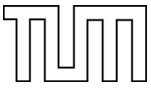


Technische Universität München

Fakultät für Mathematik

Bachelor-Arbeit

Bündelmethoden für Lagrange-Relaxationen: Anwendung auf ein Problem der ambulanten Krankenpflege

Susanne Huber

Aufgabensteller: Prof. Dr. Peter Gritzmann

Betreuer: M.Sc.(hons) Wolfgang Ferdinand Riedl, Dr. Michael Ritter

Abgabetermin: 30.09.2014

Ich erkläre hiermit, dass ich die Bachelorarbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt habe.

Garching, den

Summary

This bachelor thesis describes the Lagrange relaxation. An important step in this solving technique is the way of updating the Lagrange multipliers in it. Therefore two basic techniques, named as the subgradient method and the bundle method, and some betterment of these are introduced. The subgradient method uses line search with subgradients to reach the best multipliers. In comparison to that the bundle method uses an approximation based on a collection of subgradients. The profits and disadvantages of these two methods and their betterments are discussed in this thesis.
Additionally the Lagrange relaxation method is used to solve the optimal scheduling and routing in extramural health care. Therefore two variants of Lagrange relaxation are presented. One of them has been implemented with two different ways of updating the multipliers, the subgradient method and in comparison to that the bundle method. In this implementation the bundle method returned better results for specific parameters. The way of selecting these specific parameters is discussed in the end of this thesis.

Inhaltsverzeichnis

1 Einleitung **5**

2 Lagrange-Relaxation **6**

3 Methoden der Multiplikatoranpassung **10**
3.1 Subgradientenmethode 10
3.1.1 Einfache Subgradientenmethode 10
3.1.2 Verbesserte Verfahren 13
3.2 Bündelmethode 14
3.2.1 Grundlagen der Bündelmethode 14
3.2.2 Anpassungen der Bündelmethode 19
3.2.3 Verbesserte Bündelmethode 22
3.2.4 Vereinfachte Bündelmethode 22
3.3 Diskussion der Methoden 23

4 Anwendung auf das Vehicle Routing Problem **25**
4.1 Vehicle Routing Problem with Time-Windows 25
4.2 Vehicle Routing Problem für Pflegedienste 27
4.3 Die Lagrange Relaxationen 29
4.3.1 Relaxation 1 29
4.3.2 Relaxation 2 31

5 Auswertung **34**
5.1 Ergebnisse der Testdaten 34
5.2 Fazit der Auswertung 39

6 Zusammenfassung und Ausblick **42**

1 Einleitung

Komplexe Probleme gibt es viele in der Optimierung, daher stellt sich die Frage, wie man solche Probleme lösen kann. Eine Methode ist die Methode der Lagrange-Relaxation. Bei dieser Methode wird versucht das komplexe Problem auf bekannte beziehungsweise einfache Probleme zurückzuführen. Nebenbedingungen, die dabei stören, werden mithilfe von Lagrange-Multiplikatoren der Zielfunktion als eine Art Strafterm angehängt. Dabei stellen die Multiplikatoren eine Gewichtung des Strafterms dar. Das Problem soll gelöst und gleichzeitig die relaxierte Nebenbedingung eingehalten werden. Die Wahl der Multiplikatoren im Strafterm ist daher sehr entscheidend, da das eigentliche Ziel des Optimierungsproblems nicht verloren gehen soll.

In dieser Arbeit werden zuerst in Kapitel 2 die Grundlagen der Lagrange-Relaxation erläutert. Hierbei entsteht die Frage, wie man während des Optimierungsprozesses die Multiplikatoren anpasst. In Kapitel 3 werden daher zum einen die Subgradientenmethode, als auch die Bündelmethode, sowie Modifikationen derer vorgestellt.
Im Anschluss wird in Kapitel 4 ein Beispiel für ein solches komplexes ganzzahliges lineares Optimierungsproblem anhand der ambulanten Krankenpflege eingeführt. Hierbei müssen den Patienten, die noch in ihrem eigenen Haus leben, Pflegekräfte zugeordnet werden, die bei der täglichen Medikamentengabe oder der täglichen Hygiene helfen. Durch die vielen Rahmenbedingungen, die bei der Zuordnung beachtet werden müssen, wird das Problem sehr komplex.
Dieses Problem wurde mithilfe von Mosel implementiert. In Kapitel 5 werden die Ergebnisse der Subgradientenmethode mit denen der Bündelmethode verglichen.
In Kapitel 6 wird eine kurze Zusammenfassung und ein Ausblick auf offene Fragen, die sich aus dieser Arbeit ergeben, dargestellt.

2 Lagrange-Relaxation

Die Lagrange-Relaxation findet große Anwendung im Bereich der Optimierung, vor allem durch die Vielzahl an Einsatzmöglichkeiten. Man kann diese Lösungsmethode sowohl in der linearen, nichtlinearen und ganzzahligen Programmierung, als auch in der kombinatorischen Optimierung verwenden. Ziel der Methode ist es mit der Hilfe von Lagrange-Multiplikatoren spezielle Nebenbedingungen in die Zielfunktion einzubinden und dadurch das Optimierungsproblem auf bekannte Probleme zurückzuführen. Das Optimierungsproblem zerfällt dabei meist in mehrere Teilprobleme. Somit kann man schneller beziehungsweise überhaupt zu einem Ergebnis kommen.

Beispiel 2.1. *In der Fertigungsplanung (Job-Shop-Scheduling) ist das Problem, dass die Fertigungsschritte in ihrem zeitlichen Ablauf an allen Maschinen geplant werden müssen. Wendet man hier die Lagrange-Relaxation an, zerfällt das Problem, sodass der Fertigungsablauf für jede Maschine einzeln geplant werden kann und das Augenmerk nicht mehr auf dem Zusammenspiel der Maschinen ruht. Dies vereinfacht das Problem sehr. Für eine nähere Betrachtung siehe [1].*

Zur Einführung in die Anwendung der Lagrange-Relaxation betrachten wir folgendes lineares Optimierungsproblem:

Sei $A \in \mathbb{R}^{m \times n},\ b \in \mathbb{R}^m,\ c \in \mathbb{R}^n,\ X = \{x_1, \ldots, x_{|X|}\} \subset \mathbb{R}^n,\ |X|$ endlich

---

Problemstellung 2.1: P

$$\begin{array}{rll} & \min & c^\top x \\ \textbf{u.d.N} & & \\ & Ax & = b \\ & x & \in X \end{array}$$

Hierbei ist $\mathfrak{P} := \{x | Ax = b,\ x \in X\}$ die Menge der zulässigen Lösungn und $x^* \in \operatorname{argmin}_{x \in \mathfrak{P}}\ c^\top x$ eine optimale zulässige Lösung.
Hinter der Menge X können sich auch weitere Nebenbedingungen verbergen, wie z.B. $X = \{x \in \mathbb{R} : d^\top x \leq B\}$ oder die Einschränkung auf ganzzahlige Lösungen.
Der Wert der optimalen Lösung von $[P]$ ist folgendermaßen definiert:

Definition 2.1. *Der Wert der optimalen Lösung des Problems* (2.1) *ist definiert als*

$$Z^* = \min\{c^\top x \ :\ Ax = b\ ,\ x \in X\}$$

Die Nebenbedingung $Ax = b$ sei in diesem Problem eine spezielle Nebenbedingung, die das Problem komplex macht. Um das Problem zu vereinfachen wird diese Bedingung in der Zielfunktion mit einem Faktor addiert. Falls das Problem durch das Relaxieren der Nebenbedingung in mehrere Teilprobleme zerfällt, so können die Teilprobleme vorerst unabhängig voneinander gelöst werden. In diesem Fall wird die Lagrange-Methode auch

Zerlegungsmethode genannt, weil das ursprüngliche Problem zum Lösen in mehrere Teilprobleme zerlegt wird.

Die Zielfunktion des relaxierten Problems ist dann wie folgt:

Definition 2.2. *Die Funktion des relaxierten Problems für ein gegebenes $\lambda \in \mathbb{R}^m$ wird definiert als*

$$L(\lambda, x) = c^\top x + \lambda^\top (Ax - b),\ x \in X$$

Das neu entstandene Problem wird *Lagrange-Relaxation* des eigentlichen Problems genannt und lässt sich folgendermaßen darstellen:

Problemstellung 2.2: $P_\lambda\ (\lambda \in \mathbb{R}^m)$

$$\begin{array}{ll} & \min \quad L(\lambda, x) \\ \textbf{u.d.N} & \\ & x \in X \end{array}$$

Diesbezüglich definiert man die *Lagrangefunktion* als das Minimum der Funktion des relaxierten Problems.

Definition 2.3. *Die Lagrangefunktion ist definiert als*

$$L(\lambda) = \min_{x \in X} L(\lambda, x) = \min\{c^\top x + \lambda^\top (Ax - b)\ :\ x \in X\} \qquad ,\ \lambda \in \mathbb{R}^m$$

Durch die Einbindung der speziellen Nebenbedingung in die Zielfunktion ist diese nicht mehr strikt vorgegeben. Nur durch eine Verschlechterung der Zielfunktion nimmt eine Missachtung der entfernten Nebenbedingung Einfluss. Die Lösungsvektoren dieses neuen Optimierungsproblems $x^*_\lambda = \text{argmin}_{x \in X} L(\lambda, x)$ sind somit nicht unbedingt zulässig für das originale Optimierungsproblem. Ist jedoch $x \in \mathfrak{P}$, so verschwindet der Lagrange-Term aus der Zielfunktion des relaxierten Problems $[P_\lambda]$ und man erhält die eigentliche Zielfunktion von $[P]$.

Die Lagrangefunktion bildet jedoch immer eine untere Schranke an den optimalen Wert der originalen Zielfunktion Z^*.

Lemma 2.1. *Für alle Werte von $\lambda \in \mathbb{R}^m$ gilt*

$$L(\lambda) \leq Z^*$$

Beweis. Es gilt

$$\begin{aligned} Z^* &= \min\{c^\top x\ :\ Ax = b,\ x \in X\} \\ &= \min\{c^\top x + \lambda^\top (Ax - b)\ :\ Ax = b,\ x \in X\} \end{aligned}$$

da $Ax - b = 0$ ist. Durch das Wegnehmen einer Restriktion kann sich der Zielfunktionswert im Vergleich zu Z^* allerdings nicht verschlechtern, da hier über eine größere Menge

minimiert wird. Das heißt, zu den $x \in X$ mit $Ax = b$ werden auch diejenigen für die Minimierung in Betracht gezogen, für die $Ax \neq b$ gilt, sodass der neue Optimalwert nur kleiner als Z^* sein kann:

$$\begin{aligned} Z^* &= \min\{c^\top x + \lambda^\top (Ax - b) \ : \ Ax = b, \ x \in X\} \\ &\geq \min\{c^\top x + \lambda^\top (Ax - b) \ : \ x \in X\} = L(\lambda) \end{aligned}$$

□

Um eine möglichst gute untere Schranke für die optimale Lösung des eigentlichen Problems zu bekommen, maximieren wir die Lagrangefunktion über λ. Dieses Optimierungsproblem wird oft als das *Lagragian Dual* oder *Lagrange-Multiplikator Problem* bezeichnet.

Problemstellung 2.3: LAGRANGE-MULTIPLIKATOR PROBLEM

$L^* = \max_{\lambda \in \mathbb{R}^m} \ L(\lambda)$

Wir erhalten somit mit der gleichen Argumentation wie in Lemma (2.1) folgende Reihenfolge:

$$L(\lambda) \leq L^* \leq Z^* \leq c^\top x \qquad\qquad \forall \ x \in X, \ \lambda \in \mathbb{R}^m$$

Wir können demnach sicher sein eine optimale Lösung $x = x^*$ von $[P]$ gefunden zu haben, wenn für ein festes λ die Bedingung $L(\lambda) = c^\top x$ erfüllt ist, also Gleichheit in der obigen Gleichungskette gilt. Dies ist gleichbedeutend mit dem Test, ob eine gefundene eine Lösung L^* des dualen Lagrange Problems auch für das primale Problem $[P]$ zulässig ist, d.h. ob $x \in \mathfrak{P}$ ist und somit der Lagrange-Term wegfällt.

Auch wenn man nicht die optimale Lösung findet, kann man mit Hilfe des dualen Lagrange Problems Aussagen darüber treffen, wie weit die erhaltene Lösung von der optimalen entfernt ist. Die Differenz zwischen L^* und Z^* bezeichnet man dabei als *Dualitätslücke*. Man betrachtet hierfür

$$\frac{c^\top x - L(\lambda)}{L(\lambda)}$$

Dies ist eine obere Schranke an die maximale, prozentuale Abweichung von der optimalen Lösung.

In der Praxis kommen allerdings nicht nur Gleichheitsnebenbedingungen $Ax = b$ vor. Falls schwierige Nebenbedingungen der Form $Ax \leq b$ vorkommen, ist lediglich ein leicht verändertes duales Lagrange Problem zu betrachten. Der Multiplikator λ darf nur positive Werte annehmen.

$$L^* = \max_{\lambda \geq 0} L(\lambda)$$

Die Schreibweise $\lambda \geq 0$ ist hierbei komponentenweise zu verstehen.
Die Schrankeneigenschaft gilt mit ähnlicher Argumentation wie in Lemma (2.1) weiterhin,

da auch hier durch das Wegnehmen der Nebenbedingung sich die Lösung der Minimierungsaufgabe nicht verschlechtern kann. Jedoch muss bei dieser Variante eine optimale Lösung x^* von $L^* = L(\lambda)$ für ein $\lambda \geq 0$, das für das primale Problem zulässig ist, keine optimale Lösung für das duale sein. Hierfür muss man folglich den Optimalitätstest anpassen:

Proposition 2.2. *Existiert ein $x \in X$ mit $L(\lambda) = c^\top x + \lambda^\top (Ax - b)$, so dass $x \in \mathfrak{P}$ und $\lambda^\top (Ax - b) = 0$ erfüllt ist, so ist $L(\lambda)$ die optimale duale Schranke und x die optimale Lösung für das primale Problem.*

Beweis. Es gilt

$$L(\lambda) = c^\top x - \underbrace{\lambda^\top (Ax - b)}_{=0} = c^\top x$$

Da $x \in \mathfrak{P}$ ist, ist x zulässig für das Primale Problem $[P]$. Die Dualitätslücke ist geschlossen und somit ist dieses x optimal. □

Diese Methode kann unter den obigen Bedingungen folglich Lösungen für das Primale Problem $[P]$ finden. Der Nachteil der Lagrange-Relaxation ist jedoch, dass eine gefundene Lösung nicht zulässig für das Primale Problem sein muss, da die relaxierte Nebenbedingung und die zusätzlichen Bedingungen des Optimalitätstests nicht unbedingt erfüllt sind.
Dennoch bietet die Lagrange-Relaxation viele Vorteile. Ein Vorteil der Lösung mit der Lagrange-Methode ist zum Beispiel, dass sie für eine Vielzahl von Problemen anwendbar und dabei sehr flexibel ist. Die breit gefächerten Anwendungsmöglichkeiten basieren auf der Tatsache, dass den meisten Problemen eine spezielle Struktur zugrunde liegt. Jedem Transport- oder Zuordnungsproblem liegen zum Beispiel eine Netzstruktur aus Knoten und Kanten zugrunde. In vielen Modellen lassen sich verbindende Nebenbedingungen finden. Durch deren Relaxierung mit Lagrange-Multiplikatoren lassen sich Subprobleme bilden. Diese Subprobleme können eigenständig gelöst und meist direkt auf standardisierte Probleme zurückgeführt werden.

Auch wenn die richtige Lösung nicht gefunden wird, erhält man Schranken an die Zielfunktion und oft in kurzer Zeit eine annähernd perfekte Lösung. In vielen Fällen der ganzzahligen Optimierung sind diese Schranken besser als die durch lineare Programmierung. Durch Lagrange-Relaxationen kann man häufig effektive, heuristische Lösungsmethoden für kombinatorische Optimierung und ganzzahlige Programmierung finden.

Dis Grundidee der Lagrange-Relaxion ist nun bekannt, doch das Verfahren zur Lösung des Lagrange-Multiplikator Problems 2.3 blieb bisher offen. Hierzu werden im nächsten Kapitel einige Herangehensweisen präsentiert.

3 Methoden der Multiplikatoranpassung

Im folgenden Abschnitt wollen wir uns mit der Lösung des dualen Lagrange-Problems befassen. Um die untere Schranke zu verbessern, müssen wir über λ maximieren. Bei der Maximierung ist es dann von großem Vorteil, dass $L(\lambda)$ eine konkave Funktion ist.

Theorem 3.1. *$L(\lambda)$ ist eine konkave Funktion.*

Beweis. Aus Definition 2.3 von $L(\lambda)$ ist

$$L(\lambda) \leq \; L(\lambda, x) \qquad\qquad \forall\, x \in X$$

$L(\lambda, x)$ ist dabei für ein fixes x eine affine Funktion in λ gemäß der Definition 2.2. Daher ist $L(\lambda)$, als Minimum über affine Funktionen, konkav (siehe auch [2]). Da X endlich ist, ist $L(\lambda)$ das Minimum über begrenzt viele affine Funktionen und somit konkav und stückweise linear. □

Jedes lokale Maximum ist folglich ein globales Maximum und es reicht daher einfache Aufstiegsalgorithmen anzuwenden. Diese basieren auf der geschickten Wahl einer Aufstiegsrichtung zum Maximum hin. In dieser Richtung wird eine neue Iterierte gesucht, deren Funktionswert möglichst näher am Maximum liegen soll. Je nach Art des Aufstiegsalgorithmus kann zusätzlich eine Schrittlänge berechnet werden, die bestimmt wie weit der Algorithmus in die Aufstiegsrichtung geht.

In dem Ablauf der Problemlösung muss zunächst die relaxierte Funktion $L(\lambda_0, x)$ für ein initiales λ_0 gebildet werden. Diese wird für das fixe λ minimiert. Aus der Lösung $x_0 = \operatorname{argmin}_{x \in X} L(\lambda_0, x)$ der Minimierung wird eine Aufstiegsrichtung berechnet. Mit Hilfe dieser Aufstiegsrichtung (und eventuell einer Schrittweite) wird ein neues λ_1 berechnet. In jeder weiteren Iteration k muss daher zuerst für λ_k das Problem $[P_\lambda]$ gelöst werden, damit anschließend eine neue Aufstiegsrichtung und damit ein neues λ_{k+1} berechnet werden kann.

Im Folgenden werden einige Varianten der Bildung einer Aufstiegsrichtung vorgestellt. Die Subgradientenmethode bildet hierbei eine Art Basisverfahren, auf dem die anderen aufbauen. Jede weitere Methode versucht dann entweder die Komplexität zu verringern oder die Genauigkeit der Vorgängermethode zu verbessern.

Bemerkung 3.1. *In der Literatur findet man im Zusammenhang mit der Lagrange-Methode häufig die Conjugate-Gradient-Methode. Diese Methode der Aufstiegsalgorithmen kann zur Lösung von mehrfach differenzierbaren Problemen angewandt werden, ist jedoch nicht für lineare Probleme einsetzbar. Daher sei hier auf [3] verwiesen.*

3.1 Subgradientenmethode

3.1.1 Einfache Subgradientenmethode

Die Subgradientenmethode kann verwendet werden, um eine konkave Funktion zu maximieren. Dabei versucht die Subgradientenmethode mit Hilfe des steilsten Anstiegs möglichst schnell das Maximum zu erreichen. Hierbei muss die Funktion nicht unbedingt

differenzierbar sein. Die Aufstiegsrichtung wird durch die Subgradienten festgelegt, die wie folgt definiert sind:

Definition 3.1. *Subgradient*
Sei $L(\lambda) : \mathbb{R}^m \to \mathbb{R}$ eine konkave Funktion. Dann ist $g \in \mathbb{R}^m$ ein Subgradient der Funktion $L(\lambda)$ bei dem aktuellen $\lambda_k \in \mathbb{R}^m$ in Iteration k, falls für alle $\lambda \in \mathbb{R}^m$ gilt:

$$L(\lambda) \leq L(\lambda_k) + g^\top(\lambda - \lambda_k)$$

Der Subgradient an der Stelle λ_k, im Weiteren als $g_k := g(\lambda_k)$ bezeichnet, wird demnach aus der folgenden Menge gewählt:

$$\partial L(\lambda_k) := \{g \in \mathbb{R}^n | L(\lambda) \leq L(\lambda_k) + g^\top(\lambda - \lambda_k)\ \forall \lambda \in \mathbb{R}^n\}$$

Man erkennt, dass der Subgradient von dem gewählten beziehungsweise bisher berechneten λ_k und der Lagrangefunktion beziehungsweise den Subproblemen bei λ_k abhängt.

Zwar ist die Subgradienten-Methode nicht immer aufsteigend, doch die Richtung der Methode bildet immer einen spitzen Winkel mit der Richtung zu λ^*.

Lemma 3.2. *Zwischen dem Subgradienten an der Stelle λ_k und der Richtung zu λ^*, d.h. $\lambda^* - \lambda_k$, liegt immer ein spitzer Winkel.*

Beweis. Die optimale, also größte untere Schranke mit dem optimalen Multiplikator λ^*, im Folgenden mit $L^* = L(\lambda^*)$ bezeichnet, ist immer größer als die aktuelle Schranke $L_k = L(\lambda_k)$. Daher und aus der Definition 3.1 muss für den Subgradienten die folgende Gleichung gelten:

$$0 \leq L^* - L_k \leq g_k^\top(\lambda^* - \lambda_k) \tag{1}$$

Der Subgradient bildet mit der optimalen Richtung ein positives Skalarprodukt. Somit bildet die Aufstiegsrichtung immer einen spitzen Winkel mit der Richtung zu λ^*. □

Lemma 3.3. *Für unser lineares Optimierungsproblem [P] (vgl. Problemstellung 2.1) kann man $g_k = Ax_k - b$ als Optimierungsrichtung wählen.*

Beweis. Setzt man in die Ungleichung (1) die Definitionen für L^* und L_k, so erhält man

$$\begin{aligned} g_k^\top(\lambda^* - \lambda_k) &\geq L^* - L_k \\ &= c^\top x_k + (\lambda^*)^\top(Ax_k - b) - c^\top x_k - \lambda_k^\top(Ax_k - b) \\ &= (Ax_k - b)^\top(\lambda^* - \lambda_k) \end{aligned}$$

Diese Ungleichung ist für $g_k = Ax_k - b$ erfüllt. □

Das Update der Lagrange-Multiplikatoren in der Subgradienten-Methode erfolgt durch:

$$\lambda_{k+1} = \lambda_k + \sigma_k g_k$$

wobei σ_k die Schrittweite in der k-ten Iteration darstellt.

Der Algorithmus der Lagrange-Relaxation mit der Subgradientenmethode zum Update der Multiplikatoren ist dann wie folgt:

Algorithm 1: Subgradientenmethode

Data: $\lambda_0 \in \mathbb{R}$
Result: λ^*
$k = 0$
while $0 \notin \partial L(\lambda_k)$ **do**
  Berechnung der Aufstiegsrichtung
  $x_k = \operatorname{argmin}\{c^\top x + \lambda_k^\top (Ax - b) : x \in X\}$
  $g_k = Ax_k - b$

  Update des Multiplikators
  $\lambda_{k+1} = \lambda_k + \sigma_k g_k$

  $k = k + 1$
end

Im obigen Algorithmus ist die Schrittweite σ_k noch nicht genauer spezifiziert worden. Die richtige Justierung der Schrittweite kann Probleme bereiten. Allgemein ist es sinnvoll mit steigender Anzahl an Iterationen die Schrittweite zu verringern, jedoch sollte sie dabei nicht vor der Lösungsfindung konvergieren.

Man kann die Schrittweite zum Beispiel unabhängig von dem Problem wählen. Hierdurch erspart man sich längere Rechenprozesse während dem Algorithmus. Im Folgenden sind einige problemunabhängige Schrittweiten angegeben. Eine genauere Beschreibung und Konvergenzanalyse dieser Methoden findet man in [4].

- Konstante Schrittgröße: $\sigma_k = a > 0 \ \forall k$
- $\sigma_k \geq 0,\ \sum_{k=1}^{\infty} \sigma_k^2 < \infty,\ \sum_{k=1}^{\infty} \sigma_k = \infty$, d.h. $\sigma_k = \frac{a}{b+k},\ a > 0,\ b \geq 0$
- Verschwindende Schrittgröße: $\sigma_k \geq 0,\ \lim_{k\to\infty} \sigma_k = 0,\ \sum_{k=1}^{\infty} \sigma_k = \infty$, z.B. $\sigma_k = \frac{a}{k},\ a > 0$

Bemerkung 3.2. $\|\cdot\|$ *steht in der gesamten Arbeit für die Euklidische Norm,* $\|\cdot\|_k$ *für die entsprechende k-Norm*

Die andere Art der Schrittweitenwahl bezieht sich auf die momentane Aufstiegsrichtung. Hierdurch wird die Schrittweite genauer an das Problem angepasst, was jedoch mehr Rechenzeit verlangt. Eine genauere Beschreibung und Konvergenzanalyse zu den folgenden zwei Methoden findet man auch hier in [4].

- Konstante Schrittlänge: $\sigma_k = \frac{\gamma}{\|g_k\|},\ \gamma > 0$ d.h. $\|\lambda_{k+1} - \lambda_k\| = \gamma$
- Verschwindende Schrittlänge: $\sigma_k = \frac{\gamma_k}{\|g_k\|},\ \gamma_k \geq 0,\ \lim_{k\to\infty} \gamma_k = 0,\ \sum_{k=1}^{\infty} \gamma_k = \infty$

Die in der Literatur am meisten verwendete Methode der Schrittweitenwahl ist jedoch die folgende, problemabhängige Methode. Die Idee hinter dieser Methode ist, die Schrittweite genau so zu wählen, das die optimale untere Schranke L^* theoretisch mit dem nächsten Schritt erreicht wird.

$$\underbrace{L(\lambda_{k+1}) = c^\top x_k + \lambda_{k+1}^\top (Ax_k - b) = c^\top x_k + (\lambda_k + \sigma_k g_k)^\top (Ax_k - b)}_{\text{untere Schranke nach dem nächsten Schritt}} = \underbrace{L^*}_{\text{optimale Lösung}}$$

Aufgelöst nach σ_k und mit $g_k = Ax_k - b$ ergibt sich dann

$$\sigma_k = \frac{L^* - c^\top x_k - \lambda_k^\top (Ax_k - b)}{\|g_k\|^2} = \frac{L^* - L(\lambda_k)}{\|g_k\|^2}$$

Da wir allerdings die optimale untere Schranke L^* nicht kennen, begnügen wir uns mit einer oberen Schranke $\bar{\theta}$ an das primale Problem, was somit auch eine obere Schranke an L^* darstellt.

$$\sigma_k = \gamma \frac{\bar{\theta} - L(\lambda_k)}{\|g_k\|^2}$$

wobei $\gamma \in (0, 2]$ ein regulierender Faktor ist. Für die Wahl und die Anpassung des regulierenden Faktors sei hier auf [5] verwiesen.
In jedem Schritt muss somit die Schrittweite neu berechnet und eventuell der regulierende Faktor angepasst werden, was einen wesentlichen Mehraufwand zu den problemunabhängigen Schrittweiten darstellt. Da hier auch eine obere Schranke berechnet werden muss, ist diese Methode der Schrittweitenwahl am aufwendigsten, jedoch auch am besten.

Wir haben nun mit der Subgradientenmethode ein einfaches Verfahren kennengelernt, mit dem die Lagrange-Relaxation in jedem Schritt angepasst werden kann. Dieses Verfahren ist nun Basis für einige Verbesserungen. Diese werden im folgenden Kapitel vorgestellt.

3.1.2 Verbesserte Verfahren

Das kommende Kapitel beinhaltet zwei Verbesserungen des Subgradientenverfahrens. Diese Verbesserungen eignen sich besonders gut für Probleme, die durch die Relaxation zerfallen. Die Wirkungsweise für Probleme, die nicht diese Struktur haben, ist nicht bekannt. Für mehr Details zu diesen Verfahren wird hier auf [6] verwiesen.

Angenommen es liegt ein Problem vor, das durch die Relaxation mit Lagrange-Multiplikatoren in N Subprobleme zerfällt. In der einfachen Subgradientenmethode muss in jeder Iteration jedes dieser N Subprobleme berechnet werden. Dies ist notwendig, um x_k und anschließend den Subgradienten zu berechnen. Die Berechnung aller Subprobleme kann daher recht aufwendig werden, wenn es viele beziehungsweise schwere Subprobleme sind.

Eine Methode, die diese Schwachstelle verbessert, ist die Interleaved-Subgradientenmethode. Hierbei muss nur ein Subproblem pro Iteration gelöst werden. Mit dem Subgradienten des einen Subproblems erneuert man den Wert für λ. Das heißt, falls das Problem in N

Subprobleme zerfällt, so ist der Rechenaufwand $\frac{1}{N}$ im Vergleich zur Subgradientenmethode. In der Praxis konvergiert diese Methode zwar schneller, jedoch ist die Konvergenz bisher noch nicht bewiesen worden.

Eine Verallgemeinerung der Interleaved-Subgradientenmethode ist die Surrogate-Subgradientenmethode. Bei der Surrogate-Subgradientenmethode wird auch auf die exakte Berechnung aller Subprobleme verzichtet. Man greift hierbei auf approximative Lösungen zurück. Die Konvergenz der Surrogate-Subgradientenmethode ist dennoch gesichert, im Gegensatz zu der Konvergenz der Interleaved-Subgradientenmethode.
Man muss tatsächlich nur ein Subproblem approximativ lösen, um eine gute Suchrichtung zu erhalten. Der Rechenaufwand dieser Methode ist bei N Subproblemen kleiner als $\frac{1}{N}$ im Vergleich zur Subgradienten Methode. Diese Methode bietet sich folglich für sehr große Probleme an.

3.2 Bündelmethode

Die Bündelmethode ist auch eine Verbesserung des Subgradientenverfahrens. Hierbei werden ebenfalls die Subgradienten berechnet, jedoch werden hier die Informationen der vorangegangenen Iterationen in einer Menge gesammelt und benutzt, um eine neue Schrittrichtung zu bestimmen. Dies macht das Verfahren weniger schwankend, jedoch sind die Berechnungen komplizierter und es ist mehr Speicherplatz nötig. Die Menge der gesammelten Informationen wird im Folgenden als Bündel bezeichnet.

3.2.1 Grundlagen der Bündelmethode

Bei der Bündelmethode wird die konkave Funktion $L(\lambda)$ durch Stützgeraden angenähert. Diese Stützgeraden berühren die Funktion in den bisherigen Punkten $(\lambda_i; L(\lambda_i))$ $i \in \{1, \ldots, k\}$ (vgl. Abbildung 3.2.1). Die Speicherung der bisherigen Stützgeraden in einem Bündel ist daher nötig, um die Approximation zu berechnen. Je höher die Iterationszahl ist, desto mehr Stützgeraden werden mit einbezogen und desto genauer wird die Approximation. Der Algorithmus wird auch Cutting-Plane-Algorithmus genannt, da durch das Hinzufügen neuer Stützgeraden teilweise Funktionswerte der vorangegangenen Approximation ausgeschlossen werden.

Konkret ausgedrückt haben wir folgendes:

- eine konkave Funktion $L(\lambda)$
- k iterierte λ_i, $i \in \{1, \ldots, k\}$
- deren Funktionswerte $L(\lambda_i) = c^\top x_i + \lambda_i^\top (Ax_i - b)$
- und die jeweiligen Subgradienten
  $g_i = g(\lambda_i) \in \partial L(\lambda_i) = \{g \in \mathbb{R}^n | L(\lambda) \leq L(\lambda_i) + g^\top(\lambda - \lambda_i)\ \forall \lambda \in \mathbb{R}^n\}$.

Die Informationen $(\lambda_i; L(\lambda_i); g_i)$ werden für alle $i \in \{1, \ldots, k\}$ in einem Bündel $B_k = \{(\lambda_i, L(\lambda_i), g_i),\ i = 1, \ldots, k\}$ gespeichert, um für die Berechnung der stückweise linearen Funktion $\bar{L}_k(\lambda)$ in Iteration k zur Verfügung zu stehen. Diese ist wie folgt definiert:

Definition 3.2. *Die linearen Stützfunktionen in Iteration k sind definiert als*

$$\bar{L}(\lambda;\lambda_i) := L(\lambda_i) + g_i^\top(\lambda - \lambda_i) \qquad \forall\, i \in \{1,\dots,k\}$$

Die untere Hülle der Stützgeraden in Iteration k ist definiert als

$$\bar{L}_k(\lambda) := \min_{i\in\{1,\dots,k\}} \bar{L}(\lambda;\lambda_i)$$

Aus der Definition erkennt man, dass die Approximation $\bar{L}_k(\lambda)$ und $L(\lambda)$ an den Stützstellen λ_i übereinstimmen, da der hintere Summand der Approximation wegfällt.

Proposition 3.4. *$\bar{L}_k(\lambda)$ ist eine Approximation von oben an $L(\lambda)$.*

Beweis. Da $L(\lambda)$ konkav ist liegen alle Funktionswerte der Stützgeraden $\bar{L}(\lambda;\lambda_i)$ für alle $i \in \{1,\dots,k\}$ über denen der konkaven Funktion $L(\lambda)$. $\bar{L}_k(\lambda)$ ist nach Definition 3.2 eine untere Hülle aus diesen Stützgeraden an die konkave Funktion $L(\lambda)$. Daher müssen auch die Funktionswerte von $\bar{L}_k(\lambda)$ über denen von $L(\lambda)$ liegen. (vgl. folgende Abbildung 3.2.1) □

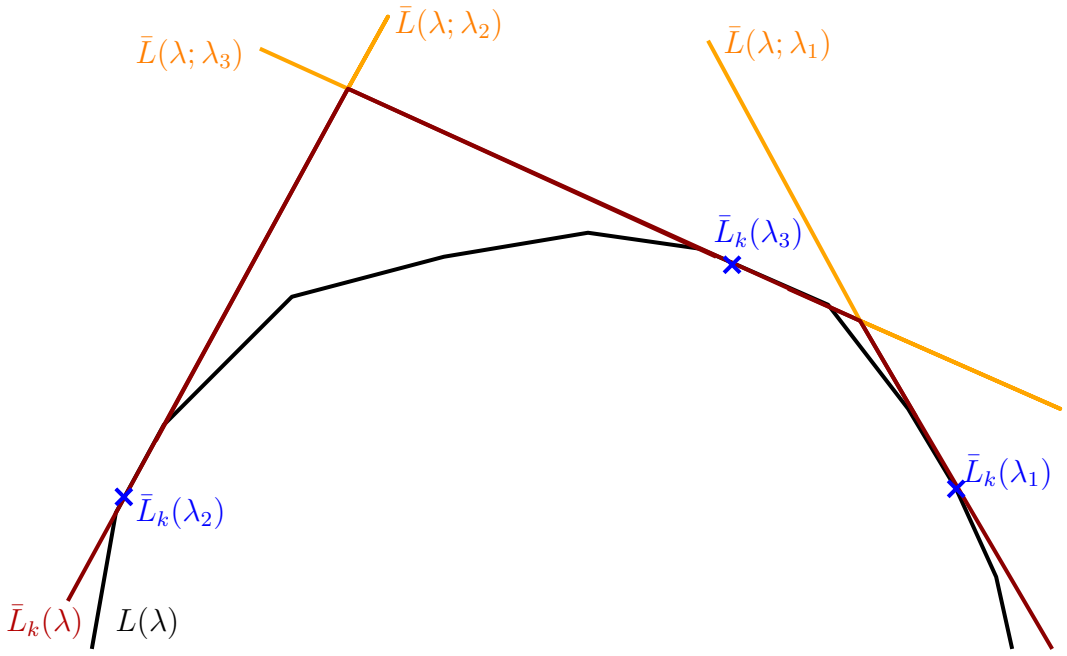


Abbildung 1: Konkave Funktion $L(\lambda)$ und Approximation $\bar{L}_k(\lambda)$

Der Zusammenhang zwischen den drei Funktionen lässt sich in der folgenden Ungleichung verdeutlichen:

$$\bar{L}(\lambda;\lambda_i) \geq \bar{L}_k(\lambda) \geq L(\lambda) \quad \text{und } \bar{L}_k(\lambda_i) = L(\lambda_i)\ \forall\, i \in \{1,\dots,k\}$$

Mit der Annahme, dass wir die Funktion nur auf einer konvexen, kompakten Menge C betrachten, erhalten wir den neuen Multiplikator durch Maximierung über $\bar{L}_k(\lambda)$.

Problemstellung 3.1: STÜTZFUNKTIONEN

$\lambda_{k+1} \in \operatorname{argmax}_{\lambda \in C} \bar{L}_k(\lambda)$

Der folgende Algorithmus löst die Lagrange-Relaxation, wobei die Multiplikatoren mit Hilfe der Stützfunktionen aktualisiert werden.

Algorithm 2: Basic Cutting Plane Algorithm

Data: C konvex; $\lambda_1 \in C$
Result: λ^*
$k := 1$
Berechne $L(\lambda_1),\ g_1$
while *Abbruchkriterium nicht erfüllt* **do**
  Berechnen der unteren Hülle der Stützgeraden
  for $i = \{1, \ldots, k\},\ \lambda \in C,\ r \in \mathbb{R}$ **do**
    $r \leq \bar{L}(\lambda; \lambda_i)$
  end
  $\bar{L}_k(\lambda) := \max r$

  Update des Multiplikators
  $\lambda_{k+1} := \operatorname{argmax} \bar{L}_k(\lambda)$
  Berechne $L(\lambda_{k+1}),\ g_{k+1}$

  $k := k + 1$
end

Als Abbruchkriterium dient der Abstand zwischen der Lagrangefunktion und der Annäherung bei dem aktuellen Multiplikator. Ist dieser kleiner als ein zuvor festgelegter Parameter $\delta > 0$, d.h.

$$\bar{L}_{k-1}(\lambda_k) - L(\lambda_k) \leq \delta$$

so wird der Algorithmus abgebrochen.

Wir haben nun einen Algorithmus, der mit steigender Iterationszahl immer bessere Approximationen für die Lagrangefunktion liefert. Für diese Approximationen kann leichter ein Maximum gefunden werden, das sich somit auch mit steigender Iterationszahl dem Maximum der Lagrangefunktion annähert.
Doch dieser Algorithmus hat ein paar Schwachstellen: Zum einen ist die Maximierung über die Stützgeraden oft unbeschränkt, da die Stützgeraden nicht unbedingt in jeder Dimension gegenläufige Steigungen haben. Zum anderen kann der Algorithmus je nach der Wahl der Startpunkte recht instabil sein.
Mit steigender Iterationszahl ergibt sich ein weiterer Nachteil im obigen Algorithmus. Je näher man dem Optimum kommt, desto eher verschlechtert sich nach einem Schritt der Funktionswert von $L(\lambda)$, da zu große Schritte gemacht werden.

Um diese Problematik zu beheben, können mehrere Varianten angewandt werden. Diese haben das Ziel dafür zu sorgen, dass der Algorithmus keine zu großen Schritte macht, die ihn weit von der bisherigen Lösung bringen würden. Hierdurch spart man sich die Einschränkung auf die kompakte Menge C.

Eine dieser Methoden zur Einschränkung ist die Trust-Region Methode. Hierbei wird festgelegt, dass bei der Maximierung von $\bar{L}(\lambda)$ nur diejenigen λ in Betracht gezogen werden, die in einem κ-Ball um ein Stabilitätszentrum $\mathfrak{Z}_k$ liegen.

Problemstellung 3.2: TRUST-REGION

$$\lambda_{k+1} \in \operatorname{argmax}\{\bar{L}_k(\lambda) : \ \|\lambda - \mathfrak{Z}_k\| \leq \kappa\}$$

Diese Art der Beschränkung ist sehr strikt und je nach Wahl der Größe von κ kann sie sehr hinderlich sein. Deswegen betrachten wir die zweite Variante. Die Beschränkung wird als Strafterm in die Maximierung eingebaut. Hierbei werden starke Verbesserungen, die knapp außerhalb der Trust-Region liegen auch betrachtet.
Das neue λ_{k+1} wird durch Lösen des quadratischen Problems $[QP_k]$ berechnet:

Problemstellung 3.3: QP_k

$$\lambda_{k+1} \in \operatorname{argmax}\{\bar{L}_k(\lambda) - \tfrac{u}{2}\|\lambda - \mathfrak{Z}_k\|^2\}$$

In Abbildung 2 wird die Veränderung durch den Strafterm sichtbar. Je weiter sich λ vom Stabilitätszentrum $\mathfrak{Z}_k$ entfernt, desto mehr nimmt der Strafterm Einfluss.
Die Umwandlung in ein mindestens quadratisches Problem ist bei dieser Art der Stabilisierung nötig. Unter Verwendung der 1-Norm würde der Einfluss des Strafterms bei steigender Abweichung nicht ansteigen. Dies ist aber zwingend notwendig, da die Verbesserung entlang steiler Stützgeraden sonst größer ist als der negative Einfluss der Abweichung vom Zentrum. Das Problem wäre somit wieder unbeschränkt. Zwar wäre das Problem unter Verwendung der 1-Norm einfacher zu lösen, doch deren Einsatz ist nur im Sinne der Trust-Region Methode als strikte Einschränkung der Abweichung möglich. Mithilfe einer mindestens quadratischen Norm werden geringe Abweichungen kaum bestraft, große Abweichungen gemäß ihrer Entfernung vom Stabilitätszentrum jedoch umso mehr. Das Problem ist hierdurch beschränkt und somit lösbar.

Im Algorithmus sieht man, dass eine neue Iterierte λ_{k+1} in der Norm nicht zu weit von dem Stabilitätszentrum $\mathfrak{Z}_k$ entfernt liegen darf, da sonst der Störterm zu groß wird. Der Einfluss des Störterms wird zudem durch die Wahl von $u > 0$ reguliert. Man kann hierdurch die Genauigkeit des Algorithmus oder die Konvergenzgeschwindigkeit anpassen. Eine Wahl von $u = 0$ würde zum Basic Cutting Plane Algorithmus zurückführen. In der Praxis wird meist $u = 1$ gewählt, dies kann jedoch zu langsamer Konvergenz führen. In [7] löst Kiwiel dieses Problem mit quadratischer Interpolation.

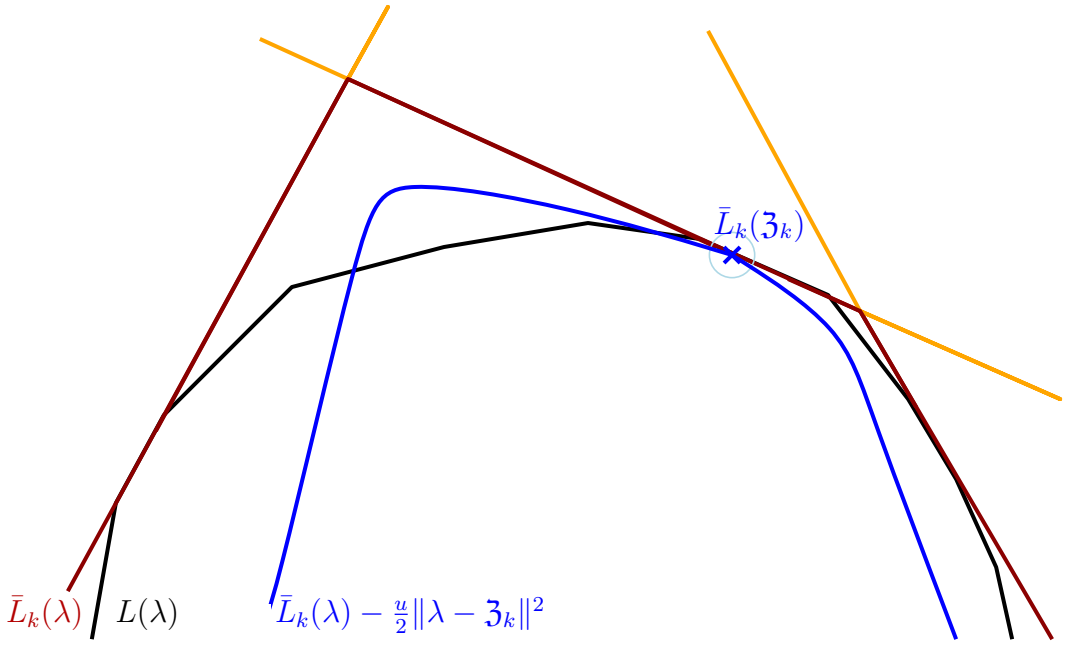


Abbildung 2: Approximation $\bar{L}_k(\lambda)$ und die Veränderung durch den Strafterm

Das Stabilitätszentrum $\mathfrak{Z}_k$ sollte das λ_i sein, für das der Funktionswert von $L(\lambda)$ bisher am größten ist. Je nachdem, ob der Funktionswert von $L(\lambda_{k+1})$ eine angemessene Verbesserung bringt, wird das Stabilitätszentrum auf λ_{k+1} verschoben oder ein Nullschritt gemacht, d.h. das Stabilitätszentrum bleibt gleich. Genauer bedeutet das, dass falls der vorhergesagte Anstieg $\bar{L}_k(\lambda_{k+1}) - L(\mathfrak{Z}_k)$ zu einem angemessenen realen Anstieg $L(\lambda_{k+1}) - L(\mathfrak{Z}_k)$ führt, wird das Stabilitätszentrum neu gesetzt, d.h. $\mathfrak{Z}_{k+1} = \lambda_{k+1}$.

Ob ein Anstieg angemessen ist, überprüft man im Algorithmus durch die folgende Ungleichung

$$\overbrace{L(\lambda_{k+1}) - L(\mathfrak{Z}_k)}^{\text{realer Anstieg}} \geq m \overbrace{(\bar{L}_k(\lambda_{k+1}) - L(\mathfrak{Z}_k))}^{\text{vorhergesagter Anstieg}}$$

Wenn diese für einen gewählten Parameter m, $0 < m < 1$, erfüllt ist, wird das Stabilitätszentrum $\mathfrak{Z}_k$ verschoben, ansonsten wird ein Nullschritt gemacht und das vorherige Zentrum bleibt erhalten.
Falls ein solcher Nullschritt gemacht werden muss, so bringt dies dennoch eine Verbesserung im Vergleich zu $\bar{L}_k(\lambda)$ mit sich. Das Bündel B_k wird durch $(\lambda_{k+1}; L(\lambda_{k+1}); g_{k+1})$ erweitert. $\bar{L}_{k+1}(\lambda)$ ist somit eine verbesserte Approximation von $L(\lambda)$, da eine neue Stützhyperebene hinzugekommen ist. Zudem wird, trotz des Nullschrittes in der Zentrumsverbesserung, mit λ_{k+1} als neuen Multiplikator weitergerechnet.

Der Algorithmus kann gestoppt werden, wenn die Approximation beim neuen Multiplikator schlechter ist als der Funktionswert der Lagrangefunktion im Zentrum, d.h. $\bar{L}(\lambda_{k+1}) - L(\mathfrak{Z}_k) \leq 0$. Als Lösung für den optimalen Multiplikator erhalten wir $\lambda^* = \mathfrak{Z}_k$.

Der folgende Algorithmus fasst die beschriebene Vorgehensweise zusammen:

Algorithm 3: Cutting Plane und Stabilisierung durch Bestrafung

Data: $Z_1;\ u > 0;\ m \in]0,1[$
Result: λ^*
$k = 1$
$\lambda_1 = \mathfrak{Z}_1$
Berechne $L(\lambda_1),\ g_1$
while *Abbruchkriterium nicht erfüllt* **do**
  Update des Multiplikators
  $\lambda_{k+1} = \text{argmax}\{\bar{L}_k(\lambda) - \frac{u}{2}\|\lambda - \mathfrak{Z}_k\|^2\}$
  Berechne $L(\lambda_{k+1}),\ g_{k+1}$

  Prüfe Update des Stabilitätszentrums
  $\tau_k = \bar{L}_k(\lambda_{k+1}) - L(\mathfrak{Z}_k)$
  if $L(\lambda_{k+1}) \geq L(\mathfrak{Z}_k) + m\tau_k$ **then**
    $\mathfrak{Z}_{k+1} = \lambda_{k+1}$
  else
    $\mathfrak{Z}_{k+1} = \mathfrak{Z}_k$
  end

  $k = k + 1$
end

Der Algorithmus soll dabei abbrechen, wenn die Verbesserung zwischen dem Maximum der unteren Hülle der Stützgeraden und dem derzeitigen Zentrum nur noch minimal ist. Die Verbesserung wird dabei in Relation zum Funktionswert im Zentrum gestellt, d.h.

$$\bar{L}_k(\lambda_k) - L(\mathfrak{Z}_k) < \gamma(1 + |L(\mathfrak{Z}_k)|)$$

wobei $\gamma > 0$ ist. Ist dies erfüllt so stoppt der Algorithmus. Die Addition von 1 zum Funktionswert im Zentrum ist hierbei nötig, damit der Algorithmus nicht fälschlicherweise frühzeitig abbricht, wenn das Stabilitätszentrum noch nicht gesetzt wurde.

Der Cutting Plane Algorithmus mit Stabilisierung durch Bestrafung ist somit ein Algorithmus, der für jede Wahl der Startpunkte funktioniert und dessen Maximierung über die untere Hülle der Stützgeraden immer beschränkt ist.

Bemerkung 3.3. *Eine weitere Methode der Stabilisierung ist die Level-Stabilisierung. Diese Methode beinhaltet, dass nun $\bar{L}_k$ in die Nebenbedingung genommen wird und durch ein Level $l(\lambda)$ begrenzt wird. Genaueres hierzu kann man in [8] nachlesen.*

Auch bei diesem Algorithmus können Anpassungen vorgenommen werden, die die Berechnungszeit verkürzen oder die Schritte einfacher machen. Im nächsten Kapitel werden ein paar Varianten vorgestellt.

3.2.2 Anpassungen der Bündelmethode

Eine Variante der Anpassung der Bündelmethode ist das Arbeiten mit ϵ-Approximationen. Diese können zum Beispiel eingesetzt werden, falls das Problem trotz der Lagrange Relaxation immer noch nicht lösbar ist, also $L(\lambda)$ und g nicht oder nur schwer berechnet

werden können. Auch wenn der normale Algorithmus zu einer falschen Lösung konvergiert, kann es hilfreich sein, zur ϵ-Methode zu wechseln. Ein Beispiel hierfür kann in [9] gefunden werden.

Die ϵ-approximativen Lösungen werden von ϵ-Subgradienten erzeugt, wobei $\epsilon > 0$ zu Beginn des Algorithmus frei gewählt werden kann, jedoch danach fix ist.

Definition 3.3. *ϵ-Subgradient*
Sei $L(\lambda) : \mathbb{R}^m \to \mathbb{R}$ eine konkave Funktion. Dann ist $g \in \mathbb{R}^m$ ein ϵ-Subgradient der Funktion $L(\lambda)$ bei λ_k, falls für alle $\lambda \in \mathbb{R}^m$ gilt:

$$L(\lambda) \leq L(\lambda_k) + g^\top(\lambda - \lambda_k) + \epsilon$$

Der ϵ-Subgradient an der Stelle λ_k, im Weiteren als $g_{\epsilon,k} := g_\epsilon(\lambda_k)$ bezeichnet, wird demnach aus der folgenden Menge gewählt:

$$\partial_\epsilon L(\lambda_k) := \{g \in \mathbb{R}^n | L(\lambda) \leq L(\lambda_k) + g^\top(\lambda - \lambda_k) + \epsilon \ \forall \lambda \in \mathbb{R}^n\}$$

Es gilt, dass $\partial L(\lambda) \subseteq \partial_\epsilon L(\lambda)$, d.h. die Auswahl der Aufstiegsrichtung bekommt einen größeren Spielraum. Richtungen, die zuvor nicht in Betracht kamen, werden nun mit einbezogen, was zu einem schnelleren Aufstieg führen kann. Subgradienten von bisher nicht berechneten $\tilde{\lambda}$ können mit einbezogen werden, die zu einer besseren Suchrichtung führen können. Dies zeigt das folgende Lemma:

Lemma 3.5. *Sei $\lambda_k \in \mathbb{R}^m$ der Multiplikator der k-ten Iteration und $\tilde{\lambda} \in \{\lambda \in \mathbb{R}^m | \ |\lambda_k - \lambda| < \mu\}$ mit $\mu \in \mathbb{R}_0^+$ ein λ in der Nähe von λ_k. Des Weiteren sei $\tilde{g} \in \partial L(\tilde{\lambda})$ ein Subgradient der Lagrangefunktion bei $\tilde{\lambda}$. Sei $\rho(\lambda_k, \tilde{\lambda}) := L(\tilde{\lambda}) - L(\lambda_k) - \tilde{g}^\top(\tilde{\lambda} - \lambda_k) \geq 0$ der Linearisierungsfehler bei $L(\lambda_k)$. Dann gilt für alle $\epsilon \geq \rho(\lambda_k, \tilde{\lambda}) \geq 0$*

$$\tilde{g} \in \partial_\epsilon L(\lambda_k)$$

Beweis. Da $\tilde{g} \in \partial L(\tilde{\lambda})$ gilt:

$$\begin{aligned} L(\lambda) \leq L(\tilde{\lambda}) + \tilde{g}^\top(\lambda - \tilde{\lambda}) = L(\lambda_k) + \tilde{g}^\top(\lambda - \lambda_k) + \rho(\lambda_k, \tilde{\lambda}) \\ L(\lambda_k) + \tilde{g}^\top(\lambda - \lambda_k) + \epsilon \ \forall \lambda \in \mathbb{R}^n, \end{aligned}$$

Folglich ist $\tilde{g} \in \partial_{\rho(\lambda_i, \tilde{\lambda})} L(\lambda_i)$. □

Die Stützfunktionen beinhalten nun den ϵ-Subgradienten.

Definition 3.4. *Die linearen Stützfunktionen mit ϵ-Subgradienten sind definiert als*

$$\tilde{L}(\lambda; \lambda_i) := L(\lambda_i) + \tilde{g}_i^\top(\lambda - \lambda_i) \qquad \forall \ i \in \{1, \ldots, k\}$$

Die untere Hülle der Stützgeraden mit ϵ-Subgradienten ist definiert als

$$\tilde{L}_k(\lambda) := \min_{i \in \{1,\ldots,k\}} \tilde{L}(\lambda; \lambda_i)$$

Bezüglich dieser neuen Stützfunktionen mit ϵ-Subgradienten kann man analog zur Problemstellung 3.3 des Quadratischen Problems ein neues Quadratisches Problem $[QP_{\epsilon,k}]$ betrachten.

Problemstellung 3.4: $QP_{\epsilon,k}$

$$\lambda_{k+1} \in \operatorname{argmax}\{\tilde{L}_k(\lambda) - \tfrac{u}{2}\|\lambda - \mathfrak{Z}_k\|^2\}$$

$[QP_{\epsilon,k}]$ ersetzt dann in Algorithmus 3 das normale Quadratische Problem $[QP_k]$ und die Aufstiegsrichtungen sind nun die ϵ-Subgradienten. Ansonsten bleibt der Ablauf des Algorithmus gleich.
Diese ϵ-approximative Lösungsmethode wurde von Hintermüller (vgl. [10]) noch dahingehend verbessert, dass die Kenntnis über den Wert von ϵ gar nicht mehr vorausgesetzt wird. Seine Lösungen sind dabei genauso gut, wie die von Kiwiel. Sie konvergieren zur optimalen Lösung, falls die linearen Approximationen dies tun.

Eine weitere Lösungsmöglichkeit des Problems ist durch die Dualisierung des quadratischen Problems $[QP_k]$ beziehungsweise $[QP_{\epsilon,k}]$. Diese ist bei sehr komplexen Problemen mit vielen Lagrange-Multiplikatoren hilfreich. Dabei wird das quadratische Problem nicht über den primalen, sondern über den dualen Weg gelöst.
Bei der Dualisierung von $[QP_k]$ werden die neuen Subgradienten aus der konvexen Hülle der bisherigen Subgradienten gewählt.

$$g_k \in \mathcal{P}_k := \{\ g \mid g = \sum_{i=1}^{k-1} \alpha_i g_i,\ g_i \in B_{k-1},\ 0 \le \alpha_i\ , \sum_{i=1}^{k-1} \alpha_i = 1\ ,$$
$$\sum_{i=1}^{k-1} \alpha_i (L(\lambda_i) + g_i^\top(\lambda - \lambda_i) - L(\lambda)) \le 0\}$$

Diese Menge ist eine Approximation von $\partial L(\lambda)$.
Analog kann bei der Dualisierung von $[QP_{\epsilon,k}]$ diese Menge gebildet werden:

$$\tilde{g}_k \in \mathcal{P}_{\epsilon,k} := \{\ g \mid g = \sum_{i=1}^{k-1} \alpha_i g_i,\ g_i \in B_{k-1},\ 0 \le \alpha_i\ , \sum_{i=1}^{k-1} \alpha_i = 1\ , \tag{2}$$
$$\sum_{i=1}^{k-1} \alpha_i (L(\lambda_i) + g_i^\top(\lambda - \lambda_i) - L(\lambda)) \le \epsilon\} \tag{3}$$

Diese Menge ist wiederum eine Approximation von $\partial L_\epsilon(\lambda)$.
Für eine genauere Betrachtung dieser Vorgehensweise sei hier auf [11] und [12] verwiesen. Die beiden vorgestellten Verbesserungen der Bündelmethode, zum einen die Verwendung von ϵ-Approximationen, zum anderen die Dualisierung des Quadratischen Problems, fließen in die folgenden Algorithmen ein. Diese sind eine Verbesserung der einfachen Bündelmethode und werden in den nächsten beiden Kapiteln genauer vorgestellt.

3.2.3 Verbesserte Bündelmethode

Neben den obigen Anpassungen der Bündelmethode wurden auch weitere Algorithmen zur Verbesserung entwickelt. Einer davon ist die Verbesserte Bündelmethode (auch Improved Bundle Method). Mit dieser Methode soll die Rechenzeit für das Update der Multiplikatoren verkürzt werden.

Die Berechnung von x_k nach jedem Schritt verbraucht viel Rechenzeit. Auch wenn nur ein Nullschritt im obigem Algorithmus gemacht wird, muss trotzdem ein neues x_k berechnet werden, d.h. viel Rechenaufwand für wenig Verbesserung.
Das Ziel der Verbesserten Bündelmethode ist es demnach die Anzahl dieser Nullschritte zu verringern, indem die Approximation aus mehr Stützgeraden berechnet wird.

Wir betrachten hier die Menge der ϵ-optimalen Lösungen.

Definition 3.5. *ϵ-optimalen Lösungen*
Die Menge der ϵ-optimalen Lösungen ist definiert als

$$X_Z(\lambda, \epsilon) := \{\tilde{x} | L(\lambda, \tilde{x}) - L(\lambda) \leq \epsilon, \tilde{x} \in Z^n\}$$

$\tilde{x}_k \in X_Z(\lambda, \epsilon)$ ist definiert als eine ϵ-optimale Lösung in der k-ten Iteration.

Diesbezüglich kann auch ein neuer, veränderter Subgradient gewählt werden.

Definition 3.6. *ϵ-Surrogate-Subdifferential*
Das ϵ-Surrogate-Subdifferential ist definiert als

$$\partial \tilde{L}_\epsilon(\lambda) = \{g | g = \sum_i \alpha_i g(\tilde{x}_i),\ \tilde{x}_i \in X_Z(\lambda, \epsilon),\ \text{und } 0 < \alpha_i,\ \sum_i \alpha_i = 1\}$$

Dies ist eine Teilmenge von $\partial L_\epsilon(\lambda)$.

Kombiniert man nun die Mengen $\mathcal{P}_{\epsilon,k}$ der dualen Lösungsmethode (vgl.(2)) und $\partial \tilde{L}_\epsilon(\lambda)$ so erhält man eine bessere Approximation von $\partial L_\epsilon(\lambda)$ und kann Nullschritte vermeiden, da die Approximation durch die Vergrößerung der Menge der Stützgeraden besser ist.
Wenn $\mathcal{P}_{\epsilon,k}$ keine gute Approximation von $\partial L_\epsilon(\lambda)$ ist, so sammelt der Algorithmus viele Subgradienten von nah beieinander liegenden Stützgeraden mit vielen Nullschritten.
Verwendet man jedoch die Verbesserte Bündelmethode, so sind die meisten Subgradienten in der näheren Umgebung schon in $\partial \tilde{L}_\epsilon(\lambda)$ enthalten. Die Nullschritte, die in der normalen Bündelmethode gemacht worden wären, fallen dadurch weg. Das Verfahren wird somit schneller.

Dieses Verfahren benötigt viele Berechnungen. Um dies zu reduzieren kann man die Vereinfachte Bündelmethode verwenden.

3.2.4 Vereinfachte Bündelmethode

Eine weitere Methode der Verbesserung ist die Vereinfachte Bündelmethode (Simplified Bundle Method). Hierbei wird der Prozess der Wahl eines neuen Subgradienten vereinfacht.

Für die Erzeugung der Subgradienten bildet dieses Verfahren eine Konvexkombination aus den Gradienten der vorangegangenen Iterationen. Die Vorfaktoren in der Konvexkombination werden dabei stets auf die folgende Weise gewählt.

$$\bar{\alpha_i} = \begin{cases} \frac{L(\lambda)+\epsilon-L(\tilde{x}_i,\lambda)}{\epsilon} = 1 + \frac{L(\lambda)-L(\tilde{x}_i,\lambda)}{\epsilon} & , \ \tilde{x}_i \in X_Z(\lambda,\epsilon) \\ 0 & , \text{ sonst} \end{cases}$$

Es gilt $0 \leq L(\lambda, \tilde{x}_i) - L(\lambda) \leq \epsilon$ für alle $i \in \{1, \ldots, k\}$ und somit $\bar{\alpha_k} \geq 0$ für alle $i \in \{1, \ldots, k\}$.
Der neue Gradient wird dann als Konvexkombination der bisherigen Gradienten erzeugt, nachdem die Vorfaktoren normalisiert worden sind, d.h.

$$\tilde{g} = \sum_{i=1}^{k} \alpha_i g(x_i), \text{ mit } \alpha_i = \frac{\bar{\alpha_i}}{\sum_{i=1}^{k} \bar{\alpha_i}}, \quad \forall i \in \{1, \ldots, k\}$$

Zur anschließenden Berechnung von λ_{k+1} wird wie im Subgradientenverfahren ein Aktualisierungsschritt durchgeführt, d.h. $\lambda_{k+1} = \lambda_k + \sigma_k \tilde{g}$.
Für die Schrittweite kann man der Einfachheit halber nicht Line Search sondern eine einfache Begrenzungsregel, wie zum Beispiel $0 < \sigma_k < \frac{L^* - L(\lambda_k)}{\|\tilde{g}\|^2}$, verwenden.

3.3 Diskussion der Methoden

Wie anfangs schon erwähnt, werden in allen Methoden die Subgradienten oder Näherungen dieser als Aufstiegsrichtungen verwendet. Die Subgradientenmethode ist dabei die meist genutzte Methode, die für wenig komplexe Modelle gute Ergebnisse liefert. In dieser Methode wird versucht das Maximum der Lagrangefunktion durch einen schrittweisen Aufstieg zu finden. Die Methode konvergiert somit in einer Zick-Zack-Linie, was mit vielen Iterationen verbunden ist. Ist ein Problem zu komplex, so werden die Berechnungen schnell zu aufwendig, da in jedem Schritt jedes Subproblem gelöst werden muss.
Hier kommen die verbesserten Versionen zum Einsatz. Die Interleaved-Subgradientenmethode braucht nur eine Lösung eines Subproblems um in einer Iteration eine geeignete Suchrichtung zu finden. Diese ist nicht so gut wie die der Subgradientenmethode, doch mit wesentlich weniger Rechenaufwand verbunden. Für große Probleme, die in gleiche Subprobleme zerfallen, ist diese Methode sehr hilfreich, da alle Subprobleme von vorn herein auf die gleiche Lösung kommen.
Falls das Lösen dieses einen Subproblems noch zu aufwendig ist, so kann man die Surrogate-Subgradient Methode anwenden, die das Subproblem nur noch approximativ löst. Ein weiterer Vorteil dieser Methode ist es, dass hier im Gegensatz zur Interleaved-Subgradientenmethode die Konvergenz bewiesen ist.

Für die Verbesserungen des Subgradientenverfahrens müssen Anpassungen in der Programmierung des Subproblems vorgenommen werden, daher werden diese Methoden nicht implementiert.

Einen anderen Ansatz bietet die Bündelmethode, die durch das Maximieren einer Annäherung der Lagrangefunktion versucht nahe an das Maximum der Lagrangefunktion zu kommen. Hierbei wird auf die Subgradienten und Multiplikatoren aus den vorherigen Iterationen aufgebaut. Die Bündelmethode ist vom Ablauf und der Speicherung aufwendiger als

die Subgradientenmethode. Bei der Subgradientenmethode kommt es allerdings darauf an, wie die Schrittweitenregelung ist. Hier müssen teilweise seperate quadratische Optimierungsprobleme gelöst werden. Dadurch kann die Schrittweitenanpassung sehr aufwendig und somit die Bündelmethode auch schneller als die Subgradientenmethode sein.
Um die Wahl der Optimierungsrichtung zu verbessern, wird in der Verbesserten Bündelmethode der Subgradient aus einer größeren Menge gewählt. Zum einen werden auch Subgradienten aus der Umgebung als Aufstiegsrichtung betrachtet, zum anderen werden zu kleine Schritte ausgeschlossen, die kaum Verbesserung bringen würden. Durch die einfachere Wahl der Aufstiegsrichtung und die Vermeidung von Nullschritten wird das Verfahren schneller, trotz des höheren Rechen- und Speicheraufwands.
Die Vereinfachte Bündelmethode versucht alle rechenaufwendigen Prozesse zu vermeiden. Die Aufstiegsrichtungen werden nach einem festgelegten Schema gewählt und das Update der Multiplikatoren durch einen einfachen Aktualisierungsschritt durchgeführt. Dies spart viel Rechenzeit und ist für große Probleme praktisch.

Da auch hier Veränderungen an den Subproblemen vorzunehmen wären, wird auch diese Methode nicht in unserem Anwendungsbeispiel implementiert.

Im Folgenden ist eine kurze Übersicht über die behandelten Methoden mit den resultierenden Suchrichtungen und Multiplikatoren in Iteration k. Für mehr Details sei auf die entsprechenden Kapitel verwiesen.

Übersicht der Methoden

Verbesserung der Optimierung

Subgradientenmethode
$g_k \in \partial L(\lambda_k),\ \lambda_{k+1} = \lambda_k + \sigma_k g_k$

Verringerung des Rechenaufwands

Bündelmethode
$g_k \in \partial L(\lambda_k),\ \lambda_{k+1} \in \operatorname{argmax}\{\bar{L}_k(\lambda) - \frac{u}{2}\|\lambda - \mathfrak{Z}_k\|^2\}$

Interleaved-Subgradientenmethode
x_k Lösung eines Subproblems,
sonst analog zu Subgradientenmethode

Verbesserte Bündelmethode
$g_k \in \partial \tilde{L}_\epsilon(\lambda_k) \cup \mathcal{P}_{\epsilon,k},\ \lambda_{k+1} \in \operatorname{argmax}\{\tilde{L}_k(\lambda) - \frac{u}{2}\|\lambda - \mathfrak{Z}_k\|^2\}$

Surrogate-Subgradientenmethode
$x_k \in \{x | L(\lambda_k, x) < L(\lambda_k, x_{k-1}),\ x \in X\}$,
sonst analog zu Subgradientenmethode

Verringerung des Rechenaufwands

Vereinfachte Bündelmethode
$g_k = \sum_{i=1}^{k-1} \alpha_i g(x_i),\ \lambda_{k+1} = \lambda_k + \sigma_k g_k$

4 Anwendung auf das Vehicle Routing Problem

In diesem Kapitel sollen die verschiedenen Vorgehensweisen zur Lösung eines Problems mit Hilfe der Lagrange-Multiplikatoren beim *Vehicle Routing Problem with Time-Windows* $\mathbb{VRPTW}$ angewandt und verglichen werden. Dieses Problem ist eine Verallgemeinerung des Travelling Salesman Problem mit zusätzlichen Einschränkungen für den Besuch der Knoten.
Im einfachen Travelling Salesman Problem müssen alle Städte auf möglichst kostengünstigsten Weg von einer Person besucht werden. Beim Vehicle Routing Problem sind es mehrere Personen, die die Städte besuchen müssen. Dabei reisen sie unabhängig voneinander. Am Ende soll auch hier jede Stadt mindestens einmal besucht worden sein. Man nennt dies dann auch ein Multiple Travelling Salesman Problem.

In der Ambulanten Krankenpflege müssen die Patienten von den Pflegekräften besucht werden, um diese zu pflegen. Da diese unabhängig voneinander, möglichst kostengünstig ihre Touren abfahren sollen, wobei sie gewisse Zeitfenster einhalten müssen, handelt es sich hierbei um eine Anwendung des $\mathbb{VRPTW}$.
Bevor jedoch konkret auf den Anwendungsfall eingegangen wird, soll zunächst das $\mathbb{VRPTW}$ kurz dargestellt werden.

4.1 Vehicle Routing Problem with Time-Windows

Wie in der Einleitung erwähnt, ist das $\mathbb{VRPTW}$ ein Multiple Travelling Salesman Problem. Jede Stadt i aus der Menge der Städte $\mathcal{J}$ muss von mindestens einem Verkäufer k aus der Menge der Verkäufer $\mathcal{V}$ besucht werden.
Für die Fahrt von einem Job zum nächsten fallen Kosten an. Diese werden in $c_{ij} : \mathcal{J} \times \mathcal{J} \to \mathbb{R}_{\geq 0}$ festgehalten. Falls ein Verkäufer zuerst Stadt i und direkt danach Stadt j besucht, müssen diese Kosten mitberechnet werden. Dieses Ereignis wird in der Binärvariable $x_{ijk} : \mathcal{J}_0 \times \mathcal{J}_0 \times \mathcal{V} \to \{0, 1\}$ festgehalten, wobei x_{ijk} genau dann 1 ist, wenn der Verkäufer von Stadt i und direkt in die Stadt j fährt.
Ziel der Optimierung der Routen ist es die Gesamtkosten für alle Fahrten zwischen den Städten möglichst gering zu halten und gewisse Rahmenbedingungen zu erfüllen.

Doch zunächst einmal das komplette Optimierungsproblem, modelliert als ganzzahliges Problem:

$\mathbb{VRPTW}$

$$\min \sum_{i \in \mathcal{J}_0} \sum_{j \in \mathcal{J}_0} c_{ij} \sum_{k \in \mathcal{V}} x_{ijk} \tag{4}$$

u.d.N.

$$\sum_{k\in\mathcal{V}}\sum_{j\in\mathcal{J}_0} x_{ijk} = 1 \qquad \forall i \in \mathcal{J} \qquad (5)$$

$$\sum_{j\in\mathcal{J}_0} x_{0jk} = 1 \qquad \forall k \in \mathcal{V} \qquad (6)$$

$$\sum_{i\in\mathcal{J}_0} x_{i0k} = 1 \qquad \forall h \in \mathcal{V} \qquad (7)$$

$$\sum_{i\in\mathcal{J}_0} x_{ihk} = \sum_{j\in\mathcal{J}_0} x_{hjk} \qquad \forall k \in \mathcal{J},\ \forall k \in \mathcal{V} \qquad (8)$$

$$s_{ik} + (d_i + t_{ij})x_{ijk} \leq s_{jk} + M(1 - x_{ijk}) \qquad \forall i, j \in \mathcal{J},\ \forall k \in \mathcal{V} \qquad (9)$$

$$a_i \sum_{j\in\mathcal{J}_0} x_{ijk} \leq s_{ik} \leq b_i \sum_{j\in\mathcal{J}_0} x_{ijk} \qquad \forall i \in \mathcal{J},\ \forall k \in \mathcal{V} \qquad (10)$$

$$x_{ijk} \in \{0,\ 1\} \qquad \forall i, j \in \mathcal{J}_0,\ \forall k \in \mathcal{V} \qquad (11)$$

In der Zielfunktion (4) steht die Minimierung der Gesamtkosten. Falls der Verkäufer k erst Stadt i und gleich im Anschluss Stadt j besucht, ist die $x_{ijk} = 1$. Falls dieser Fall nicht eintritt, so ist die Binärvariable 0. Die Bedingung, dass x_{ijk} eine Binärvariable ist, wird in (11) festgehalten.

Hierbei ist zu beachten, dass eine Fahrt von Stadt i zu Stadt i nur eine Kostensteigerung bedeuten würde und somit automatisch durch den Minimierungsanspruch ausgeschlossen wird.

Damit jede Stadt auch besucht wird, wird dies in (5) gefordert. Dabei ist es egal, von welchem Verkäufer dies erledigt wird und welche Stadt danach besucht wird.

Da die Verkäufer ihre Arbeit am selben Ort anfangen und beenden sollen, wird zu der Menge der Städte noch ein zusätzlicher Knoten $0 \in \mathcal{J}_0$ gebildet. Die Nebenbedingung (6) sorgt dafür, dass jeder Verkäufer von dort startet und Nebenbedingung (7), dass er auch wieder zum Start zurückkehrt.

In (8) wird dafür gesorgt, dass es nach jedem Besuch einer Stadt auch wieder weiter geht, d.h. dass jeder Verkäufer, der eine Stadt besucht hat, diese auch wieder verlässt. Diese Bedingung wird auch als Flusserhaltungsbedingung bezeichnet.

Das Besondere am $\mathbb{VRPTW}$ im Vergleich zu einem normalen Vehicle Routing Problem sind die Zeitfenster. In (9) und (10) werden spezielle Bedingungen an diese festgelegt.
Mit s_{ik} wird die Ankunftszeit des Verkäufers k in Stadt i codiert. d_i steht für die Zeit, die in der Stadt i für den Vertrieb benötigt wird und t_{ij} für die Fahrzeit zwischen Stadt i und Stadt j. Die Konstante M sei größer als s_{ik}. Die Nebenbedingung (9) sorgt also zum einen dafür, dass, falls Stadt j nach Stadt i besucht werden soll, ein ausreichender Puffer für die Fahrzeit und den Vertrieb der Ware in der einen Stadt und der Ankunftszeit in der anderen Stadt vorliegt.
In (10) wird dafür gesorgt, dass die Stadt i in einem festgelegten Zeitfenster $[a_i, b_i]$ besucht

wird.

Für den Anwendungsfall der Zuordnung der Pflegekräfte auf die Jobs ist dieses Modell noch nicht ausreichend. Um die Realität besser darzustellen sind noch mehr Nebenbedingungen notwendig. Diese werden im nächsten Kapitel veranschaulicht.

4.2 Vehicle Routing Problem für Pflegedienste

Nun betrachten wir das Problem für den Anwendungsfall. Hier entsprechen die Menge der Verkaufsleute der Menge der Schichten der Pflegekräfte und die Menge der Städte entspricht der Menge der Jobs, die erledigt werden müssen.
Der Start- und Zielpunkt kann in unserem Modell mehreren Orten entsprechen. Er kann zum Beispiel für das Zuhause der Pflegekraft oder für eine Station stehen, in der die Fahrzeuge für die Touren untergebracht sind. Wichtig ist nur, dass die Pflegekraft am selben Ort ihre Arbeit beginnt und beendet.

Neben den Bedingungen des $\mathbb{VRPTW}$ gibt es in unserem Modell eine Vielzahl von Nebenbedingungen, die zusätzlich gestellt werden müssen. Zum Beispiel sollte die Pflegekraft auch für den jeweiligen Patienten geeignet sein, zum einen hinsichtlich der Anforderungen, zum anderen hinsichtlich der Sprache.
Es sollte auch darauf geachtet werden, dass eine Pflegekraft nicht zu viele Überstunden macht und Pausenzeiten eingehalten werden. In dieser Erweiterung des Problems der einfachen Zuweisung, soll außerdem ein gewisses individuell vereinbartes Zeitfenster für den Besuch bei den Patienten eingehalten werden, da dies zum Beispiel für die Medikamentengabe wichtig ist. Zudem sind die Tarifverträge der Pflegekräfte unterschiedlich. Manche werden ab der Ankunft beim ersten Kunden, manche ab dem Verlassen der Wohnung manche ab der Ankunft im Büro bezahlt. Daher ergeben sich unterschiedliche Arbeitszeiten.

Die oben genannten Bedingungen sollten eingehalten werden, da sonst die Aufgabenstellung nicht erfüllt ist. Es können jedoch noch weichere Nebenbedingungen eingeführt werden, die vor allem auf die Zufriedenheit der Mitarbeiter und Kunden ausgelegt sind. Hierunter fallen Bedingungen wie persönliche Präferenzen der Kundenzuordnung und die Einhaltung spezieller Anfangs- und Endzeiten ihrer Arbeitsschicht sowie Wünsche bezüglich der Pausenzeit. Auch die Kunden können zusätzlich zu den festen Zeitintervallen Wunschzeiten und Pflegekräfte für den Besuch äußern.

Im Modell des $\mathbb{VRPTW}$ war das Ziel, die Kosten für die Fahrten zwischen den Jobs zu minimieren. Dies wird im Anwendungsfall durch mehrere Ziele ersetzt. Je nach Präferenz wird jedes dieser Ziele mit einem Faktor $\alpha_i \in [0, 1]$ versehen, damit die Zufriedenheit der Pflegekräfte und Patienten individuell gesteigert werden kann. Die Faktoren entsprechen der Präferenz in Prozent, d.h. $\sum_i \alpha_i = 1$. In unserem Anwendungsfall wurden 7 Ziele definiert. Diese Ziele mit den Faktoren sind die folgenden:

Faktor	Ziel
α_1	Minimierung der Fahr- und Wartezeiten, damit die Pflegekräfte nicht unnötig Zeit verschwenden
α_2	Überstunden der Pflegekräfte, die zu Verstimmungen führen, wenn sie zu oft geleistet werden müssen
α_3	Ignorierte Präferenzen hinsichtlich der Zuordnung
α_4	Missachtung der Wunschzeitfenster der Kunden
α_5	Missachtung der Wunschzeitfenster der Pflegekräfte
α_6	Überqualifikationen der Pflegekräfte, die durch die fachliche Unterforderung ein Motivationsdefizit erleiden
α_7	Unbezahlte Anfahrtszeiten der Pflegekräfte, die erst bei der Ankunft beim Kunden bezahlt werden.

Bemerkung 4.1. *In Wien ergaben sich die Präferenzen von Pflegekräften und Patienten durch Umfragen folgendermaßen:*

$$\alpha_1 = 0.6,\ \alpha_2 = 0.1,\ \alpha_3 = \alpha_4 = \alpha_5 = 0.05,\ \alpha_6 = 0.1,\ \alpha_7 = 0.05$$

Die neue Zielfunktion ergibt sich aus der Summation über die einzelnen Zielfunktionen mit ihrem Präferenzfaktor. Im Folgenden wird die Zielfunktion mit $\mathcal{ZF}$ abgekürzt.

Wie im Vorherigen bereits erwähnt kommen noch ein paar Nebenbedingungen hinzu. Diese bilden die Rahmenbedingungen zu den folgenden Themen ab:

- Zuordnung: nur wenn Sprache und Anforderungsniveau passen und kein Ausschluss beider Seiten besteht, kann die jeweilige Pflegekraft zu dem jeweiligen Kunden fahren.
- Zeitfenster: nur innerhalb des festgelegten Zeitfensters darf der Kunde besucht werden.
- Flussbedingungen: siehe auch $\mathbb{VRPTW}$ (Kapitel 4.1) sowie Einschränkungen bezüglich der Pausen.
- Zeitablauf: Korrektheit der Startzeiten jedes Jobs und der Schicht allgemein sowie Korrektheit der Endzeiten der Schicht.
- Vertraglich: Einhaltung der maximalen Arbeitszeit und der Pausenzeit zwischen zwei Schichten; Einhaltung der maximalen Arbeitszeit bis zu einer Pause und Ausführung der Pause, wenn diese nötig ist.
- Rahmenbedingungen für die Zielfunktion: zum Beispiel wann sind es Überstunden, wann sind die flexiblen Zeitfenster verletzt.
- Nichtnegativitäts- und Binärbedingungen der Variablen.
- weitere Nebenbedingungen, die die LP-Relaxation verbessern

Für eine Ausführung der Nebenbedingungen sowie eine genauere Beschreibung siehe [5]. Hier werden auch einige weitere veränderte Problemformulierungen betrachtet.

Im nächsten Kapitel wird nun der Einsatz der Lagrange Relaxation auf das Anwendungsbeispiel dargestellt.

4.3 Die Lagrange Relaxationen

Im Folgenden werden zwei Varianten der Relaxation vorgestellt. Zunächst wird die Relaxation auf das $\mathbb{VRPTW}$ bezogen und dann die Wirkungsweise auf das Anwendungsproblem betrachtet.
Als erstes wird die einfachere der beiden Relaxationsarten vorgestellt.

4.3.1 Relaxation 1

Im $\mathbb{VRPTW}$ verbindet die Nebenbedingung (5) die einzelnen Verkäufer, da über alle Verkäufer summiert wird. Bringt man diese Nebenbedingung in die Zielfunktion, dann zerfällt das Problem in $|\mathcal{V}|$ Teilprobleme und man kann das Problem für jeden Verkäufer einzeln lösen.

Das relaxierte Problem ist dann:

$$\begin{aligned} \min \ & \sum_{i\in\mathcal{J}_0}\sum_{j\in\mathcal{J}_0} c_{ij}\sum_{k\in\mathcal{V}} x_{ijk} + \sum_{i\in\mathcal{J}} \lambda_i (1 - \sum_{k\in\mathcal{V}}\sum_{j\in\mathcal{J}_0} x_{ijk}) \\ \text{u.d.N.} \ & (6)-(11) \end{aligned}$$

Hierbei ist der Subgradient

$$1 - \sum_{k\in\mathcal{V}}\sum_{j\in\mathcal{J}_0} x_{ijk} \ , \ i \in \mathcal{J}$$

Durch eine Umstellung der Zielfunktion erhält man:

$$\min \sum_{i\in\mathcal{J}_0}\sum_{j\in\mathcal{J}_0} \hat{c}_{ij}\sum_{k\in\mathcal{V}} x_{ijk} + \sum_{i\in\mathcal{J}} \lambda_i$$

hierbei gilt

$$\hat{c}_{ij} = \begin{cases} c_{ij} - \lambda_i & i \in \mathcal{J},\ j \in \mathcal{J}_0 \\ c_{0j} & sonst \end{cases}$$

Die λ_i sind konstant und können daher im Optimierungsschritt vernachlässigt werden. Sie werden erst nachträglich hinzugefügt.

Das neu entstandene Subproblem wird *Elementary Shortest Path Problem with Time Windows* $\mathbb{ESPPTW}$ genannt. Hierbei darf jede Stadt nur maximal einmal besucht werden. Eine

Vereinfachung, bei der die maximale Besuchszahl nicht eingeschränkt ist, heißt *Shortest Path Problem with Time Windows* $\mathbb{SPPTW}$. Dies kann in anderen Optimierungsproblemen zu Problemen führen, da negative Kanten unendlich oft genommen werden können, doch hier ist Anzahl der Benutzung einer Kante durch die Zeitfenster beschränkt.
Dieses Problem lässt sich durch dynamische Programmierung pseudo-polynomiell lösen. Da die Ganzzahligkeit der Lösung nicht vorausgesetzt war, liefert dieses Verfahren meist nur eine bessere untere Schranke.

In dem oben vorgestellten Anwendungsproblem gibt es drei Nebenbedingungen, die über die einzelnen Pflegekräfte summieren. Diese sind:

$$\sum_{k\in\mathcal{V}}\sum_{i\in\mathcal{J}_0} x_{ijk} = 1 \qquad \forall j \in \mathcal{J} \tag{12}$$

$$E_k + p_k \leq S_l \qquad \forall k,l \in \mathcal{V}:\ k<l,\ \vartheta(k)=\vartheta(l) \tag{13}$$

$$\sum_{k\in\mathcal{V}:\ \vartheta(k)=n} (E_k - S_k - p_k y_k) \leq T \qquad \forall n \in \mathcal{N} \tag{14}$$

Wobei (12) fordert, dass jeder Job einmal ausgeführt wird, wobei dies unabhängig davon ist in welcher Schicht oder in welcher Reihenfolge die Jobs ausgeführt werden. In (13) wird gefordert, dass zwischen zwei Schichten eine Pause eingelegt wird. Dabei steht E_k für den Endzeitpunkt von Schicht k und S_l für den Anfangszeitpunkt für Schicht l. p_k ist die Pause nach Schicht k. Dies soll für alle Schichten gelten, die von der gleichen Pflegekraft ausgeführt werden. $\vartheta : \mathcal{V} \to \mathcal{N}$ ist dabei die Zuordnung jeder Schicht zu ihrer Pflegekraft. Die dritte verbindende Nebenbedingung (14) stellt sicher, dass die maximale Arbeitszeit T mit allen Schichten einer Pflegekraft nicht überschritten wird. Die Binärvariable $y_k : \mathcal{V} \to \{0,1\}$ gibt dabei an, ob eine Pause in Schicht k eingelegt wird oder nicht.

Diese Nebenbedingungen werden mit den Multiplikatoren $\lambda_j \in \mathbb{R},\ j \in \mathcal{J},\ \mu_{kl} \in \mathbb{R}_{\geq 0},\ k,l \in \mathcal{V}$ mit $k<l,\ \vartheta(k)=\vartheta(l),\ \nu_n \in \mathbb{R}_{\geq 0},\ n \in \mathcal{N}$ zur Zielfunktion hinzugefügt. Dadurch entsteht folgendes Minimierungsproblem:

[LAG1]

$$\min\ \mathcal{ZF} \tag{15}$$

$$+ \sum_{j\in\mathcal{J}} \lambda_j (1 - \sum_{i\in\mathcal{J}_0}\sum_{k\in\mathcal{V}} x_{ijk}) \tag{16}$$

$$+ \sum_{k,l\in\mathcal{V}:k<l,\vartheta(k)=\vartheta(l)} \mu_{kl}(E_k + p_k - S_l) \tag{17}$$

$$+ \sum_{n\in\mathcal{N}} \nu_n (\sum_{k\in\mathcal{V}:\vartheta(k)=n} (E_k - S_k - p_k y_k) - T) \tag{18}$$

Die Nebenbedingungen sind analog zu denen in Kapitel 4.2 ohne die bereits relaxierten.

Das Problem zerfällt damit in $|\mathcal{V}|$ unabhängige Subprobleme, für jede Schicht ein eigenes Problem.

Wenn es zwischen den Schichten und den verwendeten Fahrzeugen keinen Unterschied gibt, kommen alle Subprobleme zur gleichen Lösung, da für jede Schicht die gleiche Route optimal ist. Dies ist zwar ein Vorteil für den Rechenaufwand in den Subproblemen, doch aus dem Ergebnis lässt sich nicht sofort eine zulässige Lösung für das eigentliche Problem ablesen. Falls Unterschiede bei der Ausstattung der Fahrzeuge bestehen, beziehungsweise eine Einschränkung der Qualifikationen der Fahrer oder Präferenzen der Patienten vorliegen, muss dennoch jedes Subproblem für sich gelöst werden, um diese mit einzubeziehen. Des Weiteren können sich die Subprobleme hinsichtlich der Tarifgruppen unterscheiden, d.h. je nachdem wie sich die Arbeitszeit errechnet.

Die Subprobleme zu Relaxation 1 sind folgendermaßen:

[SUB1]

$$
\begin{aligned}
\min \mathcal{Z}\mathcal{F} \\
& - \sum_{j \in \mathcal{J}} \lambda_j \sum_{i \in \mathcal{J}_0} x_{ijk} \\
& + \sum_{l \in \mathcal{V}: k<l, \vartheta(k)=\vartheta(l)} \mu_{kl} E_k \\
& - \sum_{l \in \mathcal{V}: k<l, \vartheta(k)=\vartheta(l)} \mu_{lk} S_k \\
& + \nu_{\vartheta(k)} (E_k - S_k - p_k y_k)
\end{aligned}
$$

Die Nebenbedingungen sind auch hier analog zu denen in Kapitel 4.2 ohne die bereits relaxierten.

Hierbei wurden die konstanten Therme, die unabhängig von der Wahl der Route beziehungsweise Zuordnung sind, bereits weggelassen. Die Nebenbedingungen sind hier ebenfalls analog zu denen in 4.2 ohne die bereits relaxierten.

Für dieses Subproblem werden nun die konstanten Therme addiert und unter Verwendung der in Kapitel 3 vorgestellten Methoden die initialen Multiplikatoren aktualisiert. In den nächsten Iterationen wird mit den neuen Multiplikatoren immer zuerst das Subproblem gelöst und anschließend die Multiplikatoren neu berechnet bis der Algorithmus abbricht. Dieser Vorgang kann auch mit einer anderen Art der Relaxation durchgeführt werden, die im folgenden Kapitel beschrieben wird.

4.3.2 Relaxation 2

Für diese Relaxation wird zunächst eine kleine Veränderung des $\mathbb{VRPTW}$ vorgenommen. Hierfür wird eine neue Binärvariable z_{ik} eingeführt.

$$z_{ik} = \sum_{j \in \mathcal{J}_0} x_{ijk} \ , \ \forall i \in \mathcal{J}, \ \forall k \in \mathbb{V}$$

Diese ist nur dann 1, wenn die Stadt i von Verkäufer k besucht wird.

Die Nebenbedingung (5) im $\mathbb{VRPTW}$ wird somit durch drei neue Nebenbedingungen ersetzt:

$$\sum_{k\in\mathcal{V}} z_{ik} = 1 \qquad \forall i \in \mathcal{J} \tag{19}$$

$$\sum_{j\in\mathcal{J}_0} x_{ijk} = z_{ik} \qquad \forall i \in \mathcal{J},\ \forall k \in \mathcal{V} \tag{20}$$

$$z_{ik} \in \{0,\ 1\} \qquad \forall i \in \mathcal{J},\ \forall k \in \mathcal{V} \tag{21}$$

In dem neu entstandenen Problem verknüpft nun die Nebenbedingung (20) die Subprobleme der einzelnen Verkäufer miteinander. Relaxiert man diese mit λ so erhält man das neue Problem:

$$\begin{aligned}
\min\ & \sum_{i\in\mathcal{J}_0}\sum_{j\in\mathcal{J}_0} c_{ij}\sum_{k\in\mathcal{V}} x_{ijk} + \sum_{i\in\mathcal{J}}\sum_{k\in\mathcal{V}} \lambda_{ik}\Big(\sum_{j\in\mathcal{J}_0} x_{ijk} - z_{ik}\Big) \\
\text{u.d.N.}\ & (6)-(11) \\
& \sum_{k\in\mathcal{V}} z_{ik} = 1 && \forall i \in \mathcal{J} \\
& z_{ik} \in \{0,1\} && \forall i \in \mathcal{J}, k \in \mathcal{V}
\end{aligned}$$

Hierbei ist der Subgradient

$$\sum_{j\in\mathcal{J}_0} x_{ijk} - z_{ik}\ ,\ i \in \mathcal{J},\ k \in \mathcal{V}$$

Auch hier lässt sich die Zielfunktion analog zum vorherigen umformen:

$$\min \sum_{i\in\mathcal{J}_0}\sum_{j\in\mathcal{J}_0}\sum_{k\in\mathcal{V}} \hat{c}_{ijk}x_{ijk} - \sum_{i\in\mathcal{J}}\sum_{k\in\mathcal{V}} \lambda_{ik}z_{ik}$$

dabei gilt

$$\hat{c}_{ijk} = \begin{cases} c_{ij} + \lambda_{ik} & i \in \mathcal{J},\ j \in \mathcal{J}_0 \\ c_{0j} & sonst \end{cases}$$

Falls die Stadt i von Verkäufer k nicht besucht werden soll, ist $z_{ik} = 0$. Wird die Stadt dennoch von Verkäufer k besucht, so erhöht sich der Subgradient im nächsten Schritt. Die Umkehrung gilt analog.
Die genaue Zuteilung der Städte zu den Verkäufern findet in einem *Semi-Assignement Problem* statt. Zusätzlich entsteht für jeden Verkäufer wieder ein *Elementary Shortest Path Problem with Time Windows* $\mathbb{ESPPTW}$.

Bemerkung 4.2. *Die untere Schranke, die durch dieses Problem erreicht wird, ist in keinem Fall besser als die der ersten Relaxation! (Vgl. [5])*

In unserem Anwendungsbeispiel sind die verbindenden Nebenbedingungen fast die gleichen wie in der ersten Relaxation, nur (12) wird durch die neue Nebenbedingung (19) ersetzt:

$$\sum_{k\in\mathcal{V}} z_{ik} = 1 \qquad \forall i \in \mathcal{J}$$

$$E_k + p_k \le S_l \qquad \forall k,l \in \mathcal{V}:\ k<l,\ \vartheta(k)=\vartheta(l)$$

$$\sum_{k\in\mathcal{V}:\ \vartheta(k)=n} (E_k - S_k - p_k y_k) \le T \qquad \forall n \in \mathcal{N}$$

Demnach können wir die relaxierte Lagrangefunktion schreiben als:

[LAG2]

$$\begin{aligned}
&\min \mathcal{ZF} \\
&\quad + \sum_{j\in\mathcal{J}}\sum_{k\in\mathcal{V}} \lambda_{jk}\Big(\sum_{i\in\mathcal{J}_0} x_{ij} - z_{ik}\Big) \\
&\quad + \sum_{k,l\in\mathcal{V}:k<l,\vartheta(k)=\vartheta(l)} \mu_{kl}(E_k + p_k - S_l) \\
&\quad + \sum_{n\in\mathcal{N}} \nu_n\Big(\sum_{k\in\mathcal{V}:\vartheta(k)=n} (E_k - S_k - p_k y_k) - T\Big)
\end{aligned}$$

Daraus ergeben sich wiederum $|\mathcal{V}|$ unabhängige Subprobleme. Für jede Pflegekraft kann das Subproblem einzeln gelöst werden.
[SUB2]

$$\begin{aligned}
&\min \mathcal{ZF} \\
&\quad + \sum_{j\in\mathcal{J}} \lambda_{ik} \sum_{i\in\mathcal{J}_0} x_{ijk} \\
&\quad + \sum_{l\in\mathcal{V}:k<l,\vartheta(k)=\vartheta(l)} \mu_{kl} E_k \\
&\quad - \sum_{l\in\mathcal{V}:k<l,\vartheta(k)=\vartheta(l)} \mu_{lk} S_k \\
&\quad + \nu_{\vartheta(k)}(E_k - S_k - p_k y_k)
\end{aligned}$$

Die Nebenbedingungen sind wie in der ersten Lagrange-Relaxation.

5 Auswertung

Bemerkung 5.1. *Die Zahlen der Auswertungen sind in Amerikanischer Schreibweise angegeben, d.h. die Nachkommastellen sind mit einem Punkt abgetrennt. Der Punkt stellt demnach keine Markierung der Tausender dar.*

Das obige Problem wurde in Xpress Mosel implementiert. Für das Update des Multiplikators wurde zum einen die Subgradientenmethode (Algorithmus 1), die bereits durch W.F. Riedl implementiert wurde, sowie die Bündelmethode (Algorithmus 3) mit Relaxation 1 verwendet.
Zur Auswertung standen drei Testdatensätze (T1, T2, T3) für 4 Pflegekräfte und 20 Jobs und drei Testdatensätze (T4, T5, T6) für 6 Pflegekräfte und 30 Jobs zur Verfügung.
Bei der Bündelmethode wurden mehrere Parameter getestet. Der Parameter für die Gewichtung des Strafterms u wurde aus der Menge $U = \{0.01; 0.05; 0.1; 0.25; 0.5; 1; 2\}$ gewählt. Hierbei wurden mehr kleinere Testparameter gewählt, da diese in vorherigen Tests bessere Ergebnisse lieferten. Die größeren Werte wurden gewählt, um die Veränderung von großen zu kleinen Strafparameter zu betrachten.
Der Parameter für die Anpassung des Zentrums m wurde aus der Menge $M = \{0.1; 0.25\}$ gewählt. Größere Werte für m wurden nicht genauer analysiert, da diese bei den getesteten Instanzen nur einen Anstieg der Iterationszahl zur Folge hatten. Eine Verbesserung der unteren Schranke oder der zulässigen Lösung wurde nicht erreicht.
Neben dem Abbruchkriterium aus Algorithmus 3 wurde der Algorithmus nach maximal 300 Iterationen abgebrochen, um die Laufzeit zu beschränken.

5.1 Ergebnisse der Testdaten

Dieses Kapitel präsentiert die Ergebnisse der Programmierung für die betrachteten Datensätze anhand von verschiedenen Parameterkombinationen aus U und M. Diese sind:

Parameterkombination	$\rho 1$	$\rho 2$	$\rho 3$	$\rho 4$	$\rho 5$	$\rho 6$	$\rho 7$
Parameter Zentrumsupdate (m)	0.1	0.1	0.1	0.1	0.1	0.1	0.1
Gewichtung Strafterm (u)	0.01	0.05	0.1	0.25	0.5	1	2

Parameterkombination	$\rho 8$	$\rho 9$	$\rho 10$	$\rho 11$	$\rho 12$	$\rho 13$	$\rho 14$
Parameter Zentrumsupdate (m)	0.25	0.25	0.25	0.25	0.25	0.25	0.25
Gewichtung Strafterm (u)	0.01	0.05	0.1	0.25	0.5	1	2

Abbildung 3: Kodierung der Parameterkombinationen ρ

Die dazu gehörigen Tabellen enthalten neben den jeweiligen Parameterkombinationen die folgenden Auswertungsergebnisse:

- den Wert der unteren Schranke;
  in blau: die größte untere Schranke
- die Zeit in Sekunden, die für die Berechnung dieser Schranke benötigt wurde;
  in grün: die kürzeste Zeit, die benötigt wurde, um die größte untere Schranke zu erreichen

- die Anzahl der benötigten Iterationen;
  in grün: die kleinste Anzahl an Iterationen, mit der die größte untere Schranke erzielt wurde
  in rot: falls die maximale Iterationszahl (hier 300) erreicht wurde
- die mit den erhaltenen Multiplikatoren berechnete zulässige Lösung;
  in grün: die kleinste zulässige Lösung

Testdaten 1
Durch Anwendung des Subgradientenverfahrens wurde nach 12.728 Sekunden und 53 Iterationen $L(\lambda) = 510.667$ als Maximum der Lagrangefunktion erreicht. Mithilfe des zugehörigen λ konnte somit eine zulässige Lösung $Z = 534$ berechnet werden.
Im Vergleich hierzu stehen die Ergebnisse der Bündelmethode. Die gesammelten Ergebnisse der Bündelmethode für Datensatz 1 sind in Tabelle 4 festgehalten. Als maximale untere Schranke wurde hier ein minimal besserer Wert $L(\lambda_{max}) = 511$ erzielt. Für $m = 0.1$ und $u = 0.01$ gelang dies in 9.311 Sekunden und 26 Iterationen. In Abbildung 5 sind die benötigte Zeit und die Iterationszahl für alle Parameterkombinationen nochmal grafisch dargestellt. Die minimale, zulässige Lösung des Anwendungsproblems $Z = 534$ wurde mit mehreren unterschiedlichen Parameterkombinationen erreicht.
Demnach liefert die Bündelmethode für die geeigneten Parameter eine bessere untere Schranke als die Subgradientenmethode. Die Lösung der Bündelmethode wurde zudem schneller berechnet. Doch auch bei der Bündelmethode bleibt eine Dualitätslücke $D = 23$.

Testdaten 2
Durch Anwendung des Subgradientenverfahrens wurde hier nach 5.077 Sekunden und 31 Iterationen $L(\lambda) = 493.906$ als Maximum der Lagrangefunktion erreicht. Mithilfe des zugehörigen λ konnte somit eine zulässige Lösung $Z = 655$ berechnet werden.
Die Ergebnisse der Bündelmethode für diesen Datensatz stehen in Tabelle 6. Als maximale untere Schranke wurde hier $L(\lambda_{max}) = 522$ erreicht. Für $m = 0.1$ und $u = 0.01$ gelang dies in 5.115 Sekunden und 28 Iterationen. In Abbildung 7 sind die benötigte Zeit und die Iterationszahl für alle Parameterkombinationen nochmal grafisch dargestellt. Als minimale zulässige Lösung ergab sich $Z = 522$. Zwar brauchte die Bündelmethode somit minimal länger, doch es konnte in weniger Iterationen eine bessere Lösung gefunden werden. Zudem ist bei der Bündelmethode die Dualitätslücke gleich Null und somit eine optimale Lösung $Z^* = 522$ des Anwendungsproblems gefunden.

Testdaten 3
Durch Anwendung des Subgradientenverfahrens wurde hier nach 35.754 Sekunden und 59 Iterationen $L(\lambda) = 348.881$ als Maximum der Lagrangefunktion erreicht. Mithilfe des zugehörigen λ konnte somit eine zulässige Lösung $Z = 455$ berechnet werden.
Die gesammelten Ergebnisse der Bündelmethode zu diesem Datensatz stehen in der Tabelle 8. Hier wurde eine maximale untere Schranke $L(\lambda_{max}) = 351.192$ gefunden. Diese wurde am schnellsten von $m = 0.25$ und $u = 0.05$ in 26.468 Sekunden und 32 Iterationen erreicht. In Abbildung 9 sind die benötigte Zeit und die Iterationszahl für alle Parameterkombinationen nochmal grafisch dargestellt.
Die Bündelmethode liefert demnach auch für diesen Datensatz in kürzerer Zeit und Iterationenanzahl ein besseres Ergebnis. Da die kleinste gefundene zulässige Lösung $Z = 405$

Abbildung 4: Testergebnisse mit Relaxation 1 für Datensatz 1

Parameterkombination	$\rho 1$	$\rho 2$	$\rho 3$	$\rho 4$	$\rho 5$	$\rho 6$	$\rho 7$
Maximum	511	511	511	511	510.999	511	499.182
Zeit (Sek.) nach Lagrange-Relaxation	9.311	16.152	18.846	72.277	65.037	175.589	213.569
Iterationszahl	26	32	40	93	124	251	300
Zulässige Lösung	541	541	543	534	543	543	562

Parameterkombination	$\rho 8$	$\rho 9$	$\rho 10$	$\rho 11$	$\rho 12$	$\rho 13$	$\rho 14$
Maximum	511	511	511	511	511	508.836	497.626
Zeit (Sek.) nach Lagrange-Relaxation	18.709	20.37	35.819	45.83	125.474	218.465	220.711
Iterationszahl	27	30	46	81	198	300	300
Zulässige Lösung	534	543	541	534	534	547	534

Abbildung 5: Änderung der benötigten Zeit und Iterationen der verschiedenen Parameterkombinationen für Datensatz 1

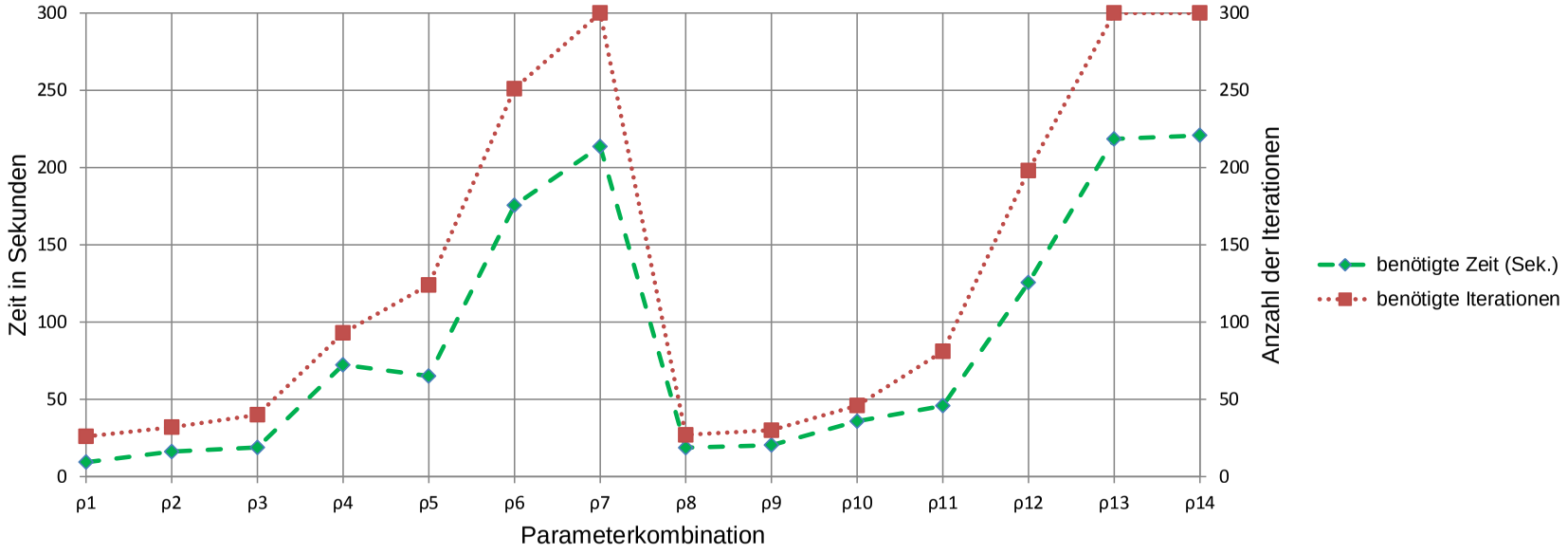


Abbildung 6: Testergebnisse mit Relaxation 1 für Datensatz 2

Parameterkombination	$\rho 1$	$\rho 2$	$\rho 3$	$\rho 4$	$\rho 5$	$\rho 6$	$\rho 7$
Untere Schranke	522	522	521.998	521.999	521.999	507.01	467.645
Zeit (Sek.) nach Lagrange-Relaxation	5.115	23.649	23.122	47.971	98.976	153.999	131.749
Iterationszahl	28	42	68	141	256	300	300
Zulässige Lösung	522	522	522	522	522	522	705

Parameterkombination	$\rho 8$	$\rho 9$	$\rho 10$	$\rho 11$	$\rho 12$	$\rho 13$	$\rho 14$
Untere Schranke	522	521.999	522	521.998	520.543	504.607	437.493
Zeit (Sek.) nach Lagrange-Relaxation	13.051	29.94	43.466	117.796	156.656	170.315	148.482
Iterationszahl	31	48	90	192	300	300	300
Zulässige Lösung	522	522	522	522	522	655	744

Abbildung 7: Änderung der benötigten Zeit und Iterationen der verschiedenen Parameterkombinationen für Datensatz 2

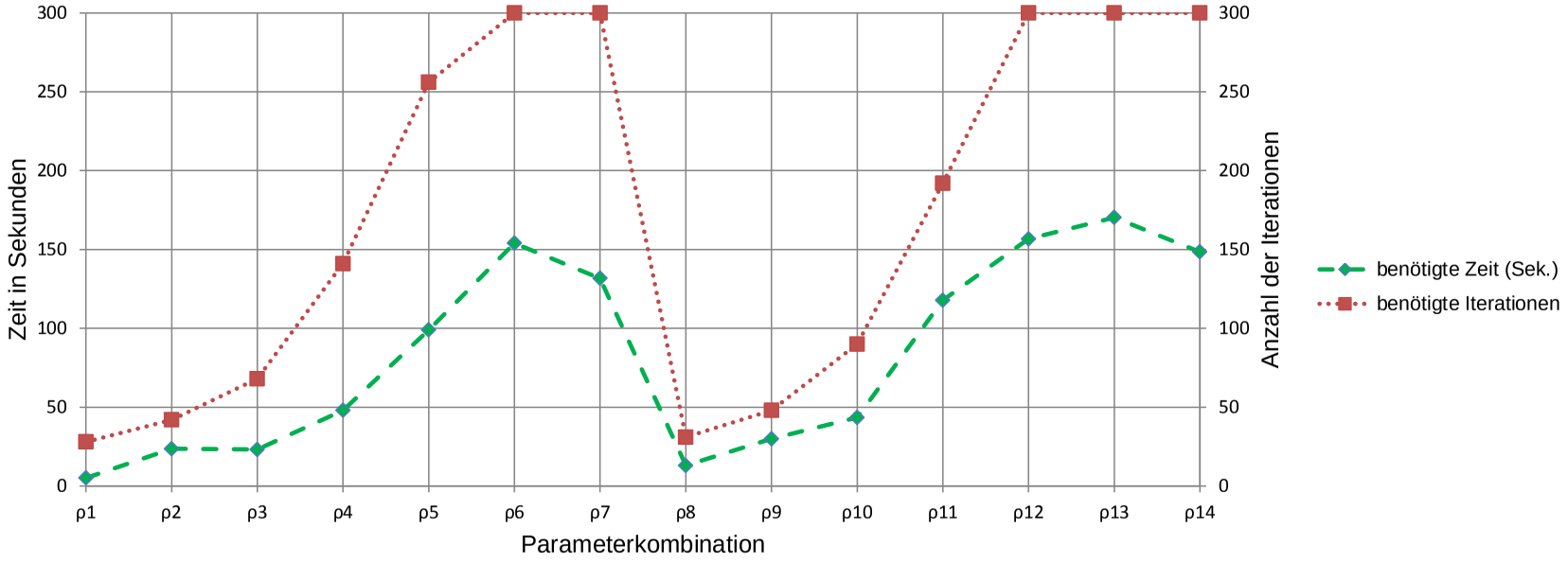
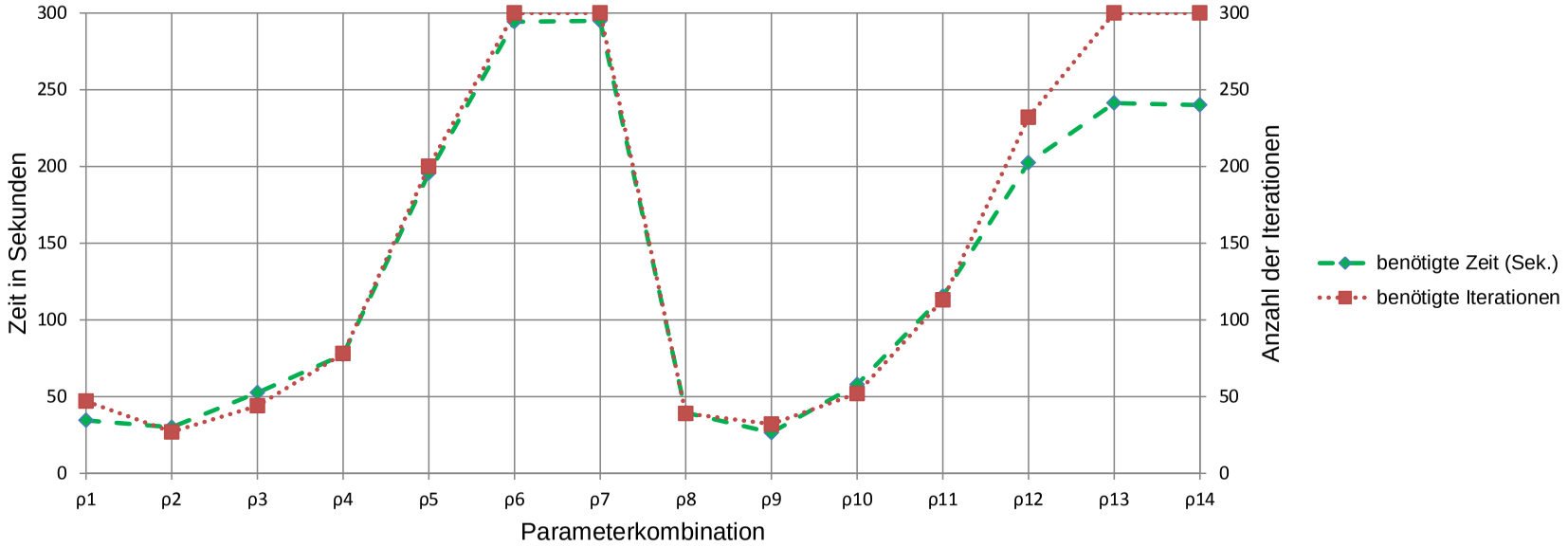


Abbildung 8: Testergebnisse mit Relaxation 1 für Datensatz 3

Parameterkombination	$\rho 1$	$\rho 2$	$\rho 3$	$\rho 4$	$\rho 5$	$\rho 6$	$\rho 7$
Untere Schranke	351.192	351.191	351.191	351.192	351.191	351.069	350.385
Zeit (Sek.) nach Lagrange-Relaxation	34.495	29.938	52.524	77.381	195.571	294.26	294.9
Iterationszahl	47	27	44	78	200	300	300
Zulässige Lösung	1E+40	430	1E+40	1E+40	1E+40	1E+40	1E+40

Parameterkombination	$\rho 8$	$\rho 9$	$\rho 10$	$\rho 11$	$\rho 12$	$\rho 13$	$\rho 14$
Untere Schranke	351.192	351.192	351.192	351.191	351.189	350.87	345.604
Zeit (Sek.) nach Lagrange-Relaxation	39.437	26.468	57.81	115.387	202.393	241.255	239.969
Iterationszahl	39	32	52	113	232	300	300
Zulässige Lösung	457	405	405	1E+40	1E+40	1E+40	445

Abbildung 9: Änderung der benötigten Zeit und Iterationen der verschiedenen Parameterkombinationen für Datensatz 3



ist, ist die erhaltene Dualitätslücke somit $D = 53.808$.

Testdaten 4 bis 6
Die Auswertung der Testdaten 4 und 6 ergab, dass auch hier mit dem Faktor $u = 0.01$ des Strafterms die besten unteren Schranken erzeugt wurden. Die Wahl des Parameters m war dabei nicht eindeutig. Mal wurden weniger Iterationen für $m = 0.1$ und mal weniger für $m = 0.25$ benötigt.
Die besten Parameter für die Testdaten 5 waren hingegen $u = m = 0.25$. Diese Parameter lieferten mit Abstand die beste untere Schranke.
Bei allen drei Datensätzen waren die unteren Schranken der Bündelmethode wesentlich besser als die der Subgradientenmethode, jedoch wurde in der Bündelmethode immer mehr Iterationen und meist doppelt so viel Zeit wie für die Subgradientenmethode benötigt.

5.2 Fazit der Auswertung

Der Vergleich zwischen Subgradientenmethode und Bündelmethode zeigt, dass die Bündelmethode bessere Ergebnisse liefert.
Es zeigt sich, dass vorallem mit den kleineren Faktoren aus der Menge U sehr ähnliche Ergebnisse erzeugt wurden. Bei stärkerer Gewichtung des Strafterms steigt auch die Anzahl der benötigten Iterationen. Für $u = 1$ und $u = 2$ wird meistens die maximale Iterationszahl erreicht. Auch im Hinblick auf die Wahl der Startmultiplikatoren ist die Wahl einer kleineren Gewichtung des Strafterms sinnvoll. Sind die Startmultiplikatoren schlecht gewählt und der Faktor u groß, so ist es schwer sich weit von diesen zu entfernen. Bessere Multiplikatoren, die eventuell weit von den Startmultiplikatoren entfernt sind, können so nur sehr langsam erreicht werden.

Beispiel 5.2. *Bisher wurde* $\lambda_{\text{initial}} = \max_{\{i,j \in \mathcal{J} | i \neq j\}} c(i,j)$ *gewählt, so dass* $\lambda_j = \lambda_{\text{initial}}$, $\forall\ j \in \mathcal{J}$ *die Multiplikatoren in der ersten Iteration waren. Mit diesem Startmultiplikator wurden für alle Faktoren u eine relativ gute untere Schranke berechnet.*
Wählt man nun $\lambda_{\text{initial}} = 0$*, so liefert der Algorithmus mit Datensatz T1 nur für* $u = 0.01$ *ein sinnvolles Ergebnis.*

In Abbildung 10 sieht man, wie die Gewichtung des Strafterms im Algorithmus wirkt. Hierbei wurden die Daten von Datensatz 2 mit $m = 0.1$ verwendet.
Je kleiner der Faktor ist, desto größere Schrittweiten können gemacht werden. Die Schwankungen in den Werten für die untere Schranke können somit sehr groß sein. Je größer der Faktor ist, desto kleiner ist der Unterschied zwischen den Werten der unteren Schranke zweier aufeinanderfolgender Iterationen.

Durch die größeren Schwankungen in der Anfangsphase kommt der Algorithmus für kleinere Faktoren u meist schneller zu einem Ergebnis. Doch je näher man dem Maximum im Laufe des Algorithmus kommt, desto mehr möchte man die Schrittweiten begrenzen, damit man sich durch den nächsten Schritt nicht weit vom bisherigen Maximum entfernt. Eine Möglichkeit der Anpassung wäre eine Veränderung des Strafterms in Abhängigkeit von der Iterationszahl k. Anfängliche Versuche mit $u_1 = 0.01$ und $u_k = u_1 \cdot k$ beziehungsweise $u_k = u_1 \cdot k^{\frac{1}{2}}$ lieferten zwar keine Verbesserung der unteren Schranke, doch konnten

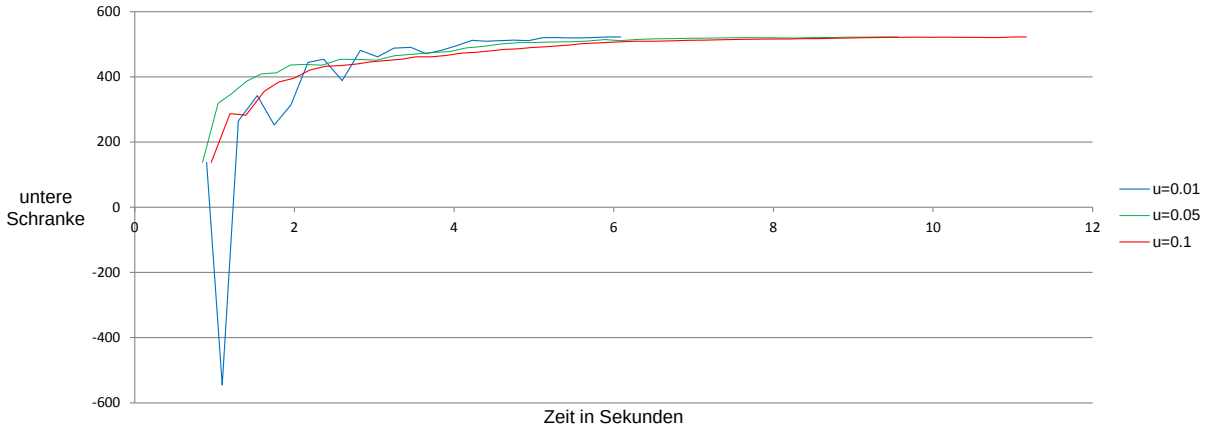


Abbildung 10: Wirkungsweise der Gewichtung des Strafterms

teilweise bessere Laufzeiten und eine geringere Anzahl an Iterationen erzielt werden. Hier könnte demnach noch Potenzial zur Verbesserung sein.

Die Wahl des Faktors m für die Zentrumsanpassung spielt im Gegensatz zu u in den meisten Fällen eine eher untergeordnete Rolle. Oft wird bei einem höheren Faktor, d.h. einer geringeren Wahrscheinlichkeit von Zentrumsanpassungen, nur eine höhere Anzahl von Iterationen gebraucht, um zum gleichen Ergebnis zu kommen. In Abbildung 11 sind für $u = 0.01$ drei verschiedene Gewichtungen für die Zentrumsanpassung basierend auf dem Testdatensatz 1 dargestellt. Wenn in einer Iteration das Zentrum angepasst wurde, so ist dies in der Abbildung mit einem senkrechten Strich vermerkt.

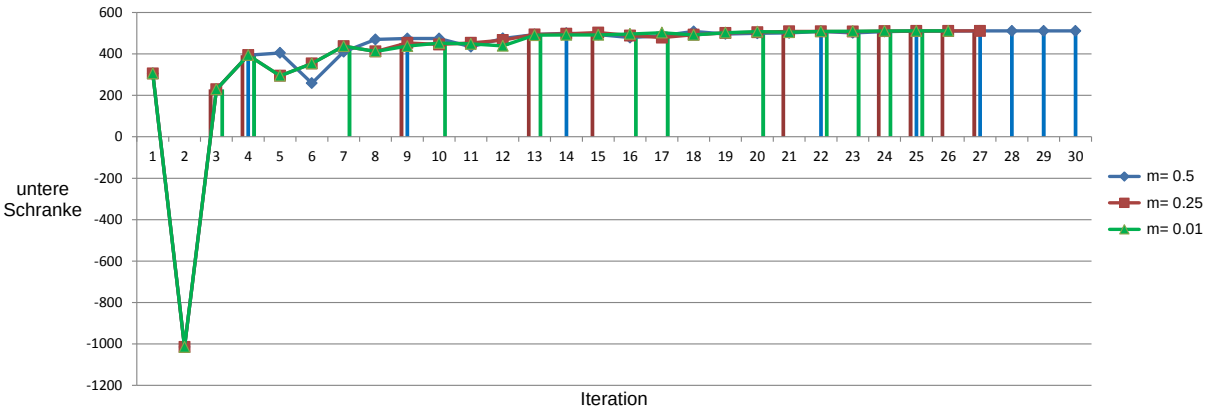


Abbildung 11: Wirkungsweise der Zentrumsanpassung

Je größer die Gewichtung der Zentrumsanpassung ist, desto seltener wird dieses angepasst. Für $m = 0.5$ liegen die senkrechten Linien zu Beginn weiter auseinander als für die anderen Parameter. Wenn mit fortschreitendem Algorithmus die Approximation der Lagrangefunktion besser wird, so ist der Unterschied zwischen vorhergesagtem und tatsächlichen Anstieg geringer. Demnach wird auch das Zentrum häufiger angepasst.

Für weitere Auswertungen empfiehlt sich daher die Gewichtung des Strafterms $u = 0.01$ zu wählen oder einen Anpassungsalgorithmus zu entwickeln. Bei der Zentrumsanpassung empfiehlt sich auch der kleinere Wert, da hier in den meisten Fällen eine geringere Anzahl von Iterationen nötig ist.

6 Zusammenfassung und Ausblick

Die Lagrange-Relaxation ist ein wichtiges Verfahren zur Optimierung diverser Probleme. Duch die Relaxierung schwieriger Nebenbedingungen mit zusätzlichen Lagrange-Multiplikatoren können Probleme vereinfacht und somit schneller oder überhaupt erst gelöst werden beziehungsweise mit der Lagrangefunktion eine untere Schranke an die optimale Lösung gefunden werden.
Da die Lagrangefunktion immer als untere Schranke existiert, ist ein wichtiger Bestandteil der Lagrange-Relaxation diese zu maximieren, um sich der optimalen Lösung anzunähern. In dieser Arbeit wurden einige Verfahren zur Maximierung vorgestellt. Zum einen das Subgradientenverfahren, das aufgrund von Subgradienten als Suchrichtungen und berechneten Schrittweiten versucht das Maximum der Lagrangefunktion schrittweise zu erreichen, zum anderen die Bündelmethode, die über die Maximierung einer Approximation, bestehend aus Stützgeraden, versucht das Maximum der Lagrangefunktion zu finden. Um diese Verfahren hinsichtlich der Rechenzeit für das Update zu verbessern, kann die Komplexität verringert werden, so wie in der Interleaved-Subgradientenmethode, der Surrogate-Subgradientenmethode oder der Vereinfachten Bündelmethode, oder die Prozedur der Optimierung verbessert werden, wie in der Verbesserten Bündelmethode.

Als Anwendungsbeispiel diente die ambulante Krankenpflege mit dem Problem der optimalen Zuordnung von Jobs bei Patienten zum Pflegepersonal. Dieses Problem wurde mithilfe des Vehicle Routing Problems modelliert, zu dem noch zusätzliche Bedingungen kamen. In diesem Problem wurde auf zwei verschiedene Arten relaxiert, wodurch das Problem in mehrere, oftmals gleiche, Subprobleme zerfallen ist. Für jede Pflegekraft entstand somit ein eigenes Problem.

Durch die Implementierung einer der Relaxationen in Mosel wurden einige gute Ergebnisse für untere Schranken erzielt. Hierbei waren die Ergebnisse der Bündelmethode für spezielle Parameter (meist m=0.1 und u=0.01) immer besser als die der Subgradientenmethode.

Bei der Auswertung fiel auf, dass eine geringe Gewichtung des Strafterms durch u besonders in den ersten Iterationen wichtig ist. Mit steigender Iterationszahl sollte der Algorithmus nicht mehr so stark schwanken, daher wäre hier eine größere Gewichtung des Strafterms nötig. Ein offener Punkt ist daher die Optimierung dieses Faktors durch eine Anpassung gemäß der Iterationszahl. Hierdurch soll ein allgemeiner Faktor u_k für Iteration k geschaffen werden, der für alle Probleme in angemessener Zeit eine untere Schranke findet.

Hier wurde die Verwendung der Lagrange Relaxation nur in Kombination mit der Subgradientenmethode und der Bündelmethode an einem linearen, ganzzahligen Optimierungsproblem getestet. Es wäre daher interessant die Wirkung anderer Methoden für ein solches Problem mit den bisherigen Ergebnissen für die untere Schranke zu vergleichen.

Ein anderer Ansatz, der bisher noch nicht beachtet wurde, ist die minimale zulässige Lösung, die bisher berechnet wurde, zu verbessern. Man könnte zum Beispiel die Art der

Berechnung, die von W.F. Riedl implementiert wurde, verändern und somit verbessern. Des Weiteren könnten für den Algorithmus andere Abbruchkriterien verwendet werden, um bessere Multiplikatoren zu bekommen. Beim Testen der Implementierung lieferte der Algorithmus bei frühzeitigem Abbrechen manchmal bessere Multiplikatoren, sodass bessere zulässige Lösungen erzeugt wurden.

Diese kurz skizzierten Ansätze bieten Raum für weitere Untersuchungen.

Literatur

[1] C.A. Kaskavelis and M.C. Caramanis. Efficient Lagrangian relaxation algorithms for industry size job-shop scheduling problems. *IIE Transactions*, 30(11):1085–1097, 1998. Kluwer Academic notes.

[2] H.M. Dietz. Konvexe Funktionen. In *Mathematik für Wirtschaftswissenschaftler*, Springer-Lehrbuch, pages 329–355. Springer Berlin Heidelberg, 2012.

[3] D.P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods*. Optimization and Neural Computation Series. Athena Scientific, 1 edition, 1996.

[4] S. Boyd and A. Mutapcic. Subgradient Methods. 2008. Notes for EE364b, Stanford University, Winter 2006-07.

[5] W.F. Riedl. Optimale Einsatz- und Routenplanung in der ambulanten Krankenpflege. Bachelorarbeit, Technische Universität München, 2011.

[6] X. Zhao, P. B. Luh, and J. Wang. Surrogate Gradient Algorithm for Lagrangian Relaxation. *Journal of Optimization Theory and Applications*, 100:699–712, 1999.

[7] K.C. Kiwiel. Proximity control in bundle methods for convex nondifferentiable minimization. *Mathematical Programming*, 46(1-3):105–122, 1990. Springer-Verlag.

[8] C. Lemaréchal and J. Zowe. A Condensed Introduction to Bundle Methods in Nonsmooth Optimization. In Emilio Spedicato, editor, *Algorithms for Continuous Optimization*, volume 434 of *NATO ASI Series*, pages 357–382. 1994. Springer Netherlands.

[9] J.F. Bonnans, J.C. Gilbert, C. Lemaréchal, and C.A. Sagastizábal. Numerical Optimization, Theoretical and Practical Aspects. 2006.

[10] M. Hintermüller. A Proximal Bundle Method Based on Approximate Subgradients. *Computational Optimization and Applications*, 20(3):245–266, 2001.

[11] K.C. Kiwiel. Approximations in proximal bundle methods and decomposition of convex programs. *Journal of Optimization Theory and Applications*, 84(3):529–548, 1995. Kluwer Academic notes-Plenum notes.

[12] R. Borndörfer, A. Löbel, and S. Weider. A Bundle Method for Integrated Multi-Depot Vehicle and Duty Scheduling in Public Transit. In Mark Hickman, Pitu Mirchandani, and Stefan Voß, editors, *Computer-aided Systems in Public Transport*, Lecture Notes in Economics and Mathematical Systems, pages 3–24. 2008. Springer Berlin Heidelberg.

[13] T. Grünert and S. Irnich. *Optimierung im Transport - Band 1: Grundlagen*. Shaker Verlag Aachen, 2005.

[14] R. K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Englewood Cliffs, NJ, 1993.

[15] X. Zhao and P.B. Luh. New Bundle Methods for Solving Lagrangian Relaxation Dual Problems. *Journal of Optimization Theory and Applications*, 113(2):373–397, 2002.

[16] C. Lemaréchal, A. Nemirovskii, and Y. Nesterov. New variants of bundle methods. *Mathematical Programming*, 69(1-3):111–147, 1995.

[17] C. Stefan. Lagrangian Relaxation Solving NP-hard Problems in Computational Biology via Combinatorial Optimization, 2009.

[18] D.P. Bertsekas. *Nonlinear programming*. 1999. Athena Scientific.