Ostfalia
Hochschule für angewandte
Wissenschaften

Mehrperiodische Standortplanung unter Berücksichtigung von Kapazitäten und Unsicherheiten mit Hilfe der Modellierungssprache GAMS und kommerzieller MIP-Löser

Eingereichte Abschlussarbeit
zur Erlangung des akademischen Grades eines Bachelor of Science (B. Sc.)
im Studiengang Logistik- und Informationsmanagement
an der Karl-Scharfenberg-Fakultät
der Ostfalia Hochschule für angewandte Wissenschaften

von

Daniel Hauer
(Matrikelnummer: 70451348)

in Zusammenarbeit mit GAMS Software GmbH

GAMS

Erstprüfer:	Prof. Dr. Ronny Hansmann
Zweitprüfer:	Dr. Michael R. Bussieck
Eingereicht am:	31. Januar 2019

Inhaltsverzeichnis

Abkürzungsverzeichnis **IV**

Abbildungsverzeichnis **V**

Tabellenverzeichnis **VI**

1 Einleitung **1**

2 Mathematische Grundlagen **3**

2.1 (Lineare) Optimierung 3

2.1.1 Dualität 4

2.1.2 Benders' Dekompositionsverfahren (grundsätzliche Idee) 7

2.2 Stochastik 10

2.2.1 Wahrscheinlichkeitsraum 10

2.2.2 Zufallsvariable und Verteilungsfunktion 10

2.2.3 Erwartungswert 12

2.2.4 Quantile, Value at Risk (VaR) und Conditional Value at Risk (CVaR) 12

2.3 Stochastische Optimierung 15

2.3.1 Deterministisches Äquivalent zum EV 15

2.3.2 Deterministisches Äquivalent zum CVaR 16

2.3.3 Expected Value of Perfect Information (EVPI) und Value of the Stochastic Solution (VSS) 17

2.3.4 Benders' Dekompositionsverfahren für SP 19

3 Das Facility Location Problem (FLP) **26**

3.1 Das klassische FLP 26

3.2 Mulitperiodisches FLP unter Berücksichtigung von Unsicherheiten und Kapazitäten 27

3.3 Vorbereitungen für Benders' Dekompositionsverfahren 31

4 Empirische Untersuchungen **36**
4.1 Die Modellierungssprache GAMS 36
4.2 Implementierungen in GAMS 37
4.3 Instanzen 38
4.4 EV und CVaR in GAMS 38
4.5 EVPI und VSS in GAMS 40
4.6 Benders' Dekompositionsverfahren in GAMS 41
4.6.1 Vorstellung von `CPLP_Risk_Benders.gms` 41
4.6.2 Vergleich Resultate 50

5 Zusammenfassung **53**

6 Fazit und Ausblick **55**

7 Danksagung **56**

Literaturverzeichnis **57**

Eidesstattliche Erklärung **59**

A Aufbau der Instanzen **60**

B Inhalt der Datei importFile.inc **61**

C Wertetabelle Single- und Multicuts **63**

Abkürzungsverzeichnis

CPLP-Risk	Mehrperiodisches FLP unter Berücksichtigung von Kapazitäten und Unsicherheiten
CVaR	Conditional Value at Risk
DE	Deterministisches Äquivalent
EMP	Extended Mathematical Programming
EV	Expected Value
EVPI	Expected Value of Perfect Information
FLP	(Klassisches) Facility Location Problem
GAMS	General Algebraic Modeling System
IP	Integer (Linear) Program
LP	Linear Program
MIP	Mixed Integer (Linear) Program
PLP	Plant Location Problem
SP	Stochastic Program
VaR	Value at Risk
VSS	Value of the Stochastic Solution

Abbildungsverzeichnis

1 Illustrierung der Dualitäts-Lücke 6
2 Veranschaulichung des VaR und CVaR für eine Verlustfunktion 13
3 Konvexkombination von EV und CVaR 40
4 Vergleich Single- und Multicuts bei `medium.txt` 51

Tabellenverzeichnis

1 Übersicht über die GAMS-Programme mit Kurzbeschreibung 37
2 Übersicht über verfügbare Instanzen 38
3 Resultate von `CPLP_Risk_EV.gms` 39
4 Resultate von `CPLP_Risk_EVPI.gms` und `CPLP_Risk_VSS.gms` 41
5 Wertetabelle zu Single- und Multicuts (`large.txt`) 52
6 Wertetabelle zu Single- und Multicuts (`medium.txt`) 63

Kapitel 1

Einleitung

Die Standortplanung einer Verwertungsstätte stellt für Unternehmen eine sehr bedeutsame Aufgabe dar, denn es wird eine strategische Entscheidung getroffen, die auch noch in vielen Jahren ihre Berechtigung finden soll. Eine ganze Reihe an s. g. Standortfaktoren, die sich von ökonomischen Faktoren (wie etwa Logistikkosten, Absatzmarktnähe, Mitarbeiterkosten, etc.) über ökologische Punkte (z. B. das Image, Stichwort *Green Logistics*[1]) bis hin zu gesellschaftlich motivierten Aspekten (Rechtssicherheit, soziales Arbeitsklima) erstrecken, müssen berücksichtigt werden. Gerade für die letzten beiden Gruppen von Faktoren, die zu den s. g. *weichen Standortfaktoren* zählen, ist es jedoch schwer, den erreichten Nutzen zu messen. Dahingegen zeichnen sich die übrigen s. g. *harten Standortfaktoren* durch eine hohe Messbarkeit aus, weshalb hier in der Entscheidungsfindung zwischen numerischen Größen abgewogen werden kann. So senkt die Wahl eines Standortes in der Nähe der Kunden zwar die anfallenden Transportkosten zu diesen, jedoch ist es denkbar, dass an diesem Standort höhere Fixkosten anfallen, da z. B. der Erwerb dieses Grundstücks durch größere Nachfrage teurer sein kann.[2]

Dieses Abwägen kann mathematisch modelliert werden und führt somit zu einer Optimierungsaufgabe, in der anfallende Kosten minimiert werden. Um der Langfristigkeit der Entscheidung zu entsprechen, wird das Modell erweitert. Dabei werden Aspekte wie Mehrperiodizität oder Kapazitäten an den Standorten berücksichtigt. Doch realistisch ist das Modell noch immer nicht. Da es um eine Entscheidung mit Einfluss in die Zukunft geht, spielt Unsicherheit eine wesentliche Rolle in der Modellierung. Diese lässt sich jedoch mathematisch schwer beschreiben. Die gängige Herangehensweise sieht die Betrachtung von endlich vielen Szenarien versehen mit einer Eintrittswahrscheinlichkeit vor. Daraus lässt sich ein in der Regel großes Modell ableiten.

Diese Arbeit setzt sich zunächst allgemein mit mathematischen Konzepten und Verfah-

[1] Vgl. [Wittenbrink, 2015].
[2] Vgl. [Daum et al., 2010, S. 188 - 189].

ren zur Lösung solcher Modelle auseinander. Dazu werden theoretische Grundlagen zur Optimierung und zur Stochastik eingeführt, welche dann in der Stochastischen Optimierung münden. Es wird aufgezeigt, wie ein Optimierungsproblem unter Verwendung der Risikomaße Erwartungswert und Conditional Value at Risk betrachtet werden kann.

Ein wesentlicher Bestandteil dieser Arbeit stellt Benders' Dekompositionsverfahren dar. Dieses wird zunächst konzeptionell für allgemeine lineare Optimierungsprobleme, die in Teilprobleme zerlegbar sind, vorgestellt. Da stochastische Optimierungsprobleme mit ihrem Deterministischen Äquivalent auf natürliche Weise dieser notwendigen Form entsprechen, wird im Abschnitt zur Stochastischen Optimierung Benders' Dekompositionsverfahren noch einmal bezogen auf diese ausführlich erläutert.

Im darauf folgenden Kapitel wird das s. g. Facility Location Problem, in dem es um Standortplanung geht, hergeleitet, welches dann u. a. um den Aspekt der Unsicherheit erweitert wird. Dazu wird wie oben beschrieben eine endliche Menge an Szenarien verwendet. Auf das entstehende Modell können nun die theoretischen Überlegungen aus dem vorherigen Kapitel angewandt werden. Dafür werden sämtliche Modelle in der Modellierungssprache GAMS implementiert und auf diese Weise miteinander verglichen. Das GAMS-Programm, in dem Benders' Dekompositionsverfahren umgesetzt ist, wird ausführlich beschrieben.

Ziel der Arbeit ist es, herauszufinden, mit welchen Verfahren die Standortplanung unter Unsicherheit am performantesten gelingt. Das Hauptaugenmerk liegt hier zum einen auf dem Vergleich der Risikomaße Erwartungswert und Conditional Value at Risk. Zum anderen wird die Lösungsfindung bei direkter Verwendung des Deterministischen Äquivalents mit der bei Verwendung von Benders' Dekompositionsverfahren verglichen.

Kapitel 2

Mathematische Grundlagen

In diesem Kapitel werden theoretische Grundlagen eingeführt. Zu linearen Optimierungsproblemen wird das Konzept der Dualität hergeleitet. Zudem wird die grundsätzliche Idee von Benders' Dekompositionsverfahren präsentiert. Im Anschluss daran werden wichtige Begriffe der Stochastik eingeführt. Die beiden vorherigen Abschnitte werden in einem dritten Abschnitt über Stochastische Optimierung miteinander verknüpft. Da stochastische Optimierungsprobleme auf natürliche Weise der für Benders' Dekompositionsverfahren notwendige Form entsprechen, wird dieses in einem eigenen Abschnitt detailliert hergeleitet. Am Ende des Kapitels wird ein Algorithmus für dieses Verfahren aufgestellt, der sich insbesondere für die spätere Implementierung eignet.

2.1 (Lineare) Optimierung

Eine allgemeine *Optimierungsaufgabe* über $\mathbb{R}$ sucht aus dem *zulässigen Bereich* $M \subset \mathbb{R}^n$ dasjenige $x^* \in M$ aus, das eine Zielfunktion $f : M \to \mathbb{R}$ optimal löst. x^* heißt *Optimalpunkt*. Im Falle einer Maximierungsaufgabe wird somit das $x^* \in M$ ausgewählt, für das $f(x)$ maximal wird, also $\forall x \in M : f(x^*) \geq f(x)$. Die Menge M ist durch Restriktionen eingeschränkt. Ist M leer, so ist die Optimierungsaufgabe *unzulässig*. Ist f linear, also von der Form

$$f(x) = \sum_{i=1}^{n} c_i x_i = c^T x, \tag{2.1}$$

wobei $c \in \mathbb{R}^n$, und sind auch alle Restriktionen linear, so wird von einer *linearen Optimierungsaufgabe*[3] gesprochen.[4]

Formal lassen sich dann die Nebenbedingungen durch eine Matrix $A \in \mathbb{R}^{m \times n}$ und einen Vektor $b \in \mathbb{R}^m$ ausdrücken, wobei m der Anzahl an Restriktionen entspricht. Es ergibt

[3] Englisch: *linear program* (LP).
[4] Vgl. [Gritzmann, 2013, S. 1 - 4].

sich in kompakter Schreibweise die s. g. *Standard-Form*[5]:

$$\begin{aligned} \max_x \; & c^T x \\ \text{s.t. } & Ax \leq b \\ & x \geq 0, \end{aligned} \tag{2.2}$$

in der zusätzlich gefordert wird, dass die Komponenten von x nicht-negativ sind.[6]

Sorgen die Restriktionen für keine Einschränkung der Zielfunktion, sodass der Zielfunktionswert gegen Unendlich geht, so ist die Optimierungsaufgabe *unbeschränkt*.[7] Damit gilt folgende Äquivalenz: Zu einem Optimierungsproblem existiert genau dann eine optimale Lösung, wenn das Optimierungsproblem weder unzulässig noch unbeschränkt ist.

Sind für x_i mit $i \in \{1, ...n\}$ nur ganzzahlige Realisierungen erlaubt, so nennt man (2.2) auch *ganzzahlige Optimierungsaufgabe*[8]. Gibt es sowohl ganzzahlige als auch stetige Variablenkomponenten, so handelt es sich um eine *gemischt-ganzzahlige Optimierungsaufgabe*[9].[10]

2.1.1 Dualität

Es sei ein lineares Optimierungsproblem in Standard-Form, vgl. (2.2), gegeben:

$$\begin{aligned} \max_x \; & c^T x \\ \text{s.t. } & Ax \leq b \\ & x \geq 0. \end{aligned}$$

Wir nennen es das *primale Problem*. Jedes primale Problem kann in ein *duales Problem* transformiert werden. Handelt es sich bei dem primalen Problem um ein Maximierungsproblem, so ist das zugehörige duale Problem ein Minimierungsproblem, und entsprechend umgekehrt. Das duale Problem einer Maximierungsaufgabe lässt sich grundsätzlich motivieren als die Suche nach einer oberen Schranke für den Zielfunktionswert des primalen Problems. Dazu wird eine Minimierungsaufgabe konstruiert, deren Zielfunktionswert, falls existent, gleich dem Zielfunktionswert des primalen Problems ist.[11]

[5] Vgl. [Boyd und Vandenberghe, 2004, S. 129], manchmal auch *kanonische Form*.
[6] Vgl. [Vanderbei, 2013, S. 7].
[7] Vgl. [Vanderbei, 2013, S. 9].
[8] Englisch: *Integer (Linear) Program* (IP oder auch ILP).
[9] Englisch: *Mixed-Integer (Linear) Program* (MIP oder auch MILP).
[10] Vgl. [Gritzmann, 2013, S. 5].
[11] Vgl. [Murphy, 2013, S. 6 - 7].

Sei $y \in \mathbb{R}^m$, mit $y_i \geq 0 \; \forall i \in \{1, ..., m\}$, ein Vektor. Multipliziere y^T von links an die Restriktionen heran:

$$y^T Ax \leq y^T b. \tag{2.3}$$

Es ist klar, dass der Zielfunktionswert jeder Lösung von (2.2) eine untere Schranke für $y^T Ax$ darstellt. Daher muss für jedes y gelten:

$$y^T A \geq c^T. \tag{2.4}$$

Um die kleinste obere Schranke für $y^T Ax$ zu finden, muss nun $y^T b$ minimiert werden. Es ergibt sich das folgende duale Optimierungsproblem:

$$\begin{aligned} \min_y \; & y^T b \\ \text{s.t. } & y^T A \geq c^T \\ & y \geq 0, \end{aligned} \tag{2.5}$$

äquivalent zu:

$$\begin{aligned} \min_y \; & b^T y \\ \text{s.t. } & A^T y \geq c \\ & y \geq 0. \end{aligned} \tag{2.6}$$

Ein Blick auf die Dimensionen offenbart, dass die Variablen des dualen Problems mit den Restriktionen des primalen Problems in Verbindung stehen. Ebenso ist jede Restriktion im dualen Problem auf eine Variable des primalen Problems zurückzuführen. Bringt man das duale Problem aus (2.6) in Standard-Form, so lässt sich die obige Konstruktion wieder anwenden. Nach erneuter Umformung in Standard-Form resultiert wieder das primale Problem[12]. Damit wird klar, dass das duale Problem zum dualen Problem wieder das primale Problem ist.[13]

Es gibt zwei wichtige Theoreme bezüglich der Dualität. Seien dafür x^* und y^* die optimalen Lösungen für das primale und das duale Problem. Das s. g. *Weak Duality Theorem* besagt, dass der Wert jeder zulässigen Lösung des primalen Problems stets kleiner ist als der Wert jeder zulässigen Lösung des dualen Problems. Dieser Zusammenhang lässt sich wie folgt veranschaulichen. Der zulässige Wertebereich einer reellen Zielfunktion ist immer ein Intervall in $\mathbb{R}$, sagen wir für das primale Problem sei das $I_P = (l_P, u_P]$, mit $l_P \in \{\mathbb{R} \cup -\infty\}$ und $u_P = c^T x^* \in \mathbb{R}$. Nach unten kann I_P

[12] Vgl. [Vanderbei, 2013, S. 57 - 58].
[13] Vgl. auch für die vorangegangenen Absätze [Murphy, 2013, S. 7].

beschränkt sein, muss es aber nicht. Möglich wäre als untere Schranke auch $-\infty$. Die obere Schranke u_P existiert, wenn das primale Problem nicht unbeschränkt ist. Gleiches gilt für das dualen Problem mit dem Intervall $I_D = [l_D, u_D)$, wobei $l_D = b^T y^* \in \mathbb{R}$ und $u_D \in \{\mathbb{R} \cup \infty\}$. Das Weak Duality Theorem liefert nun, dass

$$c^T x^* = u_P \leq l_D = b^T y^* \tag{2.7}$$

gelten muss.[14]

Abbildung 1 zeigt diesen Zusammenhang[15]. Die darin dargestellte Dualitäts-Lücke gibt den Bereich an, der zwischen I_P und I_D liegt. Die Länge dieser Lücke ist $l_D - u_P$.

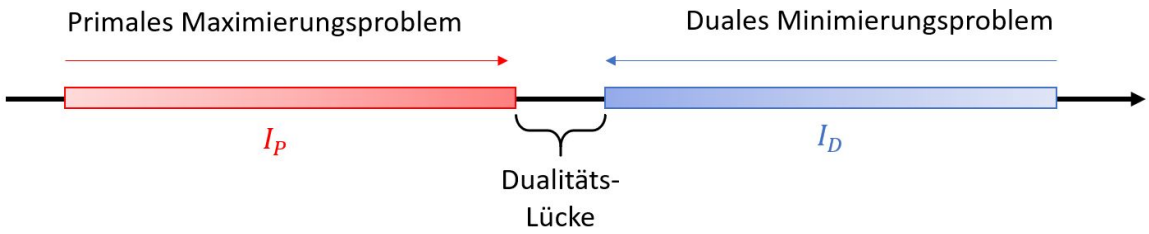

Abbildung 1: Illustrierung der Dualitäts-Lücke. Gemäß dem Weak Duality Theorem ist jede zulässige Lösung des primalen Problems kleiner als jede Lösung des dualen Problems. Die Intervalle I_P und I_D sind als disjunkt dargestellt, sodass die Dualitäts-Lücke deutlich wird.

Es stellt sich die Frage, wann es eine solche Lücke überhaupt gibt. Wie oben bereits angeklungen ist, sollen die beiden Zielfunktionswerte vom primalen und dualen Problem, wenn existent, den gleichen Wert haben, also $u_P = l_D$. Dies sichert das s. g. *Strong Duality Theorem*. Es besagt, dass sofern es für das primale Problem eine optimale Lösung x^* gibt, dass dann auch das zugehörige duale Problem eine optimale Lösung besitzt, nämlich y^* mit

$$c^T x^* = b^T y^* \tag{2.8}$$

Dementsprechend ist die Dualitäts-Lücke leer, falls es eine optimale Lösung gibt. Der Schnitt der Intervalle I_P und I_D enthält nur den gemäß Strong Duality Theorem gemeinsamen Zielfunktionswert von primalen und dualen Problem.[16]

Ist das primale Problem unbeschränkt, so ist $u_P = \infty$ und da nach dem Weak Duality Theorem $l_D \geq u_P$ gelten muss, ist das entsprechende duale Problem unzulässig. Die Dualitäts-Lücke liegt nun bei ∞. Gleiches gilt umgekehrt: wenn das duale Problem unbeschränkt ist, ist das zugehörige primale Problem unzulässig und die Dualitäts-Lücke

[14] Vgl. [Vanderbei, 2013, S. 58].
[15] Vgl. [Vanderbei, 2013, S. 59].
[16] Vgl. [Vanderbei, 2013, S. 58 - 60].

liegt bei $-\infty$.[17]

Ist das primale Problem unzulässig, dann folgt daraus jedoch nur, dass das duale Problem unzulässig *oder* unbeschränkt ist. Im ersten Fall geht die Dualitäts-Lücke von $-\infty$ bis zu ∞. Im zweiten Fall, wenn das duale Problem unbeschränkt ist, liegt die Dualitäts-Lücke, wie oben schon gesehen, bei $-\infty$. Analoges lässt sich aus der Unzulässigkeit des dualen Problems folgern.[18]

2.1.2 Benders' Dekompositionsverfahren (grundsätzliche Idee)

Jacques F. Benders[19] präsentierte 1962 das nach ihm benannte Dekompositionsverfahren, welches auch als *L-Shaped Algorithmus* bekannt ist[20]. Damals versuchte er ein MIP mit sowohl ganzzahligen als auch kontinuierlichen Variablen in kleinere Subprobleme aufzuteilen: eines sollte die ganzzahligen Variablen enthalten, also ein IP sein, und eines sollte die kontinuierlichen Variablen beinhalten, also ein LP sein. Diese beiden Optimierungsprobleme konnten wesentlich schneller gelöst werden als das größere MIP. Dieses Konzept der Dekomposition lässt sich auch auf sehr große LPs anwenden, sofern sich die Variablen in Teilprobleme aufteilen lassen.[21]

In diesem Abschnitt wird die Idee von Benders dargestellt. Das genaue Verfahren wird im Abschnitt 2.3.4 *Benders' Dekompositionsverfahren für SP* angewandt auf stochastische Optimierungsprobleme präsentiert; hier zunächst ein vereinfachter Algorithmus. Angenommen, es besteht das folgende MIP, ohne Beschränkung der Allgemeinheit ein Minimierungsproblem:

$$\begin{aligned} z^* = \min_{x,y} \; & c_1^T x + c_2^T y \\ \text{s.t. } & A_1 x + A_2 y \geq b \\ & x \geq 0 \\ & y \geq 0, \end{aligned} \tag{2.9}$$

wobei $c_1, x \in \mathbb{R}^{n_1}$, $c_2, y \in \mathbb{R}^{n_2}$, $A_1 \in \mathbb{R}^{m \times n_1}$ und $A_2 \in \mathbb{R}^{m \times n_2}$. Das ursprüngliche Problem (vgl. (2.2)) ist in Teilprobleme aufgeteilt. Ursprünglich verwendete Benders x für die kontinuierlichen und y für die ganzzahligen Variablen. Nun betrachte bezogen auf

[17] Vgl. [Vanderbei, 2013, S. 65].
[18] Vgl. [Vanderbei, 2013, S. 66].
[19] Niederländischer Mathematiker (1924-2017).
[20] Vgl. [Benders, 1962].
[21] Vgl. [Murphy, 2013, S. 9].

(2.9) das von y abhängige LP, indem y also ein fixer Wert zugewiesen wird:

$$\begin{aligned} v_0(y) = \min_x \; & c_1^T x + c_2^T y \\ \text{s.t. } & A_1 x \geq b - A_2 y \\ & x \geq 0. \end{aligned} \tag{2.10}$$

Es folgt

$$z^* \leq v_0(y) \quad \forall y, \tag{2.11}$$

wobei nur bei Wahl des optimalen y^* Gleichheit besteht. Damit existiert eine obere Schranke für den Zielfunktionswert von (2.9). Das duale Problem zu (2.10) ermöglicht das Finden von unteren Schranken für z^*. Der bezüglich fixem y konstante Term $c_2^T y$ kann in der dualen Zielfunktion enthalten bleiben:

$$\begin{aligned} w_0(y) = \max_u \; & (b - A_2 y)^T u + c_2^T y \\ \text{s.t. } & A_1^T u \leq c_1 \\ & u \geq 0. \end{aligned} \tag{2.12}$$

Dieses duale Problem ist ein LP und wird auch als *Subproblem* bezeichnet. Der zulässige Bereich dieses Problems ist von y unabhängig, da die Restriktionen von y unabhängig sind. Aufgrund der Tatsachen, dass eine optimale Lösung eines LPs immer an einem Eckpunkt des zulässigen Bereichs liegt[22], und es nur endlich viele, sagen wir K, solcher Eckpunkte gibt, können wir $w_0(y)$ auch auf andere Weise formulieren. Dazu sei U die endliche Menge an Eckpunkten des zulässigen Bereichs von (2.12):

$$w_0(y) = c_2^T y + \max_{u_k \in U} (b - A_2 y)^T u_k. \tag{2.13}$$

Nun ist es möglich, ein zu (2.9) äquivalentes Problem zu formulieren. Die Fixierung der Variable y wird dabei wieder aufgehoben, sodass über y minimiert werden kann:

$$z^* \overset{(2.11)}{=} \min_y \left[v_0(y)\right] \overset{(2.8)}{=} \min_y \left[w_o(y)\right] \overset{(2.13)}{=} \min_y \left[c_2^T y + \max_{u_j \in U} (b - A_2 y)^T u_k\right]. \tag{2.14}$$

Die Maximierungsaufgabe aus der Zielfunktion kann ersetzt werden durch K Restriktionen, die jeweils in der Form der Zielfunktion der Maximierungsaufgabe formuliert werden. Auf diese Weise ergibt sich das folgende Modell M^K, welches immer noch

[22] Vgl. [Burkard und Zimmermann, 2012, S. 33].

äquivalent zu unserem Ausgangsproblem in (2.9) ist:

$$\begin{aligned} z^* = \min_y \; & z \\ \text{s.t. } & z \geq (b - A_2 y)^T u_1 + c_2^T y \\ & \ldots \\ & z \geq (b - A_2 y)^T u_K + c_2^T y \\ & y \geq 0. \end{aligned} \tag{2.15}$$

Nun werden nicht alle K Restriktionen mit einem Mal hinzugefügt, sondern nach und nach. Das Optimierungsproblem (2.15), welches nur die ersten $R \leq K$ Restriktionen enthält, sei M^R und wird auch als *Masterproblem* bezeichnet. Der zugehörige optimale Zielfunktionswert sei z^R. Es ist klar, dass gilt: je weniger solcher Restriktionen hinzugefügt wurden, desto niedriger ist der Zielfunktionswert des Masterproblems im Vergleich zum Zielfunktionswert unseres Startproblems aus (2.9). Zusammen mit (2.11) gilt:

$$z^R \leq z^* \leq w_0(y) = v_0(y). \tag{2.16}$$

Damit existieren eine untere und eine obere Schranke für den Zielfunktionswert aus (2.9). Sobald nach Hinzufügen von $R \leq K$ Restriktionen Gleichheit zwischen z^R und $v_0(y)$ besteht, ist auch der optimale Zielfunktionswert für (2.9) gefunden, ohne dass das ursprüngliche Problem (2.9) vollständig gelöst werden musste.

Der folgende Algorithmus fasst die Funktionsweise dieses Verfahrens zusammen:

1. **Initialisierung**:
   - Setze $R = 1$.
   - Setze y^0 auf einen beliebigen zulässigen Wert.
2. **Iteration**:
   - Löse das duale Subproblem mit gegebenen y^{R-1}. Der zugehörige Zielfunktionswert sei $w_0(y^{R-1})$.
   - Löse M^R mit den ersten R Restriktionen. Der Zielfunktionswert sei z^R. Die optimale Lösung sei y^R.
   - **Abbruchbedingung**: Wenn $z^R \leq w_0(y^{R-1})$, dann beende Algorithmus mit y^R als optimaler Lösung, sonst:
   - Erhöhe R um 1 und beginne nächste Iteration.

Der Algorithmus terminiert spätestens bei $R = K$.[23]

[23] Vgl. auch für vorangegangene Absätze [Murphy, 2013, S. 10 - 13] und [Jensen, 2002, S. 1 - 5].

2.2 Stochastik

In diesem Abschnitt werden die stochastischen Konzepte Wahrscheinlichkeitsraum, Verteilungsfunktion, Dichtefunktion und Erwartungswert eingeführt. Darüber hinaus werden die Risikomaße Value at Risk und Conditional Value at Risk erläutert.

2.2.1 Wahrscheinlichkeitsraum

Ein Wahrscheinlichkeitsraum $(\Omega, \mathcal{A}, \mathbb{P})$ besteht aus einer Grundmenge Ω, einer σ-Algebra $\mathcal{A}$ und einem Wahrscheinlichkeitsmaß $\mathbb{P}$. Ω enthält alle denkbaren Ergebnisse, die ein stochastischer Prozess annehmen kann. $\mathcal{A} \subset \mathcal{P}(\Omega)$ ist eine Familie von Teilmengen von Ω, den Ereignissen. Das Wahrscheinlichkeitsmaß $\mathbb{P} : \mathcal{A} \to \mathbb{R}$ erfüllt die folgenden vier Eigenschaften:

$$\mathbb{P}(\emptyset) = 0 \tag{2.17}$$

$$\mathbb{P}(A) \geq 0 \qquad \forall A \in \mathcal{A} \tag{2.18}$$

$$\mathbb{P}(A_1 \cup A_2) = \mathbb{P}(A_1) + \mathbb{P}(A_2) \qquad A_1, A_2 \in \mathcal{A} : A_1 \cap A_2 = \emptyset \tag{2.19}$$

$$\mathbb{P}(\Omega) = 1. \tag{2.20}$$

Ist (2.20) nicht erfüllt, so handelt es sich bei $(\Omega, \mathcal{A}, \mathbb{P})$ um einen Maßraum.[24]

2.2.2 Zufallsvariable und Verteilungsfunktion

Eine reelle Zufallsvariable ξ über einen Wahrscheinlichkeitsraum $(\Omega, \mathcal{A}, \mathbb{P})$ ist eine Abbildung von der Grundmenge Ω in die reellen Zahlen. Einen Parameter, der unsicher ist, wollen wir später als eine Zufallsvariable modellieren. Im Folgenden müssen wir unterscheiden, ob Ω endlich ist oder nicht, wollen aber der Einfachheit halber annehmen, dass Ω eine Teilmenge der reellen Zahlen ist.

Ist Ω endlich, handelt es sich um eine diskrete Wahrscheinlichkeitsverteilung, andernfalls eine stetige Wahrscheinlichkeitsverteilung. In beiden Fällen wird die *kumulative Verteilung* zunächst definiert über

$$F(x) = \mathbb{P}(\{\omega \in \Omega : \xi \leq x\}) =: \mathbb{P}(\xi \leq x). \tag{2.21}$$

F ist eine monoton wachsende Funktion, da für $x_1 < x_2$ auch folgt

$$F(x_2) = \mathbb{P}(\xi \leq x_2) = \mathbb{P}(\xi \leq x_1) + \underbrace{\mathbb{P}(x_1 \leq \xi \leq x_2)}_{\geq 0 \text{ wegen } (2.18)} \geq \mathbb{P}(\xi \leq x_1) = F(x_1). \tag{2.22}$$

[24] Vgl. [Birge und Louveaux, 2011, S. 55 - 56].

Die Stetigkeit von F ist im diskreten Fall nicht gegeben, da Unstetigkeitsstellen existieren.

Im diskreten Fall existiert eine endliche Menge ξ^k an möglichen Werten für ξ. Die *Wahrscheinlichkeitsverteilung* f wird daher definiert durch

$$f(\xi^k) := \mathbb{P}(\xi = \xi^k). \tag{2.23}$$

Es wird gefordert, dass $\sum_k f(\xi^k) = 1$ gilt, was mit (2.20) zusammenhängt.

Im kontinuierlichen Fall wird eine *Dichtefunktion* f angegeben. Die Wahrscheinlichkeit für ξ, in einem Intervall $[a, b]$ zu liegen, ist gegeben durch:

$$\mathbb{P}(a \leq \xi \leq b) = \int_a^b f(\xi) d\xi \tag{2.24}$$

Auch wenn es nun möglich wäre, am Graphen von f für jeden fixen Punkt a einen Wert abzulesen, gilt: $\mathbb{P}(\xi = a) = 0$. Dies ist ein Unterschied zum diskreten Fall, wo die Wahrscheinlichkeitsverteilung gerade über einzelne Punkte definiert wurde (vgl. (2.23)).

Betrachte für $a \leq b$:

$$\int_{-\infty}^a f(\xi) d\xi = \mathbb{P}(\xi \leq x) = F(a) \ , \ \int_{-\infty}^b f(\xi) d\xi = \mathbb{P}(\xi \leq x) = F(b), \tag{2.25}$$

woraus folgt

$$\int_a^b f(\xi) d\xi = \int_{-\infty}^b f(\xi) d\xi - \int_{-\infty}^a f(\xi) d\xi = F(b) - F(a). \tag{2.26}$$

Wir können also die kumulative Verteilung F als Stammfunktion der Dichtefunktion f verstehen. Daher gilt für (2.24) auch:

$$\mathbb{P}(a \leq \xi \leq b) = \int_a^b dF(\xi) = [F(\xi)]_a^b \,. \tag{2.27}$$

Analog zum diskreten Fall fordern wir

$$\mathbb{P}(\infty \leq \xi \leq \infty) = \int_{-\infty}^{\infty} f(\xi) d\xi = \int_{-\infty}^{\infty} dF(\xi) = 1, \tag{2.28}$$

was wieder mit (2.20) zusammenhängt.[25]

[25] Vgl. [Birge und Louveaux, 2011, S. 56 - 57].

2.2.3 Erwartungswert

Der *Erwartungswert*[26] lässt sich im diskreten Fall wie folgt bestimmen:

$$\mathbb{E}[\xi] = \sum_k \xi^k f(\xi^k) \overset{(2.23)}{=} \sum_k \xi^k \mathbb{P}(\xi^k). \tag{2.29}$$

Im kontinuierlichen Fall lässt sich der Erwartungswert über

$$\mathbb{E}[\xi] = \int_{-\infty}^{\infty} \xi dF\xi = \int_{-\infty}^{\infty} \xi f(\xi) d\xi \tag{2.30}$$

bestimmen.[27]

2.2.4 Quantile, Value at Risk (VaR) und Conditional Value at Risk (CVaR)

Für $0 < \alpha < 1$ ist ein Punkt η das α-Quantil von ξ, genau dann wenn

$$\eta = \min\{x \mid F(x) \geq \alpha\} = \min\left\{x \,\middle|\, \int_{-\infty}^{x} f(\xi) d\xi \geq \alpha\right\}. \tag{2.31}$$

Das α-Quantil gibt somit den kleinsten Wert η für ξ an, für den $F(\eta)$ größer ist als α, wobei F die kumulative Verteilung sei.[28]

Für eine stetige Dichtefunktion kann ein Quantil auch durch Gleichheit definiert werden. Gesucht ist das η, für das gilt:

$$\alpha = \int_{-\infty}^{\eta} f(\xi) d\xi = F(\eta). \tag{2.32}$$

Auf diese Weise lässt sich nun auch der so genannte *Value at Risk* (VaR) definieren. Der VaR ist ein v. a. in der Finanzbranche gängiges Risikomaß, also eine Methodik zum Umgang mit Unsicherheit einer Zufallsvariable. Der VaR wird hier auf einer Verlustverteilung definiert, das bedeutet, dass das α-Quantil die besten $(\alpha \times 100)\%$ Realisierungen angibt. Die schlechten Realisierungen liegen auf der rechten Seite der Dichtefunktion. Daher wird der VaR zu einem $0 < \alpha < 1$ wie folgt definiert:

$$VaR_{1-\alpha}(\xi) = \inf\{x \mid F(x) \geq 1 - \alpha\}. \tag{2.33}$$

Verbal kann man den $VaR_{1-\alpha}$ auch als den Wert aus dem Definitionsbereich der Zufalls-

[26] Englisch: *Expected Value* (EV).
[27] Vgl. [Birge und Louveaux, 2011, S. 57].
[28] Vgl. [Birge und Louveaux, 2011, S. 57].

variable bezeichnen, der zu $((1-\alpha)\times 100)\%$ nicht unterschritten wird.[29] Abbildung 2 zeigt die Lage des VaR am Beispiel der Normalverteilung.

Der *Conditional Value at Risk* (CVaR) ist eine Erweiterung dieses Konzeptes. Hier werden für stetige Verteilungen die schlechtesten $(\alpha \times 100)\%$ der Ergebnisse aus Ω betrachtet und deren Erwartungswert bestimmt:[30]

$$CVaR_{1-\alpha}(\xi) = \mathbb{E}\left[x \mid x \geq VaR_{1-\alpha}(\xi)\right]. \tag{2.34}$$

Abbildung 2 zeigt die Lage des CVaR am Beispiel der Normalverteilung.

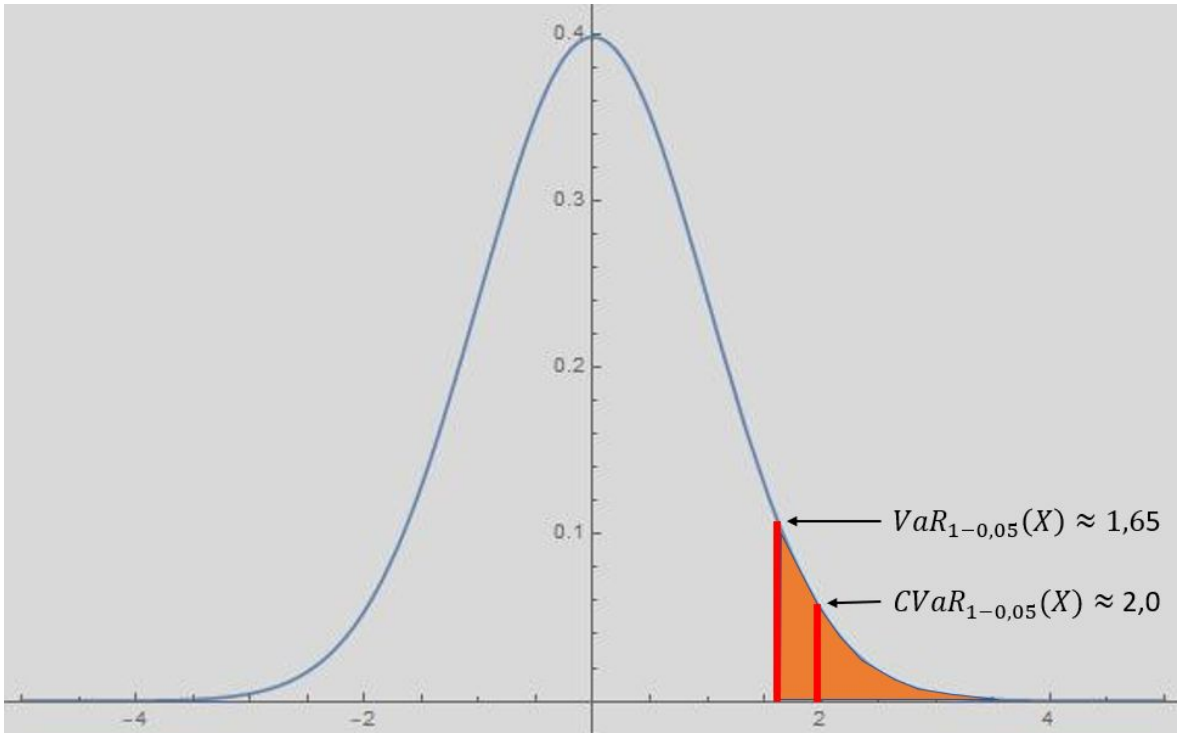

Abbildung 2: Veranschaulichung des VaR und CVaR für eine Verlustfunktion. Die Zufallsvariable X sei normalverteilt mit Erwartungswert $\mu = 0$ und Standardabweichung $\sigma = 1$. α sei 0,05. Gemäß der Sigma-Regeln liegt der $VaR_{1-0,05}(X)$ etwa bei 1,65. $CVaR_{1-0,05}(X)$ ist der Erwartungswert über alle Ergebnisse, die schlechter als $VaR_{1-0,05}(X)$ sind, hervorgehoben durch den orangenen Bereich, und liegt bei etwa 2.

Der CVaR lässt sich auch für diskrete Verteilungen geeignet wie folgt definieren:

$$CVaR_{1-\alpha}(X) = \min_{c}\left[c + \frac{1}{\alpha}\cdot \mathbb{E}\left[(X-c)^{+}\right]\right], \tag{2.35}$$

wobei $(t)^{+} = \max\{0, t\}$.[31]

[29] Vgl. [Sarykalin et al., 2008, S. 272].
[30] Vgl. [Sarykalin et al., 2008, S. 273].
[31] Vgl. [Sarykalin et al., 2008, S. 274].

Dieser Formulierung liegt offenbar eine Optimierungsaufgabe zu Grunde. Gesucht ist das c, für das der Ausdruck innerhalb der eckigen Klammern minimal ist.

Ein einfaches Beispiel zeigt den Unterschied zwischen (2.34) und (2.35) auf. Sei X eine reelle, diskrete Zufallsvariable mit drei möglichen Ereignissen:

$$\begin{aligned} \mathbb{P}(X = 8) &= 0,92, \\ \mathbb{P}(X = 9) &= 0,06, \\ \mathbb{P}(X = 10) &= 0,02. \end{aligned} \tag{2.36}$$

- Sei zunächst $\alpha = 0,08$. Dann ergibt sich gemäß der obigen Definitionen:

$$\begin{aligned} VaR_{0,92}(X) &\overset{(2.33)}{=} 9, \\ CVaR_{0,92}(X) &\overset{(2.35)}{=} 9 + \frac{0,06}{0,08} \cdot (9-9) + \frac{0,02}{0,08} \cdot (10-9) = 9,25, \end{aligned} \tag{2.37}$$

  wobei offenbar $c = VaR_{0,92} = 9$ gilt. Für $X = 8$ ergibt sich 0, weshalb dieser Summand weggelassen werden kann. Hier liefert auch die erste Definition des CVaR in (2.34) diese Lösung, weil klar ist, welches die schlechtesten 8% Ereignisse sind. Dieser erste Fall gibt zudem eine intuitive Erklärung für den Skalierungsfaktor $\frac{1}{\alpha}$ in (2.35). Die Summe der Wahrscheinlichkeiten der beiden Szenarien, die betrachtet werden, ist gerade $\alpha = 0,08$. Mit Hilfe des Skalierungsfaktors können daher bei der Berechnung des Erwartungswerts die Eintrittswahrscheinlichkeiten verwendet werden.

- Sei nun $\alpha = 0,05$. Aufgrund der diskreten Verteilung von X ergibt sich für den VaR wieder:

$$VaR_{0,95}(X) \overset{(2.33)}{=} 9. \tag{2.38}$$

  Bei der Berechnung des CVaR liefert (2.35) nun:

$$CVaR_{0,95}(X) \overset{(2.35)}{=} 9 + \frac{0,02}{0,05} \cdot (10-9) = 9,4, \tag{2.39}$$

  wobei wieder $c = VaR_{0,95} = 9$ gilt. Die Verwendung von Definition (2.34) würde hier Schwierigkeiten bereiten, weil nicht klar ist, was die schlechtesten 5% Ereignisse sind. Das atomare Ereignis $X = 9$ müsste aufgesplittet werden[32].

[32] Vgl. dazu [Sarykalin et al., 2008, S. 275 - 276].

2.3 Stochastische Optimierung

Optimierungsmodelle im Bereich der Stochastischen Optimierung[33] enthalten Parameter, deren Werte nicht konstant, also unsicher, sind. Eine gängige Ausprägung für ein SP sind s. g. *two-stage SP*, welche in den folgenden Ausführungen dieser Arbeit betrachtet werden. Bei der ersten Stufe geht es dabei um initiale Entscheidungen, meist mit der Frage verbunden, ob eine bestimmte Maßnahme getroffen werden soll. Daher bieten sich hier häufig ganzzahlige oder auch binäre Variablen an. In der zweiten Stufe wird auf diese dann *reagiert*. Die zu dieser Stufe gehörenden Variablen können kontinuierlich sein. Dieses Konzept ist wesentlich, um Benders Dekompositionsverfahren auf SP anzuwenden, was am Ende dieses Abschnitts vorgestellt wird.

Zuvor werden noch einmal die im vorherigen Abschnitt eingeführten Risikomaße EV und CVaR betrachtet, zu denen jeweils ein s. g. Deterministisches Äquivalent formuliert wird. Im Anschluss daran werden die stochastischen Indikatoren *Expected Value of Perfect Information* und *Value of the Stochastic Solution* thematisiert.

2.3.1 Deterministisches Äquivalent zum EV

Im vorherigen Abschnitt 2.2 *Stochastik* wurde die Grundmenge Ω erwähnt. Dabei handelt es sich um die Menge aller möglichen Ergebnisse eines stochastischen Prozesses. Wir haben unterschieden, ob Ω endlich ist oder nicht. Im Folgenden wollen wir nur den endlichen Fall betrachten. Daraus lässt sich schnell das s. g. *Deterministische Äquivalent* ableiten. Dabei wird jedes Szenario $\omega \in \Omega$ mit einem Subproblem identifiziert und dessen Zielfunktionswert gewichtet mit der Eintrittswahrscheinlichkeit p_ω von ω in der Zielfunktion addiert. Darüber hinaus werden sämtliche Parameter und Variablen, deren Werte von Szenarien abhängen, als solche gekennzeichnet und tauchen nur in den zugehörigen Subproblemen auf. Für die first-stage mit y als ganzzahlige first-stage-Variable entsteht allgemein das folgende Optimierungsmodell:

$$\begin{aligned} \min_y \; & c^T y + Q(y) \\ \text{s.t. } & Ay \leq b \\ & y \geq 0, \end{aligned} \tag{2.40}$$

wobei $Q(y)$ eine reellwertige Abbildung ist, welche die Subprobleme in der Zielfunktion repräsentiert.

Dazu kommen $|\Omega|$ lineare Subprobleme, die in allgemeiner Form mit passenden Vek-

[33] Englisch: *Stochastic Program* (SP).

toren q_ω, h_ω und Matrizen W_ω, T_ω sowie den second-stage Variablen v_ω beschrieben sind:

$$\begin{aligned} Q(y,\omega) = \min_{v_\omega} \; & q_\omega^T v_\omega \\ \text{s.t. } & W_\omega v_\omega \leq h_\omega - T_\omega y \\ & v_\omega \geq 0. \end{aligned} \tag{2.41}$$

Diese beiden Probleme (2.40) und (2.41) lassen sich nun verknüpfen via

$$Q(y) = \mathbb{E}\left[Q(y,\omega)\right] \overset{(2.29)}{=} \sum_{\omega\in\Omega} p_\omega \cdot q_\omega^T v_\omega. \tag{2.42}$$

Daraus lässt sich nun das Deterministische Äquivalent ableiten:[34]

$$\begin{aligned} RP = \min_{y,v_\omega} \; & c^T y + \sum_{\omega\in\Omega} p_\omega \cdot q_\omega^T v_\omega \\ \text{s.t. } & Ay \leq b \\ & y \geq 0 \\ & W_\omega v_\omega \leq h_\omega - T_\omega y \qquad \forall \omega \in \Omega \\ & v_\omega \geq 0 \qquad\qquad\qquad\quad \forall \omega \in \Omega. \end{aligned} \tag{2.43}$$

Die Lösung des DE nennt man auch *expected recourse problem solution value*, kurz RP[35], oder *here-and-now-solution*[36].

Anstelle des Erwartungswertes in (2.42) kann auch der VaR bzw. der CVaR an dieser Stelle betrachtet werden: je nachdem, welches Risikomaß von Interesse ist.

2.3.2 Deterministisches Äquivalent zum CVaR

In der Definition des CVaR in (2.35) ist ein Optimierungsproblem gegeben, welches sich zur Formulierung der Subprobleme benutzen lässt. Seien dazu p_ω die Eintrittswahrscheinlichkeit für ein $\omega \in \Omega$ und $Q(y,\omega)$ wie in (2.41). t sei die zu minimierende Variable c in (2.35). Um zu modellieren, dass nur über diejenigen ω gemittelt wird, für die $Q(y,\omega) \geq t$, also schlechter als t, ist, wird eine positive Variable $d_\omega \geq Q(y,\omega) - t$ eingeführt. Das Relationszeichen $\geq$ in dieser Ungleichung sorgt dafür, dass für $Q(y,\omega) < t$ der Wert von d_ω auf 0 gesetzt wird. Es ergibt sich zu einem gegebenen $0 < \alpha < 1$ für

[34] Vgl. auch für die vorangegangenen Absätze [Murphy, 2013, S. 14 - 19].
[35] Vgl. [Birge und Louveaux, 2011, S. 24]
[36] Vgl. [Birge und Louveaux, 2011, S. 164]

$Q(y)$:

$$\begin{aligned} Q(y) = CVaR_\alpha(Q(y,\omega)) = \quad & \min_{t,d_\omega} t + \frac{1}{\alpha}\sum_{\omega\in\Omega} p_\omega \cdot d_\omega \\ & \text{s.t. } d_\omega \geq Q(y,\omega) - t \\ & d_\omega \geq 0 \end{aligned} \tag{2.44}$$

Im Vergleich zu (2.42) wurde also der Erwartungswert durch CVaR ersetzt. Es ist leicht zu sehen, dass insgesamt das folgende DE entsteht:[37]

$$\begin{aligned} \min_{y,v_\omega,t,d_\omega} \quad & c^T y + \left(t + \frac{1}{\alpha}\sum_{\omega\in\Omega} p_\omega \cdot d_\omega \right) \\ \text{s.t. } & Ay \leq b \\ & y \geq 0 \\ & W_\omega v_\omega \leq h_\omega - T_\omega y && \forall \omega \in \Omega \\ & v_\omega \geq 0 && \forall \omega \in \Omega \\ & d_\omega \geq q_\omega^T v_\omega - t && \forall \omega \in \Omega \\ & d_\omega \geq 0 && \forall \omega \in \Omega. \end{aligned} \tag{2.45}$$

2.3.3 Expected Value of Perfect Information (EVPI) und Value of the Stochastic Solution (VSS)

Bei SPs stellt sich immer die Frage, ob die Verwendung von stochastischen Konzepten an dieser Stelle überhaupt zielführend ist. Es kann durchaus vorkommen, dass der Einfluss der stochastischen Parameter geringer ist als der der deterministischen Parameter, sodass die tatsächliche Realisierung eines stochastischen Parameters für das Optimierungsproblem keine oder nur eine geringe Rolle spielt. Um dies festzustellen, werden im Folgenden die beiden stochastischen Indikatoren EVPI und VSS präsentiert, die ein Maß darüber geben, wie viel die Unsicherheit in einem gegebenen Modell kostet.

EVPI

Zunächst ist von Interesse, herauszufinden, was unter optimalen Umständen für jedes Szenario der optimale Zielfunktionswert ist. Von diesen wird dann mit der zugehörigen Eintrittswahrscheinlichkeit des Szenarios der Erwartungswert bestimmt. Dieses Ergebnis wird mit RP verglichen. Es lässt sich schnell berechnen, indem die szenariounabhängigen Entscheidungsvariablen aus der first-stage einen Szenario-Index erhalten.

[37] Vgl. [GAMS, 2018, S. 1472 - 1473].

Bleibt das restliche Modell unverändert, so sieht (2.43) nun wie folgt aus:

$$\begin{aligned} WS = \min_{y_\omega, v_\omega} & \sum_{\omega \in \Omega} p_\omega \cdot (c^T y_\omega + q_\omega^T v_\omega) \\ \text{s.t. } & Ay_\omega \leq b \\ & y_\omega \geq 0 \\ & W_\omega v_\omega \leq h_\omega - T_\omega y \qquad \forall \omega \in \Omega \\ & v_\omega \geq 0 \qquad \forall \omega \in \Omega. \end{aligned} \tag{2.46}$$

Diese Lösung wird auch *wait-and-see-solution*, kurz WS[38], genannt. Es sollte klar sein, dass RP im Sinne der Optimierungsaufgabe nicht besser, also in dem Fall nicht kleiner, ist als WS, weil dieser Wert unter optimalen Umständen für jedes Szenario bestimmt wurde. Die folgende Differenz ist demnach positiv und definiert den EVPI:

$$EVPI = RP - WS. \tag{2.47}$$

Der EVPI entspricht somit den Kosten, die entstehen, dass die Szenariowahl nicht vorhergesagt werden kann.[39]

VSS

Bei der Berechnung des EVPI wurde der Erwartungswert der optimalen Zielfunktionswerte für jedes Szenario ω bestimmt. Es gibt aber auch den Ansatz, dass mit dem Anwenden des Erwartungswerts begonnen wird, d. h. jedem szenarioabhängige Parameter wird sein Erwartungswert zugewiesen. Mit diesem nun szenariounabhängigen Wert kann ein wesentlich kleineres MIP gelöst werden, da in (2.43) die Summe in der Zielfunktion auf einen einzigen Summanden zusammenfällt und bei den Restriktionen einiges wegfällt. Zudem genügt eine second-stage-Variable v ohne Szenarioindex. Das aus (2.43) hervorgehende MIP sieht wie folgt aus:

$$\begin{aligned} EV = \min_{y,v} & \; c^T y + \mathbb{E}[q]^T v \\ \text{s.t. } & Ay \leq b \\ & y \geq 0 \\ & \mathbb{E}[W]v \leq \mathbb{E}[h] - \mathbb{E}[T]y \\ & v \geq 0. \end{aligned} \tag{2.48}$$

[38] Vgl. [Birge und Louveaux, 2011, S. 164].
[39] Vgl. auch für alle vorangegangenen Absätze [Birge und Louveaux, 2011, S. 163 - 165].

Diese Lösung wird auch *Expected Value Solution*, kurz EV[40], genannt. Nun werden die Variablen der first-stage festgehalten und mit dieser Entscheidung wird die second-stage optimiert. Sei dazu $\bar{y}$ die optimale Lösung, die in (2.48) zu EV geführt hat. Dann sieht das zu lösende MIP wie folgt aus:

$$\begin{aligned} EEV = \min_{v_\omega} & \sum_{\omega \in \Omega} p_\omega \cdot (c^T \bar{y} + q_\omega^T v_\omega) \\ \text{s.t. } & W_\omega v_\omega \leq h_\omega - T_\omega \bar{y} && \forall \omega \in \Omega \\ & v_\omega \geq 0 && \forall \omega \in \Omega. \end{aligned} \tag{2.49}$$

Das Ergebnis dieser Optimierung sei EEV, abgeleitet von *expected result of using the EV solution*[41]. Der Wert VSS wird wie schon der EVPI durch den Abstand zu RP ermittelt. Dabei sollte klar sein, dass die Lösung EEV nicht besser sein kann als RP.

$$VSS = EEV - RP \tag{2.50}$$

Der Wert VSS entspricht somit den Kosten, die entstehen, wenn die Unsicherheit bei der Entscheidung in der first-stage ignoriert wird.[42]

Diese beiden Konzepte helfen nun, um zu entscheiden, ob ein gegebenes Problem als SP gelöst werden sollte: Dies sollte nur dann der Fall sein, wenn $EVPI$ und VSS groß sind, einen konkreten relativen oder absoluten Richtwert gibt es allerdings nicht.[43]

2.3.4 Benders' Dekompositionsverfahren für SP

In Abschnitt 2.1.2 wurde das grundsätzliche Prinzip von Benders' Dekompositionsverfahren vorgestellt. Wie dabei anfänglich beschrieben, sollte das MIP in getrennte Teilprobleme aufzuteilen sein. Das DE aus (2.43) entspricht genau dieser Form, wobei es nun $|\Omega|$ viele Teilprobleme gibt. Die Verfahrensidee bleibt aber die gleiche: anstatt der Restriktionen des Originalproblems werden dem Masterproblem iterativ s. g. *Cuts* betrachtet. Dabei sind zwei Sorten von Cuts zu unterscheiden. Zum einen gibt es *Feasibility Cuts*, die dafür sorgen, dass die Entscheidung in der first-stage auch in der second-stage zulässig ist. Zum anderen ermöglichen *Optimality Cuts* das Finden der optimalen Lösung.[44]

In einem ersten Teil geht es nun um das Ermitteln der Feasibility Cuts. Anschließend

[40] Vgl. [Birge und Louveaux, 2011, S. 165].
[41] Vgl. [Birge und Louveaux, 2011, S. 165].
[42] Vgl. auch für alle vorangegangenen Absätze [Birge und Louveaux, 2011, S. 165].
[43] Vgl. [Birge und Louveaux, 2011, S. 170].
[44] Vgl. [Murphy, 2013, S. 20].

werden die Optimality Cuts hergeleitet. Darauf aufbauend lassen sich untere und obere Schranken für den gesuchten Zielfunktionswert finden. Schließlich wird in diesem Abschnitt noch ein Algorithmus für Benders' Dekompositionsverfahren angegeben.

Zunächst sei noch einmal das DE aus (2.43), unser Ausgangs-MIP, gegeben:

$$\begin{aligned} RP = \min_{y,v_\omega} \; & c^T y + \sum_{\omega \in \Omega} p_\omega \cdot q_\omega^T v_\omega \\ \text{s.t. } & Ay \leq b \\ & y \geq 0 \\ & W_\omega v_\omega \leq h_\omega - T_\omega y \qquad \forall \omega \in \Omega \\ & v_\omega \geq 0. \qquad\qquad\qquad \forall \omega \in \Omega \end{aligned}$$

Nun wird ein y fest aber beliebig gewählt; dies stellt die initiale first-stage-Entscheidung dar. Bezogen auf diese Entscheidung werden die Teilprobleme betrachtet. Existiert ein Teilproblem, für das die Entscheidung aus der first-stage dieses Teilproblem unzulässig lassen werden, so müssen Feasibility Cuts dem Masterproblem hinzugefügt werden. Andernfalls, wenn also alle Teilprobleme zulässig sind, werden Optimality Cuts hinzugefügt.[45]

Feasibility Cuts

In (2.41) wurden die Teilprobleme abhängig für ein gegebenes y angegeben. Um festzustellen, ob das Teilproblem zulässig ist, können s. g. nichtnegative Schlupfvariablen v_ω^+ und v_ω^- in den Restriktionen eingefügt werden. Es wird das folgende neue Problem formuliert:

$$\begin{aligned} V(y,\omega) = \min_{v_\omega, v_\omega^+, v_\omega^-} \; & e^T (v_\omega^+ + v_\omega^-) \\ \text{s.t. } & W_\omega v_\omega + v_\omega^+ - v_\omega^- \leq h_\omega - T_\omega y \quad \forall \omega \in \Omega \\ & v_\omega \geq 0 \qquad\qquad\qquad\qquad\qquad \forall \omega \in \Omega \\ & v_\omega^+ \geq 0 \qquad\qquad\qquad\qquad\qquad \forall \omega \in \Omega \\ & v_\omega^- \geq 0 \qquad\qquad\qquad\qquad\qquad \forall \omega \in \Omega \end{aligned} \tag{2.51}$$

wobei der Vektor e nur Einsen enthält. Sind alle Restriktionen $\leq$-Ungleichungen, so ist v_ω^+ in allen Komponenten Null, da die Erhöhung dieser Schlupfvariable erst recht nicht für die Erfüllung dieser Restriktion sorgen würde. Ist dahingegen eine Restriktion für eine first-stage-Entscheidung nicht erfüllt, muss die linke Seite verkleinert werden, was durch das Abziehen des positiven v_ω^- gelingt. Es gilt nun: Ist $V(y,\omega) = 0$, so ist

[45] Vgl. [Murphy, 2013, S. 20].

das zu ω gehörende Teilproblem zulässig, da offenbar alle Restriktionen erfüllt sind. Andernfalls, wenn $V(y,\omega) > 0$, ist das zu ω gehörende Teilproblem unzulässig in der ursprünglichen Form. Betrachte nun das duale Problem zu (2.51):

$$\begin{aligned} V(y,\omega) = \max_{\sigma} \; & (h_\omega - T_\omega y)^T \sigma \\ \text{s.t. } & W_\omega^T \sigma \leq 0 \\ & \sigma \leq e \\ & -\sigma \leq e. \end{aligned} \tag{2.52}$$

Gemäß des Strong Duality Theorem müssen die Zielfunktionswerte dieser beiden Probleme gleich sein. Das bedeutet, wenn $V(y,\omega) > 0$ im dualen Problem, so ist das entsprechende Teilproblem unzulässig. Um genau dies auszuschließen, wird die optimale Lösung σ^* von (2.52) gewählt und gefordert:

$$\begin{aligned} 0 &\geq (h_\omega - T_\omega y)^T \sigma^* \\ &= h_\omega^T \sigma^* - (T_\omega y)^T \sigma^* \\ &= \underbrace{h_\omega^T \sigma^*}_{=:d} - \underbrace{(T_\omega \sigma^*)^T}_{=:D} y. \end{aligned} \tag{2.53}$$

Die Ungleichung

$$0 \geq d - Dy \tag{2.54}$$

wird dem Masterproblem als Restriktion hinzugefügt.[46]

Wenn in (2.43) keine Wahl von y zu unzulässigen Teilproblemen führt, werden dem Masterproblem natürlich keine Feasibility Cuts hinzugefügt.[47]

Optimality Cuts

Betrachte wieder (2.40) und (2.41). Wir suchen nun iterativ nach unteren Schranken für einen zu minimierenden Wert z in unserem Masterproblem:

$$\begin{aligned} \min_{y} \; & c^T y + z \\ \text{s.t. } & Ay \leq b \\ & y \geq 0. \end{aligned} \tag{2.55}$$

Dabei soll y eine zulässige Entscheidung für die second-stage-Probleme sein. z ersetzt $Q(y)$ in (2.40), wobei $z \geq Q(y)$ gelten soll, weshalb wir eine untere Schranke für $Q(y)$

[46] Vgl. auch für alle vorangegangenen Absätze [Rico-Ramirez, 2002, S. 9 - 10].
[47] Vgl. [Murphy, 2013, S. 25].

suchen werden. Dazu werden die dualen Probleme der Teilprobleme in (2.41)

$$\begin{aligned} Q(y^i, \omega) = \min_{v_\omega} \; & q_\omega^T v_\omega \\ \text{s.t. } & W_\omega v_\omega \leq h_\omega - T_\omega y^i \\ & v_\omega \geq 0 \end{aligned} \tag{2.56}$$

betrachtet:

$$\begin{aligned} Q(y^i, \omega) = \max_{\pi^i} \; & (h_\omega - T_\omega y^i)^T \pi^i \\ \text{s.t. } & W_\omega^T \pi^i \leq q_\omega \\ & \pi^i \leq 0, \end{aligned} \tag{2.57}$$

wobei y^i die gewählte Entscheidung in der i-ten Iteration des Benders' Algorithmus sei. Das π^i ist nicht-positiv, da das primale Problem eine Minimierungsaufgabe ist und die Restriktionen durch eine $\leq$-Relation beschrieben werden. Seien darüber hinaus π_ω^i die Optimalpunkte für π^i, um $Q(y^i, \omega)$ in (2.57) zu maximieren. Es gilt also:

$$Q(y^i, \omega) = (h_\omega - T_\omega y^i)^T \pi_\omega^i. \tag{2.58}$$

Damit lässt sich $Q(y^i)$ wie folgt schreiben:

$$Q(y^i) = \sum_{\omega \in \Omega} p_\omega \cdot Q(y^i, \omega) = \sum_{\omega \in \Omega} p_\omega \cdot (h_\omega - T_\omega y^i)^T \pi_\omega^i. \tag{2.59}$$

Nun muss noch ein Zusammenhang zwischen $Q(y)$ und $Q(y^i)$ gefunden werden. Für jedes ω ist die nur von y abhängige Funktion $Q(y, \omega)$ konvex. Sie kann außerdem stückweise über die Restriktionen beschrieben werden. Jedes π_ω^i repräsentiert genau eine Restriktion, und zwar jene, die verhindert, dass $Q(y^i, \omega)$ in (2.56) weiter minimiert wird. Diese Restriktion gilt für alle y und kann daher auch auf alle y erweitert werden. Die dadurch gegebene Restriktion bildet wegen der Konvexität von $Q(y, \omega)$ eine untere Abschätzung für $Q(y, \omega)$:

$$Q(y, \omega) \geq (h_\omega - T_\omega y)^T \pi_\omega^i. \tag{2.60}$$

Aufsummieren liefert eine untere Schranke für $Q(y)$:

$$Q(y) = \sum_{\omega \in \Omega} p_\omega \cdot Q(y, \omega) \geq \sum_{\omega \in \Omega} p_\omega \cdot (h_\omega - T_\omega y)^T \pi_\omega^i \tag{2.61}$$

Damit muss für z gelten:

$$\begin{aligned} z &\geq \sum_{\omega \in \Omega} p_\omega \cdot (h_\omega - T_\omega y)^T \pi_\omega^i \\ &= \sum_{\omega \in \Omega} p_\omega \cdot \left(h_\omega^T \pi_\omega^i - (T_\omega y)^T \pi_\omega^i \right) \\ &= \sum_{\omega \in \Omega} p_\omega \cdot h_\omega^T \pi_\omega^i - \sum_{\omega \in \Omega} p_\omega \cdot (T_\omega y)^T \pi_\omega^i \\ &= \underbrace{\sum_{\omega \in \Omega} p_\omega \cdot h_\omega^T \pi_\omega^i}_{=:g} - \underbrace{\sum_{\omega \in \Omega} p_\omega \cdot (T_\omega \pi_\omega^i)^T}_{=:G} y. \end{aligned} \tag{2.62}$$

Daraus lassen sich die Optimality Cuts der Form

$$z \geq g - Gy \tag{2.63}$$

ableiten, welche dem Masterproblem hinzugefügt werden.[48]

Obere und untere Schranken für den gesuchten Zielfunktionswert

Während jeder Iteration werden das Masterproblem (2.55) und die Subprobleme in (2.56) gelöst für ein y^i. Das y^i ergibt sich allgemein durch ein Lösen des Masterproblems in der $(i-1)$-ten Iteration. In der Startiteration wird y^1 beliebig gewählt.

Zunächst werden mit dem gegebenen y^i sämtliche Subprobleme gelöst. Es resultiert eine obere Schranke UB für die Lösung des Masterproblems:

$$UB = \underbrace{z^{i-1} - z}_{\text{aus dem Masterproblem}} + \underbrace{\sum_{\omega \in \Omega} p_\omega \cdot q_\omega^T v_\omega^i}_{\text{aus den Subproblemen}} , \tag{2.64}$$

wobei v_ω^i die optimale Lösung des zu ω gehörenden Subproblems in der i-ten Iteration sei. Das $z^{i-1} - z$ lässt sich auf einfache Weise über

$$z^{i-1} - z = c^T y^i \tag{2.65}$$

bestimmen. Der gesuchte Zielfunktionswert mit einem optimalen y^* kann nicht über UB liegen, da hier ein beliebiges y^i betrachtet wurde.

Nachdem die Subprobleme ausgewertet wurden und dem Masterproblem ein Feasibility oder Optimality Cut hinzugefügt wurde, kann das Masterproblem optimiert werden.

[48] Vgl. auch für alle vorangegangenen Absätze [Murphy, 2013, S. 29 - 33].

Der Zielfunktionswert z^i des Masterproblems bildet eine untere Schranke für den tatsächlich gesuchten Zielfunktionswert, also

$$LB = z^i, \tag{2.66}$$

da sich durch fehlende Restriktionen nur ein besseres Ergebnis einstellen kann.[49]

Benders Algorithmus (Übersicht)

1. **Initialisierung**:
   - Setze $i = 1$.
   - Setze $LB = -\infty$.
   - Setze $UB = \infty$.
   - Setze y^1 auf einen beliebigen zulässigen Wert. Dieser kann auch der optimalen Lösung des (für z noch restriktionsfreie) Masterproblems entsprechen.
   - Bestimme z^0 über $z^0 = c^T y^1 + z$, also den Zielfunktionswert des Masterproblems. Durch die Minimierung dieser sollte klar sein, dass hier $z = 0$ gilt.
2. **Iteration**:
   - Löse die Subprobleme mit gegebenen y^i. Seien v_ω die optimale Lösung für das zu ω gehörende Subproblem.
   - Füge dem Masterproblem Cuts (entweder Feasibility oder Optimality Cuts) in der oben beschriebenen Weise zu.
   - Aktualisiere die obere Schranke $UB = z^{i-1} - z + \sum_{\omega \in \Omega} p_\omega \cdot q_\omega^T v_\omega^i$.
   - **Abbruchbedingung**: Wenn $UB \leq LB$, dann beende Algorithmus mit y^i als optimaler Lösung, sonst:
   - Löse das aktualisierte Masterproblem und aktualisiere die untere Schranke $LB = z^i$. Merke dir den ermittelten Wert für z und setze y^{i+1} als Optimalpunkt des Masterproblems.
   - Erhöhe i um 1 und beginne nächste Iteration.

Um eine u. U. lange Rechenzeit zu vermeiden, kann die Anzahl an gewünschten Iterationen beschränkt werden. Alternativ kann in der Abbruchbedingung auch eine positive Toleranz ε angegeben werden via $UB \leq LB + \varepsilon$.[50]

[49] Vgl. [Birge und Louveaux, 2011, S. 184 - 185].
[50] Vgl. [Birge und Louveaux, 2011, S. 183 - 184].

Multicuts

In (2.55) wurde *eine* Variable z verwendet, um die Zielfunktionswerte der Subprobleme im Masterproblem einzubauen. Alle Optimality Cuts lieferten eine untere Schranke für dieses z, wobei beim Bestimmen der Parameter g und G über alle Szenarien iteriert wurde. Die Cuts werden daher auch *Singlecuts* genannt. Alternativ lassen sich dafür auch $|\Omega|$ viele Variablen z^ω verwenden, die jeweils durch jeden Optimality Cut nach unten abgeschätzt werden. Diese szenarioabhängigen Cuts heißen *Multicuts*. Das allgemeine Masterproblem aus (2.55) lässt sich dann wie folgt formulieren:

$$\begin{aligned} \min_{y,z^\omega} \; & c^T y + \sum_{\omega \in \Omega} z^\omega \\ \text{s.t. } & Ay \leq b \\ & y \geq 0. \end{aligned} \tag{2.67}$$

Die Optimality Cuts aus (2.62) werden ebenfalls angepasst. Für jedes $\omega \in \Omega$ ergibt sich nun

$$\begin{aligned} z^\omega &\geq p_\omega \cdot (h_\omega - T_\omega y)^T \pi_\omega^i \\ &= p_\omega \cdot \left(h_\omega^T \pi_\omega^i - (T_\omega y)^T \pi_\omega^i \right) \\ &= p_\omega \cdot h_\omega^T \pi_\omega^i - p_\omega \cdot (T_\omega y)^T \pi_\omega^i \\ &= \underbrace{p_\omega \cdot h_\omega^T \pi_\omega^i}_{=:g} - \underbrace{p_\omega \cdot (T_\omega \pi_\omega^i)^T}_{=:G} y. \end{aligned} \tag{2.68}$$

Daraus lassen sich wieder wie in gewohnter Weise die folgenden Optimality Cuts ableiten, die dieses Mal für jedes Szenario angegeben werden können:

$$z^\omega \geq g - Gy. \tag{2.69}$$

Die Hinzunahme solcher Cuts lässt spezifischere Informationen zu jedem Szenario zu, sodass die Anzahl an notwendigen Iterationen als geringer zu erwarten ist. Dies bestätigen auch empirische Untersuchungen[51]. Allerdings wird durch die Multicuts auch das Masterproblem umfangreicher, da weitaus mehr Cuts hinzugefügt werden. Für jedes konkrete Problem muss hier ein Kompromiss gefunden werden, der es möglicherweise auch vorsieht, die beiden vorgestellten Konzepte miteinander zu vermischen.[52]

[51] Vgl. [Birge und Louveaux, 2011, S. 200].
[52] Vgl. [Birge und Louveaux, 2011, S. 198 - 200] und [Murphy, 2013, S. 37].

Kapitel 3

Das Facility Location Problem (FLP)

In diesem Kapitel wird zunächst das klassische FLP hergeleitet, welches einer Standortplanung entspricht. Anschließend wird es um die Aspekte Kapazität, Mehrperiodizität und Unsicherheit erweitert. Schließlich wird das Modell erarbeitet, das als Grundlage für die empirischen Untersuchungen im nächsten Kapitel dient.

3.1 Das klassische FLP

Im klassischen FLP wird nach optimalen Standorten für Verwertungsstätten gesucht. Dabei gibt es eine Menge an denkbaren Einrichtungen[53] I gekoppelt an einem bestimmten geographischen Ort, von denen eine Teilmenge ausgewählt werden soll, sowie eine Menge an Kunden J, deren Bedarf d_j für $j \in J$ bestmöglich erfüllt werden soll. Ziel ist zudem, einen Transportplan zu erstellen, aus dem hervorgeht, wie viel jeder Kunde j von einer bestimmten Einrichtung $i \in I$ erhält.

Die Eröffnung einer Einrichtung bringt Fixkosten f_i mit sich. Zwischen jeder Einrichtung i und jedem Kunden j sind stückzahlabhängige Transportkosten $c_{i,j}$ angegeben. Die Erfüllung einer Einheit des Bedarfs bei einem Kunden i gibt den Ertrag r_i.

[53] Englisch: *facilities*, manchmal auch *plants*, weshalb das FLP manchmal auch als *Plant Location Problem* (PLP) bezeichnet wird.

Hieraus lässt sich schnell das folgende Optimierungsmodell ableiten:

$$\begin{aligned}
\min \quad & \sum_{i \in I} f_i y_i + \sum_{j \in J} (c_{i,j} - r_j) d_j x_{i,j} \\
\text{s.t.} \quad & \sum_{i \in I} x_{i,j} = 1 && \forall j \in J \\
& \sum_{j \in J} x_{i,j} \leq y_i && \forall i \in I \\
& 0 \leq x_{i,j} \leq 1 \\
& y_i \in \{0, 1\}.
\end{aligned} \tag{3.1}$$

Die Zielfunktion minimiert die entstehenden Kosten. Die Entscheidungsvariablen $x_{i,j}$ geben den Anteil am Gesamtbedarf bei j an, der von i erfüllt wird. Diese Variable muss somit zwischen 0 und 1 liegen, kann dort aber jeden Wert annehmen. Um den Bedarf bei j zu erfüllen, muss für jedes j die Summe der $x_{i,j}$ Eins ergeben, was die erste Restriktion begründet. Die binäre Entscheidungsvariable y_i gibt an, ob Einrichtung i eröffnet werden soll, wobei $y_i = 1$ genau dann gilt, wenn i eröffnet wird. Die zweite Restriktion soll sicherstellen, dass nur von eröffneten Einrichtungen Stücke geliefert werden. Ist etwa $y_i = 0$, so müssen alle $x_{i,j}$ auch 0 sein. Da sowohl ganzzahlige als auch kontinuierliche Variablen vorhanden sind, ist das Modell ein MIP.

3.2 Mulitperiodisches FLP unter Berücksichtigung von Unsicherheiten und Kapazitäten

In der Realität kann es immer wieder vorkommen, dass etwa der Kundenbedarf Schwankungen und Unsicherheiten untersetzt ist. Es ist auch denkbar, dass die Einrichtungen durch Belegschaftsstreik oder Baumaßnahmen nur über eine bestimmte Kapazität verfügen, die über verschiedene Zeitperioden variieren kann. Damit all das auch mathematisch formuliert werden kann, muss das in (3.1) gegebene Modell um diverse Parameter und Variablen, aber auch Mengen erweitert werden.

Berücksichtigung von Kapazitäten

Jede Einrichtung i erhält einen Parameter K_i, der ausdrückt, wie viele Stückzahlen maximal in dieser Einrichtung verwertet werden können. Durch die Hinzunahme einer weiteren Restriktion lässt sich die Erfüllung dieser Kapazität erzwingen:

$$\sum_{j \in J} d_{i,j} x_{i,j} \leq K_i y_i \qquad \forall i \in I. \tag{3.2}$$

Beachte, dass $x_{i,j}$ nur den Anteil am Gesamtbedarf eines Kunden j angibt, der von i erfüllt wird.[54]

Mehrperiodizität

Um eine mehrperiodische Betrachtungsweise zu ermöglichen, sollte eine Menge T an Zeitperioden eingebaut werden. Eine Zeitperiode ist meistens ein Intervall bzw. ein Zeitraum, etwa eine Woche oder ein Monat.

Es ist denkbar, Einrichtungen gekoppelt an eine Zeitperiode zu eröffnen und wieder zu schließen, oder andererseits die Bedingung zu formulieren, dass eine einmal geöffnete Einrichtung diesen Zustand beibehält.

In dem hier präsentierten Modell soll ein wiederum anderer Ansatz verfolgt werden: die Eröffnung einer Einrichtung i soll zu Beginn der ersten Periode feststehen, d. h. die y-Variablen sind zeitunabhängig. Dahingegen kann es sein, dass der Kundenbedarf schwankt. Daher müssen $d_{t,j}$ und insbesondere auch $x_{t,i,j}$ zeitabhängig werden.[55]

Des Weiteren wird dem Modell noch der Parameter $\gamma_{t,i}$ hinzugefügt, der den Anteil der verfügbaren Kapazität bei Einrichtung i im Zeitraum t angibt. Dieser Parameter kann wie oben bereits erwähnt etwa durch die gegebene Gefahr von Streiks, Umweltkatastrophen oder Baumaßnahmen motiviert werden.[56]

Servicegrad

Auf einfache Art und Weise lässt sich nun auch ein Servicegrad einbauen. In den obigen Varianten war stets gefordert, dass der gesamte Bedarf der Kunden erfüllt werden muss. Wegen der Fixkosten wäre es äußerst unrentabel für einen möglicherweise sehr unbedeutenden Kunden mit geringem Bedarf eine eigene Einrichtung zu eröffnen. Daher ist es durchaus nicht unüblich, einen bestimmten Servicegrad vorzugeben, der nicht bei 100% liegt und nicht unterschritten werden soll. Der Servicegrad wird über alle Kunden aggregiert bestimmt[57]:

$$\frac{\text{Menge an erfüllte Lieferungen}}{\text{Gesamtbedarf der Kunden}} \times 100. \tag{3.3}$$

Bei Betrachtung der Modellparameter und -variablen gilt für den erreichten Servicelevel β:

[54] Vgl. [Heckmann, 2016, S. 316 u. 319].
[55] Vgl. [Heckmann, 2016, S.319 - 320].
[56] Vgl. [Heckmann, 2016, S. 322].
[57] Vgl. [Nyhuis und Wiendahl, 2012, S. 276 - 277].

$$\beta = \frac{\sum_{t \in T} \sum_{j \in J} \sum_{i \in I} d_{t,j} \; x_{t,i,j}}{\sum_{t \in T} \sum_{j \in J} d_{t,j}}. \tag{3.4}$$

Es empfiehlt sich, eine positive Schlupfvariable φ_{tj} einzubauen, die für jeden Kunden zu jeder Zeit die nicht erfüllte Nachfrage als Anteil der Gesamtnachfrage zu diesem Zeitpunkt vermerkt. Nun kann (3.4) umformuliert werden[58]:

$$\beta = 1 - \frac{\sum_{t \in T} \sum_{j \in J} d_{t,j} \; \varphi_{t,j}}{\sum_{t \in T} \sum_{j \in J} d_{t,j}}. \tag{3.5}$$

Um der Realität zu entsprechen, wird ein nicht eingehaltener Servicegrad bestraft. Der gewünschte Servicegrad wird zunächst als skalarer Parameter β_0 mitgeteilt. Der Parameter u gibt darüber hinaus die Strafkosten an, falls β_0 nicht eingehalten wird. Die positive Variable Δ ist die Abweichung des erreichten Servicegrades β von dem erwünschten Servicegrad β_0.[59]

Expandieren der Einrichtungen

Wir wollen für unser Modell auch die Möglichkeit der Expansion einer Einrichtung bereithalten. Dies lässt sich durch eine endliche Menge H an Erweiterungsstufen für die Einrichtungen erreichen. Die Variable z_i gibt dann an, ob bei Einrichtung i eine Erweiterung entstehen soll. Der Parameter b_i gibt die fixen Kosten an, die bei Einrichtung i entstehen, falls eine Erweiterung eingerichtet wird. Diese Kosten könnten etwa durch erforderliche Umbaumaßnahmen entstehen, die für jede Ausbaustufe konstant sind. Mit jeder Erweiterungsstufe $h \in H$ entstehen neben der zusätzlichen Kapazität o_h aber auch zusätzliche variable Kosten v_h.

Um sicherzustellen, dass eine Erweiterung nur an einer geöffneten Einrichtung installiert wird, wird als Nebenbedingung benötigt:

$$z_i \leq y_i. \tag{3.6}$$

Die Variable $\omega_{t,i,h}$ legt fest, welche Erweiterungsstufe h in Periode t bei Einrichtung i gewählt wurde. Es ist klar, dass zu jedem Zeitpunkt nur eine Erweiterung eingerichtet werden kann, was in folgender Bedingung vermerkt wird:

$$\sum_{h \in H} \omega_{t,i,h} \leq z_i \qquad \forall i \in I, t \in T. \tag{3.7}$$

Dabei lässt sich über die Zeitabhängigkeit dieser Entscheidungsvariable diskutieren.

[58] Vgl. [Heckmann, 2016, S. 321].
[59] Vgl. [Heckmann, 2016, S. 317].

Denkbar wäre die Forderung, dass eine gewählte Erweiterung ab dem Zeitpunkt erhalten bleibt, wenn sie das erste Mal benötigt wurde, also für $T = \{t_1, ..., t_n\}$:

$$\omega_{t,i,h} \leq \omega_{t+1,i,h} \quad \forall t \in T \backslash \{t_n\}. \tag{3.8}$$

Auch wenn diese Nebenbedingung von praktischem Interesse ist, lassen wir sie in unserer Betrachtung heraus, sodass wir flexibler pro Zeitraum optimieren können.[60]

Unsicherheit

Eine Möglichkeit, Unsicherheit zu modellieren, ist gegeben durch eine endliche Menge Szenarien S, angelehnt an das DE (vgl. Abschnitt 2.3.1). Jedem Szenario s wird eine Eintrittswahrscheinlichkeit π_s zugewiesen. Dazu werden die Parameter, die vom Szenario abhängen, für jedes Szenario separat angegeben.

Das sind in diesem Fall der Bedarf $d_{s,t,j}$ und die Kapazitätsreduktion der Einrichtungen $\gamma_{s,t,i}$. Beide erhalten somit den Index s.[61] Des Weiteren werden sämtliche Entscheidungsvariablen, die szenarioabhängig sind, mit einem s im Index versehen. Das sind der Anteil der transportierten Menge $x_{s,t,i,j}$, der Anteil der nicht transportierten Menge $\varphi_{s,t,j}$, die Erweiterungsstufe $\omega_{s,t,i,h}$ und der erreichte Servicelevel β_s sowie die Abweichung vom erwünschten Servicelevel Δ_s. Damit sind y_i und z_i die einzigen verbleibenden szenariounabhängigen Variablen.[62]

[60] Vgl. auch für alle vorangegangenen Absätze [Heckmann, 2016, S. 323].
[61] Vgl. [Heckmann, 2016, S. 317].
[62] Vgl. [Heckmann, 2016, S. 318].

Resultierendes Modell

Das resultierende Modell lässt sich wie folgt formulieren[63]:

$$\min \sum_{i \in I} (f_i y_i + b_i z_i) + \sum_{s \in S} \pi_s \left(u \ \Delta_s + \sum_{i \in I} \sum_{h \in H} \sum_{t \in T} \sum_{j \in J} (v_h \omega_{t,i,h}) + (c_{ij} - r_j) d_{s,t,j} x_{s,t,i,j} \right) \tag{3.9}$$

$$\text{s.t.} \quad z_i \leq y_i \qquad \forall i \in I \tag{3.10}$$

$$\sum_{i \in I} x_{s,t,i,j} + \varphi_{s,t,j} = 1 \qquad \forall s \in S, t \in T, j \in J \tag{3.11}$$

$$\sum_{j \in J} x_{t,i,j} \leq y_i \qquad \forall s \in S, t \in T, i \in I \tag{3.12}$$

$$\sum_{j \in J} d_{s,t,j} \ x_{s,i,j} \leq \gamma_{s,t,i} \ K_i \ y_i + \sum_{h \in H} o_h \ \omega_{s,t,i,h} \qquad \forall s \in S, t \in T, i \in I \tag{3.13}$$

$$\sum_{h \in H} \omega_{s,t,i,h} \leq z_i \qquad \forall s \in S, t \in T, i \in I \tag{3.14}$$

$$\beta = 1 - \frac{\sum_{t \in T} \sum_{j \in J} d_{s,t,j} \ \varphi_{s,t,j}}{\sum_{t \in T} \sum_{j \in J} d_{s,t,j}} \qquad \forall s \in S \tag{3.15}$$

$$\Delta_s \geq \beta_0 - \beta \qquad \forall s \in S \tag{3.16}$$

$$0 \leq x_{s,t,i,j} \leq 1 \tag{3.17}$$

$$0 \leq \varphi_{s,t,j} \leq 1 \tag{3.18}$$

$$0 \leq \Delta \leq 1 \tag{3.19}$$

$$y_i \in \{0, 1\} \tag{3.20}$$

$$z_i \in \{0, 1\} \tag{3.21}$$

$$\omega_{s,t,i,h} \in \{0, 1\}. \tag{3.22}$$

3.3 Vorbereitungen für Benders' Dekompositionsverfahren

Benders' Dekompositionsverfahren kann auf das angegebene Modell in der Form noch nicht angewandt werden. Dazu sind noch die folgenden Änderungen notwendig.

Einordnung als two-stage-SP

Grundsätzlich kann das obige Modell jedoch bereits als two-stage-SP eingeordnet werden, denn offenbar besitzt es genau die allgemeine Form eines DE, vgl. (2.43). Die

[63] Vgl. [Heckmann, 2016, S. 326].

szenariounabhängigen Variablen y_i und z_i sind first-stage-Variablen. Die Nebenbedingung (3.10) stellt die einzige Restriktion dar, die ebenfalls der ersten Stufe zugeordnet werden kann. Dagegen sind alle anderen Variablen, die ein s als Index tragen und damit szenarioabhängig sind, der zweiten Stufe zuzuordnen. Auch alle anderen Restriktionen, die für jedes s gelten sollen, gehören der zweiten Stufe an.

Eliminieren der ganzzahligen Variablen in der second-stage

In der Einführung für SP wurde nahegelegt, dass die Variablen der zweiten Stufe kontinuierlich sind, sodass die späteren Subprobleme LP sind. Die binäre Variable $\omega_{s,t,i,h}$ für die Erweiterungsstufen widerspricht dieser Forderung offenbar. Daher bietet es sich hier an, eine Linearisierung der Erweiterungsstufen vorzunehmen. Der Nachteil, dass dadurch nicht mehr explizit vorgegeben werden kann, welche Erweiterungsstufen es geben kann, wird durch den Vorteil der schnelleren Lösbarkeit der Subprobleme ausgeglichen. Die Informationen über die endlich vielen Erweiterungsstufen, gegeben durch die Parameter o_h und v_h, lassen sich allerdings dazu nutzen, eine funktionelle Abhängigkeit zwischen Ausbaumenge und Ausbaupreis herzuleiten. Denkbar wäre, dass für größer werdende Erweiterungsmengen der dafür anstehende Preis relativ gesehen abnimmt (vgl. *economies of scales*). Der Einfachheit halber bestehe aber ein linearer Zusammenhang zwischen Ausbaumenge und Ausbaupreis mit der Proportionalitätskonstanten ce, die sich durch den gemittelten Quotienten $\frac{v_h}{o_h}$ über alle $h \in H$ ergibt. Um einen unendlichen Ausbau zu vermeiden wird der Parameter m als die größte Erweiterungsmenge o_h aus allen h definiert. Anstelle der binären Variable $\omega_{s,t,i,h}$ wird nun eine kontinuierliche Variable $\epsilon_{s,t,i}$ verwendet, deren Wertebereich zwischen 0 und 1 liegt und damit angibt, welcher Anteil des maximal möglichen Ausbaus tatsächlich umgesetzt wurde.

In der Zielfunktion (3.9) werden die entstehenden variablen Kosten der Erweiterungen nun anders bestimmt. Anstelle des Summanden $(v_h \omega_{t,i,h})$ sind die Kosten nun durch

$$ce \cdot m \cdot \epsilon_{s,t,i} \tag{3.23}$$

gegeben. In (3.13) wird die rechte Seite angepasst:

$$\sum_{j \in J} d_{s,t,j}\ x_{s,i,j} \leq \gamma_{s,t,i}\ c_i\ y_i + \boldsymbol{m \cdot \epsilon_{s,t,i}} \quad \forall s \in S, t \in T, i \in I. \tag{3.24}$$

Durch die obere Beschränktheit von $\epsilon_{s,t,i}$ kann zudem auf einfache Art und Weise dafür gesorgt werden, dass nur an den Einrichtungen ein Ausbau besteht, an denen dies erlaubt ist, d. h. deren z-Variable Eins ist, was der Funktion der Restriktion (3.14)

entspricht:

$$\boldsymbol{\epsilon_{s,t,i}} \leq z_i \quad \forall s \in S, t \in T, i \in I. \tag{3.25}$$

Zuletzt muss noch (3.22) ersetzt werden durch:

$$0 \leq \epsilon_{s,t,i} \leq 1. \tag{3.26}$$

Damit sind die ω-Variablen aus dem Modell entfernt. Die second-stage besteht nun ausschließlich aus kontinuierlichen Variablen.

Anpassen der Restriktionen

Ein Blick auf die allgemeine Vorschrift der Restriktionen in (2.43)

$$W_\omega v_\omega \leq h_\omega - T_\omega y \tag{3.27}$$

zeigt, dass die second-stage-Restriktionen jeweils $\leq$-Ungleichungen sein sollen. Des Weiteren müssen die second-stage-Variablen auf die linke Seite sowie die konstanten Terme bzw. solche die mit einer first-stage-Variable zusammenhängen auf die rechte Seite gebracht werden. Es werden im Folgenden alle second-stage-Restriktionen in diese Form gebracht. Diese Umformungen sind nicht essentiell, um Benders' Dekompositionsverfahren anzuwenden, erleichtern jedoch später das Ablesen der Cut Parameter.

Die Gleichheit in (3.11) kann umgangen werden, indem aus der einen Restriktion zwei gemacht werden:

$$\sum_{i \in I} x_{s,t,i,j} + \varphi_{s,t,j} \leq 1 \qquad \forall s \in S, t \in T, j \in J \tag{3.28}$$

$$\sum_{i \in I} x_{s,t,i,j} + \varphi_{s,t,j} \geq 1 \qquad \forall s \in S, t \in T, j \in J \tag{3.29}$$

was nach Multiplikation der zweiten Ungleichung mit (-1) äquivalent ist zu

$$\sum_{i \in I} x_{s,t,i,j} + \varphi_{s,t,j} \leq 1 \qquad \forall s \in S, t \in T, j \in J \tag{3.30}$$

$$\sum_{i \in I} -x_{s,t,i,j} - \varphi_{s,t,j} \leq -1 \qquad \forall s \in S, t \in T, j \in J \tag{3.31}$$

und damit jeweils (3.27) genügt.

Restriktion (3.12) ist bereits in dieser Form. Für (3.13), die in (3.24) neu formuliert

ist, muss der hintere Summand auf die linke Seite subtrahiert werden:

$$\sum_{j \in J} d_{s,t,j}\ x_{s,i,j} - m \cdot \epsilon_{s,t,i} \leq \gamma_{s,t,i}\ c_i\ y_i \quad \forall s \in S, t \in T, i \in I. \tag{3.32}$$

Die Restriktion (3.14), neu formuliert in (3.25), ist bereits in entsprechender Form. Die nächste Nebenbedingung (3.15) muss umgeformt werden, denn mit $\varphi_{s,t,j}$ taucht in dem Quotienten auf der rechten Seite eine second-stage-Variable auf. Da außerdem eine Gleichung vorliegt, könnten analog zum Vorgehen bei (3.11) wieder zwei neue Ungleichungen aufgestellt werden. Hier allerdings kann ausgenutzt werden, dass die Zielfunktion bestrebt, Δ zu minimieren, was wegen (3.16) dafür sorgt, dass β maximiert wird. Von daher genügt es, das Gleichheitszeichen zu einem $\leq$ zu ändern:

$$\beta + \frac{\sum_{t \in T} \sum_{j \in J} d_{tj}\ \varphi_{s,t,j}}{\sum_{t \in T} \sum_{j \in J} d_{s,t,j}} \leq 1 \quad \forall s \in S, \tag{3.33}$$

was nun (3.27) entspricht. Die Restriktion (3.16) muss zu einer $\leq$-Ungleichung transformiert werden:

$$\begin{aligned} & \Delta_s \geq \beta_0 - \beta \\ \Leftrightarrow \quad & \beta_0 - \beta \leq \Delta_s \\ \Leftrightarrow \quad & -\beta - \Delta_s \leq -\beta_0. \end{aligned} \tag{3.34}$$

Damit sind alle Restriktionen in passender Form.

Das neue Modell

Durch eine Vielzahl von kleinen Änderungen hat sich das DE optisch gewandelt. Das Modell soll im Folgenden als CPLP-Risk bezeichnet werden.

$$\min \sum_{i \in I} (f_i y_i + b_i z_i) + \sum_{s \in S} \pi_s \left(u \ \Delta_s + \sum_{i \in I} \sum_{t \in T} \sum_{j \in J} ce \cdot m \cdot \epsilon_{s,t,i} + (c_{ij} - r_j) d_{s,t,j} x_{s,t,i,j} \right) \tag{3.35}$$

$$\text{s.t.} \quad z_i \leq y_i \qquad \forall i \in I \tag{3.36}$$

$$\sum_{i \in I} x_{s,t,i,j} + \varphi_{s,t,j} \leq 1 \qquad \forall s \in S, t \in T, j \in J \tag{3.37}$$

$$\sum_{i \in I} -x_{s,t,i,j} - \varphi_{s,t,j} \leq -1 \qquad \forall s \in S, t \in T, j \in J \tag{3.38}$$

$$\sum_{j \in J} x_{t,i,j} \leq y_i \qquad \forall s \in S, t \in T, i \in I \tag{3.39}$$

$$\sum_{j \in J} d_{s,t,j} \ x_{s,i,j} - m \cdot \epsilon_{s,t,i} \leq \gamma_{s,t,i} \ K_i \ y_i \qquad \forall s \in S, t \in T, i \in I \tag{3.40}$$

$$\epsilon_{s,t,i} \leq z_i \qquad \forall s \in S, t \in T, i \in I \tag{3.41}$$

$$\beta + \frac{\sum_{t \in T} \sum_{j \in J} d_{tj} \ \varphi_{s,t,j}}{\sum_{t \in T} \sum_{j \in J} d_{s,t,j}} \leq 1 \qquad \forall s \in S \tag{3.42}$$

$$-\beta - \Delta_s \leq -\beta_0 \qquad \forall s \in S \tag{3.43}$$

$$0 \leq x_{s,t,i,j} \leq 1 \tag{3.44}$$

$$0 \leq \varphi_{s,t,j} \leq 1 \tag{3.45}$$

$$0 \leq \Delta \leq 1 \tag{3.46}$$

$$y_i \in \{0, 1\} \tag{3.47}$$

$$z_i \in \{0, 1\} \tag{3.48}$$

$$0 \leq \epsilon_{s,t,i} \leq 1. \tag{3.49}$$

Kapitel 4

Empirische Untersuchungen

In diesem Kapitel wird unter Verwendung der Modellierungssprache GAMS (*General Algebraic Modelling System*) das im vorherigen Kapitel erarbeitete Modell untersucht. Es werden insgesamt neun Programme, in denen dieses Modell unter Verwendung verschiedener methodischer Konzepte angegangen wird, vorgestellt. Zur empirischen Untersuchung stehen neun Instanzen zur Verfügung, die in einem Abschnitt kurz beschrieben werden. Anschließend werden die Resultate der Programme nacheinander aufgeführt. Der Programmcode, in dem Benders' Dekompositionsverfahren implementiert ist, wird dabei ausführlich erklärt.

4.1 Die Modellierungssprache GAMS

Die Programmiersprache GAMS nahm ihre Anfänge in den 1970er Jahren, als sich der Befarf nach einer kompakten Programmiersprache entwickelte, die in der Lage war, mathematische Modelle modellnah zu implementieren und auch zu lösen. Die damals bestehenden Möglichkeiten waren begrenzt und aufwendig. Denn viel Zeit ging dabei für Datenaufbereitung oder Datenumwandlung verloren. Um ein mathematisches Optimierungsproblem zu lösen, musste stets eine weitere Programmiersprache bemüht werden, um Daten für bestehende Löser umzuwandeln. Mit GAMS wurde versucht, diesen fehleranfällige Umweg zu umgehen. Im Softwarepaket GAMS sind mittlerweile diverse Löser wie Cplex oder Gurobi bereits integriert, sodass auf weitere Installationen ebendieser verzichtet werden kann und die Löser schon im Programmcode gestartet werden können.[64]

Das s. g. *Extended Mathematical Programming* (EMP) ermöglicht die automatisierte Neuformulierung von Modellen zu solchen, welche dann mit üblichen Lösern gelöst werden können. Im Falle von stochastischen Programmen stellt EMP etwa Möglichkeiten

[64] Vgl. [Kallrath, 2012, S. 3] und [Rosenthal, 2017, S. 3 - 4].

bereit, zu den Risikomaßen EV und CVaR das zugehörige deterministisches Modell automatisiert zu formulieren und lösen zu lassen. Ein EMP-Modell besteht zum einen aus den üblichen Variablen und Gleichungen (dem *core model*) und zum anderen werden einem Text-Dokument Annotationen übergeben. Bei stochastischen Programmen lässt sich an dieser Stelle angeben, welches Risikomaß verwendet werden soll oder aber auch, welche Variablen und Restriktionen der zweiten Stufe zugeordnet werden.[65]

4.2 Implementierungen in GAMS

Sämtliche Implementierungen sind im Grundaufbau gleich. Auf die Implementierung von Benders' Dekompositionsverfahren wird in Abschnitt 4.6.1 noch genauer eingegangen. In Tabelle 1 ist eine Übersicht über alle GAMS-Programme gegeben, aus der auch hervorgeht, welcher Löser jeweils eingesetzt wird[66].

Dateiname	Beschreibung	Löser
`CPLP_Risk_EV.gms`	Implementierung von CPLP-Risk, gegeben auf Seite 35.	Cplex
`CPLP_Risk_EVPI.gms`	Implementierung des EVPI (vgl. Abschnitt 2.3.3, Seite 17).	Cplex
`CPLP_Risk_VSS.gms`	Implementierung des VSS (vgl. Abschnitt 2.3.3, Seite 18).	Cplex
`CPLP_Risk_CVaR.gms`	Implementierung des DE zum CVaR in (2.45).	Cplex
`CPLP_Risk_EMP_EV.gms`	Implementierung des DE zum EV mit EMP.	EMP/DE
`CPLP_Risk_EMP_CVaR.gms`	Implementierung das CVaR mit EMP.	EMP/DE
`CPLP_Risk_EMP_Convex.gms`	Implementierung einer Konvexkombination von EV und CVaR mit EMP.	EMP/DE
`CPLP_Risk_Benders.gms`	Implementierung von Benders' Dekompositionsverfahren.	Cplex
`CPLP_Risk_Benders_Multicuts.gms`	Implementierung von Benders' Dekompositionsverfahren unter Verwendung von Multicuts.	Cplex

Tabelle 1: Übersicht über die GAMS-Programme mit Kurzbeschreibung. Zu jedem Programm wird die eingesetzte Löser aufgeführt. Bei den Programmen, die EMP nicht verwenden, ist die Wahl von Cplex (Version 12.8) als Löser am performantesten. Gurobi (Version 8.0.0) ist nur marginal langsamer. Bei EMP wird der Löser DE verwendet, der ebenfalls Cplex zur Lösung nutzt (vgl. [GAMS, 2018, S. 1465 - 1467]).

[65] Vgl. [GAMS, 2018, S. 851].
[66] Die verwendete GAMS-Version ist 25.1.2, veröffentlicht im August 2018.

4.3 Instanzen

Insgesamt stehen neun Instanzen zur Verfügung[67]. In Tabelle 2 ist eine Übersicht gegeben. Die Instanzen mit dem Präfix `hv` zeichnen sich durch eine hohe Variation in den Szenarien aus. Dahingegen deutet der Präfix `hf` auf hohe Fixkosten in dieser Instanz hin. Die Instanzen beziehen sich auf das Originalmodell auf Seite 31. Der grundsätzliche Aufbau der Instanzen ist im Anhang A auf Seite 60 abgebildet. Die Verarbeitung der Instanzen wird in GAMS durch einen s. g. `$batinclude` nach `importFile.inc` ausgelagert. Darin wird Python verwendet, um die Daten der Instanzen auf die GAMS-Parameter zu übertragen. Der Inhalt dieser Datei ist in Anhang B auf Seite 61 gegeben.

Nr.	Dateiname	$\|S\|$	$\|T\|$	$\|I\|$	$\|J\|$
(1)	`small.txt`	4	5	3	5
(2)	`medium.txt`	32	10	30	50
(3)	`large.txt`	64	20	60	100
(4)	`hvSmall.txt`	3	5	2	3
(5)	`hvMedium.txt`	10	50	18	30
(6)	`hvLarge.txt`	20	100	36	60
(7)	`hfSmall.txt`	3	5	2	3
(8)	`hfMedium.txt`	10	50	18	30
(9)	`hfLarge.txt`	20	100	36	60

Tabelle 2: Übersicht über die verfügbare Instanzen. Es wird für jede Instanz die Kardinalität der Menge der Szenarien S, der Zeitperioden T, der Einrichtungen I sowie der Kunden J angegeben.

4.4 EV und CVaR in GAMS

Der EV und CVaR einer Zielfunktion sind zwei der in Abschnitt 2.2 *Stochastik* vorgestellten Risikomaße. Beide sind auf zwei verschiedene Weisen implementiert.

Zum EV ist das CPLP-Risk auf Seite 35 in `CPLP_Risk_EV.gms` direkt umgesetzt. In `CPLP_Risk_EMP_EV.gms` wird dazu EMP angewandt. Dabei ist zu beachten, dass der EV auch ohne explizite Angabe als Risikomaß verwendet wird. Die für beide Programme übereinstimmenden Ergebnisse sind in Tabelle 3 gegeben.

Es zeigt sich, dass dem Programm die hohe Anzahl an Szenarien in den Instanzen (2) und (3) deutliche Schwierigkeiten bereitet. Die Instanzen mit Präfix `hv` und `hf` sind

[67] Da keine Instanzen frei verfügbar sind, wurden die aufgeführten Instanzen vom *Institut für Operations Research: Diskrete Optimierung und Logistik* des *Karlsruher Institut für Technologie* (KIT) zur Verfügung gestellt. Der dort verwendete Generator ist jedoch noch in einer beta-Version, wodurch bislang nur diese geringe Anzahl an Instanzen zu Stande kam.

Nr.	Dateiname	Zeit	Bestes Ergebnis	Beste ob. Schranke	rel. Abstand
(1)	`small.txt`	<1s	-1.500,3	-1.500,3	0,00%
(2)	`medium.txt`*	1000s	-62.501,1	-62.533,1	0,05%
(3)	`large.txt`*	1000s	66,6	-29.533.971,4	>100%
(4)	`hvSmall.txt`	<1s	-1.520,7	-1.520,7	0,00%
(5)	`hvMedium.txt`	6s	-2.233.772,0	-2.233.772,0	0,00%
(6)	`hvLarge.txt`	120s	-26.795.023,7	-26.795.023,7	0,00%
(7)	`hfSmall.txt`	<1s	-1.200,1	-1.200,1	0,00%
(8)	`hfMedium.txt`	75s	-1.934.124,0	-1.934.124,0	0,00%
(9)	`hfLarge.txt`	11s	-24.835.491,2	-24.835.491,2	0,00%

Tabelle 3: Resultate von `CPLP_Risk_EV.gms`. Die mit * markierten Instanzen konnten nach 1000s nicht optimal gelöst werden. Neben dem resultierenden Zielfunktionswert ist auch die beste obere Schranke angegeben sowie deren relativer Abstand.

dagegen in annehmbarer Zeit optimal gelöst, weshalb für diese Instanzen der Einsatz anderer Methoden nicht zwingend erforderlich ist.

Für den CVaR ist in `CPLP_Risk_CVaR.gms` das zugehörige DE (2.45) auf Seite 17 implementiert, wobei das dortige α im Programm `theta` ist. Genauso wie beim EV ist in `CPLP_Risk_EMP_CVaR.gms` ein Programm gegeben, in dem EMP verwendet wird. Hierbei ist aber zu beachten, dass das bei EMP verwendete `theta` immer von der linken Seite ausgeht. Während das in `CPLP_Risk_CVaR.gms` verwendete `theta` die Verteilung direkt von der rechten Seite auswertet, geschieht dies bei `CPLP_Risk_EMP_CVaR.gms` durch `1-theta`.

Die folgende Aussage bezieht sich auf `CPLP_Risk_CVaR.gms`. Es gilt, dass die Verwendung des CVaRs nie zu einem besseren Gesamtergebnis führen kann. Sonderfälle stellen `theta=1` bzw. `theta=0` dar. Für `theta=1` ist die Lösung des CVaR gleich dem EV, da die schlechtesten 100% der Szenarien im arithmetischen Mittel verbessert werden, was offenbar allen Szenarien entspricht. Wird im Programm `theta` auf `0` gesetzt, so wird in (2.45) auf Seite 17 durch 0 geteilt, was bei Programmausführung eine korrespondierende Fehlermeldung bewirkt.

In `CPLP_Risk_EMP_Convex.gms` ist eine Konvexkombination dieser beiden Risikomaße implementiert. Das gegebene Problem wird darin sowohl mit EV als auch mit CVaR als Risikomaß unter Verwendung von EMP gelöst. Um den CVaR zu 0,05 zu ermitteln, wird `theta` in den EMP-Annotation auf `0.95` gesetzt. Die beiden Zielfunktionswerte werden in einer neuen Zielfunktion durch einen Faktor `lambda`, der zwischen `0` und `1` liegt, gewichtet:

```
lambda*EV_obj + (1-lambda)*CVaR_obj;
```

Abbildung 3 zeigt den zur Instanz `hvMedium.txt` resultierenden Zielfunktionswert in Abhängigkeit von `lambda`. Je stärker der EV in der Zielfunktion gewichtet wird, desto besser ist der Zielfunktionswert. Eine verstärkte Gewichtung des CVaR sorgt hingegen dafür, dass die schlechtesten 5% der Szenarien bestmöglich sind. Diese pessimistische Herangehensweise führt allerdings wie oben bereits erwähnt zu einem schlechteren Gesamtergebnis.

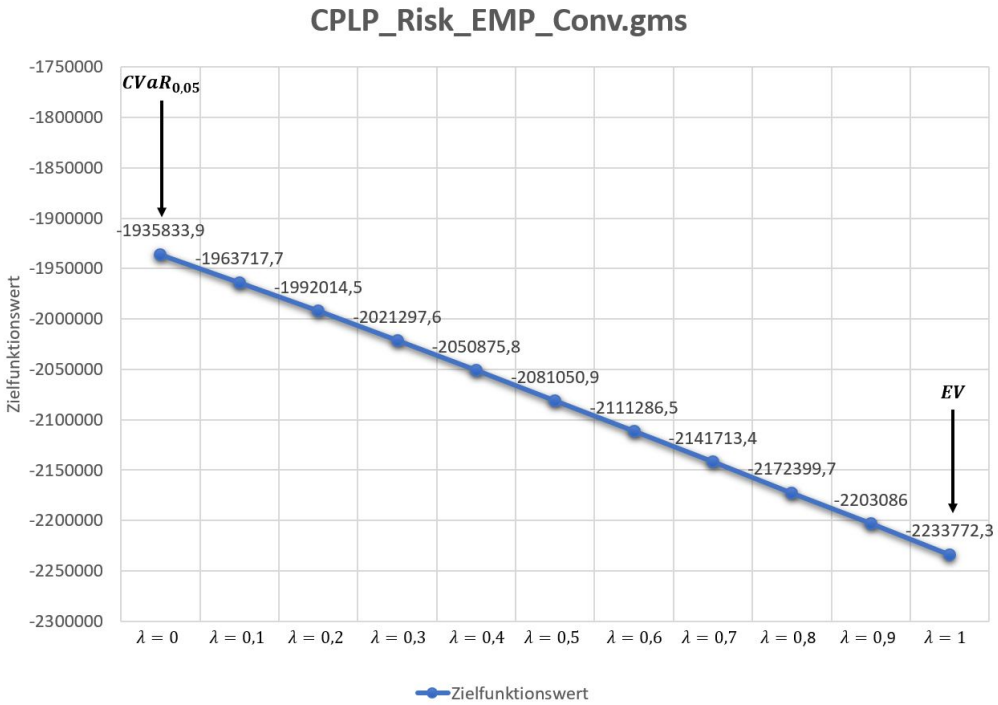

Abbildung 3: Konvexkombination von EV und CVaR. In diesem Diagramm sind die Zielfunktionswerte der Instanz `hvMedium.txt` abhängig vom Faktor `lambda` abgebildet. Für `lambda=0` wird der CVaR bestimmt, für `lambda=1` der EV.

4.5 EVPI und VSS in GAMS

Mit dem Programm `CPLP_Risk_EVPI.gms` kann der EVPI zu einem stochastischen Optimierungsproblem bestimmt werden. Gleiches gilt mit `CPLP_Risk_VSS.gms` für den VSS. Die Programme liefern jedoch nur die Lösungen WS (vgl. (2.46) auf Seite 18) bzw. EEV (vgl. (2.49) auf Seite 19).

In Tabelle 4 sind diese Werte für die Instanzen angegeben. Offenbar ist für fünf der neun Instanzen sowohl $EVPI$ als auch VSS 0. Daraus kann für RP, also den Zielfunk-

tionswert des Original Problems, sofort abgeleitet werden:

$$RP = WS = EEV. \tag{4.1}$$

Auch bei den anderen Instanzen sind weder $EVPI$ noch VSS relativ zum Zielfunktionswert besonders groß. Somit liefern WS bzw. EEV gute untere bzw. obere Schranken für RP. Zumindest EEV wird zu jeder Instanz in weniger als einer Minute bestimmt und stellt daher eine bereits gute Startlösung für einen Warmstart des DE dar[68].

Nr.	Dateiname	WS	EEV	Differenz
(1)	`small.txt`	-1.500,3	-1.500,3	0,0
(2)	`medium.txt`	-62.744,2	-62.448,2	296,0
(3)	`large.txt`	-496.793,0 *	-496.988,9	-195,9
(4)	`hvSmall.txt`	-1.520,7	-1.520,7	0,0
(5)	`hvMedium.txt`	-2.237.788,4	-2.232.368,8	5.419,6
(6)	`hvLarge.txt`	-26.795.023,7	-26.795.023,7	0,0
(7)	`hfSmall.txt`	-1.200,1	-1.200,1	0,0
(8)	`hfMedium.txt`	-1.937.023,7	-1.934.054,6	2.969,1
(9)	`hfLarge.txt`	-24.835.491,2	-24.835.491,2	0,0

Tabelle 4: Resultate von `CPLP_Risk_EVPI.gms` und `CPLP_Risk_VSS.gms`. WS und EEV werden bestimmt. Bei den mit * markierten Einträgen konnte nach 1000 Sekunden keine optimale Lösung gefunden werden. Es wird die beste gefundene Lösung angegeben. Die angegebene Differenz stellt i. A. eine obere Schranke für $EVPI$ und VSS dar.

4.6 Benders' Dekompositionsverfahren in GAMS

Im ersten Unterabschnitt wird der Programmcode von `CPLP_Risk_Benders.gms` dargestellt und erklärt. Anschließend werden Resultate des Programms unter Verwendung von Singlecuts und Multicuts miteinander verglichen.

4.6.1 Vorstellung von `CPLP_Risk_Benders.gms`

Bei der Anwendung von Benders' Dekompositionsverfahren wollen wir annehmen, dass jede First-Stage-Entscheidung auch für die second-stage gültig ist. Dann kann in der Implementierung des Algorithmus der Einfachheit halber auf die Ermittlung von Feasibility Cuts verzichtet werden. Das Programm `CPLP_Risk_Benders.gms` wird gemäß der folgenden Abschnitte vorgestellt.

[68] Für genauere Untersuchungen zum EVPI und VSS von CPLP-Risk sei auf [Heckmann, 2016, S. 345 - 353] verwiesen.

1. File einlesen und Datenstrukturen deklarieren
2. Datenstrukturen initialisieren
3. Vorbereitungen für Benders' Verfahren
4. First-Stage Master Problem definieren
5. Second-Stage Subprobleme definieren
6. Benders' Algorithmus, finale Vorbereitungen
7. Benders' Algorithmus

1. Abschnitt: File einlesen und Datenstrukturen deklarieren

Im ersten Abschnitt wird ein Textdokument angegeben, aus welchem die Daten einer Instanz ermittelt werden. Außerdem werden sämtliche Datenstrukturen deklariert.

```
$set file instances\hclarge.txt
$if not set file $set file instances\Small.txt

***Declare required sets, parameters and scalars
Sets
   I                 'facilities'
   J                 'customers'
   T                 'Timeslots'
   H                 'extension levels'
   S                 'scenarios';

Parameters
   f(i)              'fix opening costs for i'
   K(i)              'capacity of i'
   b(i)              'costs if capacity is exceeded'
   c(i,j)            'transport costs from i to j per unit'
   r(j)              'profit at j fulfilling one demanded unit'
   v(h)              'costs of an extension level'
   o(h)              'amount of extra capacity of an extension level'
   p(s)              'probability of a scenario'
   gamma_s(s,t,i)    'percentage of capacity remains at t in s'
   d_s(s,t,j)        'demand of customer j at t in s';
```

```
Scalars
   u                 'costs if desired service level is not fulfilled'
   beta0             'desired service level';
```

2. Abschnitt: Datenstrukturen initialisieren

Im zweiten Abschnitt werden die in Abschnitt 1 deklarierten Datenstrukturen mit den in `file` gegebenen Daten initialisiert. Die Hauptarbeit erledigt das Include-File `importFile.inc`. Durch das Schlagwort `$batinclude` wird der Inhalt dieser externen Datei so ausgeführt, als würde er im Hauptprogramm stehen. Im Anhang B ist `importFile.inc` aufgeführt.

In Abschnitt 3.3 *Vorbereitungen auf Benders' Dekompositionsverfahren* auf Seite 32 ist das Modell so umformuliert, dass es in der second-stage keine ganzzahligen Variablen mehr gibt. Da sich die Datensätze noch auf das ursprüngliche Modell beziehen, müssen die beiden neuen Parameter `c4e` und `maxEC` wie auf Seite 32 beschrieben berechnet werden. Um `c4e` zu bestimmen, wird über die Ausbaumenge `h` iteriert. Wegen des linearen Zusammenhangs zwischen Ausbaumenge und Ausbaukosten reicht es, den ersten Eintrag des GAMS-Parameters `v(h)` durch den des Parameters `o(h)` zu teilen. `maxEC` wird mit dem unter allen Ausbaustufen maximalen `o(h)` initialisiert.

```
***Import of data via EmbeddedCode Section Python
$batinclude importFile.inc
       %file% I J T H S f K b r c u beta0 v o p gamma_s d_s

Scalars
   c4e               'Costs for extension'        / 0 /
   maxEC             'maximal extra capacity'     / 0 /;

loop(h$(ord(h)=1), c4e = v(h) / o(h); );
maxEC = smax(h,o(h));
```

3. Abschnitt: Vorbereitungen für Benders' Verfahren

Es soll genau der in Abschnitt 2.3.4 *Benders' Dekompositionsverfahren für SP* beschriebene Algorithmus umgesetzt werden. Es ist sinnvoll eine gewisse Anzahl an Iterationen vorzugeben, falls der Algorithmus erst spät terminiert. Entsprechend dieses Werts wird eine Set `iter` deklariert. Die dynamische Teilmenge `itActive` beschreibt die bereits durchgeführten Iterationen, für die ein Optimality Cut schon bestimmt wurde. Des

Weiteren werden drei Parameter `g`, `G_y` und `G_z` für jede Iteration deklariert. Diese repräsentieren die Konstante g und den Vektor G in (2.63) auf Seite 23. Ein Optimality Cut ist somit von folgender Form:

$$\texttt{theta} \geq \texttt{g} - \texttt{G_y} \cdot \texttt{y} - \texttt{G_z} \cdot \texttt{z}, \tag{4.2}$$

wobei `theta` dem z in (2.63) entspricht. Da es zwei first-stage-Variablen gibt, müssen zwei Vektoren angelegt werden, die sich jeweils auf eine der beiden Variablen beziehen.

```
***benders loop specifications
$if not set maxiter $set maxiter 200
Sets
 iter                'max Benders iterations' / 1*%maxiter% /
 itActive(iter)      'active Benders cuts';

Parameters
 g(iter)       'constants in opt. cuts'                / #iter 0 /
 G_y(iter,i)  'Vectors referring to y in opt. cuts' / #iter.#i 0 /
 G_z(iter,i)  'Vectors referring to z in opt. cuts' / #iter.#i 0 /;
```

4. Abschnitt: First-Stage Master Problem definieren

In diesem Abschnitt werden zunächst die binären first-stage-Variablen `y(i)` und `z(i)` deklariert. Der Zielfunktionswert des Masterproblems heißt `masterObj` und die Schranke z in (2.55) ist `theta`. Im Anschluss werden drei first-stage-Gleichungen deklariert. `defMaserObj` beschreibt die Zielfunktion. `defOpe` entspricht der first-stage-Restriktion (3.36). `defOptCuts` sind die Optimality Cuts in der Form (4.2), die je Iteration hinzugefügt werden. Dabei ist darauf zu achten, dass in der Initialisierung dieser Gleichung die dynamische Teilmenge `itActive` als Domäne angegeben wird. Dies hat zur Folge, dass nur bereits abgeschlossene Iterationen einen Optimality Cut liefern. Die letzte Zeile speichert diese drei Gleichungen im Modell `masterproblem`.

```
***first stage variables
Binary Variables
   y(i)                'facility i is opened once'
   z(i)                'extension is possible at facility i';
Free Variables
   masterObj           'objective value of Master Problem'
   theta               'threshold in master problem';
```

```
***First stage equations
Equations
   defMasterObj      'objective function of the masterproblem'
   defOpe(i)         'only extensions possible if facility is opened'
   defOptCuts(iter) 'Opt. Cuts being inserted in Benders loop';

defMasterObj..
   masterObj =E= sum(i, f(i)*y(i)+b(i)*z(i)) + theta;

defOpe(i)..
   z(i) =L= y(i);

defOptCuts(itActive)..
   theta =G= g(itActive)
   - sum(i, G_y(itActive,i)*y(i))
   - sum(i, G_z(itActive,i)*z(i));

model masterproblem / defMasterObj, defOpe, defOptCuts /;
```

5. Abschnitt: Second-Stage Subprobleme definieren

Zunächst werden die second-stage-Variablen, bei denen es sich jeweils um nicht-negative Variablen handelt, deklariert. Der Zielfunktionswert des Subproblems `subObj` ist eine freie Variable, deren Wertebereich also nicht eingeschränkt ist. Die beiden szenarioabhängigen Parameter `gamma(t,i)` und `d(t,j)` werden ebenfalls deklariert. Anschließend werden der Reihe nach die Restriktionen des Modells CPLP-Risk auf Seite 35 deklariert und darunter initialisiert. Alle hier definierten Gleichungen werden dem Modell `subproblem` zugewiesen.

```
***second stage variables and parameters
Positive Variables
   x(t,i,j)          'amount transported from i to j'
   phi(t,j)          'not fulfilled amount of the demand'
   delta             'difference current and desired service level'
   ex_am(t,i)        'relative extra amount for facility i'
   beta              'current service level';
Free Variable
   subObj            'objective value of subproblem';
```

```
Parameters
   gamma(t,i)        'percentage of capacity remains at timeslot t'
   d(t,j)            'demand of customer j at timeslot t';

***second stage equations
Equations
   defSubObj         'objective function of the subproblem'
   defDemL(t,j)      'fulfill demand at customer j, lower Bound'
   defDemG(t,j)      'fulfill demand at customer j, upper Bound'
   defCap(t,i)       'fulfill capacity requirements at facility i'
   defExp(t,i)       'fix costs for extensions'
   defSLc            'calculates the current service level'
   defSLd            'calculates delta'
   defUB1            'defines an upper bound for delta'
   defUB2(t,i)       'defines an upper bound for ex_am';

defSubObj..
   subObj =E= u*delta + c4e*sum((i,t), ex_am(t,i)*maxEC)
   + sum((t,i,j), (c(i,j)-r(j))*x(t,i,j)*d(t,j));

defDemL(t,j)..
   sum(i,  x(t,i,j)) + phi(t,j) =L=  1;

defDemG(t,j)..
   sum(i, -x(t,i,j)) - phi(t,j) =L= -1;

defCap(t,i)..
   sum(j, d(t,j)*x(t,i,j)) - maxEC*ex_am(t,i)
   =L= gamma(t,i)*K(i)*y.l(i);

defExp(t,i)..
   ex_am(t,i) =L= z.l(i);

defSLc..
   beta + (sum((t,j), d(t,j)*phi(t,j)) / sum((t,j), d(t,j)))
   =L= 1;

defSLd..
   -beta - delta =L= -beta0;
```

```
defUB1..
   delta =L= 1;

defUB2(t,i)..
   ex_am(t,i) =L= 1;

model subproblem / all - defMasterObj - defOpe - defOptCuts /;
```

6. Abschnitt: Benders' Algorithmus, finale Vorbereitungen

In diesem Abschnitt werden die Skalare für Benders' Schleifen deklariert und initialisiert. Die Zahl `rgap` gibt den relativen Abstand von der oberen Schranke `UB` und unteren Schranke `LB` an. `objMaster` ist der Zielfunktionswert des aktuellen Masterproblems. `objSub` aggregiert die Zielfunktionswerte der Subprobleme. Anschließend werden einige Optionen gesetzt. Die ersten beiden Zeilen sorgen dafür, dass weder Variablen (`limCol`) noch Gleichungen (`limRow`) gelistet werden. Um bei der Vielzahl an Teilproblemen nicht jedes Mal einen Lösungsreport zu erhalten, wird `solprint=silent` gesetzt. `optcr` legt fest, ab wann ein Problem als gelöst angesehen wird. Wird diese Option auf 0 gesetzt, so wird das Problem bis zu einer Dualitäts-Lücke von 0 gelöst. Der Löser `cplexd` stellt eine nützliche Erweiterung des Lösers `cplex` dar und wird für Benders' Algorithmus als zu verwendender Löser gesetzt. Zuletzt wird via `solveLink=%solveLink.loadLibrary%` die Verwendung von temporären Dateien abgestellt.

```
***declaration and initialization of required scalars
Scalars
   rgap        'relative gap'                                  / 0 /
   LB          'global lower bound'                          / -inf /
   UB          'global upper bound'                          / +inf /
   objMaster   'objective value of masterproblem'              / 0 /
   objSub      'aggregated objective values of all subproblems' / 0 /;

***option settings
option limRow = 0;
option limCol = 0;
option solPrint = silent;
option optcr = 0;
option solver = cplexd;
option solveLink = %solveLink.loadLibrary%;
```

7. Abschnitt: Benders' Algorithmus

Zunächst wird die Initialisierung abgeschlossen, indem die first-stage-Variablen fest gewählt werden. Werden sie auf `0` gesetzt, so ist keine Einrichtung geöffnet. Wie am Anfang dieses Abschnitts schon beschrieben, wird davon ausgegangen, dass jede first-stage-Entscheidung auch für die second-stage zulässig ist.

```
***arbitrary initialization of first stage variables:
***no facility opened
y.l(i) = 0;
z.l(i) = 0;
```

Nun wird über die Set `iter` iteriert. Mit Hilfe der s. g. *Dollar-Condition* ist es möglich, anzugeben, bis zu welcher Iteration die Schleife tatsächlich gehen soll. `objSub` wird auf Null gesetzt, denn für jede Iteration soll der Zielfunktionswert der Subprobleme neu berechnet werden. Im Abschluss daran wird über jedes Szenario iteriert. Die beiden unsicheren Parameter `gamma` und `d` werden mit den entsprechenden Szenariodaten belegt. Damit kann via `solve` dieses Subproblem, ein LP, gelöst werden. Die aggregierten Subproblem-Zielfunktionswerte `objSub` werden um den ermittelten Zielfunktionswert `subObj.l` erhöht. Anschließend wird die Zahl g und der Vektor G in (2.63) auf Seite 23 bestimmt. Als Erinnerung sei deren Definition noch einmal gegeben:

$$g := \sum_{\omega \in \Omega} p_\omega \cdot h_\omega^T \pi_\omega^i, \tag{4.3}$$

$$G := \sum_{\omega \in \Omega} p_\omega \cdot \left(T_\omega \pi_\omega^i\right)^T. \tag{4.4}$$

Da in unserem Modell zwei first-stage-Variablen enthalten sind, entstehen wie oben schon erwähnt zwei Vektoren `G_y` bzw. `G_z` bezogen auf y_i bzw. z_i. Die Restriktionen sind bereits in der Form $W_\omega v_\omega \leq h_\omega - T_\omega y$[69]. Daher kann für jede Restriktion ein Beitrag für `g`, `G_y` und `G_z` abgeleitet werden. Die zu einer Gleichung gehörende duale Variable entspricht dem s. g. *marginal*-Wert der Gleichung, auf welche mit dem Suffix `.m` zugegriffen werden kann.

Beispielsweise ergibt sich für Restriktion `defDemL` der Wert `1` für alle `t` und `j`. Demnach enthält der Vektor h in allen Komponenten eine 1. Die Matrix T ist eine Nullmatrix, weil keine first-stage-Variable in dieser Restriktion vorkommt. Daher muss nur der

[69] Vgl. Gleichung (3.27) aus dem Abschnitt *Vorbereitungen für Benders' Dekompositionsverfahren*, Unterabschnitt *Anpassen der Restriktionen* auf Seite 32.

Parameter g erhöht werden, und zwar gemäß:

$$\mathtt{g} = \mathtt{g} + \mathtt{p(s)} \cdot \sum_{\mathtt{t,j}} \mathtt{1} \cdot \mathtt{defDemL.m(t,j)}. \tag{4.5}$$

Für die Restriktion defCap hingegen ist offenbar h in allen Komponenten Null. Dafür besteht ein Vorfaktor für y_i, weshalb G_y erhöht werden muss. Es gilt:

$$\mathtt{G_y(i)} = \mathtt{G_y(i)} + \mathtt{p(s)} \cdot \sum_{\mathtt{t}} -\mathtt{gamma(t,i)} \cdot \mathtt{K(i)} \cdot \mathtt{defCap.m(t,i)}. \tag{4.6}$$

Beachte, dass der Vorfaktor $-$gamma(t,i) $\cdot$ K(i) ein negatives Vorzeichen trägt, da in $W_\omega v_\omega \leq h_\omega - T_\omega y$ der Vektor $T_\omega y$ abgezogen wird. G_y wird somit ein von i abhängiger Vektor. Alle anderen Restriktionen werden in gleicher Weise behandelt. Insgesamt ergeben sich für jede Iteration die Zahl g sowie die beiden Parameter G_y und G_z.

Anschließend wird die gerade behandelte Iteration aktiviert, was durch itActive(iter) = yes geschieht. Die obere Schranke ergibt sich gemäß (2.64) auf Seite 23 durch die Addition von den Zielfunktionswerten des Master, abzüglich z bzw. theta.l, und der Subprobleme. Da diese Schranke nicht monoton fallend ist, empfiehlt sich die Überprüfung einer Verbesserung, bevor dieser Wert aktualisiert wird. Mit dem display Statement wird die obere Schranke ausgegeben. Daraufhin wird die relative Lücke zwischen oberer und unterer Schranke berechnet. Im Falle einer Unterschreitung von 0.001 wird die Schleife abgebrochen (vgl. Abbruchbedingung auf Seite 24). Andernfalls geht es weiter mit dem Lösen des Masterproblems, welches mittlerweile um einen Optimality Cut ergänzt wurde. Gemäß (2.66) auf Seite 24 stellt der Zielfunktionswert des aktuellen Masterproblems eine untere Schranke für den Zielfunktionswert des Originalproblems dar, welche anders als die obere Schranke aber monoton steigend ist. Auch die untere Schranke wird mit display ausgegeben.

Nach Ablauf aller Iterationen wird nochmals überprüft, ob die relative Lücke zwischen oberer und unterer Schranke 0.001 überschreitet. Wenn ja, wird das Programm mit der Meldung 'need more iterations' abgebrochen.

```
***iterations of Benders' Decomposition Method
loop(iter$(ord(iter)<30),
   objSub = 0;
   loop(s,
      gamma(t,i) = gamma_s(s,t,i);
      d(t,j) = d_s(s,t,j);
```

```
      solve subproblem MIN subObj USING LP;
      objSub = objSub + p(s) * subObj.l;
      g(iter) = g(iter)
        + p(s) * sum((t,j), 1 * defDemL.m(t,j))
        + p(s) * sum((t,j), -1 * defDemG.m(t,j))
        + p(s) * 1 * defSLc.m
        + p(s) * (-beta0) * defSLd.m
        + p(s) * 1 * defUB1.m
        + p(s) * sum((t,i), 1 * defUB2.m(t,i));
      G_y(iter,i) = G_y(iter,i)
        + p(s) * sum(t, -gamma(t,i)*K(i) * defCap.m(t,i));
      G_z(iter,i) = G_z(iter,i)
        + p(s) * sum(t, -1 * defExp.m(t,i));
   );
   itActive(iter) = yes;
   if(objMaster + objSub < UB,
      UB = objMaster + objSub;
    );
   display UB;
   rgap = (UB - LB)/(1 + abs(UB));
   break$(rgap < 0.001);
   solve masterproblem MIN masterObj USING MIP;
   LB = masterObj.l;
   objMaster = masterObj.l - theta.l;
   display LB;
);
abort$(rgap >= 0.001) 'need more iterations', LB, UB;
```

Verwendung von Multicuts

Um Multicuts zu verwenden, müssen die Schranke `theta` und die Optimality Cuts szenarioabhängig werden. In der Datei `CPLP_Risk_Benders_Multicuts.gms` ist dies umgesetzt. Auf die geringfügigen Veränderungen wird an dieser Stelle nicht eingegangen.

4.6.2 Vergleich Resultate

In diesem Abschnitt wird das Verhalten der drei Programme `CPLP_Risk_EV.gms`, `CPLP_Risk_Benders.gms` und `CPLP_Risk_Benders_Multicuts.gms` miteinander verglichen.

Aufgrund der Ergebnisse von `CPLP_Risk_EV.gms`, die in Tabelle 3 aufgeführt sind, werden die beiden Benders-Implementationen nur bei den dort nicht optimal gelösten Instanzen `medium.txt` und `large.txt` herangezogen[70]. Es sei noch einmal betont, dass die im Folgenden dargestellten Zusammenhänge nur anhand des Verhaltens von zwei Instanzen abgeleitet wurden, wodurch die allgemeine Aussagekraft eingeschränkt wird.

In Abbildung 4 sind zu `medium.txt` die Verläufe der oberen und unteren Schranke für die beiden zuletzt genannten Programme dargestellt[71].

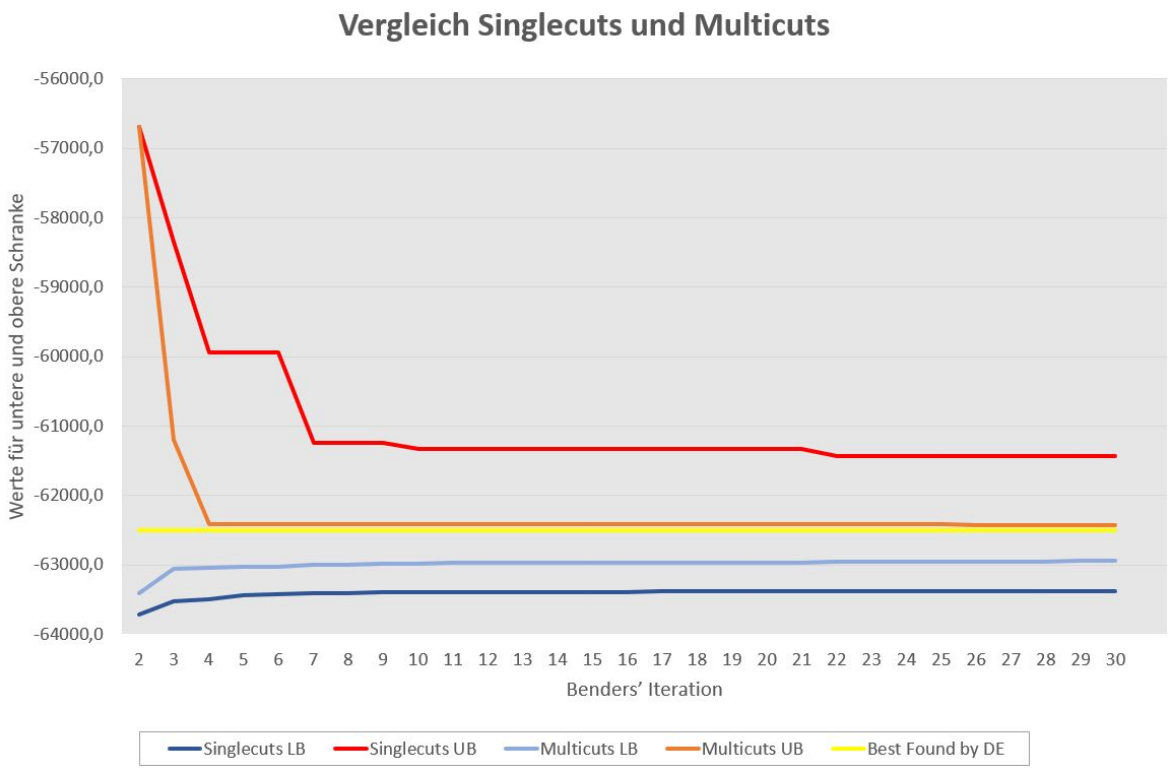

Abbildung 4: Vergleich Single- und Multicuts bei `medium.txt`. In dem Diagramm werden die unteren und oberen Schranken aus den ersten 30 Iterationen aus `CPLP_Risk_Benders.gms` und `CPLP_Risk_Benders_Multicuts.gms` bezogen auf die Instanz `medium.txt` gegenübergestellt (vgl. Tabelle 6 in Anhang C). Die gelbe Linie liegt bei `-62501,1`, dem Zielfunktionswert der besten durch `CPLP_Risk_EV.gms` gefundenen Lösung, und dient der Orientierung. Die erste Iteration ist in dem Diagramm nicht berücksichtigt.

Es wird klar, dass die Multicuts wesentlich bessere Schranken liefern. Außerdem wird deutlich, dass spätestens ab Iteration 10 keine großen Verbesserungen der Schranken erreicht werden. Daraus geht hervor, dass die benötigte Anzahl an Iterationen, um die Abbruchbedingung zu erfüllen, sehr hoch ist. Dazu kommt, dass für eine steigende Anzahl an Iterationen das Masterproblem auch schwieriger zu lösen ist und damit mit

[70] Bei den übrigen Instanzen ergibt sich in der Regel, dass Benders' Verfahren länger für die Optimalität braucht.

[71] Im Anhang C auf Seite 63 sind in Tabelle 6 die genauen Werte für die oberen und unteren Schranken angegeben.

steigender Anzahl an Iterationen die Rechenzeit je Iteration zunimmt.

In der Abbildung ist zudem zur Orientierung der beste Zielfunktionswert von `CPLP_Risk_EV.gms` als gelbe Linie dargestellt. Bei Verwendung von Multicuts und erst recht bei Verwendung von Singlecuts wird dieser Wert nicht erreicht.

Anders verhalten sich die Beobachtungen bei der Instanz `large.txt`. Hier liefert das Programm `CPLP_Risk_EV.gms` in 1.000s keine gute Lösung (vgl. Tabelle 3 auf Seite 39). Dahingegen zeigt Tabelle 5, dass beide Benders-Implementationen gute Lösungen finden. Wie bei `medium.txt` sind die Schranken bei Verwendung von Multicuts wesentlich besser.

Iteration	Singlecuts LB	Singlecuts UB	Multicuts LB	Multicuts UB
1	-1,73135E+6	66,5	-1,73135E+6	66,6
2	-501.948,8	-475.123,0	-500.366,8	-475.123,0
3	-500.759,8	-475.123,0	-498.580,0	-479.300,7
4	-500.658,9	-475.123,0	-498.522,9	-493.153,4
5	-500.607,8	-475.123,0	-498.492,9	-493.153,4
6	-500.575,8	-475.123,0	-498.439,0	-493.153,4
7	-500.569,8	-475.123,0	-498.342,9	-493.153,4
8	-500.558,8	-475.123,0	-498.307,6	-493.153,4
9	-500.553,8	-475.123,0	-498.280,5	-493.153,4

Tabelle 5: Wertetabelle zu Single- und Multicuts (`large.txt`). Aufgeführt sind die unteren und oberen Schranken der ersten neun Iterationen in `CPLP_Risk_Benders.gms` und `CPLP_Risk_Benders_Multicuts.gms` bezogen auf die Instanz `large.txt`.

Interessant ist auch ein Blick auf die Rechenzeit. Für die in der Wertetabelle angegebenen neun Iterationen benötigte die Programmvariante mit Singlecuts ca. 647s. Dahingegen brauchte das Programm, in dem die Multicuts implementiert sind, lediglich 562s. Wie oben bereits erwähnt, liefert `CPLP_Risk_EV.gms` nach 1.000s keine Lösung. Es zeigt sich, dass die Verwendung von Multicuts nicht nur bessere Schranken liefern sondern auch eine geringere Rechenzeit aufweisen.

Kapitel 5

Zusammenfassung

Diese Arbeit thematisiert das mehrperiodische Standortproblem unter Berücksichtigung von Kapazitäten und Unsicherheiten, einer Erweiterung des klassischen *Facility Location Problem*. Die Unsicherheit kann in dem dazugehörigen mathematischen Modell CPLP-Risk durch eine endliche Menge an Szenarien mit Eintrittswahrscheinlichkeiten ausgedrückt werden.

Es werden die Risikomaße Erwartungswert (EV) und Conditional Value at Risk (CVaR) zunächst allgemein eingeführt und später mittels der Modellierungssprache GAMS angewandt auf CPLP-Risk analysiert. Diese beiden Risikomaße legen dem Anwender zwei verschiedene Konzepte an die Hand, um seine Standortentscheidung zu treffen. Im Gegensatz zum EV, bei dem der Erwartungswert über alle Szenarien optimiert wird, verbessert der CVaR zu einem gegebenen Wert $0 < \alpha < 1$ den Erwartungswert der schlechtesten $(100 \times \alpha)\%$ Szenarien. Durch diese pessimistische Herangehensweise wird beim CVaR ein schlechterer Zielfunktionswert erreicht als bei EV. Beide Risikomaße können als Konvexkombination in einem Programm ausgewertet werden, was die flexible Verknüpfung von zwei Programmen in einem ermöglicht. Es bleibt dem Anwender also überlassen, welches Risikomaß stärker gewichtet wird.

Aufgrund der Größe des entstandenen Modells stellt sich jedoch die Frage, wie das Lösen dieses Problems beschleunigt werden kann. Dazu wird Benders' Dekompositionsverfahren für allgemeine Optimierungsprobleme hergeleitet und anschließend auf CPLP-Risk zu dem Risikomaß EV angewandt. Die Grundidee Benders' ist es, statt des umfangreiches Originalproblems viele kleine Teilprobleme zu lösen. Dazu muss das zugrundeliegende Problem in zwei getrennte Teilprobleme zerlegbar sein. Dies ist bei stochastischen *two-stage-Problemen* in natürlicher Weise gegeben. Die erste Stufe enthält Entscheidungen, die zu Beginn also vor der ersten Periode getroffen werden müssen. Zugehörige Variablen und Gleichungen werden sofort dem *Masterproblem*, welches anschließend iterativ erweitert wird, zugeordnet. Die szenarioabhängigen Entscheidun-

gen gehören zur zweiten Stufe. Die *Subprobleme* beziehen sich jeweils auf ein Szenario und liefern die s. g. *Cuts*. In jeder Iteration wird ein solcher Cut dem Masterproblem hinzugefügt und dessen Zielfunktionswert neu bestimmt. Handelt es sich um ein Minimierungsproblem, so stellt dieser Wert eine untere Schranke für den gesuchten Zielfunktionswert des Originalproblems dar. Als obere Schranke dient die gewichtete Summe der Zielfunktionswerte der Subprobleme zuzüglich dem Term des Masterproblems, der nur zur ersten Stufe gehört. Besteht zwischen diesen Schranken Gleichheit, so kann das Verfahren beendet werden.

Es stehen einige Instanzen zur Verfügung, die sich in den GAMS-Programmen, in denen Benders' Dekompositionsverfahren und das ursprüngliche Deterministische Äquivalent implementiert sind, verschieden verhalten. Es zeigt sich, dass sich Instanzen mit einer hohen Anzahl an Szenearien in der ursprünglichen Form als Deterministisches Äquivalent nur schwer lösen lassen. Dahingegen liefert insbesondere Benders' Dekompositionsverfahren mit s. g. *Multicuts* schnell sehr gute Schranken für den Zielfunktionswert. Es fällt aber auch auf, dass mit zunehmender Anzahl an Iterationen das Lösen des Masterproblems langsamer wird und auch die Verbesserungen der Schranken geringfügig ausfallen. Sofern das Deterministische Äquivalent bereits in annehmbarer Zeit eine optimale Lösung liefert, so ist das Heranziehen von Benders' Verfahren nicht notwendig und in der Regel auch langsamer.

Des Weiteren werden die stochastischen Indikatoren *Expected Value of Perfect Information* ($EVPI$) und *Value of the Stochastic Solution* (VSS) vorgestellt. Zur Berechnung des $EVPI$ wird jedes Szeanrio unabhängig von den anderen Szenarien optimal gelöst. Der $EVPI$ ist dann die Differenz zwischen dem über alle Szenarien aggregierten Zielfunktionswert gewichtet mit der Eintrittswahrscheinlichkeit und dem Zielfunktionswert des ursprünglichen Deterministischen Äquivalent. Dahingegen wird beim VSS zunächst der Erwartungswert aller unsicheren Parameter ermittelt und mit diesen das gesamte Problem gelöst. Anschließend wird mit der gefundenen Lösung für die first-stage-Variablen jedes Szenario für sich optimiert. Auch hier ist der VSS die Differenz zwischen der gewichteten Summe der Zielfunktionswerte aller Szenarien und dem Zielfunktionswert des zugrunde liegenden Deterministischen Äquivalents. Die Berechnungen zum $EVPI$ und VSS zeigen, dass für viele Instanzen beide Werte 0 sind. Bei anderen ist das Lösen des zu $EVPI$ gehörenden Modells sehr aufwendig und führt unter Umständen nach 1.000s nicht zu einer Lösung.

Kapitel 6

Fazit und Ausblick

Die Standortplanung ist für jeden Unternehmer wegen der Langfristigkeit dieser Entscheidung von großer Bedeutung. Steht ein mathematisches Modell wie CPLP-Risk zur Verfügung, müssen die Parameter sorgfältig mit passenden Daten belegt werden. Die Unsicherheit kann dabei nie vollständig in die Modellierung aufgenommen werden, obwohl dies durch die Szenarien annähernd gelingen kann. Je mehr Szenarien jedoch eingebunden werden, desto umfangreicher ist die Optimierungsaufgabe und desto zeitintensiver ist die Lösung dieser. An dieser Stelle kann das präsentierte Benders' Dekompositionsverfahren mit Multicuts das Finden einer guten Lösung erheblich beschleunigen, ohne aber die optimale Lösung zu erreichen. Aber auch das Deterministische Äquivalent in seiner ursprünglichen Form kann bereits zu einer optimalen Lösung führen. Es ist im Einzelfall daher sinnvoll, mit beiden Programme zu arbeiten, um herauszufinden, welches für das konkrete Problem geeigneter ist.

Dieser Zusammenhang kann Grundlage für weitere Untersuchungen sein. Von Interesse wäre eine zuverlässige Vorhersage, wann welches Programm eine bessere Lösung ergibt. In dieser Arbeit konnte ein solcher Zusammenhang aus Mangel an Instanzen empirisch nicht genauer untersucht werden.

Ein anderer Ansatzpunkt für weitere Untersuchungen ist die Verbesserung der Performance von Benders' Dekompositionsverfahren. In der jetzigen Form des Verfahrens liegt das Problem vor, dass sich die Schranken bereits nach einigen Iterationen nur noch minimal verbessern.

Kapitel 7

Danksagung

Diese Bachelorarbeit wurde in Zusammenarbeit mit GAMS Software GmbH erarbeitet. Ich danke allen Mitarbeitern für deren Unterstützung und investierte Zeit, insbesondere bei dem praktischen Teil der Arbeit.

Des Weiteren danke ich Prof. Dr. Stefan Nickel und Hannah Bakker vom *Institut für Operations Research: Diskrete Optimierung und Logistik* am *Karlsruher Institut für Technologie* (KIT), die mit dem Thema dieser Arbeit an GAMS Software GmbH herangetreten sind. Auch im weiteren Verlauf der Bearbeitung standen mir beide bei Fragen zur Verfügung und unterstützten meine empirischen Untersuchungen mit Instanzen.

Literaturverzeichnis

[Benders, 1962] Benders, J. F. (1962). *Partitioning Procedures for Solving Mixed-variables Programming Problems.* Technischer Bericht 1, Utrecht, Secaucus, NJ, USA.

[Birge und Louveaux, 2011] Birge, J. R. und Louveaux, F. (2011). *Introduction to Stochastic Programming.* Springer Publishing Company, Incorporated. URL: `http://ie.sharif.edu//%7Esp/%5BJohn_R._Birge,_Fran%C3%A7ois_Louveaux%5D_Introduction_t(BookFi.%20org).pdf`, Stand: 30.01.2019.

[Boyd und Vandenberghe, 2004] Boyd, S. und Vandenberghe, L. (2004). *Convex Optimization.* Cambridge University Press. URL: `https://web.stanford.edu/%7Eboyd/cvxbook/bv_cvxbook.pdf`, Stand: 30.01.2019.

[Burkard und Zimmermann, 2012] Burkard, R. E. und Zimmermann, U. (2012). *Einführung in die Mathematische Optimierung.* Springer Berlin Heidelberg, Berlin, Heidelberg.

[Daum et al., 2010] Daum, A., Greife, W. und Przywara, R. (2010). *BWL für Ingenieure und Ingenieurinnen: Was man über Betriebswirtschaft wissen sollte.* Vieweg+Teubner Verlag / GWV Fachverlage GmbH, Wiesbaden, Wiesbaden.

[GAMS, 2018] GAMS, Development Corporation. (2018). *GAMS-Documentation.* GAMS Software GmbH, URL: `https://www.gams.com/latest/docs/gams.pdf`, Stand: 23.01.2019.

[Gritzmann, 2013] Gritzmann, P. (2013). *Grundlagen der Mathematischen Optimierung: Diskrete Strukturen, Komplexitätstheorie, Konvexitätstheorie, Lineare Optimierung, Simplex-Algorithmus, Dualität.* Aufbaukurs Mathematik. Springer Fachmedien Wiesbaden.

[Heckmann, 2016] Heckmann, I. (2016). *Towards Supply Chain Risk Analytics: Fundamentals, Simulation, Optimization.* Springer Fachmedien Wiesbaden.

[Jensen, 2002] Jensen, P. A. (2002). *Bender's Method.* Technischer Bericht 1. URL: https://www.me.utexas.edu/%7Ejensen/ORMM/methods/unit/integer/subunits/teach_int/benders.pdf, Stand: 30.01.2019.

[Kallrath, 2012] Kallrath, J. (2012). *Algebraic Modeling Systems.* Springer, Heidelberg.

[Murphy, 2013] Murphy, J. (2013). *Benders, Nested Benders and Stochastic Programming: An Intuitive Introduction.* Technischer Bericht 1, Cambridge University Engineering Department. URL: http://www.optimization-online.org/DB_FILE/2013/12/4157.pdf, Stand: 30.01.2019.

[Nyhuis und Wiendahl, 2012] Nyhuis, P. und Wiendahl, H.-P. (2012). *Logistische Kennlinien: Grundlagen, Werkzeuge und Anwendungen.* VDI-Buch. Springer, Berlin, Heidelberg, 3. Auflage.

[Rico-Ramirez, 2002] Rico-Ramirez, V. (2002). *Two-Stage Stochastic Linear Programming.* Technischer Bericht 1, Instituto Tecnologico de Celaya, Mexico. URL: https://www.mcs.anl.gov/%7Eleyffer/views/13-1.pdf, Stand: 30.01.2019.

[Rosenthal, 2017] Rosenthal, R. E. (2017). *GAMS - a User's Guide.* Washington D.C., USA. URL: https://www.gams.com/24.8/docs/userguides/GAMSUsersGuide.pdf, Stand: 30.01.2019.

[Sarykalin et al., 2008] Sarykalin, S., Serraino, G. und Uryasev, S. (2008). Kapitel Value-at-Risk vs. Conditional Value-at-Risk in Risk Management and Optimization, Seiten 270–294. URL: https://www.researchgate.net/publication/200798611_Value-_at-Risk_vs_Conditional_Value-at-Risk_in_Risk_Management_and_Optimization, Stand: 30.01.2019.

[Vanderbei, 2013] Vanderbei, R. J. (2013). *Linear Programming: Foundations and Extensions.* International Series in Operations Research & Management Science. Springer US. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.111.1824&rep=rep1&type=pdf, Stand: 30.01.2019.

[Wittenbrink, 2015] Wittenbrink, P. (2015). *Green Logistics: Konzept, aktuelle Entwicklungen und Handlungsfelder zur Emissionsreduktion im Transportbereich.* Essential. Springer Gabler, Wiesbaden.

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides Statt, dass ich die vorliegende Arbeit selbstständig und ohne unerlaubte Hilfe angefertigt, andere als die angegebenen Quellen nicht benutzt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Ort, Datum

(Unterschrift)

Anhang A

Aufbau der Instanzen

```
|I| - number of facilities, |J| - number of customers,
  |H| - number of capacity extension levels
For each facilites i = 1...I
f_i - fixed cost, K_i - capacity,
  b_i - fixed capacity expansion costs
For each customer j = 1...J
r_j - reward for statisfying one unit of demand
c_ij - transport costs from facilities i=1...I to customer j
For each expansion level  h = 1...H
v_h - expansion costs, o_h - expansion capacity
u - value of service level adherence
beta - target service level
|T| - number of stages, |S| number of scenarios
For each scenario s=1...S
p_s - probability of scenario s
gamma_11s, gamma_12s ...gamma_1Ts
gamma_21s, gamma_22s ...gamma_2Ts
gamma_I1s, gamma_I2s ...gamma_ITs
d_11s, d_12s ...d_1Ts
d_21s, d_22s ...d_2Ts
d_J1s, d_J2s ...d_JTs
```

Anhang B

Inhalt der Datei importFile.inc

```
$onEmbeddedCode Python:

#load file
with open('%1') as file:
   content = [tuple(x.strip().split(',')) for x in file.readlines()]
currentLine=0

#sets I,J,K
I = int(content[currentLine][0])
J = int(content[currentLine][1])
H = int(content[currentLine][2])
Iset = ['i%d' %i for i in range(1,I+1)]
Jset = ['j%d' %j for j in range(1,J+1)]
Hset = ['h%d' %h for h in range(1,H+1)]
currentLine+=1

#parameters of facilities
f=[tuple(['i%d' %(i+1),int(content[currentLine+i][0])]) for i in range(I)]
K=[tuple(['i%d' %(i+1),int(content[currentLine+i][1])]) for i in range(I)]
b=[tuple(['i%d' %(i+1),int(content[currentLine+i][2])]) for i in range(I)]
currentLine+=I

#parameters of customers
r=[tuple(['j%d' %(j+1),float(content[currentLine+2*j][0])]) for j in range(J)]
c=[tuple(['i%d'%(i+1),'j%d'%(j+1),float(content[currentLine+2*j+1][i])]) for i in range(I) for j in range(J)]
currentLine+=2*J

#parameters of expansion levels
v=[tuple(['h%d'%(h+1),float(content[currentLine+h][0])]) for h in range(H)]
o=[tuple(['h%d'%(h+1),float(content[currentLine+h][1])]) for h in range(H)]
currentLine+=H

#scalars refering to the service level
u=[tuple([float(content[currentLine][0])])]
beta0=[tuple([float(content[currentLine+1][0])])]
currentLine+=2

#sets T and S
T=int(content[currentLine][0])
S=int(content[currentLine][1])
Tset = ['t%d' %t for t in range(1,T+1)]
Sset = ['s%d' %t for t in range(1,S+1)]
currentLine+=1


#parameters of the scenarios
p_pre=[['s%d'%(s+1),float(content[currentLine+s*(I+J+1)][0])] for s in range(S)]
summe_data = 0.0
for i in p_pre:
   summe_data += i[1]
diffPerS = (1-summe_data) / len(Sset)
for i in p_pre:
   i[1] = round(i[1] + diffPerS,4)
summe_norm = 0.0
for i in p_pre:
   summe_norm += i[1]
p = [tuple(i) for i in p_pre]
```

```
if (diffPerS > 0.001) or (diffPerS < 0.001 and diffPerS > 0):
    gams.printLog('sum over all probabilities was %f, normalized to %f' %(summe_data,summe_norm))
gamma=[tuple(['s%d'%(s+1),'t%d'%(t+1),'i%d'%(i+1),float(content[currentLine+s*(I+J)+(s+1)+i][t])]) for s in range(S)
    for t in range(T) for i in range(I)]
d=[tuple(['s%d'%(s+1),'t%d'%(t+1),'j%d'%(j+1),float(content[currentLine+s*(I+J)+(s+1)+I+j][t])]) for s in range(S)
    for t in range(T) for j in range(J)]

#push values to gams
gams.set('%2',Iset)
gams.set('%3',Jset)
gams.set('%4',Tset)
gams.set('%5',Hset)
gams.set('%6',Sset)
gams.set('%7',f)
gams.set('%8',K)
gams.set('%9',b)
gams.set('%10',r)
gams.set('%11',c)
gams.set('%12',u)
gams.set('%13',beta0)
gams.set('%14',v)
gams.set('%15',o)
gams.set('%16',p)
gams.set('%17',gamma)
gams.set('%18',d)

$offEmbeddedCode %2 %3 %4 %5 %6 %7 %8 %9 %10 %11 %12 %13 %14 %15 %16 %17 %18
```

Anhang C

Wertetabelle Single- und Multicuts

Iteration	Singlecuts LB	Singlecuts UB	Multicuts LB	Multicuts UB
1	-239.114,2	16,6	-239.114,2	16,6
2	-63.717,2	-56.695,8	-63.402,6	-56.695,8
3	-63.524,5	-58.349,4	-63.058,4	-61.198,1
4	-63.493,6	-59.935,2	-63.047,6	-62.409,5
5	-63.439,6	-59.935,2	-63.027,6	-62.409,5
6	-63.422,2	-59.935,2	-63.020,6	-62.409,5
7	-63.403,2	-61.238,3	-63.000,5	-62.409,5
8	-63.399,2	-61.238,3	-62.994,3	-62.409,5
9	-63.396,0	-61.238,3	-62.986,1	-62.409,5
10	-63.395,2	-61.332,6	-62.985,0	-62.409,5
11	-63.392,2	-61.332,6	-62.974,4	-62.409,5
12	-63.391,2	-61.332,6	-62.974,1	-62.409,5
13	-63.390,2	-61.332,6	-62.972,6	-62.409,5
14	-63.389,5	-61.332,6	-62.970,6	-62.409,5
15	-63.386,5	-61.332,6	-62.970,5	-62.409,5
16	-63.385,2	-61.332,6	-62.969,4	-62.409,5
17	-63.383,2	-61.332,6	-62.966,6	-62.409,5
18	-63.383,2	-61.332,6	-62.965,1	-62.409,5
19	-63.381,2	-61.332,6	-62.964,0	-62.409,5
20	-63.380,2	-61.332,6	-62.961,4	-62.409,5
21	-63.379,2	-61.335,8	-62.960,1	-62.409,5
22	-63.378,2	-61.431,8	-62.959,8	-62.409,5
23	-63.378,2	-61.431,8	-62.959,3	-62.409,5
24	-63.377,2	-61.431,8	-62.958,4	-62.409,5
25	-63.376,2	-61.431,8	-62.958,4	-62.409,5
26	-63.375,2	-61.431,8	-62.950,2	-62.422,0
27	-63.375,2	-61.431,8	-62.949,2	-62.422,0
28	-63.374,5	-61.431,8	-62.948,3	-62.422,0
29	-63.374,5	-61.431,8	-62.945,1	-62.422,0
30	-63.374,5	-61.431,8	-62.944,0	-62.422,0

Tabelle 6: Wertetabelle zu Single- und Multicuts (`medium.txt`). Aufgeführt sind die die unteren und oberen Schranken der ersten 30 Iterationen in `CPLP_Risk_Benders.gms` und `CPLP_Risk_Benders_Multicuts.gms` bezogen auf die Instanz `medium.txt`.