Technische Universität Berlin
Energieversorgungsnetze und Integration
Erneuerbarer Energien

Abschlussbericht zum Projekt

Mixed integer stochastic linear programming

Vorgelegt von:
Name Daniele Berardo
Matr.-Nr. 379925

Name Imad Abu Abed
Mat.-Nr. 329543

Vorgelegt am:
31. Juli 2016

Betreuer:
Sebastian Wildenhues

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1 Grundlagen

In diesem Kapitel soll die stochastische Programmierung bzw. Optimierung erläutert werden. Dabei wird auf die optimale Entscheidungsfindung unter Unsicherheit sowie Modelle und Lösungsverfahren der stochastischen Programmierung eingegangen. Abschließend erfolgt ein Vergleich zu deterministischen Problemformulierung.

1.1 Stochastische Programmierung

Die stochastische Optimierung ist ein Gebiet bestehend aus Modellen und Methoden zur optimalen Entscheidungsfindung bei unsicheren Eingangsdaten. Das Modell wird durch Zufallsvariablen mit bekannten Zufallsverteilungen abgebildet. Das folgende Beispiel soll diese Eigenschaft verdeutlichen:

Beispiel:*Ein Landwirt hat eine gewisse Fläche zur Verfügung, auf der er unterschiedliche Getreidesorten (z.B. Hafer, Weizen und Roggen) anbauen kann. Der Ertrag, den er durch den Anbau einer Sorte erwirtschaftet, ist sowohl von er Größe der Fläche, als auch vom Wetter abhängig.*
Es sei angenommen, dass der Landwirt eine bestimmte Menge von jeder Getreidesorte für den Eigenbedarf benötigt. Nach der Ernte hat er die Möglichkeit Überschüsse, die er über den Eigenbedarf benötigt hinaus produziert hat, u verkaufen, genauso kann er Fehlmeldungen, die bei einer Produktion unterhalb des Eigenbedarfs auftreten durch den Nachverkauf von Getreide ausgleichen.

In diesem Beispiel stellt das Wetter die Zufallsvariable dar, weil dem Landwirt dieses Ereignis nicht im Voraus bekannt ist. Des Weiteren können die Preise für den Verkauf bzw. Nachkauf von Getreide unbekannt sein. Der Einfachheit halber soll in dieser Arbeit die Unsicherheit nur auf das Wetter beschränkt werden [1].
Die unterschiedlichen Wetterlagen werden in Szenarien eingeteilt. Jedes Szenario bekommt eine bekannte Eintrittswahrscheinlichkeit zugeordnet.
Ziel der stochastischen Optimierung in diesem Beispiel ist es die Anbauflächen so zu wählen, dass die erwarteten Erträge über alle möglichen Wetterentwicklungen optimiert werden.
Anhand des Beispiels lässt sich die typische Struktur von stochastischen Problemen erkennen. Das Problem lässt sich in zwei Stufen unterteilen:

1. Stufe*: Der Landwirt entscheidet sich, wie groß die Anbauflächen für die einzelnen Getreidesorten gewählt werden und bestellt die Felder entsprechend.*

2. Stufe:*Ernte zeigt sich, wie viel tatsächlich von welchem Getreide geerntet wurde. Mit dieser Information kann der Landwirt die Fehlmengen nach- bzw. die Überschüsse verkaufen.*

Das heißt also, dass nach der ersten Entscheidung (1. Stufe), welche Flächen mit welchem Getreide angebaut werden, die Realisierung der zufälligen Ereignisse (Wetter) eintritt. Nach der Beobachtung der Realisierung (geerntete Menge) wird in der zweiten Stufe entschieden, ob und wenn ja, welche Handlungen nötig sind, um das Ziel (Eigenbedarf abdecken) zu erreichen. Dieses zweistufige Verfahren wird in Abbildung 1verdeutlicht [1].

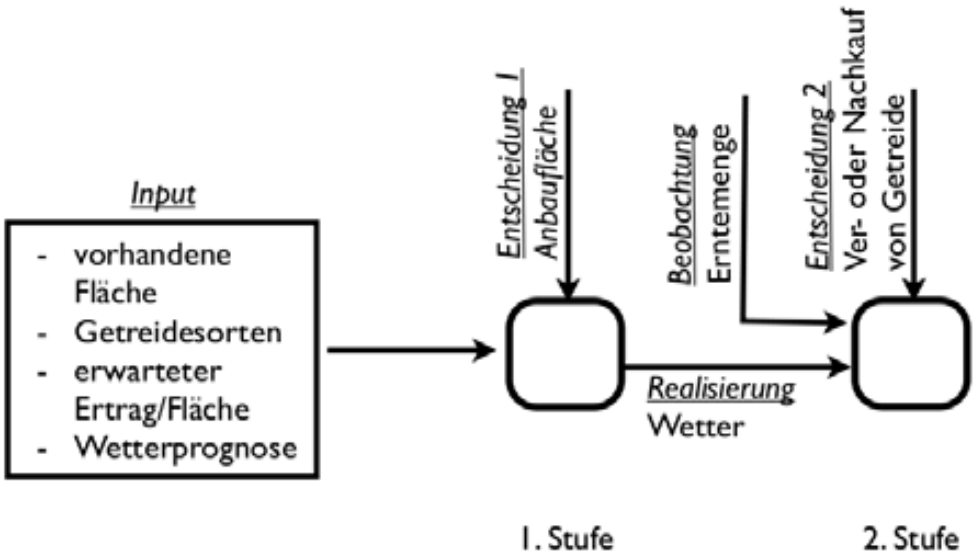

Abbildung 1: Entscheidungen in einem zweistufigen stochastischen Problem am Farmerbeispiel

Das Problem, das sich in diesem Beispiel ergibt, wird als zweistufiges stochastisches Problem bezeichnet. Es gibt aber auch stochastische Probleme, die über mehrere Stufen gehen. Dies ist immer dann der Fall, wenn die Ereignisse in einer zeitlichen Abfolge auftreten.

In einem Problem mit mehr als zwei Stufen lassen sich Entscheidungen nach eintreten eines Ereignisses neu treffen. Dadurch fallen zwar erneut Kosten an, allerdings besteht die Chance auf eine bessere Lösung des Optimierungsproblems. Bezogen auf das Farmerbeispiel würde es beuteten, dass der Landwirt sein Ackerland neu sähen kann, nachdem ein Unwetter die erste Saat zerstört. Durch den zusätzlichen Arbeitsaufwand und des neuen Saatgutes fallen zwar zusätzliche Kosten an, aber der Landwirt könnte eine bessere Ernte erzielen.

Es verschiebt sich die getroffene Entscheidung aus der zweiten Stufe in die dritte Stufe, bei der es gilt wieder eine Entscheidung zu treffen in Abhängigkeit unvorhersehbarer Wetterereignisse. Bei mehrstufigen Modellen ergibt sich wiederholt die Situation, dass die Realisierungeines Teils der zufälligen Ereignisse beobachtet wird, danach neue Entscheidungengetroffen werden, bevor der nächste Teil der zufälligen Ereignissebeobachtet werden kann.Abbildung 2 verdeutlicht diesen Ablauf:

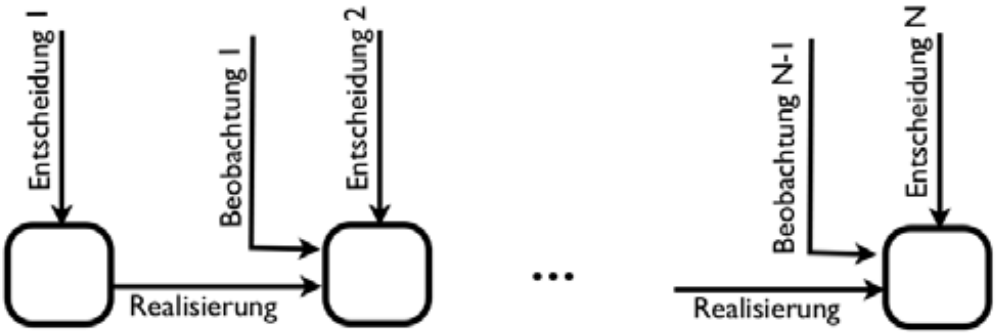

Abbildung 2: Entscheidung in einem mehrstufigen stochastischen Problem

1.1.1 Entscheidugsformulierung

Die Realisierung der zufälligen Ereignisse lässt sich beim zweistufigen Modell wie in Abbildung 3 dargestellt verstehen.

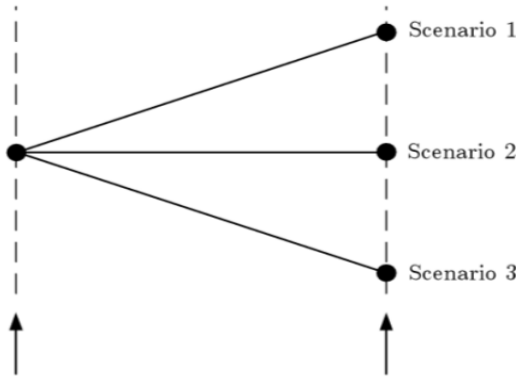

Abbildung 3: Zweistufiger Entscheidungsbaum

Eine solche Darstellung wird Entscheidungsbaum genannt. Die grafische Darstellung als Baumdiagramm veranschaulicht hierarchisch aufeinander folgende Entscheidungen. Bei gleichen Entscheidungen in der ersten Stufe können bei unterschiedlicher Realisierung der zufälligen Ereignisse verschiedene Zustände in der zweiten Stufe erreicht werden. Es tritt eines der möglichen Szenarios für die Parameter ein, daraus resultiert der Zustand in der zweiten Stufe.

Für den mehrstufigen Fall ergibt sich ein Entscheidungsbaum, wie in Abbildung 4 dargestellt [1].

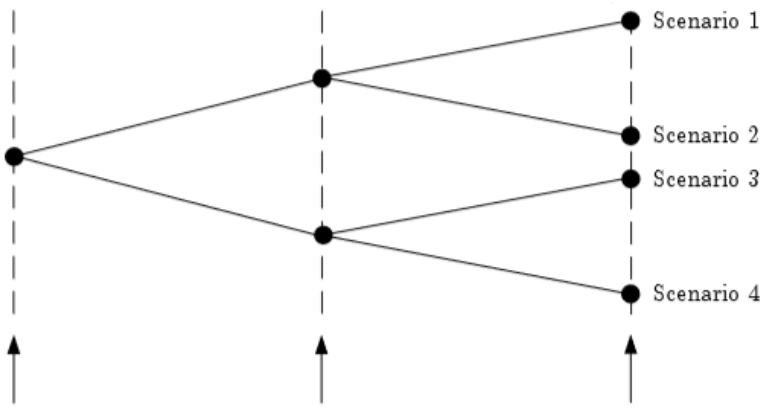

Abbildung 4: Mehrstufiger Entscheidungsbaum

Während des Entscheidungsprozesses, werden zwei unterschiedliche Arten unterschieden, um mit Unsicherheit in Entscheidungen umzugehen:

1. „First-stage"-or „here-and-now"-Entscheidungen:

Die Lösung des „here-and-now"-Problems (HN) liefert die zum Zeitpunkt der Entscheidungsfindung optimale Entscheidung, die für jedes mögliche Szenario zulässig ist. Diese Art von Entscheidung wird vor Beginn des stochastischen Prozesses getroffen. Optimalität wird meist über den Erwartungswert der Zielfunktion über alle Szenarien definiert. Es sind aber auch andere Optimalitätsbegriffe denkbar, beispielsweise, wenn Risikomaße mitberücksichtigt werden sollen. Dann entsteht ein multikriterielles Optimierungsproblem.

2. „Second-stage"-or „wait-and-see"-Entscheidungen:

Die Grundannahme dieses Ansatzes ist es, dass der Entscheidungsträger die Möglichkeit hat, solange abzuwarten, bis sich die Unsicherheit aufgelöst hat, und dann die optimale Entscheidung zu treffen. Das „wait-and-see"-Problem besteht also darin, die Verteilung der optimalen Zielfunktionswerte des deterministischen Modells für jedes mögliche Szenario zu berechnet. Diese Art von Entscheidung wird, anders als bei der HN Entscheidung, erst getroffen, nachdem der stochastische Prozess realisiert worden ist. Der Erwartungswert dieser Verteilung heißt „wait-and-see"-Wert (WS). Der Nachteil des WS-Problems besteht darin, dass es keine Planungslösung liefert, die zum eigentlichen Zeitpunkt der Entscheidungsfindung umsetzbar ist.

Durch den Vergleich von WS-, und HN-Wert kann der sog. Wert der vollständigen Information (EVPI) und der Wert der stochastischen Lösung (VSS) berechnet werden.

Diese Werte geben Aufschluss über den Mehrwert, der durch eine explizite Berücksichtigung der Unsicherheit bei der Entscheidungsfindung erzielt wird.In dieser Arbeit ist der VSS von Bedeutung [2].

1.1.1.1 Value oft the Stochastic Solution

Der „Value of the Stochastic Solution“ist ein quantitatives Maß, um den Vorteil der stochastischen Programmierung gegenüber der deterministischen aufzuzeigen. Bei deterministischen Problemen in Verbindung mit stochastischen Größen werden die Zufallsvariablen des stochastischen Teils ersetzt durch den jeweiligen Erwartungswert. Die Lösung dieses deterministischen Problems liefert eine optimale Lösung für die Variablen der ersten Stufe. Das ursprüngliche stochastische Problem kann dann durch die Bestimmung der Werte der erst-stufen Variablen gelöst werden. Das Problem lässt sich in mehreren Szenarien unterteilen und ist somit einfach zu lösen.
Der Wert der stochastischen Lösung lässt wie folgt berechnen

$$VSS_{max} = z^{S*} - z^{D*},$$

bzw. für Minimierungsprobleme gilt

$$VSS_{min} = z^{D*} - z^{S*}.$$

Der optimale Wert der Zielfunktion des modifizierten stochastischen Problems wird durch z^{D*} beschrieben, wobei der Exponent D für deterministisch steht. z^{S*} bezeichnet den optimalen Wert der Zielfunktion des ursprünglichen Problems [3].

1.1.2 Modelle und Lösungsverfahren der stochastischen Programmierung

Die Modelle und Lösungsverfahren der stochastischen Optimierung beschäftigen sich vorwiegend mit dem HN-Problem. Die verschiedenen Modellklassen ergeben sich aus der Typologie der mathematischen Programmierung (linear, nicht-linear, ganzzahlig etc.), aus der zeitlichen Struktur der Entscheidungsfindung und der Zulässigkeitsdefinition der Lösung. Die wichtigsten Modellklassen sind:
Zwei- und mehrstufige stochastische Programme mit Kompensation – Bei diesem Modelltyp, der auf [4] zurückgeht, werden die Entscheidungsvariablen denjenigen diskreten Zeitpunkten innerhalb des Planungszeitraumes zugeordnet, an denen die Entscheidungsfindung stattfindet. Entscheidungen der Stufe 1 müssen immer zu Beginn des Planungszeitraumes getroffen werden und sind dann für alle weiteren Stufen fixiert. Zweistufige lineare stochastische Programme können für eine diskrete Menge von Szenarien in vielen Fällen effizient gelöst werden. Die verwendeten Lösungsverfahren bauen dabei auf Dekompositionsverfahren der linearen Programmierung auf (Benders-Dekomposition). Mehrstufige-, ganzzahlige und nicht-lineare stochastische Programme

sind oft nicht oder nur unter hohem Aufwand (spezialisierte Lösungsverfahren) für praktisch relevante Problemgrößen lösbar.
Stochastische Programme mit probabilistischen Nebenbedingungen – Bei diesem von Charnes und Cooper [5] entwickelten Modelltyp wird die Forderung fallengelassen, dass die Lösung für alle Nebenbedingungen und möglichen Szenarien zulässig sein muss. Stattdessen können sog. probabilistische Nebenbedingungen definiert werden, die in einem Teil der Szenarien verletzt sein dürfen. Die effiziente Lösbarkeit dieses Modelltyps hängt davon ab, ob ein effizient lösbares äquivalentes deterministisches Modell abgeleitet werden kann. Ist dies nicht der Fall, dann sind diese Modelle sehr schwer lösbar.

Diese Arbeit beschäftigt sich mit der mehrstufigen stochastischen gemischt-ganzzahligen linearen Programmierung.

1.1.2.1 Mehrstufige stochastische gemischt-ganzzahlige lineare Optimierung

In einigen Fällen ist eine zweistufige Optimierung in Entscheidungsproblemen nicht ausreichend. Deshalb macht es durchaus Sinn, das Problem auf ein mehrstufiges zu übertragen. Ein allgemeines Beispiel für ein stochastisches Entscheidungsproblem mit r-Stufen kann wie folgt formuliert werden:

1. Entscheidung x^1 wird getroffen.
2. Das Ereignis λ^1 wird realisiert bei $\lambda^1(\omega^1)$.
3. Entscheidung $x^2(x^1, \omega^1)$ wird getroffen.
4. Das Ereignis λ^2 wird realisiert bei $\lambda^2(\omega^2)$.
5. Entscheidung $x^3(x^1, \omega^1, x^2, \omega^2)$ wird getroffen.

2r-2. Das Ereignis λ^{r-1} wird realisiert bei $\lambda^{r-1}(\omega^{r-1})$.

2r-1. Entscheidung $x^r(x^1, \dots, x^{r-1}, \omega^{r-1})$ wird getroffen.

Dieses Beispiel orientiert sich an den Entscheidungsbaum, wie er bereits in Abbildung 4 beschrieben wurde.
Um eine angemessene Problemformulierung zu finden, müssen die Kausalitätsbedingungen zwischen den Variablen berücksichtigt werden.
In stochastischen Problemen, besonders bei mehrstufigen Entscheidungsproblemen ist es wichtig die Kausalitätsbedingung zu berücksichtigen. Eine Variable wird als kausal bezeichnet, wenn die Realisierung des Ereignisses der Stufe k, identisch zu den Entscheidungsvariablen in der Stufe k sind. Die Kausalität nimmt schon in der ersten Entscheidungsstufe x^1 Einfluss. x^1 ist unabhängig gegenüber jeder Realisation des zukünftigen Ereignisses $\{\lambda^1, \dots, \lambda^{r-1}\}$. In der nächsten Stufe ist x^2 abhängig von der

Realisation des Ereignisses aus der ersten Stufen λ^1. Folglich wird x^2 als „wait-and-see"-bzgl. λ^1und als „here-and-now"-Entscheidung bzgl. $\{\lambda^2, \dots, \lambda^{r-1}\}$ bezeichnet.

Es gibt unterschiedliche Methoden um ein gemischt-ganzzahliges lineares Problem zu lösen. Um solch ein Problem nummerisch lösen zu können, werden sogenannte Solver, wie z.B. CPLEX verwendet.
In dieser Arbeit soll näher auf Progressive Hedging (PH) eingegangen werden, da mit Hilfe des PH-Algorithmus das stochastische Problem gelöst werden soll [5].

1.1.2.2 Progressive Hedging

Progressive Hedging ist eine Technik zur Lösung linearer, diskreter stochastischer Programme. Es eignet sich besonders gut bei Problemen mit großem Maßstab, was in der Regel bei stochastischen Problemen relativ schnell der Fall ist. Beim PH-Algorithmus wird das Problem in mehren kleinen Problemen zerlegt und anschließend für jedes Szenario individuell gelöst. Abbildung 5 veranschaulicht dieses Funktionsprinzip [6].

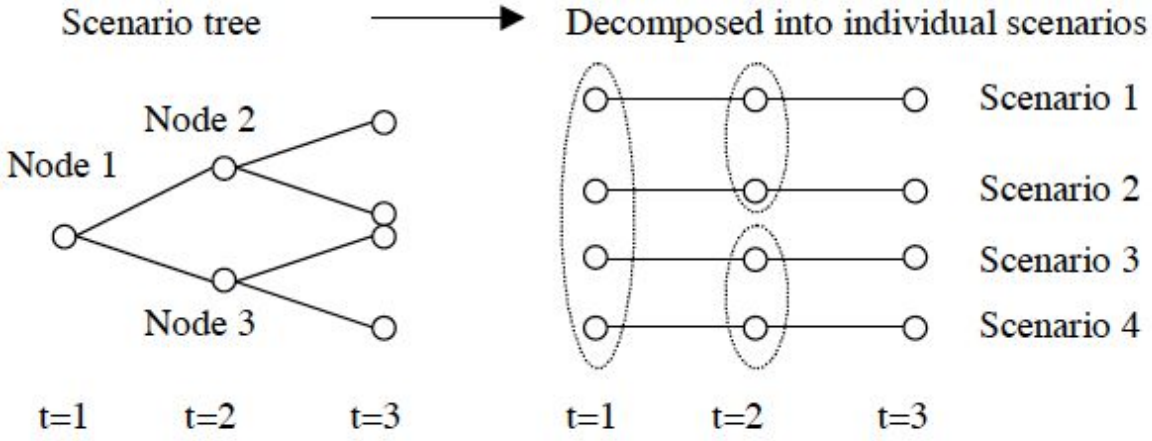

Abbildung 5: Funktionsprinzip des PH-Algorithmus

1.1.2.3 Computerbasierte Umsetzung von Progressive Hedging

Um den PH-Algorithmus zu implementieren, kann Pyomo verwendet werden. Pyomo ist eine extra dafür vorgesehene Open Source Software, die ein Bestandteil des Python Software Pakets bildet. Ebenso im Python Software Pakets enthalten, ist PySP, wodurch es möglich ist mehrstufige stochastische ganzzahlige lineare Probleme zu lösen. Das geschieht mit Hilfe der Pyomo-Modellierungssprache. Das besondere an Pyomo ist, dass die Problemformulierung in eine ähnliche Art und Weise geschieht, wie sie tatsächlich mathematisch formuliert werden. Pyomo bietet eine unterschiedliche Zahl an

Komponenten an, die exakt das beschriebene Problem abbilden können. Dazu gehören u.a. Entscheidungsvariablen, Zielfunktion sowie Nebenbedingungen.
Zur Problemlösung unterstützt Pyomo freizugängliche oder kommerziell genutzte Solvers, welche von AMPL, PICO, CBC, CPLEX, IPOPT, Gurobi und GLPK unterstützt werden.

2 Test Case

Die Basis eines Fallbeispiels bildet das „IEEE Reliability Test System“ (RTS-96), welches in diesem Abschnitt näher beschrieben werden soll. Dabei wird auf die gegeben Netztopologie eingegangen, sowie auf die Modifikationen die durchgeführt wurden, um ein entsprechenden Fallzu erzeugen.

2.1 Netztopologie

Abbildung 5 zeigt ein System bestehen aus zwei RTS-96 Zonen [7] (Zone A und B), die durch folgende drei Querverbindungen miteinander verbunden sind:

- Verbindung 1: Bus 23 und Bus 17
- Verbindung 2: Bus 13 und Bus 15
- Verbindung 3: Bus 7 und 3

Das Netz besteht aus insgesamt 48 Busse. Die Busse 1-24 gehören der Zone A an, Bus 25-48 bilden die Zone B.

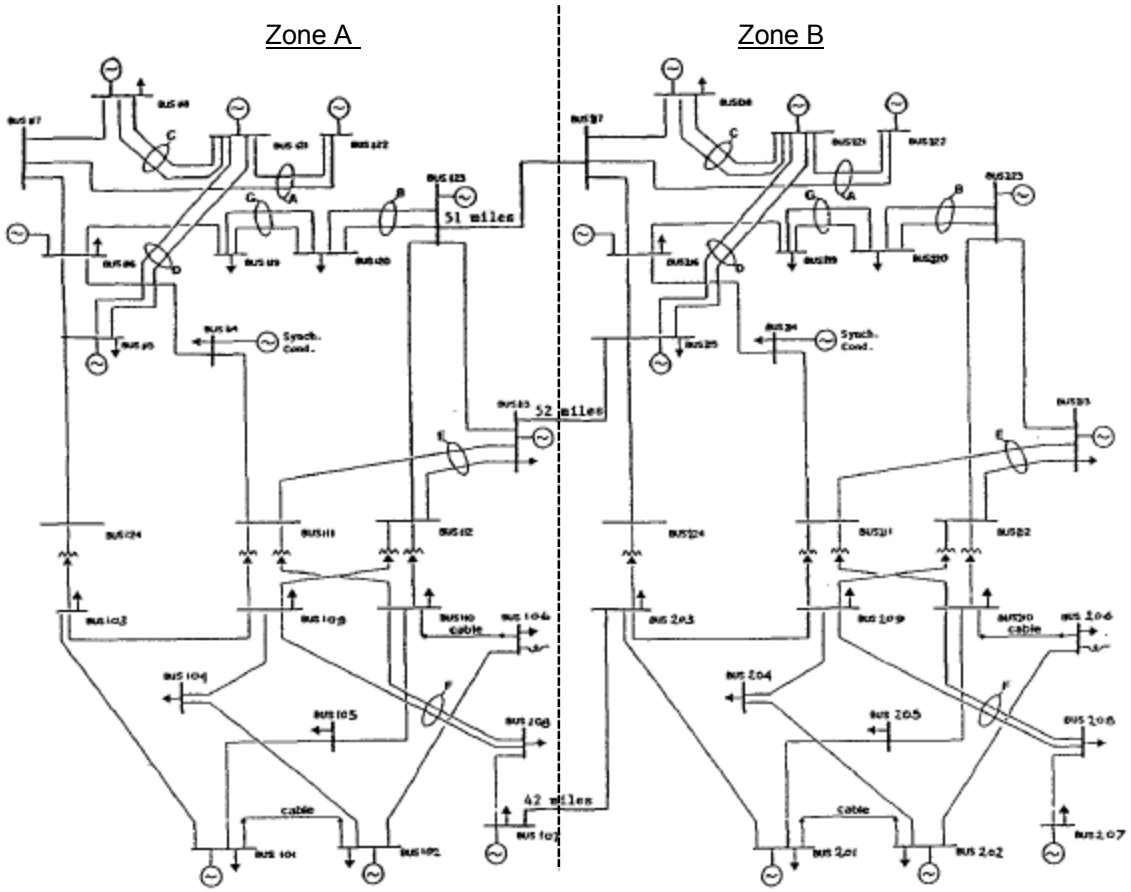

Abbildung 6: IEEE 2-Zonen RTS-96 (Zone A und B)

Des Weiteren sind folgende Daten zur Beschreibung der Bussenotwendig:

#	Bus Name	Bus_P_gen	Bus_P_load	Bus_area	Bus_cost	Bus_P_max	Bus_P_min
1	Bus1	172	108	1	29,3	1	0,2
2	Bus2	172	97	1	29,3	1	0,2
3	Bus3	0	180	1	0	1,1	0,5
4	Bus4	0	74	1	0	1,1	0,5
5	Bus5	0	71	1	0	1,1	0,5
6	Bus6	0	136	1	0	1,1	0,5
7	Bus7	240	125	1	43,6	1	0,3
8	Bus8	0	171	1	0	1,1	0,5
9	Bus9	0	175	1	0	1,1	0,5
10	Bus10	0	195	1	0	1,1	0,5
11	Bus11	0	0	1	0	1,1	0,5
12	Bus12	0	0	1	0	1,1	0,5
13	Bus13	285,3	265	1	48,6	1	0,7
14	Bus14	0	194	1	0	1,1	0,5
15	Bus15	215	317	1	24,7	1	0,3
16	Bus16	155	100	1	12,4	1	0,3
17	Bus17	0	0	1	0	1,1	0,5
18	Bus18	400	333	1	4,42	1	0,25
19	Bus19	0	181	1	0	1,1	0,5
20	Bus20	0	128	1	0	1,1	0,5
21	Bus21	400	0	1	4,42	1	0,25
22	Bus22	300	0	1	0,001	1	1
23	Bus23	660	0	1	12,1	1	0,4
24	Bus24	0	0	1	0	1,1	0,5
25	Bus25	172	108	2	29,3	1	0,2
26	Bus26	172	97	2	29,3	1	0,2
27	Bus27	0	180	2	0	1,1	0,5
28	Bus28	0	74	2	0	1,1	0,5
29	Bus29	0	71	2	0	1,1	0,5
30	Bus30	0	136	2	0	1,1	0,5
31	Bus31	240	125	2	43,6	1	0,3
32	Bus32	0	171	2	0	1,1	0,5
33	Bus33	0	175	2	0	1,1	0,5
34	Bus34	0	195	2	0	1,1	0,5
35	Bus35	0	0	2	0	1,1	0,5
36	Bus36	0	0	2	0	1,1	0,5
37	Bus37	285,3	265	2	48,6	1	0,7
38	Bus38	0	194	2	0	1,1	0,5
39	Bus39	215	317	2	24,7	1	0,3
40	Bus40	155	100	2	12,4	1	0,3
41	Bus41	0	0	2	0	1,1	0,5
42	Bus42	400	0	2	4,42	1	0,25
43	Bus43	0	181	2	0	1,1	0,5
44	Bus44	0	128	2	0	1,1	0,5
45	Bus45	400	0	2	4,42	1	0,25

46	Bus46	300	0	2	0,001	1	1
47	Bus47	660	0	2	12,1	1	0,4
48	Bus48	0	0	2	0	1,1	0,5

Tabelle 1: Test Case Bus-Daten (2 Zonen)

In der Tabelle 1 sind die Werte dargestellt, die zur Berechnung des optimalen Leistungsflusses verwendet wurden.
Die Werte an den Bussen 41, 42 und 46 wurden gegenüber dem RTS-96 verändert. An diesen Bussen erfolgt eine Energieerzeugung ausschließlich aus Windkraftanlagen.
Zu den Bus-Daten sind zusätzlich noch die Branch-Daten notwendig, welche als Excel-Datei mit dem Namen „myIEEE_48_branches.xlsx“ hinterlegt sind.

2.2 Wettereinfluss

Die Stromerzeugung aus Windkraft ist stark wetterabhängig. Zuverlässige meteorologische Vorhersagen für die Steuerung der Stromnetze und zur Festlegung des Strompreises sind daher unverzichtbar.
Im späteren Verlauf dieser Arbeit ergeben sich dadurch unterschiedliche Szenarien, die im Kapitel 4betrachtet werden.

2.2.1 Windvorhersage

Die Energieerzeugung durch Windkraftanlagen ist abhängig von der Windstärke. Je stärker der Wind ist, desto größer ist die Energieerzeugung. In der Praxis ist es wichtig, die erwartete Energieerzeugung im Vorfeld zu wissen, um am Vortag an der Börse den Strompreis zu verhandeln.
Abbildung 6 zeigt den Verlauf der prognostizierten und der tatsächlichen Energieerzeugung aus Windkraft für einen Zeitraum von 24 Stunden. Die Werte wurden aus der Ursprungsquelle um ca. 12% reduziert, um es an die Größenverhältnisse des RTS-96 anzugleichen.
In der Grafik ist zu erkennen, dass die prognostizierten Werte unterschiedlich stark von den tatsächlichen Werten abweichen.

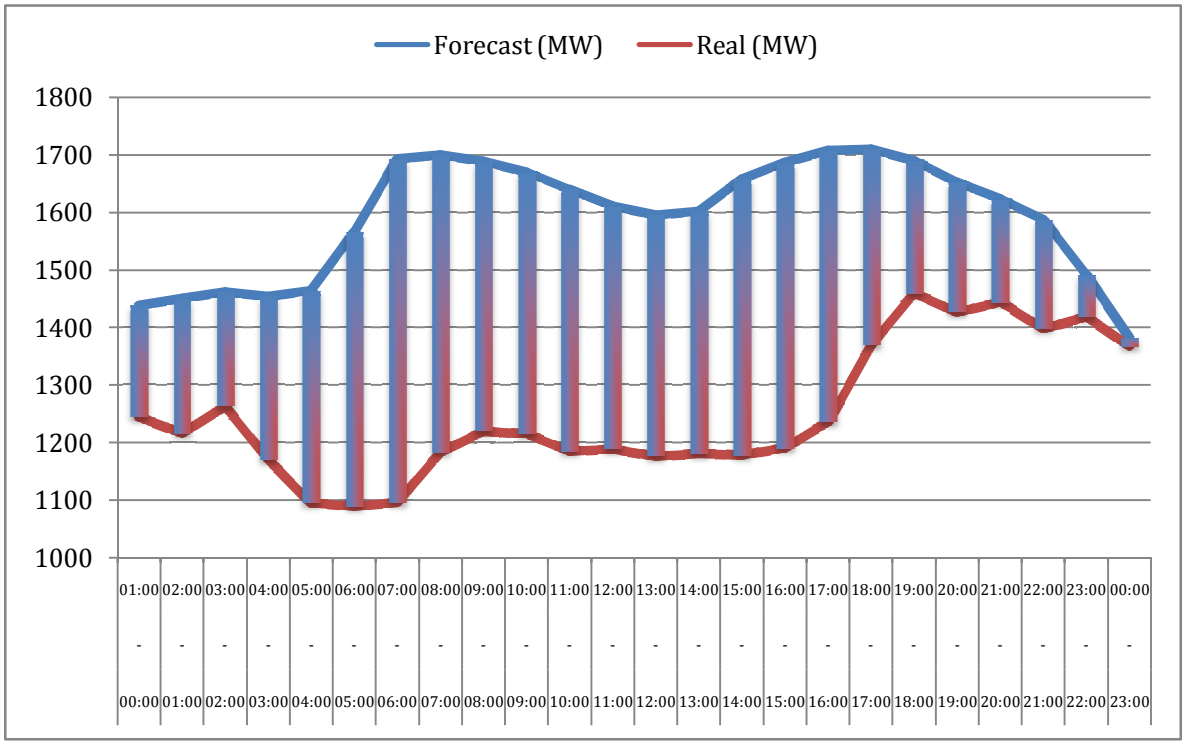

Abbildung 7: Prognostizierte Erzeugung vs. Ist-Erzeugung[1]

Auf Basis dieses Fehlers ergeben sich Ungenauigkeiten, die Einfluss auf die optimale Lösung hat. Um diese Abweichung zu berücksichtigen wurde der durchschnittliche Fehler ermittelt, wenn die Prognose 24, sechs und drei Stunden im Voraus passiert. Die Werte sind in Tabelle 2 dargestellt.

Prognose im Voraus (h)	24	6	3
Durchschnittlicher Fehler (%)	+- 21,16%	+-10,53%	+- 6,92%

Tabelle 2: Durchschnittlicher Fehler bei Prognose

Es ist zu erkennen, dass die Abweichung zwischen prognostizierter und Ist-Erzeugung deutlich kleiner wird. Im Umkehrschluss beutetet dies, dass eine genauere Prognose der zukünftigen Stromerzeugnisse durch Windkraftanlagen möglich ist.

2.2.2 Strompreis

Der Strompreis und die resultierenden Kosten der Energieerzeugung sind vom Wetter abhängig. Dies gilt insbesondere für die Windkraftanlagen.

Wird an windstarken Tagen viel aus Windenergie erzeugter Strom eingespeist, sinkt der Großhandelspreis an der Strombörse. Ist wenig Windenergie vorhanden, steigt der Preis an der Strombörse. Die Strompreissenkung durch Windenergie entsteht durch die gesetzliche Abnahmepflicht für produzierten Windstrom. Ist viel Strom aus Windenergie verfügbar, wird

[1]Daten stammen aus www.50hertz.com (01.12.2014)

der Einsatz teurer konventioneller Kraftwerke, was zu einem Absinken der Preise an der Strombörse führt.

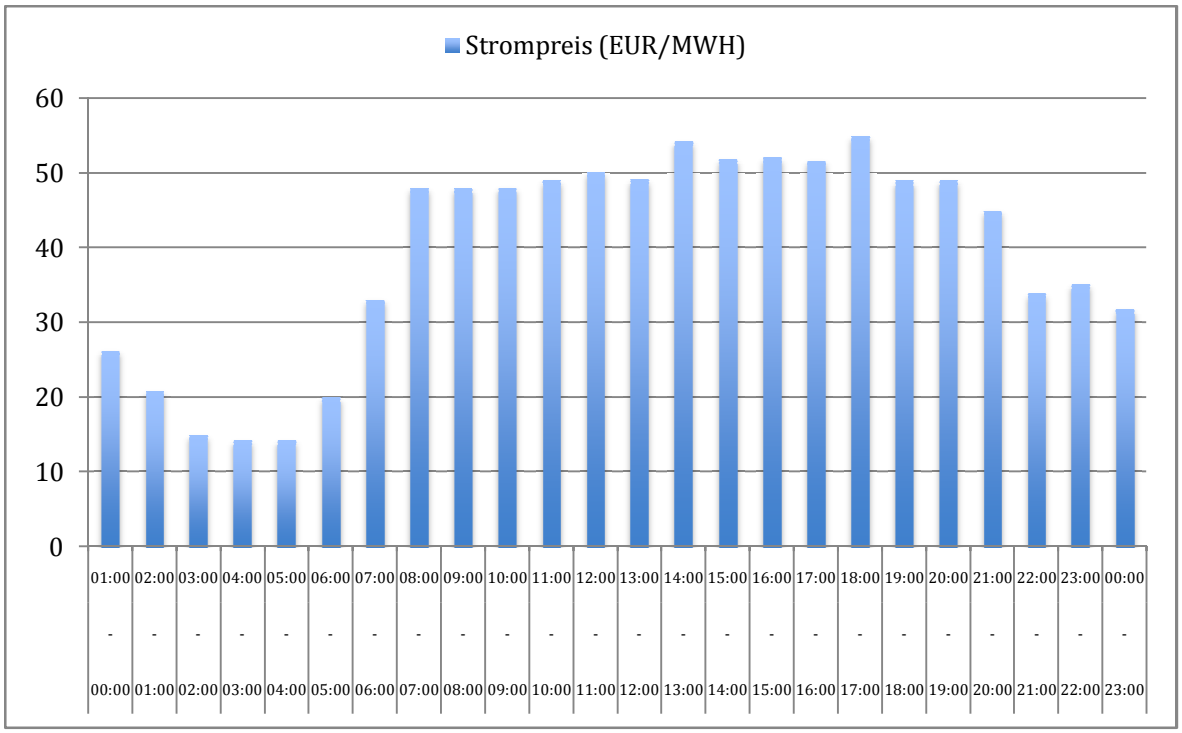

Abbildung 8: Day-Ahead Strompreise[2]

Abbildung 7 zeigt die stündlichen Strompreise. Verglichen mit der Abbildung 6 wird der Zusammenhang zwischen Erzeugung und Strompreis noch einmal verbildlicht.

Die Preise müssen in diesem Fallbeispiel berücksichtig werden, weil diese erheblichen Einfluss auf die Kosten haben.

[2]Daten stammen aus www.epexspot.com (01.12.2010)

3 Problemformulierung

3.1 Vorstellung des Problems

In diesem Kapitel wird die Aufgabestellung in Form eines Optimierungsproblems erläutert. Hierbei handelt es sich um die Beseitigung von Leitungsüberlastungen in einem elektrischen Netz mit starker Durchdringung erneuerbarer Energieanlagen (Windparks). Dies erfolgt unter Berücksichtigung von Abweichungen bei den Wetterprognosen.
Da der Anfangszustand des Netzes sich aus einer Optimierung der Generatorleistung ergibt, werden zunächst die Grundlagen dieses Algorithmus im Wesentlichen erwähnt.

3.1.1 Optimaler Lastfluss

Die allgemeine Form der DC-Lastfluss-Gleichung aufgelöst nach dem Vektor der Knotenleistungen P_{inj} lautet:

$$P_{inj} = \boldsymbol{B'} \cdot \delta \qquad 3\text{-}1$$

mit $B'_{nm} = -\frac{1}{x_{nm}}$ und $B'_{nn} = \sum_{m\in\Omega_n} -\frac{1}{x_{nm}}$

In Gleichung **3-1** sind die Größen x_{nm} und δ jeweils die Admittanz der Leitung zwischen den Knoten *n* und *m*, und der Vektor der Spannungswinkel aller Netzknoten.
In Abhängigkeit von der gegebenen Netztopologie und den Einspeisungsprofilen stellt sich ein LastflussP_{nm} über der Verbindung zwischen Knoten *n* und *m* ein, für welchen die unten stehende Bedingung jederzeit erfüllt sein muss[3]:

$$-1 < \frac{P_{nm}}{P_{nm}^{STE}} < 1 \qquad 3\text{-}2$$

Mit dem Begriff „Optimaler DC-Lastfluss“ (DCOPF) bezeichnet man das Einspeisungsprofil in einem Netz, die als Lösung des Optimierungsproblems:

$$\min_{\mathrm{P_k}} \sum_{k\in N_G} c_k \cdot P_k \qquad 3\text{-}3$$

unter Berücksichtigung der Lastfluss-Gleichungen gilt. $N_G$kennzeichnet dabei die Menge der Generatoren im Netzwerk und $P_k$die Leistung an den jeweiligen Einspeisungspunkten. Gleichung 3-3 stellt eine vereinfachte Version eines DCOPF dar, wofür die Einspeisungskosten $c_k \cdot P_k$ linear mit der Leistung ansteigen.
Die Lösung des DCOPF kann darüber hinaus, weiteren Bedingungen (engl. Constraints) erfüllen. Unter diesen zählen die Berücksichtigung von der maximalen Generatorleistung, sowie dem konstanten Spannungswinkel δ (z.B. bei PV-Busses) und der Belastbarkeit der Leitungen (vgl. Gleichung 3-2), wodurch die Komplexität des Problems zunimmt.

[3]Der Wert P_{mn}^{STE} (STE: Short-time emergency rate) entspricht dem Betrag der maximalen Scheinleistung, die durch eine Leitung für bis zu 15 Minuten fließen darf.
Da im entworfenen Algorithmus, das minimale beobachtete Zeitintervall 3 *h* beträgt und keine Rücksicht auf Blindleistung genommen wurde, muss Bedingung 3-2 in jedem Zeitpunkt gelten.

Die Lösung zum DCOPF wurde mittels eines vorgegebenen AMPL-Skripts ermittelt und als günstigste Einspeisungskombination (Best-Case) für das untersuchte Netz betrachtet.

3.1.2 Versorgungsungewissheit durch Windenergie

Bei konventionellen Versorgungssystemen basierend auf „fossiler Energie“ und herkömmlicher Last, lassen sich durch geeignete „Market clearing“-Mechanismen [3] sowohl der Energiebedarf decken, als auch die Netzstabilität gewährleisten. Die zunehmende Durchdringung erneuerbarer Energieanlagen ins Netz, führt allerdings zu einer Erhöhung der gesamten Unvorhersehbarkeit im Versorgungssystem. Abbildung 9 zeigt beispielsweise die Wahrscheinlichkeitsfunktion der Windgeschwindigkeit an einem Ort nach der Weibull Verteilung. Es fällt dabei auf, dass ,aufgrund der Abhängigkeit der Windleistung von der dritten Potenz seiner Geschwindigkeit, ein selbst nur leicht abweichender Betrag der Windenergie für Engpässe ins Netz innerhalb weniger Stunden sorgen kann.

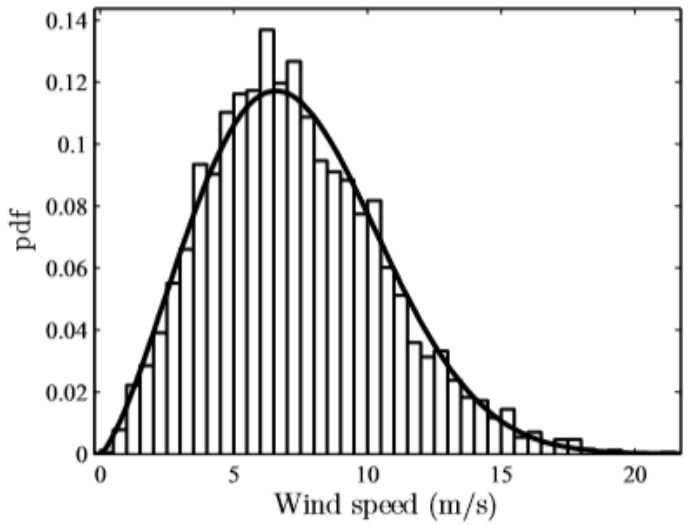

Abbildung 9: Statistische Verteilung der Windgeschwindigkeit

Zur Sicherstellung eines sicheren Netzzustandes sind Überlastungen daher schnell zu beseitigen.

Selbst wenn die Abtrennung der Windanlagen vom Netz, die eine übermäßige Leistung liefern, als die einfachste Lösung betrachtet werden könnte, ist diese, in Anbetracht der daraus entstehenden Entschädigung, die dem Anlagebetreiber zu bezahlen wäre, unerwünscht [6]. Folglich müssen weitere Ansätze vorgezogen werden, bei denen die Abtrennung von EE-Anlagen nur als Notfalllösung vorgesehen ist.

3.2 Study case

Eine weiter denkbare Lösung zu den Engpässen wäre es, alle Leitungen nach dem Worst-Case-Prinzip auf die höchstauftretende Leistungsspitze auszulegen. Das würde

allerdings unnötige Kosten verursachen. Um wirtschaftliche Verluste aufgrund einer Überdimensionierung des Betriebsmittels zu vermeiden, sind Leitungsüberlastungen „online“ beseitigen.

3.2.1 Unbilanzierte Energieerzeugung

Zwei Netze des Typs IEEE 24 wurden mit einander gekoppelt, wie in der Veröffentlichung [7] hingewiesen. Abbildung 10 (links) zeigt das sich ergebende Verbundnetzsystem, worauf ein DCOPF durchgeführt wurde [4] . Als Bezugspunkt für die Leitungsbelastungsskala wurde der STE-Wert der einzelnen Verbindungen genommen. Es fällt auf, dass das Netz im „normalen Fall“in einem erlaubten Zustand betrieben ist.

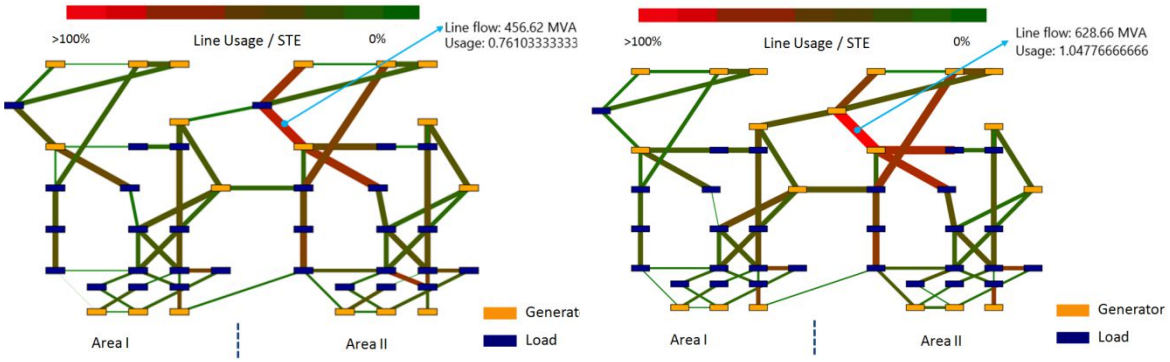

Abbildung 10: Netz im normalen Betriebzustand (links) und bei auftretenden Engpässen (rechts)

Bei der nächsten Durchführung des DCOPF wurden die Einspeisungsprofile geändert, um die Durchdringung von Windenergie ins Verbundsystem abzubilden. Das ergibt ein Netz im unsicheren Betriebszustand in dem lokale Engpässe erscheinen (rechtes Bild im obigen Diagramm). Die Generatorleistungen wurden außerdem so ausgewählt, dass es keinen Engpass auf den Verbundleitungen besteht.
Dieser Betriebszustand des Netzes gilt als Ausgangspunkt für das stochastische Optimierungsproblem.

3.2.2 Lösungsansatz

Zwei Möglichkeiten wurden für die Lösung des oben vorgestellten Problems in Betracht gezogen:

a) Drosselung der übermäßigen Windenergie an den Generator Anschlusspunkten;
b) Außerbetriebnahme einer oder mehrerer der drei Verbindungen zwischen den zwei Netzen.

Für beide Fällen müssen Entschädigungskosten angenommen werden, die nachfolgend Costsfor power variation (abk. c_{CPV}) und Contract condition violation costs (abk. c_{CCV})

[4] Für die Einspeisungsprofilen und die Leitungsmerkmale für den „normalen Fall“ sowie für den „Fall mit Engpässen“ wird auf Kapitel 3 verwiesen.

genannt werden. Die ersten beziehen sich auf die Netzknoten, während die zweiten den Verbundleitungen zugewiesen sind.

Ein mögliches Aussehen des Lastflusses nach Anwendung der Maßnahme b) wird in Abbildung 11 angezeigt.

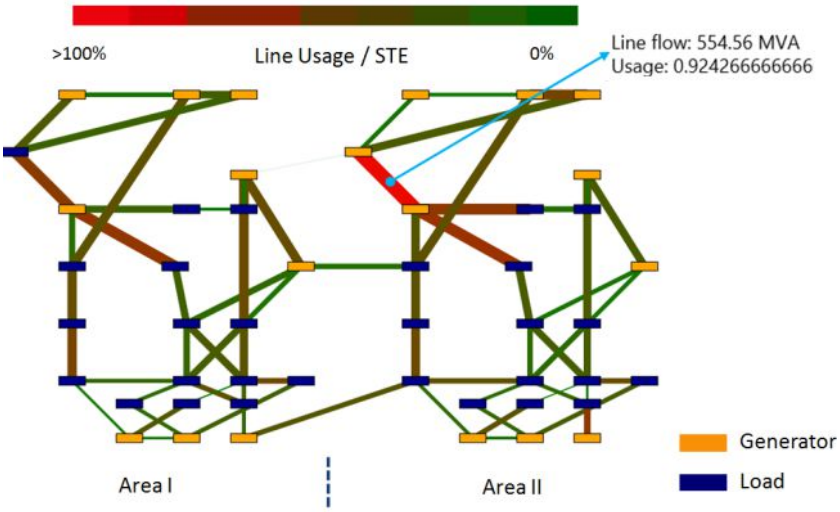

Abbildung 11: Effekt der Außerbetriebsnahme einer Verbundleitungzur Beseitigung von Engpässen

Um die Ungewissheit, die durch Windenergie entsteht, zu berücksichtigen, wurde ein beispielhaftes Zeitfenster von 6 *h*, aufgeteilt in 2 dreistündigen sog. „Stages“, gewählt, innerhalb dessen das Netz auf einem sicheren Betriebszustand zu halten ist. Zwischen den verschiedenen Stages, unterliegt die Windgeschwindigkeit einer Änderung stochastischer Natur, die weitere Leistungsschwankungen an den Anschlusspunkten zufolge haben kann.

Aus diesem Grund, muss eine besondere (weil stochastisch)Instanz des DCOPF „online[5]“ von einer zentralen Recheneinheit ausgeführt werden. Dies wird im Folgenden näher beschrieben.

3.3 Stochastische Problemformulierung

Die fluktuierende Natur der Windenergie sorgt dafür, dass die Teilnahme von den Betreibern solcher Anlagen an herkömmlichen „Redispatch“ Märkten unmöglich bzw. uninteressant ist. Die Förderung der Windenergie findet deswegen auf andere Art und Weise statt und wird gesondert reguliert [3][7].

Da die gesamte installierte Leistung aus Windkraft ein nicht vernachlässigbares Niveau in Deutschland erreicht hat, können ungünstige Wetterlagen an einzelnen Orten zu Versorgungsstörungen sämtlicher Regionen führen. Um ein sowohl wirtschaftlich, als auch technisch vorteilhaftes Energiemanagement zu entwerfen, eignet sich eine

[5]Mit dem Begriff online ist hier gemeint, dass das 6 h-Zeitfenster um mehreren Stages verschoben werden kann, um die möglichen zukünftige Betriebszustände des Netzes bei genaueren Wetterprognosen überprüfen zu können.

Untersuchung des Verhaltens der optimalen Lösung unter veränderlichen Einspeisungsbedingungen, die in Form ein stochastisches linearen Problems abgebildet werden kann.

3.3.1 Mathematische Form

Das Netz wurde über einem Zeitintervall von 6 *h*beobachtet. Dies wurde wiederum in 3 Stages aufgeteilt, nämlich der First Stage (A, t = 0 *h*), der Second Stage (B,t = 3 *h*) und der Third Stage (C, t = 6 *h*). Jedem Stage sind gewissen Variablen und Kosten zugeordnet, die als Xth-Stage variables (oder decisions) und costs bezeichnet werden. Die Zielfunktion π^S enthält die Summe aller Kosten eines Szenarios *s*, sodass das dazugehörige Optimierungsproblem in Szenario-basierte Form so lautet:

$$\min_{a^0,P_1^A,P_1^B,P_4^B,P_7^B,P_1^C,P_2^C,P_3^C,P_4^C,P_5^C,P_6^C,P_7^C,P_8^C,P_9^C} \pi^S \qquad 3\text{-}4$$

mit $\pi^S = FS_s + SS_s + TS_s = (1-a^0)c_{ccv} + \Pr(s)\cdot\sum_{k\in G}[(P_{ks}^A - P_{ks}^B)c_{ks}^B + (P_{ks}^B - P_{ks}^C)c_{ks}^C]$

und entwickelt:

$$\begin{aligned} &= (1-a^0)c_{ccv} + \sum_{k\in G}\{\Pr(s=1)\left[(P_{k1}^A - P_{k1}^B)c_{k1}^B + (P_{k1}^B - P_{k1}^C)c_{k1}^C\right] \\ &+ \Pr(s=2)\left[(P_{k2}^A - P_{k2}^B)c_{k2}^B + (P_{k2}^B - P_{k2}^C)c_{k2}^C\right] + \\ &\vdots \\ &+ \Pr(s=9)\left[(P_{k9}^A - P_{k9}^B)c_{k9}^B + (P_{k9}^B - P_{k9}^C)c_{k9}^C\right]\} \end{aligned}$$

Die bei der Formulierung 3-4 verwendeten Größen werden hier erläutert:

- P_{ks}^X: die gesamte Leistung, die vom Generator *k* bei der Realisierung von Szenario *s* am Stage *X*eingespeist wird
- a^0: der binäre „Aktivierungsvektor“, der bestimmt, welche Verbundleitungen im ersten Stage außer Betrieb (Wert 0) gesetzt werden
- $\Pr(s=n)$: die Wahrscheinlichkeit einer Realisierung von Szenario *n*
- c_{ccv}: der Kostenkoeffizient einer Contractconditonviolation (bestimmt die First stage costs)
- c_{ks}^X : die Costsfor power variation am Knoten *k* bei der Realisierung von Szenario *s* am Stage *X* (bestimmt die Second und Third stagecosts)
- G : die Menge aller Generatoren

Bei der Berechnung der obigen Zielfunktion sind weitere Nebenbendingungen zu berücksichtigen, die zum einen die Systembeschaffenheiten darstellen:

1. $a^0 \in \{0,1\}$
2. $P^{A,B,C} - \sum_{l\in L} a^0(s)\cdot b_{m,n}(\delta_m - \delta_n) \quad \forall\, s \in S$
3. $-1 \le \frac{a^0(s)\cdot b_{m,n}(\delta_m - \delta_n)}{P_{m,n}^{max}} \le 1$

4. $$P_k^A - P_k^C \leq P_k^A \forall\, k \in G$$

zum anderen die nicht Antizipativität der B- und C-Stages sicherstellen:

$$P_1^A = P_2^A = \cdots = P_9^A$$
$$P_1^B = P_2^B = P_3^B$$
$$P_4^B = P_5^B = P_6^B$$
$$P_7^B = P_8^B = P_9^B$$

Die letztenNebenbendigungen, auf Englisch „non-anticipativityconstraints“, lassen sich in grafischer Form mittels sog. Szenario Bäume (oder Scenario Trees) darstellen.

3.3.2 Scenario Tree

Der dem obigen Problem dazugehörige Scenario-Tree zeigt Abbildung 12. Von jedem Knoten des Baums in den Stages A und B sind 3 Realisierungen möglich :

1. Below Average: die Windgeschwindigkeit sinkt unter dem Durchschnittswert (Gutes Wetter)
2. Average: die Windgeschwindigkeit liegt beim Durchschnittswert
3. Above Average: Windgeschwindigkeithöher als derDurchschnittswert (Schlechtes Wetter).

Zu jeder Realisierung sind Werte der Einspeisung an der Knoten *G* zugewiesen, die bei der Implementierung als Parameter hereinzuladen sind. Die Wahrscheinlichkeit jeder Realizierung beträgt 7 bzw. 11 %. Das ergibt eine unterschiedliche Realisierungswahrscheinlichkeit $\Pr(s)$ der jeweiligen Szenarien (wie auch in Abbildung 12 zu sehen ist).

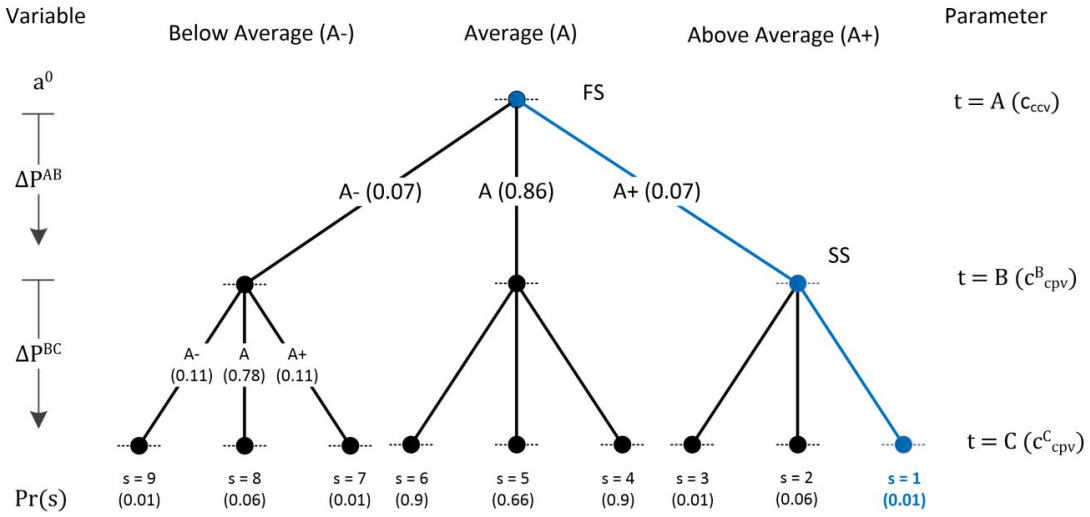

Abbildung 12 Scenario tree des analysierten Optimierungsproblems.Der hellbrau markierte Pfad kennzeichnet die Realisierung von Szenario 1

Bei der Ralisierung von Stages A nach B und von B nach C findet eine Leistungsänderung ΔP^{AB} bzw. ΔP^{BC} an den Anlagenknoten statt; diese stellen die

Optimierungsvariablen dar. Die Stage-abhängigen Parameter entsprechen den zugrunden gelegten Kosten für die FS-Decisions und für die Drosselung der Windleistung.

4 Implementierung

4.1 Problemstruktur in Pyomo

Die Implementierung eines Optimierungsproblems in Pyomo erfolgt unter Beachtung einer bestimmten Datei-Struktur. Die einzelnen Modelle und Parameter müssen in Dateien definiert werden, die nach einer gewissen Syntax zu kodieren sind und mit der richtigen Benennung abzuspeichern.

4.1.1 Deterministischer Ansatz

Der erste Schritt zur Implementierung eines Optimierungsproblems ist die Definition seines abstrakten Modells als .py Datei. `Abstract` ist ein Begriff der objekt-orientierten Programmierung, der alle Klassen umfasst, welche nur Muster und Hierarchie von ihren Attributen definieren, ohne diese zu initialisieren.
Unabhängig davon, ob es sich bei der Optimierung um ein stochastisches Problem handelt oder nicht, muss die allgemeine Formulierung einem deterministischen Ansatz folgen. Dass heißt, dass Optimierungsskripten, die in Pyomo programmiert worden sind, für jeden deterministischen Solver ausführbar ist.
Neben der Modelldefinition, muss eine zusätzliche .dat Datei angelegt werden, wodurch Werte sich den im Modell deklarierten Parametern zuweisen lassen.

4.1.2 Stochastische Erweiterung

Das Pysp Modell eines stochastischen Problems baut auf seine deterministische Formulierung in Pyomo auf, die als .py Format vorhanden sein muss. Pysp bildet dann eine stochastische Instanz des Problems, sofern die Definition eines Szenario-Baums in dem ausführenden Befehl mit eingegeben wurde.
Dabei reicht die ScenarioStructure.py Datei, wo die Struktur des Baums enthalten ist, allein nicht aus, da sie wiederum ein abstraktes Objekt ist. Aus diesem Grund müssen alle Knoten- bzw. Szenarien-bezogenen Parametersätze (in .dat Format) im selben Ordner angelegt werden, aus dem die ScenarioStructure.py Datei in Pysp geladen wird. Zudem müssen all diese Komponenten, so benannt werden, wie sie in der Baum-Struktur schon definiert worden sind.
Zur Veranschaulichung der oben genannten Beziehung (Dependencies)-Reihe dient Abbildung 13.

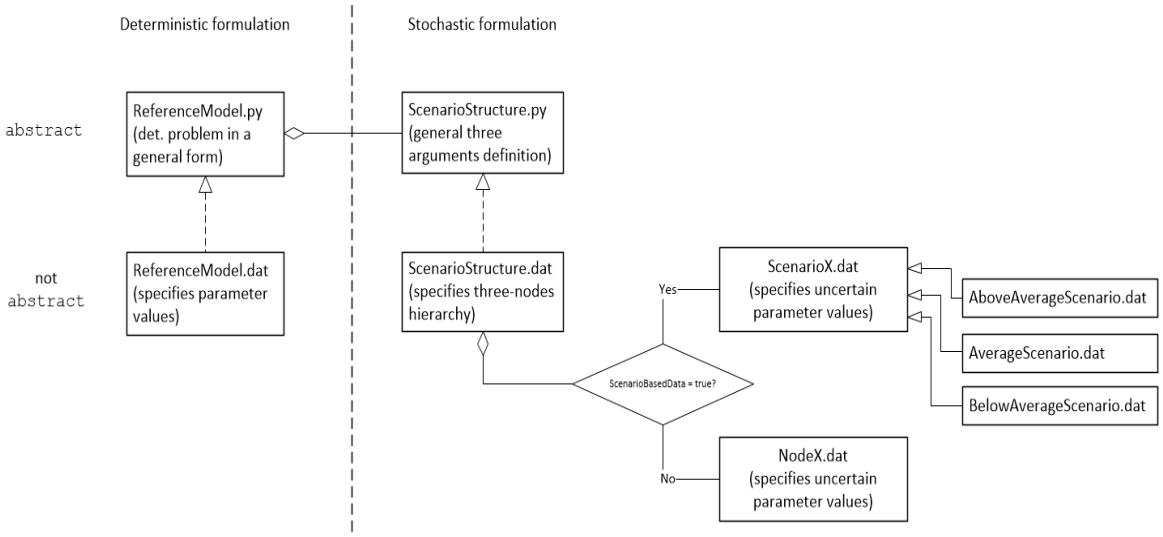

Abbildung 13 Modellaufbau eines stochastischen Optimierungsproblems in Pyomo/Pysp (Verbindungen nach UML)

4.2 Modellierung und Lösung des Optimierungsproblems

Der allererste Schritt zur Implementierung eines Problems in Pyomo ist das Einfügen von den folgenden Befehlen in die ReferenceModel.py Datei. Dadurch lässt sich die Pyomo Bibliothek laden, und das Model-Objekt instanziieren:

```
from pyomo.core import*
model = AbstractModel()
```

4.2.1 Definition des AbstractModel-Objekts

Nachdem das Modell deklariert wurde, sind die einzelnen Variablen und Parameter zu definieren. Dabei muss es beachtet werden, dass die Variablen (`Var`) die Größen sind, die zur Ermittelung der optimalen Lösung verändert werden dürfen, im Gegensatz zu den Parametern (`Param`), die die Konstanten in den definierten Szenarien darstellen. Außerdem können einzelne Variablen fixiert werden, was zugunsten der Rechengeschwindigkeit sich auswirken kann.

Da bei der Kompilierung des Codes, das Modell noch nicht initialisiert wurde, sondern nur instanziiert, sind die einzelnen Elemente dem Programm noch unbekannt und können somit nicht referenziert werden. Demzufolge dürfen einzelne Attribute nur mithilfe bestimmter „Callback“-Funktionen bearbeitet werden. Das kommt z.B. bei der Fixierung von Variablen in Pyomo vor:

```
def init_P_deltaB(model,n):#Types und P_gen wurden schon definiert
        if model.Types[n]==0 and model.P_gen[n]==0 :
                model.P_deltaB[n].fixed=True
        return 0
model.P_deltaB=Var(model.Nodes,initialize=init_P_deltaB,within=NonNegativeReals)
```

Die Namen der Modell-Parameter müssen mit den entsprechenden Angaben des Datensatzes in der .dat Datei übereinstimmen. Nur so ist Python in der Lage, die Werte den richtigen Elementen zuzuweisen. Bei fehlenden Einträgen werden die Parameter als `null` initialisiert. Der Parameter `cpvB` des im Rahmen dieses Berichtes betrachteten Problems wird beispielsweise so definiert:

In der ReferenceModel.py Datei	In der ReferenceModel.dat Datei

`model.cpvB=Param(model.Nodes,default=1000000)`	`Param cpvB :=`
	`W 10`
	`p 10`
	`v 10;`

Um die Bedingung 1 bei der Problemformulierung 3-4 aus Abschnitt 4.3.1 in Pyomo richtig einzugeben, muss der Definitionsbereich der Variable a^0 auf binär eingestellt werden. Das erfolgt durch Übergeben des Arguments `within=Binary` dem Konstruktor von a^0:

```
model.a0=Var(model.Lines,within=Binary,initialize=a0_initialize)
```

4.2.2 Die Constraint-rules

Nebenbedingungen bei Optimierungsproblemen werden in Pyomo üblicherweise als sog. „Rules“ definiert. Diese sind, wie bei der Variableninitialisierung, ebenfalls Callback-Funktionen, die vom Compiler als mathematische Zusammenhänge umgesetzt werden.
Die zwei Nebenbendigungen 2 und 3 in Problemformulierung **3-4**, nämlich die Lastfluss-Gleichung und die maximale Leitungsbelastbarkeit sind auf die folgende Art dem AbstractModel einzufügen:

```
def powerFlow_firstStage_rule(model,i):
        totalFlow = model.P_gen[i]-model.P_load[i]
        totalFlow += sum((model.anglesA[n]-model.anglesA[m])/
                model.X[n,m] for (n,m) in model.Lines if n==i and model.a0[n,m]!=0)
        totalFlow -= sum((model.anglesA[n]-model.anglesA[m])/
                model.X[n,m] for (n,m) in model.Lines if m==i and model.a0[n,m]!=0)
        return totalFlow==0

def max_line_flowA_rule(model,n,m):
        return (-1,(model.anglesA[n]-model.anglesA[m])*model.a0[n,m]/
                model.X[n,m]/(model.Plines_max[n,m]/model.bmva),1)

model.PowerFlowConstraintA=Constraint(model.Nodes,rule=powerFlow_firstStage_rule)
model.MaxLineFlowMaxA = Constraint(model.Lines,rule=max_line_flowA_rule)
```

Das muss allerdings in jedem Stage gelten, sodass der obigen Codeabschnitt dreimal zu wiederholen ist, jeweils für Stage B und C.

4.2.3 Angabe der Zielfunktion

Die allgemeine Zielfunktion des Problems **3-4**enthält die Kosten aller Stage-Decisions:

```
def total_cost_rule(model):
return model.FirstStageCost + model.SecondStageCost + model.ThirdStageCost
model.Total_Cost_Objective = Objective(rule=total_cost_rule, sense=minimize)
```

Diese sind in Form sogenannter Expression-Objekte in Pyomo zu definieren. Während bei einer deterministischen Formulierung des Problems die Aufteilung der Kosten in Stages auf einem ersten Blick für Redundanz im Code sorgt, ist diese für die Ermittelung der stochastischen Lösung entscheidend. Die Kostendefinitionen werden der Vollständigkeit halber hierbei angegeben:

```
def ComputeFirstStageCost_rule(model):
      return sum((1-model.a0[n,m])*model.ccv[n,m] for (n,m) in model.Lines)
def ComputeSecondStageCost_rule(model):
      return sum(model.cpvB[n]*model.P_deltaB[n]
            for n in model.Nodes if model.Types[n]!=0)
def ComputeThirdStageCost_rule(model):
      return sum(model.cpvC[n]*model.P_deltaC[n]
            for n in model.Nodes if model.Types[n]!=0)

model.FirstStageCost = Expression(rule=ComputeFirstStageCost_rule)
model.SecondStageCost = Expression(rule=ComputeSecondStageCost_rule)
model.ThirdStageCost = Expression(rule=ComputeThirdStageCost_rule)
```

Ein der größten Vorteile der Pyomo-Bibliothek ist, dass komplexe Optimierungsprobleme sich in vergleichsweise wenige Zeile Codes modellieren lassen. Das gelingt, dank einer guten und dafür strengten Strukturierung der Problem-Elemente.
Die oben erwähnte Redundanz des Codes kommt allerdings nur bei der Definition von Funktionen vor, da die szenario-abhängigen Parameter nicht einzeln zu definieren sind. Daraus erfolgt eine richtige „Abstrahierung“ des Modells, da dieses (wenn es vernünftig kodiert wurde) für eine beliebig hohe Anzahl an Szenarien verwendbar ist und sich für mehrere Lösungsalgorithmen (z.B. Progressive Hedging) eignet.

4.2.4 Ausführen des Skripts

.py Skripten lassen sich in der Rechner-Konsole durch Eingabe folgendes Befehls ausführen:

```
pyomo ReferenceModel.py ReferenceModel.dat #solver Default ist cplex
```

Pyomo bietet außerdem die Möglichkeit, die Optimierungsprobleme einem frei brauchbaren Server weiterzuleiten (Namens „Neos“), wo mehrere Solvers vorhanden ist,

damit der Algorithmus nicht unbedingt lokal durchgeführt werden muss. Die Ergebnisse werden dann in drei Arten ausgegeben:

1. als .sol Datei (Default)
2. aufgeführt in der Konsole (durch Übergabe des Arguments `--summary`)
3. abgespeichert in einer externen Datei (z.B. mittels `--json`)

Die Bibliothek Pysp ergänzt die Funktionalitäten von Pyomo, durch Definition u.A. von den Executables `runef` und `runph`, die für Erzeugung und Lösung der „Extensive Form“ sowie „Progressive Hedging“-Verfahren zuständig sind [8].
Diese Funktionen werden bei der Beschreibung der Ergebnisse im nächsten Kapitel benutzt.

5 Ergebnisse

5.1 Deterministischer Ansatz

Das Modell für den deterministischen Lösungsansatz wurde so implementiert wie im Kapitel 5 beschrieben. Der Quellcode wurde bei der Einreichung dieses Berichtes beigefügt[6].

5.1.1 Einfluss der zunehmenden Windgeschwindigkeit auf das Netz

Der Wert der stochastischen Lösung soll nach dem VSS-Verfahren mit dem Wert der deterministischen validiert werden. Um die Letzte zu ermitteln, sind für alle stochastischen Parameter, Durchschnittswerte anzunehmen. Es fällt in Grafik Abbildung 12 Scenario tree des analysierten Optimierungsproblems.Der hellbrau markierte Pfad kennzeichnet die Realisierung von Szenario 1Abbildung 12 auf, dass der Parameter-Datensatz entsprechend Szenario 5 dem deterministischen Modell, einzugeben ist, da dieses den „Average Case“ darstellt.
Aus dieser Wahl der Parameter stellt sich ein Lastfluss nach der Anfangsberechnung eines DCOPF ein, den in Abbildung 14 angezeigt wird. Die Leistungsschwankungen, die durch die Änderung der Wetterlage nach 6 *h* auftreten, verursachen einen unzulässigen Netzbetriebszustand .

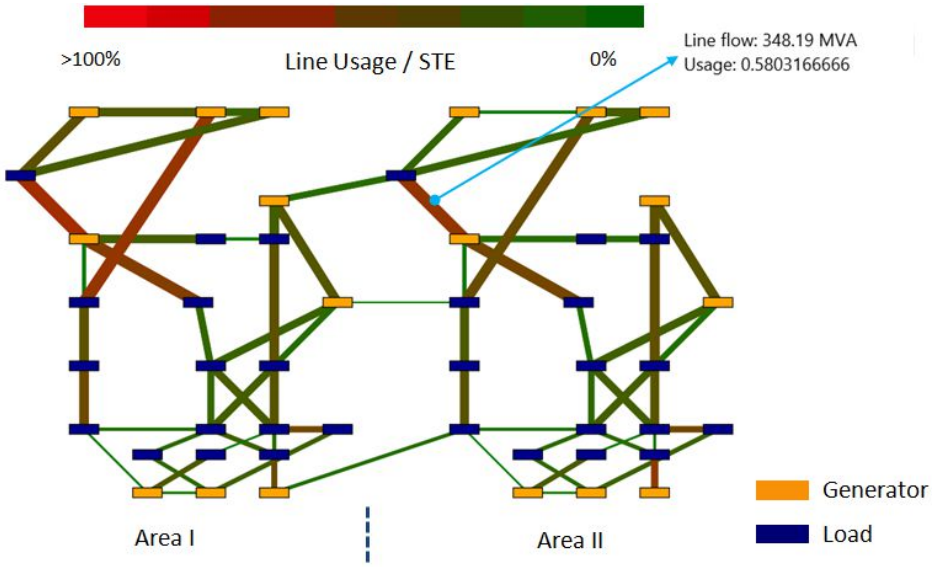

Abbildung 14 Zulässiger Betriebszustand des untersuchten Netzes bei t = 0 *h*

[6]Eine aktualisierte (Open Source) Version des Projekts kann auf https://github.com/dberardo/stochastic_problem_EEN1 abgerufen werden.

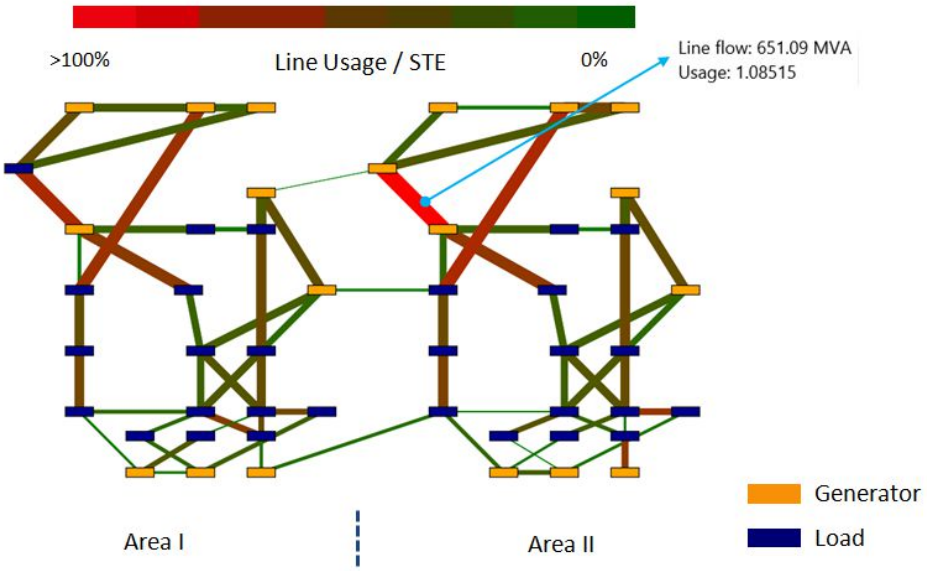

Abbildung 15 Unzulässiger Betriebszustand des untersuchten Netzes bei t = 6 h

Der dabei erscheinende Engpass muss auf die wirtschaftlichste Art gelöst werden. Dazu wird eine Optimierung nach dem entworfenen Algorithmus durchgeführt, deren Ergebnisse hiermit beschrieben werden.

5.1.2 Lösung des Optimierungsproblems

Zum Starten der Optimierung wurde der folgende Befehl verwendet:

```
$pyomo ProblemFormulation.py Scenario5.dat
      --solver-manager=neos --solver=cbc --json–summary
```

Das führt zu einer „feasible“ Lösung die in der Datei *results_deterministic_problem.json* gespeichert ist. Der wesentliche Teil dieser Ausgabe wird unten angegeben und lautet:

Präambel

```
[…]
"Message":   "CBC   2.9.4   optimal,
objective275.646519999999; 0 nodes,
0 iterations, 0.019996 seconds",
            "Objective": {

"Total_Cost_Objective": {
                    "Value":
275.646519999999
                }
            },
            "Problem": {},
            "Status": "optimal",
```

Auszug aus der Variablen-Auflistung

```
"Variable": {
                "P_deltaB[t]": {
                    "Value":
2.6360000000000134
                },
                "P_deltaC[p]": {
                    "Value": 1.8
                },
                "P_deltaC[t]": {
                    "Value":
0.8359999999999757
                },
                "a0[G,a]": {
                    "Value": 1.0
                },
                "a0[M,m]": {
                    "Value": 1.0
                },
                "a0[W,o]": {
                    "Value": 1.0
                },
```

Hier ist zu erkennen, dass die Windeinspeisung an den Knoten *t* und *p* um 2,636 bzw. 1,8 *p.u.* im Stage B reduziert wurde. Zusätzlich wurde die Leistung am Generator *t* um weitere 0,836 *p.u.* im Stage C heruntergefahren. Keine Verbundleitung wurde gekappt, weil alle binären Werte von a^0 1 betragen. Da die Kosten jeweils auf 51,72 *€/MWh*(C^B_{cpv}), 52,85 *€/MWh*(C^C_{cpv}), und 1000 *€* (c_{ccv}) liegen, wird die Trennung von Verbundleitungen vom Algorithmus als ungünstige Lösung betrachtet.

Um die Funktionsweise des Algorithmus weiter zu validieren, wurden die Kosten c_{ccv}bei einer darauffolgenden Ausführung des Skripts auf 0 gebracht. Das ergibt die folgende feasible Lösung:

Präambel

```
[...]
"Message": "CBC 2.9.4 optimal,
objective 275.646519999998; 0
nodes, 0 iterations, 0.021996
seconds",
            "Objective": {

"Total_Cost_Objective": {
   "Value": 275.64651999999836
                }
            },
            "Problem": {},
            "Status":
"optimal",
```

Auszugaus der Variablen-Auflistung

```
"Variable": {
                "P_deltaB[t]": {
                    "Value": 2.6360000000000134
                },
                "P_deltaC[p]": {
                    "Value": 1.8
                },
                "P_deltaC[t]": {
                    "Value": 0.8359999999999558
                },
                "a0[M,m]": {
                    "Value": 1.0
                },
```

Dabei ist zu bemerken, dass die Werte der Variablen `a0[W,o]` und `a0[G,a]` nicht ausgegeben wurden, weil diese 0 betragen. Das heißt, dass die (kostenfreie) Trennung von zwei Verbundleitungen, zur einer insgesamt besseren Lösung führen, selbst wenn der Unterschied in diesem Beispiel sehr gering ist.

Die Lastflussverteilung für diedeterministische Lösung zeigt Abbildung 16. Es wäre zu erwarten, dass die vorher überlastete Leitung nun an ihrer Grenze betrieben werden

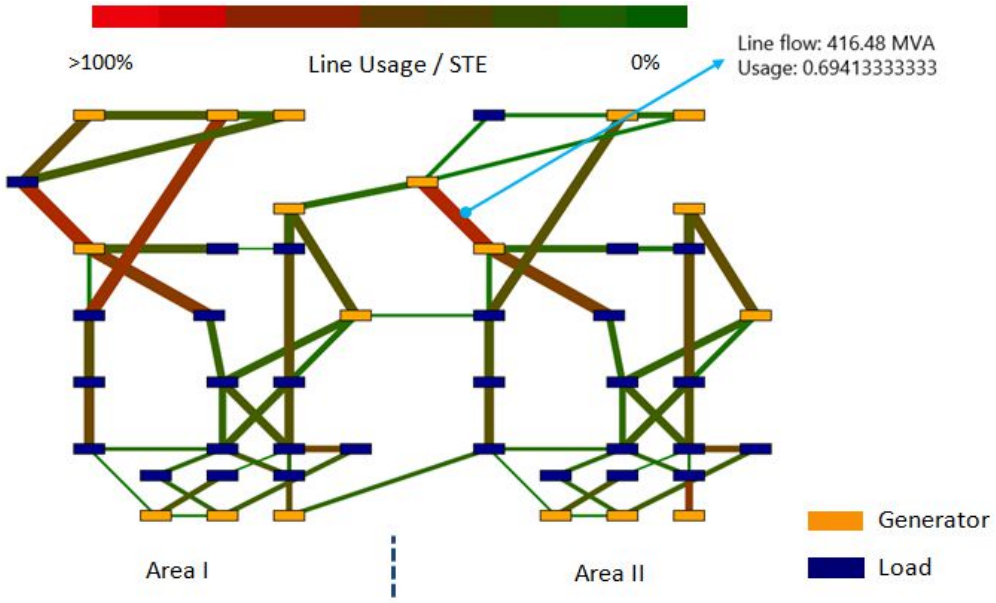

Abbildung 16 Leistungsverteilung auf den Leitungen im Testnetz aus der deterministischen Problemlösung

würde. Das trifft allerdings in der Grafik nicht zu, da neben der Leitungsbelastung noch zusätzlicher Bedingungen gleichzeitig erfüllt sein müssen, wobei die Lastfluss Bilanz die wichtigste für eine feasible Lösung ist.

5.2 Stochastischer Ansatz

Die Vorteilhaftigkeit von Pysp zeigt sich dadurch, dass das deterministische sowie stochastische Problem sich von demselben Modell beschreiben lassen. Für die Durchführung einer stochastischen Problemoptimierung ist eine Szenario-Struktur dem `runef` oder `runph` Kommando zu übergeben. Die Definition des Szenario-Tree beim untersuchten Problem wird im Anhang A angegeben.

5.2.1 Lösung mit `runph` – Initializierung

Als erstes Lösungsverfahren wurde das PH gewählt, wofür der folgende Befehl ausgeführt wurde:

```
$runph -m ProblemFormulation.py -i . --solver-manager=neos --default-rho=1
```

Selbst wenn das Programm erfolgreich kompiliert wurde, was die Vollständigkeit vom Code nachweist, liefert die Ausführung des `runph` Skripts die folgende Fehlermeldung:

```
Initializing PH
Starting PH
Initiating PH iteration=0
runph: KEY ERROR:
'Componentwithname: Total_Cost_Objective (id=139763641087488)'
```

Die Dokumentation zum Pysp hat sich nicht hinreichend genug gezeigt, sodass der Fehler in kurzer Zeit nicht zu beseitigen war.

5.2.2 Lösung mir runef

Als zweiter Versuch wurde das Skript mittels des `runef` gestartet, dafür wurde der unten stehende Befehl verwendet:

```
$runef -m ProblemFormulation.py -i . --solver-manager=neos --solver=cbc-solve
```

Die Extensive Form des stochastischen Problems wird damit efolgreich erzeugt und gelöst. Das liefert trotzdem nicht das erwünschte Ergebnis, da nur die Variable a^0 vom Programm, aus unbekannten Gründen, als StageVariable erkannt wird. Alle StageVariables wurden allerdings richtig definiert, wie auch aus anderen Beispielen fest zu stellen ist [9] (siehe auch Anhang B). Die Ausführung des Skriptes liefert im Gegensatz zum `runph` keine Fehlermeldung, weshalb der Fehler sich nicht adressieren lässt. Ein Debug ist demzufolge nicht möglich gewesen.

Nichtsdestotrotz führt die Ausgabe des runef Befehls zur erfolgreichen Validierung der implementierten Szenario-Tree, der nämlich von Python richtig interpretiert wurde. Für Details wird auf Anhang B verwiesen.

Quellenverzeichnis

[1] S. Konrad, Stochastische Optimierung, Bestandsoptimierung in mehrstufigen Lagernetzwerken, Vieweg+Teubner Verlag, 2012

[2] Jun.-Prof. Dr. A. Koberstein, Stochastische Optimierung, 2013

[3] A.J.Conejo, M.Carríon, Decision making under uncertainty in electricity markets,Springer-Verlag, 2010.

[4] Dantzig, G.B., Linear Programming under uncertainty, Management Science 1, 1995

[5] Charnes, A., Cooper, W.W., Chance-constrained programming, Management Science 5, 1959

[6] BDEW, Ermittlung von Entschädigungszahlungen nach § 12 Abs. 1 EEG 2009, Jan. 2012.

[7] Application of Probability Methods Subcommittee, The IEEE Reliability Test System-1996, IEEE Transactions on Power Systems, Vol. 14, Aug. 1999

[8] J.P. Watson, C. Laird et alii, Pyomo-Optimization Modeling in Python, Springer-Verlag, 2012.

[9] J.P. Watson, D.L. Woodruff, W.E. Hart, PySP: modeling and solving stochastic programs in Python, Springer and Mathematical Optimization Society, Sept. 2012.

Anhang A

Definition des Scenario-Tree (ScenarioDefinition.dat)

```
set Stages := FirstStage SecondStage ThirdStage;

set Nodes := RootNode

            BANode0
            ANode1
            AANode2

              BANode00
              ANode01
              AANode02
              BANode10
              ANode11
              AANode12
              BANode20
              ANode21
              AANode22 ;

param NodeStage := RootNode      FirstStage

                   BANode0 SecondStage
                   ANode1  SecondStage
                   AANode2 SecondStage

                      BANode00   ThirdStage
                      ANode01    ThirdStage
                      AANode02   ThirdStage
                      BANode10   ThirdStage
                      ANode11    ThirdStage
                      AANode12   ThirdStage
                      BANode20   ThirdStage
                      ANode21    ThirdStage
                      AANode22   ThirdStage ;

set Children[RootNode] := BANode0
                                ANode1
                                AANode2 ;

set Children[BANode0] :=  BANode00
                                ANode01
                                AANode02 ;

set Children[ANode1] :=         BANode10
                                ANode11
                                AANode12 ;

set Children[AANode2] :=  BANode20
                                ANode21
                                AANode22 ;

param ConditionalProbability :=  RootNode     1.0
                                 BANode0      0.07
                                    ANode1    0.86
                                    AANode2   0.07
                                    BANode00  0.11
                                    ANode01   0.78
                                    AANode02  0.11
                                    BANode10  0.11
                                    ANode11   0.78
                                    AANode12  0.11
                                    BANode20  0.11
                                    ANode21   0.78
                                    AANode22  0.11 ;

set Scenarios := Scenario1
```

```
                Scenario2
                Scenario3
                Scenario4
                Scenario5
                Scenario6
                 Scenario7
                Scenario8
                Scenario9 ;

param ScenarioLeafNode := Scenario1         BANode00
                          Scenario2         ANode01
                          Scenario3         AANode02
                          Scenario4         BANode10
                          Scenario5         ANode11
                          Scenario6         AANode12
                          Scenario7         BANode20
                          Scenario8         ANode21
                          Scenario9         AANode22 ;

set StageVariables[FirstStage] :=  a0[*,*]
                                    anglesA[*];

set StageVariables[SecondStage] := P_deltaB[*]
                                    anglesB[*] ;

set StageVariables[ThirdStage] :=  P_deltaC[*]
                                    anglesC[*] ;

param StageCost := FirstStage  FirstStageCost
                   SecondStage SecondStageCost
                   ThirdStage ThirdStageCost ;
```

Anhang B

Ausgabe des `runef` Befehls (Auszug)

```
Extensive form solution:
---------------------------------------------------
Tree Nodes:
       Name=AANode02
       Stage=ThirdStage
       Parent=BANode0
       Variables:

       [...]

       Name=BANode20
       Stage=ThirdStage
       Parent=AANode2
       Variables:

       Name=RootNode
       Stage=FirstStage
       Parent=None
       Variables:
              a0[A,B]=1.0
       [...]
              a0[r,u]=1.0
              a0[s,t]=1.0


Extensive form costs:
Scenario Tree Costs
---------------------------------------------------
Tree Nodes:

       Name=AANode02
       Stage=ThirdStage
       Parent=BANode0
       Conditional probability=0.1100
       Children:
              None
       Scenarios:
              Scenario3
       Expected cost of (sub)tree rooted at node=    0.0000

       Name=AANode12
       Stage=ThirdStage
       Parent=ANode1
       Conditional probability=0.1100
       Children:
              None
       Scenarios:
              Scenario6
       Expected cost of (sub)tree rooted at node=    0.0000

       Name=AANode2
       Stage=SecondStage
       Parent=RootNode
       Conditional probability=0.0700
       Children:
              AANode22
              ANode21
              BANode20
       Scenarios:
              Scenario7
              Scenario8
              Scenario9
       Expected cost of (sub)tree rooted at node=    0.0000

       Name=AANode22
```

```
Stage=ThirdStage
Parent=AANode2
Conditional probability=0.1100
Children:
       None
Scenarios:
       Scenario9
Expected cost of (sub)tree rooted at node=    0.0000

Name=ANode01
Stage=ThirdStage
Parent=BANode0
Conditional probability=0.7800
Children:
       None
Scenarios:
       Scenario2
Expected cost of (sub)tree rooted at node=    0.0000

Name=ANode1
Stage=SecondStage
Parent=RootNode
Conditional probability=0.8600
Children:
       AANode12
       ANode11
       BANode10
Scenarios:
       Scenario4
       Scenario5
       Scenario6
Expected cost of (sub)tree rooted at node=    0.0000

Name=ANode11
Stage=ThirdStage
Parent=ANode1
Conditional probability=0.7800
Children:
       None
Scenarios:
       Scenario5
Expected cost of (sub)tree rooted at node=    0.0000

Name=ANode21
Stage=ThirdStage
Parent=AANode2
Conditional probability=0.7800
Children:
       None
Scenarios:
       Scenario8
Expected cost of (sub)tree rooted at node=    0.0000

Name=BANode0
Stage=SecondStage
Parent=RootNode
Conditional probability=0.0700
Children:
       AANode02
       ANode01
       BANode00
Scenarios:
       Scenario1
       Scenario2
       Scenario3
Expected cost of (sub)tree rooted at node=    0.0000

Name=BANode00
Stage=ThirdStage
```

```
Parent=BANode0
Conditional probability=0.1100
Children:
       None
Scenarios:
       Scenario1
Expected cost of (sub)tree rooted at node=    0.0000

Name=BANode10
Stage=ThirdStage
Parent=ANode1
Conditional probability=0.1100
Children:
       None
Scenarios:
       Scenario4
Expected cost of (sub)tree rooted at node=    0.0000

Name=BANode20
Stage=ThirdStage
Parent=AANode2
Conditional probability=0.1100
Children:
       None
Scenarios:
       Scenario7
Expected cost of (sub)tree rooted at node=    0.0000

Name=RootNode
Stage=FirstStage
Parent=None
Conditional probability=1.0000
Children:
       AANode2
       ANode1
       BANode0
Scenarios:
       Scenario1
       Scenario2
       Scenario3
       Scenario4
       Scenario5
       Scenario6
       Scenario7
       Scenario8
       Scenario9
Expected cost of (sub)tree rooted at node=    0.0000
```