHP	110.38325451120002	110.4708702969	111.71337571150002
Hard Coal CHP	-0.0	-0.0	-0.0
Heat Plant	-0.0	-0.0	-0.0
Heat Storage	40.98802986731	41.28711068178	41.661096663399995
Power To Heat	83.589193351	83.5052195872	82.861319174832
Heat Demand	-170.31774723	-170.31774730800004	-170.31774723
Heat Storage	-64.64273000457	-64.94545328699999	-65.9180446049

Tabelle A.15.: Optimierungsergebnisse des Wärmesektors des Expansion Problems

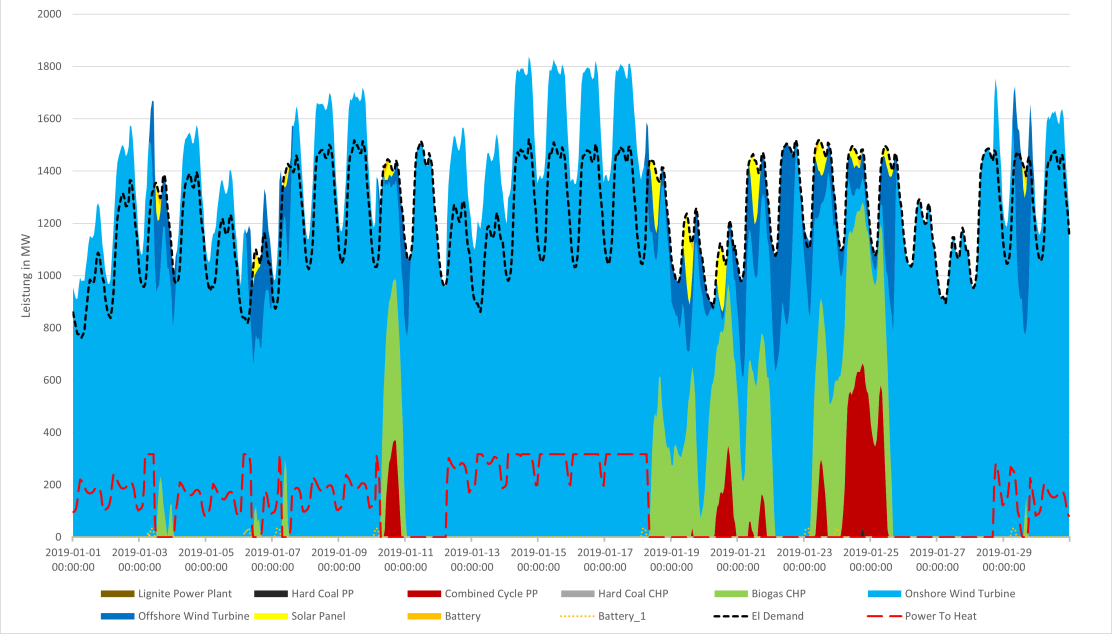


Abbildung A.15.: Optimierungsergebnisse des Stromsektors des Expansion Problems mit Oemof

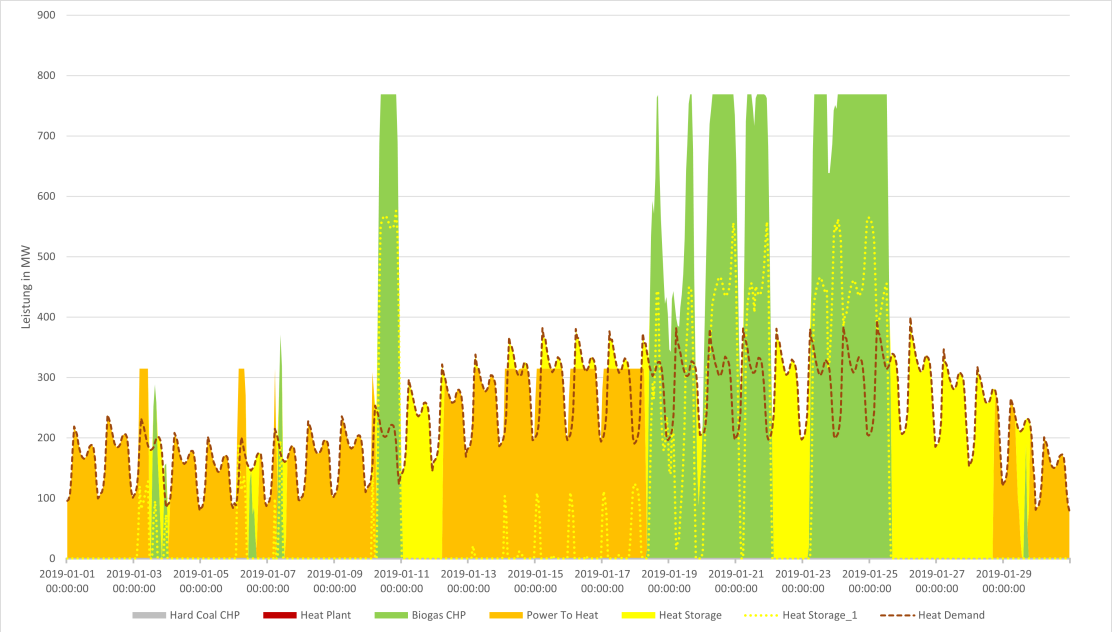


Abbildung A.16.: Optimierungsergebnisse des Wärmesektors des Expansion Problems mit Oemof

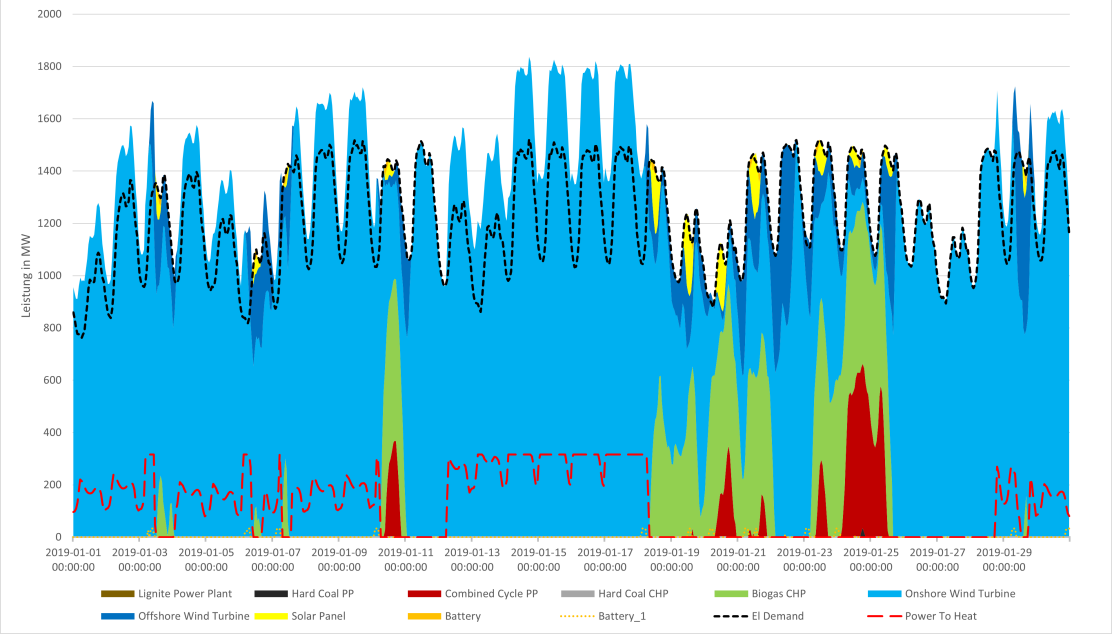


Abbildung A.17.: Optimierungsergebnisse des Stromsektors des Dispatch Problems mit PyPSA

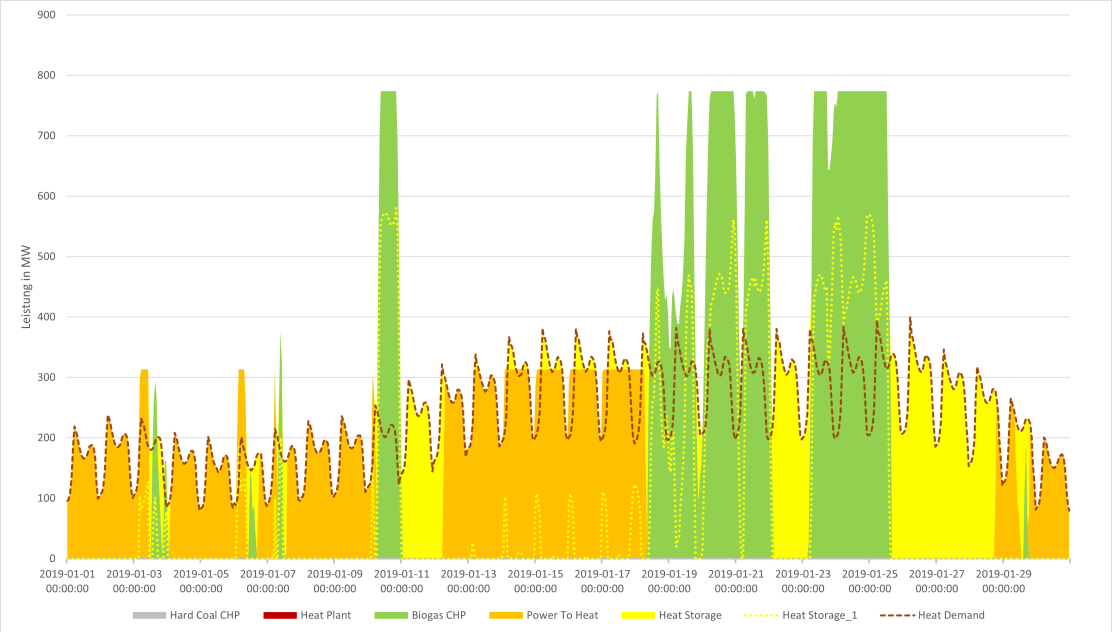


Abbildung A.18.: Optimierungsergebnisse des Wärmesektors des Expansion Problems mit PyPSA

A.5.4. Demand Side Management

Powerline	Urbs ohne DSM	Urbs mit DSM
Battery	1.66969245135	0.2803897778
Biogas CHP	121.36839545800001	121.43167260800001
Combined Cycle PP	348.5854418107	378.88041694200007
Hard Coal CHP	10.5365259694	10.2731488968
Hard Coal PP	81.64466717450001	52.6625831251
Lignite Power Plant	0.6420707426000001	-0.0
Offshore Wind Turbine	47.920072499999996	47.920072499999996
Onshore Wind Turbine	258.6662232	258.6662232
Solar Panel	15.847040000000002	15.847040000000002
Battery	-1.8927371706580003	-0.34337548739199997
El Demand	-884.9419399999999	-885.6181715512299
Power To Heat	-0.045452067	-0.0

Tabelle A.16.: Optimierungsergebnisse des Stromsektors mit und ohne Demand Side Management

Heatline	Urbs ohne DSM	Urbs mit DSM
Biogas CHP	151.71049431799997	151.78959071900005
Hard Coal CHP	10.5365259694	10.2731488968
Heat Plant	8.5499600952	8.8028289432
Heat Storage	3.5559199076300003	3.6474551429499993
Power To Heat	0.044997546	-0.0
Heat Demand	-170.31774723	-170.31774723
Heat Storage	-4.080150528	-4.195276413599999

Tabelle A.17.: Optimierungsergebnisse des Wärmesektors mit und ohne Demand Side Management