Globale Optimierung-Ausgewählte Kapitel aus der Optimierung

Vorlesungsmitschrift SS 2009

Prof. Hermann Schichl, Karl Stelzhammer

12. April 2011

Inhaltsverzeichnis

1 Einführung in die globale Optimierung

1.1 Klassifizierung der betrachteten Probleme

Wir werden in dieser Vorlesung Probleme folgender Natur betrachten: Seien f und F zwei Funktionen mit $f : \mathbb{R}^n \mapsto \mathbb{R}$ und $F : \mathbb{R}^n \mapsto \mathbb{R}^m$.

$$\begin{aligned} &\min f(x) \\ &s.t.\ F(x) \in \mathbf{F} \\ &\quad x \in \mathbf{x} \end{aligned} \tag{1.1}$$

Definition 1.1. Seien $x, \underline{x}, \overline{x} \in \mathbb{R}^n$, $\mathbf{x} := [\underline{x}, \overline{x}]$. Der Ausdruck $x \in \mathbf{x}$ bedeutet, dass $x \in [\underline{x}, \overline{x}]$.

Das Ziel der Vorlesung ist es, Methoden zu entwickeln, mit deren Hilfe das globale Optimum für das Problem (1.1) gefunden werden kann.

1.2 Inhalte der Vorlesung

Im Zuge dieser Vorlesung werden folgende Punkte betrachtet werden:

- Welche unterschiedlichen Arten und Typen von Algorithmen es gibt und eine Klassifizierung dieser Algorithmen.
- Betrachtung einiger Anwendungsprobleme die zu globalen Optimierungsproblemen führen.
- Ideen der unvollständigen, auch genannt heuristischen, und asymptotisch vollständigen Algorithmen.
- Theorie für die Entwicklung vollständiger Algorithmen, unterteilt in folgende Kapitel:
  - Branch and Bound
  - Intervallanalysis
  - konvexe und DC-Funktionen
  - Relaxationen
  - Constraint Propagation
- Theorie zur Lösung polynomialer globaler Optimierungsprobleme.

1.3 Komplexitätstheorie

Im Folgenden gibt es nun einen kleinen Auszug aus der Komplexitätstheorie: Globale Optimierung ist *schwieriger* als lokale Optimierung, lineare Optimierung oder konvexe Optimierung. Um in diesem Zusammenhang den Begriff *schwieriger* genauer definieren zu können braucht man Komplexitätstheorie: Die Standard-Komplexitätstheorie kümmert sich um Entscheidungsprobleme: Probleme dieser Form können die Antwort Ja oder Nein haben.

Beispiel 1.2. Ist 3413961271 eine Primzahl? Ist $\widehat{x}$ ein globales Minimum von (1.1)?

Wir bestimmen das worst-case Verhalten des besten Algorithmus zur Lösung des Problems und sein Verhalten: $A(n)$.
Gibt es ein Polynom, sodass $\forall\, n \in \mathbb{N} : A(n) \leq p(n)$, dann sagt man, dass das Entscheidungsproblem zur Klasse P-resource gehört.
PTIME =: P...Klasse der in polynomialer Zeit lösbaren Probleme
PSPACE...Klasse der mit polynomialem Speicherbedarf lösbaren Probleme

Weil jeder Speicherzugriff Zeit benötigt gilt: P $\subseteq$ PSPACE

Gibt es ein Polynom $p(n)$, sodass $\forall\, n \in \mathbb{N} : A(n) \leq e^{p(n)}$, dann liegt das Entscheidungsproblem in EXP-resource (also EXPTIME oder EXPSPACE), die Klasse der mit exponentiellem Ressourcenaufwand lösbaren Probleme. Weiter kann man zeigen:

$$\text{P} \subseteq \text{PSPACE} \subseteq \text{EXPTIME} \subseteq \text{EXPSPACE}$$

Definition 1.3. Sei $e \in \mathbb{R}^n$. Der Vektor e ist in dieser Vorlesung definiert als: $e := (1, ..., 1)^T$.

Nun werden wir anhand einiger Beispiele eine Motivation geben globale Optimierungsprobleme zu betrachten:

- Das erste Beispiel ist die Keplersche Vermutung:
  Die Aussage dieser Vermutung lautet, dass das kubisch flächenzentrierte Gitter die dichteste Kugelpackung im $\mathbb{R}^3$ erzeugt. Bewiesen wurde die Keplersche Vermutung von Hales.

- Ein weiteres Beispiel kommt aus dem Sektor der Graphentheorie: Das Maximum-Clique-Problem. Wieviele Ecken enthält die maximale Clique in einem Graphen (Clique ist ein vollständiger Teilgraph; vollständig bedeutet, dass je zwei Ecken miteinander verbunden sind)?
  Beispiele für vollständige Graphen:

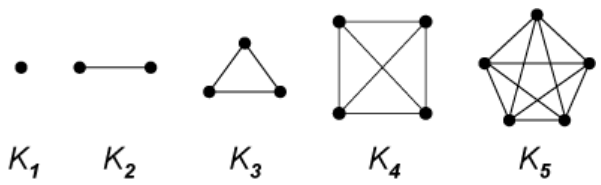


Abbildung 1: Die obige Abbildung zeigt die vollständigen Graphen $K_1, ..., K_5$.

Im Jahr 1965 wurde von Motzkin-Strauss gezeigt: Äquivalent zu diesem Problem ist es, das globale Maximum von

$$\begin{aligned} &\max\, x^T A x \\ &s.t.\ e^T x = 1 \\ &\quad x \geq 0 \end{aligned}$$

zu finden, wobei A die Adjazenz-Matrix des Graphen ist.

$$A_{ij} \begin{cases} = 1 \text{ , Ecken i und j sind verbunden} \\ = 0 \text{ , Ecken i und j sind nicht verbunden} \end{cases}$$

Das ist ein QP (quadratic problem). Im Allgemeinen ist es indefinit.

Eine Frage ist, ob dieses Problem (genauer die Frage, ob n die Größe der maximalen Clique ist) in P liegt.
Angenommen, wir haben ein „Orakel“, das uns die (Ja-)Lösung eines Entscheidungsproblems mitsamt Begründung verrät. Wir können jetzt analog zur Definition der Klassen P, PSPACE, EXPTIME, EXPSPACE messen, was die worst-case-Komplexität für die Überprüfung der Begründung ist.
Ist der Aufwand $\widehat{A}(n)$ zum Überprüfen der Lösung durch ein Polynom abschätzbar, d.h. $\widehat{A}(n) \leq p(n) \; \forall \; n$, so sagen wir, das Problem liegt in der Klasse NP-resource (nicht deterministisch polynomial).
Ist die Abschätzung exponentiell, d.h. $\widehat{A}(n) \leq e^{p(n)} \; \forall \; n$, dann ist das Problem Teil der Klasse NEXP-resource. Auf diese Weise formen wir die Klassen NP=NPTIME, NPSPACE, NEXPTIME, NEXPSPACE.
Offensichtlich gilt:

$$\begin{gathered} \text{P} \subseteq \text{NP} \\ \text{PSPACE} \subseteq \text{NPSPACE} \\ \text{EXPTIME} \subseteq \text{NEXPTIME} \end{gathered}$$

Überraschender (und auch nicht-trivial) sind die folgenden Resultate:

$$\text{NP} \subseteq \text{PSPACE}$$

Außerdem gilt:

$$\begin{gathered} \text{PSPACE} = \text{NPSPACE} \\ \text{NPSPACE} \subseteq \text{EXPTIME} \\ \text{P} \subseteq \text{NP} \subseteq \text{PSPACE} = \text{NPSPACE} \subseteq \text{EXPTIME} \end{gathered}$$

Das Zeithierarchie-Theorem in der Komplexitätstheorie impliziert, dass

$$\text{P} \subset \text{EXPTIME}$$

Für die Beantwortung der Frage, ob „P=NP ?“ gilt, ist ein Preis von 1 Million US Dollar ausgesetzt.

Ein Problem P heißt K-hart, wenn sich jedes Problem in Komplexitätsklasse K mit passendem Ressourcenaufwand auf die Lösung von P zurückführen lässt. P heißt K-vollständig, wenn es selbst in K liegt.
Es genügt, um P=NP ? zu lösen, ein NP-vollständiges Problem zu analysieren. Findet man einen Algorithmus, der dieses Problem in polynomialer Zeit löst, dann gilt P=NP. Beweist man, dass sich das Problem nicht in polynomialer Zeit lösen lässt, dann ist P $\neq$ NP.
Das Maximum-Clique-Problem ist NP-vollständig. Daher ist auch das Lösen von QPs mit indefiniten Matrizen NP-hart.
Das erste Problem, für das bewiesen wurde, dass es NP-vollständig ist, ist 3SAT.

Beispiel 1.4. 3SAT: Gegeben sei eine Menge logischer Ausdrücke in den binären Variablen $x_1, ..., x_N$, sodass in jedem logischen Ausdruck höchstens drei Variablen vorkommen. Gibt es eine Belegung aller Variablen mit 0 oder 1, sodass alle logischen Ausdrücke wahr sind?

Nachdem schon das Lösen von QPs (nicht konvex!) NP-hart ist, muss natürlich auch das allgemeine Globale Optimierungsproblem NP-hart sein. Es kann also (außer der Fall P=NP tritt ein) nur Algorithmen geben, deren worst-case-Komplexität exponentiell von der Problemgröße abhängt. Es ist unbekannt, ob das allgemeine Optimierungsproblem in NP liegt. Es liegt sicher in EXPTIME.

1.4 Wozu globale Optimierung?

Als Ausgang für den kommenden Abschnitt betrachten wir den Spezialfall von Problem (1.1) mit $f(x) \equiv 0$. In diesem Fall ist jeder zulässige Punkt globales Optimum. Solch ein Problem heißt **Zulässigkeitsproblem** (oder auch **CSP**: **constraint satisfaction problem**).
Sei x^* ein zulässiger Punkt für (1.1). Dann muss ein globales Optimum von (1.1) jedenfalls $f(x) \leq f(x^*)$ erfüllen. Ist x^* ein globales Minimum, dann müssen alle anderen globalen Minima $f(x) = f(x^*)$ erfüllen. In jedem Fall liefert das Zulässigkeitsproblem

$$\begin{aligned} f(x) &\leq f(x^*) \\ F(x) &\in \mathbf{F} \\ x &\in \mathbf{x} \end{aligned}$$

Informationen über das globale Optimierungsproblem und über x^*. Sei etwa $\hat{\mathbf{x}}$ eine kleine Box, die x^* enthält. Wenn das Zulässigkeitsproblem

$$\begin{aligned} f(x) &\leq f(x*) \\ F(x) &\in \mathbf{F} \\ x &\in \mathbf{x} \setminus \hat{\mathbf{x}} \end{aligned}$$

keine Lösung besitzt, dann liegt das globale Minimum von (1.1) in $\hat{\mathbf{x}}$.

1.5 Anwendungsprobleme, die auf globale Optimierung führen

Im Folgenden werden wir jetzt ein paar Beispiele aus verschiedenen Richtungen betrachten, die aber immer auf ein globales Optimierungsproblem zurück geführt werden können.

Beispiel 1.5. Unsere ersten beiden Beispiele führen in die Chemie:

- Ein Problem bei dem unter anderem die globale Optimierung als Hilfsmittel genommen wird ist das Auffinden von Phasenübergängen. (MEHR INFOS EV...)
- Eine zweite Anwendung in der Chemie ist es, Reaktionsgleichgewichte vorherzusagen. Wenn man eine chemische Reaktion, die unter anderem mit der Gibbs-freien Energie zusammenhängt, betrachtet, bei der man mit den Stoffen A und B die Stoffe C und D herstellt, will man in unserem Fall o.B.d.A. wissen unter welchen Vorraussetzungen (welche Temperatur, wieviel braucht man von welchem Stoff) man möglichst viel von Stoff C bekommt.

Beispiel 1.6. Das dritte Beispiel stammt aus der Robotik. Wir betrachten hier sogenannte Gough-Platforms (auch genannt: Parallelroboter).

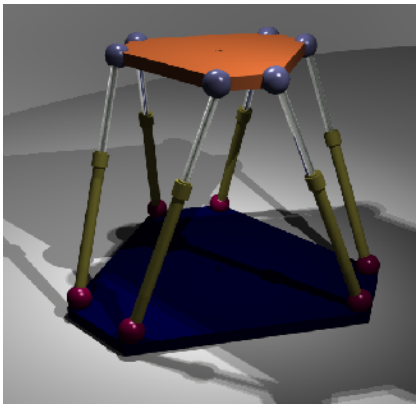


Abbildung 2: Dies ist ein Modell einer Gough-Platform. Der zu bearbeitende Teil liegt auf der oberen Platte.

Bei dieser Art von Roboter können alle Beine ihre Länge ändern. Bei diesem Problem geht es darum, Singularitäten in der Konstruktion zu finden und dann gegebenenfalls den Roboter anzupassen um diese zu vermeiden (wenn man nur in die Nähe der Singularität kommt hat das in Vergangenheit zur Zerstörung solcher Roboter geführt). J.P. Merlet hat gezeigt, dass man das Singulärwertproblem durch ein Zulässigkeitsproblem beschreiben kann und dann auch mit globaler Optimierung dieses lösen kann.

Beispiel 1.7. Dieses Problem ist nun mal eines anderer Natur als die vorherigen. Es ist auch bekannt als das Knapsack-Problem: Bei diesem Problem ist die Annahme die, dass ein Dieb sich in einem Haus befindet und einen Sack dabei hat, den er gerne mit möglichst wertvollen, aber leichten Sachen füllen würde. Also etwas formalisierter ausgedrückt: haben Objekte $O_i = (v_i, w_i)$ mit v_i...Werte der einzelnen Objekte, w_i...Gewichte der einzelnen Objekte. Ziel des Diebes ist nun:

$$\begin{aligned} \max & \sum_{i=1}^{n} x_i v_i \\ s.t. & \sum_{i=1}^{n} x_i w_i \leq W \\ & x_i \in \{0, 1\} \end{aligned}$$

wobei W das Gewicht ist, das der Dieb noch tragen kann.

Beispiel 1.8. Eine weitere ganz wesentliche Klasse von Problemen, in denen globale Optimierung zum Zug kommt, sind Zeitplanungsprobleme (oder auch scheduling problems). Hierbei geht es darum, für eine Fluglinie oder Bahnlinie einen Fahrplan zu erstellen, der alle relevanten Gegebenheiten berücksichtigt und auch noch robust genug ist (zum Beispiel falls mal ein Zug ausfallen sollte, dass dann nicht das ganze Bahnnetz zum Erliegen kommt). Oder aber auch den Stundenplan für eine Universität oder Schule zu erstellen gehört zu dieser Klasse dazu.

Generell ist es sinnvoll, bei der globalen Optimierung zu unterscheiden, aus welchem Bereich das Problem kommt. Kommt es aus der Wirtschaft, so reicht oftmals eine Lösung die „gut genug“ (oder auch robust genug wie wir gesehen haben) ist aus. Stammt das Problem aber aus der Chemie, Mathematik (Keplersche Vermutung) oder Robotik, so ist es in der Tat meistens notwendig, wirklich das globale Optimum zu finden (und auch zu beweisen, dass man eines gefunden hat).

Beispiel 1.9. Das mit Abstand bekannteste Problem das mit Mitteln der globalen Optimierung behandelt wird ist die Proteinfaltung. Es ist mittlerweile nicht mehr schwierig, das Genom eines Lebenwesens zu sequenzieren. Das ergibt

$$\text{A} \quad \underbrace{TCG}_{\text{je 3 gehören zusammen}} \quad \text{SATTACCAAAGC... (also eine DNA-Sequenz).}$$

3 Buchstaben bestimmen eine von 20 möglichen Aminosäuren. Ketten von Aminosäuren bilden Proteine. Wir kennen von jedem Protein die Primärstruktur, das ist die Aminosäurensequenz, aus der das Protein besteht. Die Aminosäurensequenz bildet eine 3D-Struktur, die Tertiärstruktur, die die chemischen Eigenschaften des Proteins bestimmt. Die Bestimmung der Tertiärstruktur aus der Primärstruktur ist ein schwieriges Problem. Zur Zeit ist dies auf „chemischem Weg“ möglich, benötigt allerdings 2-6 Monate pro Protein! Die Tertiärstruktur ist jene Form, in der das Protein minimale chemische Energie hat. Das heißt wir bestimmen das chemische Potential von $V(x), x \in \mathbb{R}^3, x$ stellt die Koordianten der Atome dar, und suchen das globale Minimum (ca. 10000-100000 Variable).

2 Grundlagen

2.1 Die mathematische Formulierung

Das Optimierungsproblem werden wir nun wie folgt formulieren, weiter untersuchen und auch klassifizieren: Seien f, F Funktionen mit $f : \mathbb{R}^n \mapsto \mathbb{R}, F : \mathbb{R}^n \mapsto \mathbb{R}^m$, $I \subseteq \{1, ..., n\}$.

$$\begin{aligned} &\min f(x) \\ &s.t.\ F(x) \in \mathbf{F} \\ &\quad x \in \mathbf{x} \\ &\quad x_I \in \mathbb{Z}^{|I|} \end{aligned} \tag{2.1}$$

(2.1) ist ein gemischt-ganzzahliges Optimierunsproblem, f die Zielfunktion, F die Nebenbedingungen und $x \in \mathbf{x}$ sind die Schrankennebenbedigungen (bound contraints). Gilt $\mathbf{F}_i = \left[\widetilde{F}_i, \widetilde{F}_i\right]$, dann heißt $F_i \in \mathbf{F}_i$ Gleichungsnebenbedigung, sonst Ungleichungsnebenbedingung. Ist $\mathbf{F}_i$ bei einer Ungleichungsnebenbedingung beschränkt, dann heißt sie zweiseitige Ungleichungsnebenbedingung.
Falls $I = \{1, ..., n\}$, dann liegt ein diskretes oder kombinatorisches Optimierungsproblem vor. Ist $I = \emptyset$, so sprechen wir von einem stetigen Optimierungsproblem.
$\mathcal{F} := \{x \in \mathbb{R}^n | F(x) \in \mathbf{F}, x \in \mathbf{x}, x_I \in \mathbb{Z}\}$ ist die zulässige Menge und alle $x \in \mathcal{F}$ heißen zulässige Punkte. x_i mit $i \in I$ heißen diskrete Variable, x_j mit $j \notin I$ heißen stetige Variable.

Das Problem (2.1) ist ein Problem mit einfachen Schranken (bound constrained), falls $m = 0$. Oft wird, wie hier, in der Literatur verlangt, dass für ein bound constrained problem auch $I = \emptyset$ gilt.

Eine Funktion f heißt **separabel**, falls sie von der Gestalt $f(x) = \sum_{k=1}^{n} f_k(x_k)$ (also als Summe eindimensionaler Funktionen) ist. Sie heißt **faktorabel**, falls sie als endlicher algebraischer Ausdruck in $(+, -, *, /,$[1] und Elementarfunktionen $\sin, \cos, \exp, \log, \arctan 2$,etc.) geschrieben werden kann. Ein Optimierungsproblem heißt separabel (faktorabel), falls alle f_i, F_i separabel (faktorabel) sind. Ein globales Optimierungsproblem heißt **DC (difference of convex)**, falls f und alle F_i als Differenz je zweier konvexer Funktionen geschrieben werden können.

2.2 Klasseneinteilung der Algorithmen zur Lösung globaler Optimierungsprobleme

Man kann die Menge der Algorithmen zur Lösung globaler Optimierungsprobleme in 4 verschiedene Klassen einteilen:

(i) Unvollständige Algorithmen (incomplete)
Diese Algorithmen verwenden clevere Heuristiken, um nach dem globalen Optimum zu suchen, haben jedoch keinerlei beweisbare Konvergenzeigenschaften und keinerlei Absicherung dagegen, in einem lokalen Minimum (oder gar keinem Minimum!) zu stoppen. Beispiele hierfür sind: genetische Algorithmen oder auch der sogenannte „Ameisenkolonie-Algorithmus“.

(ii) Asymptotisch vollständige Algorithmen (asymptotically complete)
Solch ein Algorithmus findet die globale Lösung mit Sicherheit oder wenigstens mit Wahrscheinlichkeit 1, wenn man ihm erlaubt, unendlich lange zu laufen. Der Algorithmus erlaubt zu keinem endlichen Zeitpunkt festzustellen, ob das globale Optimum bereits gefunden wurde, oder auch nur wie weit der Zielfunktionswert des globalen Optimums bereits angenähert wurde.
Beispiel hierfür ist das sog. „simulated annealing“ (besonders beliebt in der Wirtschaft).

(iii) Vollständige Verfahren (complete)
Ein vollständiges Verfahren erreicht das globale Optimum mit Sicherheit, vorausgesetzt es darf unendlich lange laufen und alle durchgeführten Rechnungen sind exakt (also ohne Rundungsfehler). Das Verfahren weiß zu jedem endlichen Zeitpunkt, ob ein approximatives globales Optimum erreicht wurde und wie weit dessen Zielfunktionswert höchstens vom Zielfunktionswert am globalen Optimum abweicht.

(iv) Rigorose Verfahren
Solch ein Algorithmus findet mit Sicherheit das globale Minimum innerhalb festgeschriebener Toleranz innerhalb endlicher Zeit trotz des Vorhandenseins von Rundungsfehlern. Ausgenommen sind numerisch degenerierte Probleme, bei denen die vorgeschriebenen Toleranzen überschritten werden.

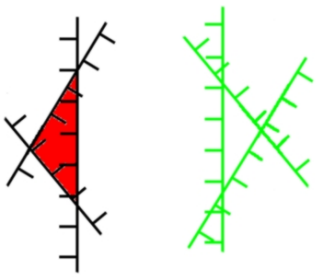


Abbildung 3: Diese Beispiel zeigt, wie Rundungsfehler die Sache numerisch degeneriert und unlösbar machen: Der rote Bereich zeigt die Menge der eigentlich zulässigen Punkte an. Mit Rundungsfehler gibts es aber auf einmal keinen zulässigen Punkt mehr (Zeichnung in grün).

3 Unvollständige und asymptotisch vollständige Verfahren

Alle bekannten unvollständigen und asymptotisch vollständigen Verfahren wurden ursprünglich entwickelt, um Probleme mit Schrankennebenbedingungen zu lösen:

$$\begin{aligned} \min \quad & f(x) \\ s.t. \quad & x \in \mathbf{x} \end{aligned} \tag{3.1}$$

Die einfachste Methode, eigentlich die erste die erfunden wurde, basiert auf dem numerischem Grundprinzip: „Ist etwas zu komplex, um es zu bearbeiten, versuche es mit Zufallsmethoden“.

3.1 Multiple random start

Erzeuge eine Zufallsfolge auf $\mathbf{x}$ gleichverteilter Punkte und verwende jedes Folgenglied als Startwert einer lokalen Optimierung. Generiert der Algorithmus Startwerte, die dicht liegen im zulässigen Bereich, dann ist das Verfahren asymptotisch vollständig. Die meisten heu-

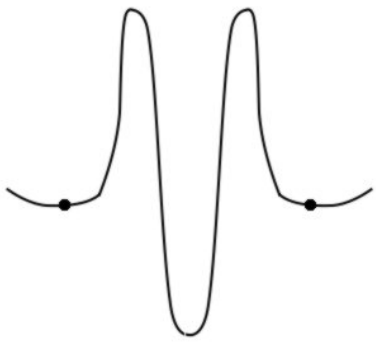


Abbildung 4: Dieses Problem ist ungut für multiple random start da die Chance klein ist, in das „tiefe Loch“ hineinzufallen

te gebräuchlichen asymptotischen Verfahren sind im Prinzip Beschleunigungen von Multiple random start. Die Überlegungen, die zur Erfindung der meisten unvollständigen Algorithmen geführt haben, sind aus der Natur entlehnt.
Üblicherweise werden sie in Anwendungen eingesetzt, wo gute lokale Minima aber nicht notwendigerweise das globale Minimum benötigt werden (zum Beispiel in der Wirtschaft).

3.2 Glättungsmethoden (Homotopie-Methoden)

Als Motivation für diesen Abschnitt versuchen wir, in den Bergen den Berg auszumachen, der den höchsten Gipfel hat. In den Bergen findet man die höchsten Gipfel aus großer Entfernung leichter, also dann, wenn man nur die grobe Form der Berge betrachtet. In Homotopieverfahren werden die „mikroskopischen“ Variationen der Zielfunktion zugunsten der „makroskopischen“ Variationen vernachlässigt.
Mathematisch etwas präziser formuliert: Betrachten wir (3.1) und konstruieren uns eine Homotopie von f auf eine einfachere Funktion g.

$$H(x,t); x \in \mathbb{R}^n, t \in [0,1] \text{ stetig (eventuell auch differenzierbar)}$$
$$H(x,0) = f(x)$$
$$H(x,1) = g(x)$$

Beispiel 3.1. Ist f DC, $f(x) = g(x) - h(x)$ mit g,h konvex: $H(x,t) := g(x) - (1-t)h(x)$
Löse für eine geeignete Nullfolge $(t_n)_{n\in\mathbb{N}}$ sukzessive die Probleme

$$\begin{aligned} &\min H(x, t_n) \\ &s.t.\ x \in \mathbf{x} \end{aligned} \tag{3.2}$$

lokal, wobei der Startwert für $(HO_n)(=(3.2))$ die Lösung von (HO_{n-1}) ist. Das Problem (HO_0) für $t_0 = 1$ kann von einem beliebigen Startwert aus gelöst werden.

Gibt es Nebenbedingungen, dann kann man Homotopie-Methoden auch einsetzen:

$$\begin{aligned} &\min x_N \\ &s.t.\ F(x) = 0 \\ &x \in \mathbf{x} \end{aligned} \tag{3.3}$$

ist eine andere, äquivalente Form für (1.1). Wir definieren $H(x,t)$ als Homotopie zwischen F und G, der linearen Approximation von F. Dann lösen wir die Probleme

$$\begin{aligned} &\min x_N \\ &s.t. H(x,t_n) \in [-\alpha t_n, \alpha t_n], \alpha \in \mathbb{R}^+ \end{aligned}$$

Homotopie-Methoden sind unvollständig. Sie liefern meist gute lokale Minima mit relativ wenigen Funktionsauswertungen, haben aber kaum beweisbare Konvergenzeigenschaften.

3.3 Responce-Surface-Methoden

Diese Verfahren sind speziell entwickelt, um Probleme mit aufwändigen Funktionen zu lösen (zum Beispiel den Auftrieb eines Schiffsrumpfes berechnen). Die Idee ist, eine Folge von so genannten Surrogatfunktionen zu erzeugen, die die bereits bekannten Funktionswerte von f möglichst gut approximieren, im Idealfall interpolieren und zur Theorie passen, aus der f entwickelt wurde. Anstelle der Originalfunktion f wird die dann die Surrogatfunktion ϕ_k über dem zulässigen Bereich optimiert:

$$\begin{aligned} &\min \phi_k \\ &s.t.\ x \in \mathbf{x} \end{aligned} \tag{3.4}$$

($F(x) \in \mathbf{F}$ ist möglich, falls F leicht auszuwerten ist).
An der Lösung x_k von (3.4) wertet man dann f das nächste Mal aus. Diesen neuen Funktionswert verwendet man, um ϕ_{k+1} zu bilden. Der Vorgang wird so lange iteriert bis
$\phi_m(x_m) \approx f(x_m)$ gilt. Das ist ein unvollständiger Algorithmus. Erzeugt man zusätzlich in jeder Iteration Punkte so, dass die Punktmenge bei unendlicher Laufzeit dicht im zulässigen Bereich liegt, ergibt das einen asymptotisch vollständigen Algorithmus.

3.4 Simulated Annealing

Der Bereich der Algorithmen, die Simulated Annealing verwenden, wurde schon in ungefähr 5000-10000 wissenschaftlichen Arbeiten abgehandelt. Man kann aber zusammenfassend sagen: Alle Simulated Annealing Verfahren erzeugen Folgen x_k die mit sehr hoher Wahrscheinlichkeit absteigen, manchmal aber auch nicht. Schritte, bei denen der Funktionswert ansteigt, werden mit einer Wahrscheinlichkeit zugelassen, die mit Hilfe der Boltzmann-Verteilung bestimmt wird. Die Dichte dieser Verteilung lautet: $e^{-\frac{K}{T}}$
$K \ldots$ ist eine Konstante (in der Physik die Boltzmann-Konstante)
$T \ldots$ ist die Temperatur
Die Hintergrundidee dieses Verfahrens ist Folgende: Erhitzen eines Stoffes und langsames Abkühlen führt zu einer sehr regelmäßigen Kristallstruktur. Die Energie einer Kristallstruktur wird mit einer Potentialstruktur beschrieben. Der perfekte Kristall entspricht dem globalen Minimum der Potentialfunktion. Die verschiedenen Simulated Annealing Verfahren unter-

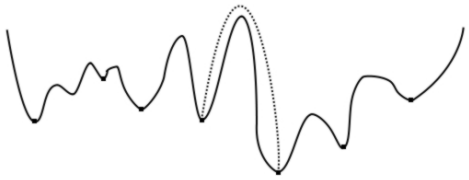


Abbildung 5: Diese Grafik veranschaulicht den Fall, dass der Algorithmus ein lokales Minimum gefunden hat, aber trotzdem über den „Hügel“ zu einem besseren Minimum kommt, indem er den Bereich um den Punkt untersucht.

scheiden sich dadurch, wie die neuen Punkte erzeugt werden und mit welchen Wahrscheinlichkeiten Punkte mit höheren Funktionswerten akzeptiert werden. Im Originalverfahren werden die Punkte gleichverteilt erzeugt und x_{neu} wird akzeptiert, wenn

$$f(x_{neu}) - f(x_{alt}) < e^{-\frac{K}{T}}$$

gilt. Die Temperatur T wird im Verlauf des Algorithmus von Startwert T_0 aus in jedem Schritt verkleinert. Man kann beweisen, dass Simulated Annealing Verfahren asymptotisch vollständig sind. Das Originalverfahren ist übrigens furchtbar langsam. Simulated Annealing Verfahren sind die am meisten verbreiteten Verfahren zur Globalen Optimierung. Es gibt SEHR viele Varianten, die alle typischerweise in ihrer Performance stark von den Parametern abhängen. Die „beste“ Parameterwahl muss meist für jede Problemklasse getrennt gefunden werden.
Simulated Annealing Verfahren sind üblicherweise sehr schlecht, wenn andere als Schranken-Nebenbedingungen existieren. Grund sind die großen Unterschiede in den Funktionswerten bei Straffunktionen. Das verleitet Simulated Annealing Verfahren dazu, einfach zulässige Punkte zu finden und dann aufzuhören.

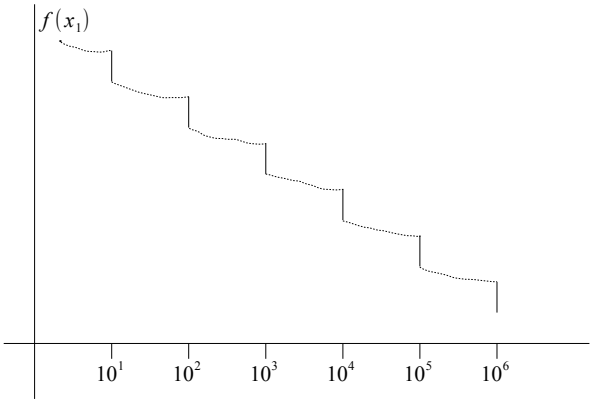


Abbildung 6: Auf der x-Achse sind die Anzahl an Funktionsauswertungen logarithmisch und auf der y-Achse die Funktionswerte dargestellt. Das heißt also, dass man beim Originalverfahren ca 10x so viele Auswertungen braucht für einen echt besseren Punkt.

3.5 Genetische Algorithmen

Genetische Algorithmen sind die wohl populärsten Algorithmen in der Globalen Optimierung: Es gibt mehr als 1000 Arbeiten darüber und unzählige Software-Pakete. Die Idee dieser Algorithmen ist Folgende: Die Punkte bilden eine „Population“, die nach dem Prinzip „Ändern durch Mutation“ oder „Ändern durch Sex“ und Überleben der Fähigsten „lebt“. Es gibt eine Populationsgröße N (zu Beginn werden zufällige N Punkte erzeugt). Ein gewisser Prozentsatz $p \in (0, 1)$ wird festgelegt und zu Beginn jeder Iteration „sterben“ pN Individuen, abhängig von ihren Funktionswerten. Niedrige Werte haben höhere Überlebenschancen als höhere Funktionswerte. Mit Hilfe von „Mutationen“, „cross-overs“ (Sex) und anderen Methoden („Spontangeburten“) wird die Population wieder auf „N“ Individuen aufgefüllt.
Statistisch untersucht sind Genetische Algorithmen besser als Multiple-Random-Start Algorithmen, wenn man am Ende vom besten Punkt aus noch eine lokale Optimierung durchführt (warum das so ist ist noch nicht geklärt). Genetische Algorithmen sind unvollständige Algorithmen. Genetische Algorithmen erzeugen meist robuste Minima, die auch kleine Parameteränderungen in der Zielfunktion überleben. Beim Auftreten von Nebenbedingungen sind sie meist auch nicht sehr gut.
Eine wichtige Idee, die Genetische Algorithmen beschleunigt, ist „parallele Evolution“(MEHR INFOS EV...). Für die Lösung des Problems (3.1) ist zur Zeit kein Verfahren mit paralleler Evolution bekannt.

4 Transformation allgemeiner Probleme auf einfache Schranken

In diesem Abschnitt wollen wir uns kurz folgendem Punkt widmen: Das Problem ist, (1.1) in geeigneter Form auf

$$\begin{aligned} &\min g(y) \\ &s.t.\ y \in \mathbf{y} \end{aligned}$$

zu transformieren. Dazu haben wir in der Vorlesung Optimierung und Variationsrechnung schon einige Wege kennengelernt, hier nochmal eine kurze Zusammenfassung:

4.1 Barriere- und Straffunktionen

Bei dieser Form der Transformation betrachten wir statt der eigentlichen Zielfunktion f folgende Funktion: $g(y) = f(y) + \phi(\sigma, F(y), \mathbf{F})$, σ ist hierbei der Strafparameter.
Bei vielen unvollständigen Algorithmen treten Konvergenzprobleme auf, wenn Straffunktionen zur Entfernung der Nebenbedingungen verwendet werden. Die Steilheit und Enge der Täler

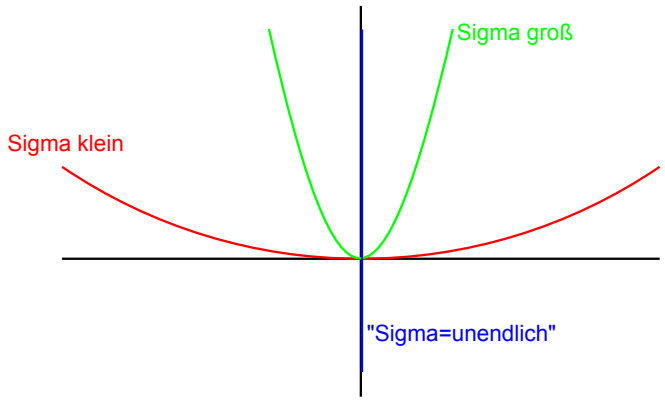


Abbildung 7: Wie die Täler aussehen, je nachdem welches σ man gewählt hat.

bei großem Strafparameter behindert Simulated Annealing und Genetische Algorithmen sowie alle Verfahren, die durch zufällige Suche Punkte erzeugen.

Satz 4.1. *Soft Optimality Theorem*
Seien $\Delta, \underline{\sigma_i}, \overline{\sigma_i}, f_0 \in \mathbb{R}$. *Setzt man*

$$q(x) := \frac{f(x) - f_0}{\Delta + |f(x) - f_0|}$$

$$\sigma_i(x) := \begin{cases} \frac{F_i(x) - \underline{F_i}}{\underline{\sigma_i}} & , \text{falls } F_i(x) \leq \underline{F_i} \\ \frac{F_i(x) - \overline{F_i}}{\overline{\sigma_i}} & , \text{falls } F_i(x) \geq \overline{F_i} \\ 0 & \text{sonst} \end{cases}$$

$$r(x) := \frac{2\sum_i \sigma_i^2(x)}{1 + \sum_i \sigma_i^2(x)}$$

dann ist die Funktion $f_{merit}(x) := q(x) + r(x)$ *beschränkt mit Wertebereich* $(-1, 3)$ *und das globale Minimum* $\hat{x}$ *von* f_{merit} *in* $\boldsymbol{x}$ *erfüllt entweder*

$$F(\hat{x}) \in \left[\underline{F_i} - \underline{\sigma_i}, \overline{F_i} + \overline{\sigma_i}\right] \forall\, i$$
$$f(\hat{x}) \leq \min\left\{f(x) | F(x) \in \boldsymbol{F}, x \in \boldsymbol{x}\right\}$$

oder eine der beiden folgenden Bedingungen ist erfüllt:

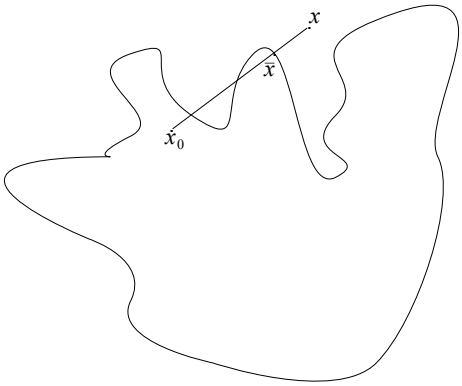
- $\mathcal{F} = \emptyset$
- $f_0 < \min\left\{f(x) | F(x) \in \boldsymbol{F}, x \in \boldsymbol{x}\right\}$

Beweis. Ohne Beweis. □

Man könnte also sagen, dass man die Nebenbedingungen mithilfe dieses Theorems etwas „aufweichen“ kann. Nachteil dieses Vorgehens: Auch Gleichungsnebenbedingungen werden aufgeweicht und im Allgemeinen in der Lösung nicht erfüllt.
Lineare Gleichungen hingegen sind kein Problem, da sie wegsubstituiert werden können. Im Notfall kann man von der Lösung aus eine lokale Optimierung starten, die danach die Gleichungsnebenbedingungen erzwingt.

4.2 Projektionsverfahren

Die in diesem Abschnitt beschrieben Verfahren wurden von Janosch Pinter entwickelt. Die Idee dahinter is folgende: Man projeziere unzulässige Punkte in den zulässigen Bereich. Die Grundannahme ist: Ein Punkt x_0 ist im relativen Inneren von $\mathcal{F}$ bekannt (funktioniert also nur bei linearen Gleichungen und allgemeinen nichtlinearen Ungleichungen). Man definiert sich eine neue Zielfunktion $\overline{f}(x) := f(\overline{x}) + \gamma \left\|\overline{x} - x\right\|^2$ wobei $\gamma > 0$ und beliebig sein kann, $\overline{x} := \lambda x_0 + (1-\lambda)x$ mit $\lambda \in [0,1]$ und λ minimal, sodass $\overline{x} \in \mathcal{F}$. Für lineare und quadratische



Nebenbedingungen ist die Berechnung von $\overline{x}$ leicht. Für allgemeine nichtlineare Nebenbedingungen kann die Berechnung aufwändig werden (etwa für $a^T x \leq \alpha \Rightarrow \lambda \geq \frac{a^T x - \alpha}{a^T x - a^T x_0}$, falls Zähler und Nenner > 0). Für allgemeine nichtlineare Nebenbedingungen kann man ein eindimensionales Nullstellenverfahren zur Bestimmung von λ verwenden. Im zulässigen Bereich $\mathcal{F}$ gilt $f = \overline{f}$. Außerhalb von $\mathcal{F}$ wächst $\overline{f}$ an, sofern γ wesentlich größer ist als der betragsmäßig größte Eigenwert der Hesse-Matrix von f. Die Lösung von

$$\min \overline{f}(x)$$
$$s.t.\ x \in \mathbf{x}$$

ist dann Lösung des Originalproblems. Ein Problem ist, dass $\overline{f}$ im Allgemeinen nicht $\mathcal{C}^2$ sondern nur Lipschitz-stetig ist. Das macht das lokale Optimieren von $\overline{f}$ schwierig. Verwendet man $\overline{f}$ in unvollständigen oder asymptotisch vollständigen Verfahren, wie Simulated Annealing oder Genetische Algorithmen, dann wiegt der Nachteil der Lipschitz-Stetigkeit nicht so schwer.

5 Vollständige und rigorose Verfahren

5.1 Reine Branching-Methoden

Reine Branching-Methoden sind auch unter reinen Teilungs-Methoden bekannt. Das sind die einzigen Verfahren, die noch funktionieren, wenn von den Funktionen nur lokale Informationen $(f(x), F(x), \nabla f(x), \nabla F(x), \nabla^2 f(x), \nabla^2 F(x), ...)$ bekannt sind (also Punktauswertungen).

Satz 5.1. *Dichheitstheorem von Törn und Zinlinskas*
Jedes Verfahren, das nur lokale Informationen verwendet und für jede $\mathcal{C}^\infty$-Funktion f gegen ein lokales Minimum von f in einem zulässigen Bereich C konvergiert, muss eine Folge $(x_i)_{i\in\mathbb{N}}$ von Punkten erzeugen, die dicht in C liegt. Es gilt: $\underline{\lim}_{l\to\infty} f(x_l) = \min\{f(x)|x \in C\}$.

Beweis. Angenommen, $(x_i)_i$ liegt nicht dicht in C. Dann gibt es eine offene Menge $U \subseteq C$, in der kein Element von x_i liegt. O.B.d.A. ist $U = B_r(x_0), x_0 \in C, r > 0$.
Sei g $\mathcal{C}^\infty$ mit *supp* $g \subseteq B_r(x_0), g(x_0) = 1$. Dann sind für das Verfahren die Funktionen $f - \lambda g \ \forall\ \lambda \in \mathbb{R}$ ununterscheidbar und für λ groß genug liegt das globale Minimum von $f - \lambda g$ in $B_r(x_0)$. Das widerspricht der Konvergenz des Verfahrens und
$\underline{\lim}_{l\to\infty} f(x_l) = \min\{f(x)|x \in C\}$. □

Definition 5.2. Ein Verfahren, das wie im obigen Theorem beschrieben eine dichte Folge von Punkten erzeugt, heißt **konvergent**.

Im Folgenden werden jetzt 5 verschiedene Branching-Methoden vorgestellt:

(i) Vollständige Gittersuche
Die Idee ist sehr einfach: Wir teilen den Suchbereich $\mathbf{x}$ in immer kleinere Teile durch Halbierung in allen Dimensionen. Nach der n-ten Unterteilung ist die Kantenlänge der einzelnen Boxen, der Gitterabstand, $\frac{1}{2^n}$ Mal des Originalabstandes. Das Verfahren ist offensichtlich konvergent. Man kann den Abstand zum globalen Optimum durch den Gitterabstand in jedem Schritt abschätzen. Der Aufwand ist exponentiell.

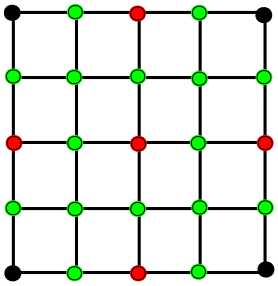


Abbildung 8: Dies ist eine mögliche Unterteilung der Box. Die schwarzen Punkte sind dabei gegeben im „0." Schritt, die roten entstehen im 1. Schritt und die grünen im 2. Schritt.

(ii) Andere Verfahren, die nur lokale Informationen verwenden
Alle diese Methoden basieren auf dem Branching-Prinzip

$$\begin{matrix}\min\ f(x) \\ s.t.\ x \in \mathbf{x}\end{matrix} \Leftrightarrow \min \begin{pmatrix} \min\ f(x) & , & \min\ f(x) \\ s.t.\ x \in \mathbf{x}_1 & & s.t.\ x \in \mathbf{x}_2 \end{pmatrix}$$

falls $\mathbf{x}_1 \cup \mathbf{x}_2 = \mathbf{x}$.

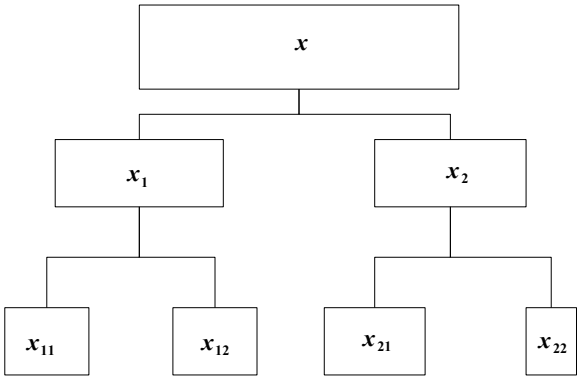


Abbildung 9: Beispiel eines Suchbaumes, der die Unterteilungsstruktur darstellt.

Die Unterteilungsstruktur ergibt in natürlicher Weise einen Baum-den Suchbaum. Nun können wir auch eine exaktere Definition des Begriffs **Branching** geben: Teilung des Gebiets $\mathbf{x}$ in endlich viele Teile T_i, sodass $\bigcup_{i=1}^{n} T_i = \mathbf{x}$ und $T_i \cap T_j$ hat Maß 0 für $i \neq j$ (besteht also nur aus Rand). Danach wird jedes Problem auf den Gebieten T_i getrennt

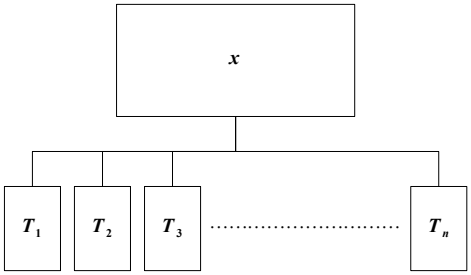


Abbildung 10: Darstellung der Aufteilung der Gebietes in die Gebiete T_i.

betrachtet. Wenn man rekursiv das Verfahren auf die T_i anwendet ergibt sich ein Baum von Problemen, dessen Wurzel zum Gebiet $\mathbf{x}$ gehört und dessen Blätter Gebiete B_k haben mit der Eigenschaft $\bigcup_{k=1}^{N} B_k = \mathbf{x}$ und $B_k \cap B_l$ hat Maß 0 für $k \neq l$.
Wird in jedem Gebiet mindestens ein Punkt ausgewertet und ist sichergestellt, dass das Maß der Blätter mit Fortdauer des Algorithmus gegen 0 konvergiert, dann ist das entstehende Verfahren konvergent. Alle Verfahren die nur diesem Prinzip folgen heißen reine Branching-Verfahren.
Das wesentliche Kriterium, das die Geschwindigkeit eines reinen Branching-Verfahrens

bestimmt, ist eine geschickte Balance zwischen lokaler Suche (dient dazu, möglichst rasch lokale Minima zu entdecken) und globaler Suche (stellt sicher, dass das Maß der Blätter gegen 0 konvergiert).

(iii) DIRECT
DIRECT wurde ursprünglich entwickelt, um Probleme mit Schrankennebenbedingungen zu behandeln:

$$\begin{aligned} &\min f(x) \\ &s.t.\ x \in \mathbf{x} \end{aligned}$$

Die Idee dieses Verfahrens ist folgende: DIRECT erzeugt eine Liste von Teilboxen B_i, die jeweils durch ein Paar (f_i, v_i) bewertet werden, wobei $v = vol(B_i)$ und f der Funktionswert am Mittelpunkt von B_i ist ($\bigcup_{i=1}^{n} B_i = \mathbf{x}$). Diese Liste wird nun nach minimalen Elementen durchsucht bezüglich der folgenden Ordnungsrelation:
$(v, f) < (v', f') :\Leftrightarrow (v > v') \wedge (f < f')$. Alle minimalen Elemente bezüglich dieser Halbordnung werden in der Liste gesucht. Alle dazugehörigen B_i werden in drei Teile geteilt an ihrer längsten Seitenkante.

Abbildung 11: Veranschaulichung des geteilten B_i, wobei „alt“ in der Mitte den schon gegebenen Funktionswert darstellt und bei „neu“ werden die neuen Funktionswerte bestimmt.

Das benötigt zwei neue Funktionsauswertungen (für 3 neue Boxen, da man eine Auswertung ja schon hat). In jedem Schritt wird das maximale Volumen verkleinert. Daher ist das Verfahren konvergent.

(iv) MCS (Multilevel Coordinate Search)
Dieses Verfahren wurde von Huyer und Neumaier entwickelt und ist auch in der NAG (Numcerical Analysis Group)-library zu finden. Dieses Verfahren behandelt folgende Probleme:

$$\begin{aligned} &\min f(x) \\ &s.t.\ x \in \mathbf{x} \end{aligned}$$

Jeder Box wird nach Anzahl der Unterteilungen, durch die sie entstanden ist, ein Level l zugewiesen und anschließend eine Ordnungsrelation eingeführt:
$(l, f) < (l', f') :\Leftrightarrow f < f'$ und $l < l'$. Von den minimalen Elementen aus wird eine lokale Optimierung gestartet in jedem Sweep (Liniensuche und quadratische Optimierung mit Schrankennebenbedigungen). Die Boxen werden geteilt, indem die bekannten Punkte eindimensional quadratisch interpoliert werden und am Minimum geteilt wird.

(v) LGO (Lipschitz Global Optimization)
Diese Methode stammt von Janos Pintér und behandelt Probleme folgender Natur:

$$\begin{aligned} &\min f(x) \\ &s.t.\ F(x) \in \mathbf{F} \\ &x \in \mathbf{x} \end{aligned} \tag{5.1}$$

Dabei ist F linear oder quadratisch konvex. LGO verwendet die Projektionsmethode aus 4.2, um eine Lipschitz-stetige Funktion $\overline{f}$ zu erzeugen und löst dann

$$\begin{aligned} &\min \overline{f}(x) \\ &s.t.\ x \in \mathbf{x} \end{aligned}$$

Um den Algorithmus konvergent zu machen muss auch dieses Programm branchen. Dabei werden Abschätzungen an die Lipschitz-konstanten von $\overline{f}$ verwendet:

$$L \geq \max_{k,l} \frac{|\overline{f}(x_k) - \overline{f}(x_l)|}{|x_k - x_l|} \text{ für } x_k \neq x_l$$

wobei die x_j zuvor bestimmte Funktionswerte sind. $|\overline{f}(x) - \overline{f}(y)| \leq L \|x - y\|$ gilt, falls L die Lipschitz-konstante ist. $\forall\, x, x_0 \in \mathbf{x}_0 : f(x) \geq \overline{f}(x_0) - L \|x - x_0\|$. Man kann darauf verzichten, alle jene Boxen zu betrachten, in denen
$\overline{f}(x_0) - L \|x - x_0\| \geq f(x^*) \,\forall\, x \in \mathbf{x}_0, x_0 \in \mathbf{x}_0$ für irgendeinen Punkt $x^* \in \mathbf{x}$.

5.2 Das Branch and Bound Prinzip

Die Idee die dahinter steckt ist die, dass man die reinen Branching-Methoden verbessert. Man betrachtet wieder Probleme folgender Art:

$$\begin{aligned} &\min f(x) \\ &s.t.\ x \in \mathbf{x} \end{aligned}$$

Beim Pure Branching wird nun $\mathbf{x}$ immer weiter unterteilt, siehe Abbildung 12.
Ohne Beschränkung der Allgemeinheit haben wir eine lokale Optimierung durchgeführt und kennen daher ein lokales Optimum x^* und dazu $f(x^*)$ (Der Vollständigkeit halber sei an dieser Stelle erwähnt, dass es eigentlich reichen würde einen zulässigen Punkt zu verwenden, ein lokales Minimum für die weiteren Überlegungen aber angenehmer in der Handhabung ist). Im Folgenden werden wir nun zwei Annahmen tätigen:

Annahme 5.3. *Die erste Annahme ist die, dass wir auch globale Informationen von f kennen, zum Beispiel: Für alle $\boldsymbol{y} \subseteq \boldsymbol{x}$ können wir $\boldsymbol{x}(\boldsymbol{y})$ bestimmen mit $\boldsymbol{x}(\boldsymbol{y}) \supseteq \{f(x) | x \in \boldsymbol{y}\}$.*

Annahme 5.3 bezeichnet man auch als Range- oder Wertebereichsabschätzung für f. Jetzt wissen wir, dass ein globales Optimum $\hat{x}$ $f(\hat{x}) \leq f(x^*)$ erfüllt. Daraus folgt aber unmittelbar:

Lemma 5.4. *Gilt $\underline{z(\boldsymbol{y})} > f(x^*)$, dann enthält $\boldsymbol{y}$ kein globales Optimum.*

Beweis. Ohne Beweis. □

Der Schritt des Beweisens mit Lemma 5.4 oder ähnlichen Aussagen heißt *Bounding*. Gilt für ein Blatt B_i und das zugehörige Gebiet $\mathbf{x}_i$, dass $\hat{x} \notin \mathbf{x}_i$ liegen kann, so kann man B_i aus dem Suchbaum streichen. Sind alle Kinder eines Knotens N des Suchpfades gestrichen, so kann man auch N streichen.

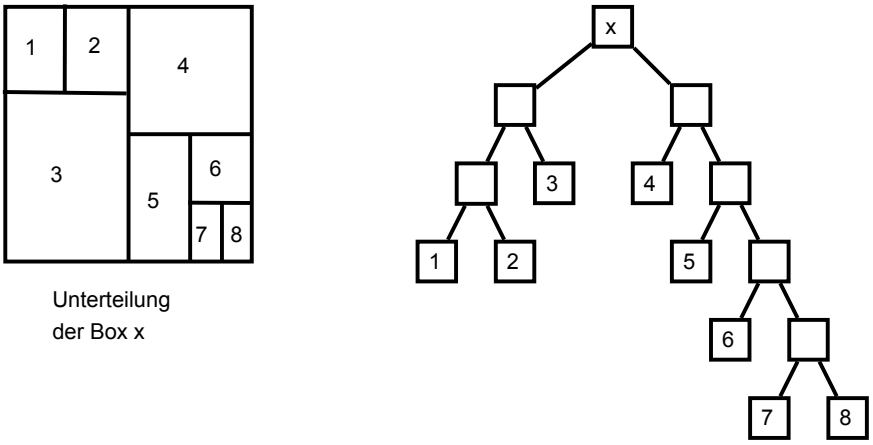


Abbildung 12: Links die Unterteilung von $\mathbf{x}$ und rechts der dazu passende Suchbaum, 1 bis 8 . . . Blätter im Suchbaum.

5.3 Das Branch and Reduce Prinzip

Was passiert in Branch and Bound, wenn $\underline{z(\mathbf{y})} \leq f(x^*)$? Antwort: Nichts spezielles, es muss geteilt werden (und das ist ja schlecht, denn dann muss man branchen und das verursacht ja den exponentiellen Rechenaufwand!). Wenn $\overline{z(\mathbf{y})} < f(x^*)$ (und $\mathbf{y}$ einen zulässigen Punkt enthält-ist bei 3.1 immer der Fall), dann ist x^* nicht das globale Minimum und es lohnt sich, in $\mathbf{y}$ nach einem besseren Punkt zu suchen.
Gilt $\overline{z(\mathbf{y})} \geq f(x^*)$, dann wissen wir immerhin: $f(\hat{x}) \in \left[\underline{z(\mathbf{y})}, f(x^*)\right]$, falls $\hat{x} \in \mathbf{y}$ gilt.

Annahme 5.5. *Die zweite Annahme die wir machen ist die, dass wir eine Methode haben, die zu jeder Box und jedem Intervall $\boldsymbol{w}$ eine Box $\widetilde{\boldsymbol{y}}$ bestimmt mit*
$\boldsymbol{y} \supseteq \widetilde{\boldsymbol{y}} \supseteq \{x \in \boldsymbol{y} | x \text{ ist zulässig und } f(x) \in \boldsymbol{w}\}$.

Daraus können wir sofort folgenden Schluss ziehen:

Lemma 5.6. *Gilt für ein globales Optimum $\hat{x} \in \boldsymbol{y}$, dann ist $\hat{x} \in \widetilde{\boldsymbol{y}}$ für $\boldsymbol{w} = \left[\underline{z(\boldsymbol{y})}, f(x^*)\right]$.*

Beweis. Ohne Beweis. □

Lemma 5.4 ist eine besondere Form des Lemmas 5.6: Gilt $\hat{x} \in \mathbf{y}$, dann ist $\hat{x} \in \emptyset$. Sei B_i ein Blatt des Suchbaumes mit Gebiet $\mathbf{x}_i$. Dann dürfen wir B_i durch $\tilde{B}_i$ ersetzen mit Gebiet $\tilde{\mathbf{x}}_i$. Ist $\tilde{\mathbf{x}}_i = \emptyset$, so dürfen wir $\tilde{B}_i$ streichen. So ein Schritt heißt Reducing-Schritt.

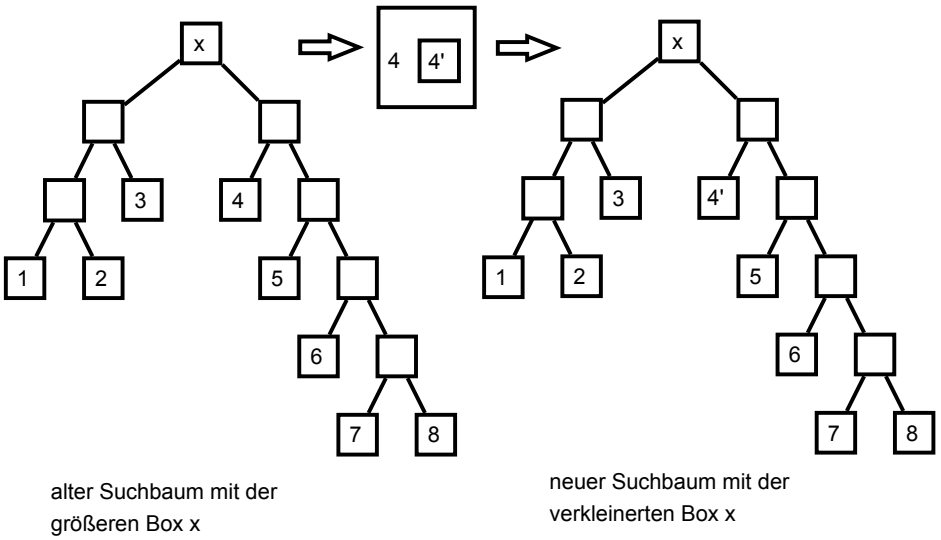


Abbildung 13: Wenn das globale Minimum nur in $4 \setminus 4'$ liegen kann, so reicht es, nur noch den Bereich $4'$ zu untersuchen und kann diese Tatsache dann auch dementsprechend im Baum adaptieren.

6 Intervallmethoden

6.1 Grundlagen

In diesem Abschnitt werden wir nun der folgenden Idee nachgehen: Wir weiten die Analysis und Lineare Algebra auf eine neue Grundmenge von „Zahlen“ zum Zwecke der automatischen Abschätzung aus.

Definition 6.1. (i) $\mathcal{I}\mathbb{R} := \{[\underline{a}, \overline{a}] \,|\underline{a}, \overline{a} \in \mathbb{R}^*, \underline{a} \subseteq \overline{a}\}, \mathbb{R}^* = \mathbb{R} \cup \{-\infty, \infty\}$
(Semantisch : $[\underline{a}, \overline{a}] := \{a \in \mathbb{R} | \underline{a} \subseteq a \subseteq \overline{a}\}$)

(ii) Für $\circ : \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}, \circ \in \left\{+, -, *, /, ^1\right\}$ definieren wir $\circ : \mathcal{I}\mathbb{R} \times \mathcal{I}\mathbb{R} \mapsto \mathcal{I}\mathbb{R}$ durch $\mathbf{a} \circ \mathbf{b} := \square \{a \circ b | a \in \mathbf{a}, b \in \mathbf{b}\}$. $\square$ ist die Intervallhülle, also das kleinste Intervall.

(iii) $\inf \mathbf{a} := \underline{a}; \sup \mathbf{a} := \overline{a}; S \subseteq \mathbb{R} : \square S := \bigcap_{\mathbf{x} \in \mathcal{I}\mathbb{R}, S \subseteq \mathbf{x}} \mathbf{x}$

(iv) Für $\phi : \mathbb{R} \mapsto \mathbb{R}$ sei $\phi(\mathbf{a}) := \square \{\phi(a) | a \in \mathbf{a}, a \in Def(\phi)\}$

(v) $\mathbf{a} + \mathbf{b} = \left[\underline{a} + \underline{b}, \overline{a} + \overline{b}\right]$

(vi) $\mathbf{a} - \mathbf{b} = \left[\underline{a} - \underline{b}, \overline{a} - \overline{b}\right]$

(vii) $\mathbf{a} \cdot \mathbf{b} = \square \left\{\underline{a} \cdot \underline{b}, \underline{a} \cdot \overline{b}, \overline{a} \cdot \underline{b}, \overline{a} \cdot \overline{b}\right\}$

(viii) $\mathbf{a}/\mathbf{b} = \square \left\{\underline{a}/\underline{b}, \underline{a}/\overline{b}, \overline{a}/\underline{b}, \overline{a}/\overline{b}\right\}$ falls $0 \notin \mathbf{b}$.

(ix) $\mathbf{a}/\mathbf{b} = [-\infty, +\infty]$ falls $0 \in \mathbf{a}$ und $0 \in \mathbf{b}$, sonst ist es kompliziert.

(x) $\sqrt{\mathbf{a}} = \left[\sqrt{\underline{a}_+}, \sqrt{\overline{a}}\right]$, falls $\overline{a} \geq 0$ und $\emptyset$ sonst.

Nun ist folgender Sachverhalt zu berücksichtigen: $\sin([-2, 2]) = [-1, 1] \neq [\sin(-2), \sin(2)]$. Betrachtet man nun das Beispiel $f(x) = x^2$ so lässt sich diese Funktion analytisch gleichwertig schreiben als $f(x) = x \cdot x$. Für die Intervalle gilt aber zum Einen $f([-1, 1]) = [-1, 1]^2 = [0, 1]$ und zum Anderen $f\,[-1, 1] = [-1, 1] \cdot [-1, 1] = [-1, 1]$, das heißt also $[0, 1] \subset [-1, 1]$, man erhält also unterschiedliche Intervalle. Man nennt die Tatsache, dass $[0, 1] \subset [-1, 1]$ auch eine **Überschätzung**.
Wird eine Funktion durch einen arithmetischen Ausdruck beschrieben und werden in diesen Ausdruck Intervalle eingesetzt, dann ist das Ergebnis ein Intervall, das den Wertebereich der Funktion enthält. Das Ergebnis hängt vom arithmetischen Ausdruck ab. Als einfachstes Beispiel hierfür dient: $f(x) = 0 = x - x; f([\underline{x}, \overline{x}]) = [\underline{x}, \overline{x}] - [\underline{x}, \overline{x}] = [\underline{x} - \overline{x}, \overline{x} - \underline{x}] \supset [0]$ falls $\underline{x} \neq \overline{x}$. Um Überschätzungen zu vermeiden kann man sich an folgendem Resultat aus der Intervallanalysis halten:

Tatsache 6.2. *Wenn in einem Ausdruck jede Variable nur einmal vorkommt, wird durch Intervallrechnung genau der Wertebereich bestimmt. Es gibt also keine Überschätzung.*

Bezüglich der Distributivität gilt Folgendes: $\mathbf{a} \cdot (\mathbf{b} + \mathbf{c}) \subseteq \mathbf{a} \cdot \mathbf{b} + \mathbf{a} \cdot \mathbf{c}$, man bezeichnet dies auch als **Subdistributivität**.
Nun wird anhand eines Beispiels erklärt, wie man den Wertebereich bei Funktionen bestimmen kann:

Beispiel 6.3.

$$\begin{aligned} f(x,y) &= x^2 + 3xy + y^2 + 4x + 3y - 9, x \in [-1,1], y \in [0,2] \\ f([-1,1],[0,2]) &= [-1,1]^2 + 3 \cdot [-1,1] \cdot [0,2] + [0,2]^2 + 4 \cdot [-1,1] + 3 \cdot [0,2] - 9 = \\ &= [0,1] + 3 \cdot [-2,2] + [0,4] + [-4,4] + [0,6] - 9 = \\ &= [0 - 6 + 0 - 4 + 0 - 9, 1 + 6 + 4 + 4 + 6 - 9] = [-19,12] \end{aligned}$$

also $f(x,y) \in [-19,12] \; \forall \; x \in [-1,1], \forall \; y \in [0,2]$. Dies ist nun aller Wahrscheinlichkeit nach aber eine Überschätzung. Um nun das „richtige" Intervall zu finden kann man nun Folgendes probieren mit: $\nabla_f = \begin{pmatrix} 2x+3y+4 \\ 3x+2y+3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ das Minimum der Funktion aufzufinden. In unserem Fall liegt das Minimum ausserhalb des Definitionsbereiches, also bleibt noch folgende Möglichkeit:

- Betrachte die Gleichung für $y = 0 : f(x,0) = x^2 + 4x - 9$. Bei dieser Funktion liegt das Minimum auch ausserhalb des Definitionsbereiches, also bleiben 2 mögliche Werte für unser Intervall übrig: $f(-1,0) = -12, f(1,0) = -4$.
- Für $y = 2 : f(x,2) = x^2 + 10x + 1$ liegt das Minimum ebenfalls ausserhalb des Definitionsbereiches $\Rightarrow f(-1,2) = -8, f(1,2) = 12$.
- Für $x = -1 : f(-1,y) = y^2 - 12$ hat das Minimum bei $y = 0$, das liefert aber nur den hier schon bekannten Wert -12.
- Für $x = 1 : f(1,y) = y^2 + 6y - 4$ liegt das Minimum ebenfalls ausserhalb des Definitionsbereiches $\Rightarrow f(1,0) = -4, f(1,2) = 12$

Nun kann man sagen, dass der „echte Bereich" $[-12,12]$ ist.

Anhand des folgendes Beispiels werden noch zwei weitere Methoden behandelt, einen Bereich zu finden, der einmal überschätzt ist und einmal nicht überschätzt ist:

Beispiel 6.4. Betrachten wir für unser Beispiel die Funktion $f(x) = 3x^2 + 12x + 17, x \in [-1,1]$

- Methode der Ergänzung auf ein vollständiges Quadrat:
  $3x^2 + 12x + 17 = 3(x^2 + 4x + \frac{17}{3}) = 3(x+2)^2 + 5 = 3((x+2)^2 + \frac{5}{3})$.
  Wenn wir nun mit der ursprünglichen Funktion rechnen so erhalten wir:
  $3 \cdot \underbrace{[-1,1]^2}_{=[0,1]} + 12 \cdot [-1,1] + 17 = [5,32]$ während wir mit der ergänzten Funktion
  $3 \cdot \underbrace{(\underbrace{([-1,1] + 2)}_{=[1,3]})^2}_{=[1,9]} + \frac{5}{3}) = [8,32]$ erhalten.
- Verwendung des Horner-Schemas: $\underbrace{\underbrace{(3 \cdot [-1,1] + 12)}_{=[9,15]} \cdot [-1,1]}_{=[-15,15]} + 17 = [2,32]$

6.2 DAG-Repräsentationen globaler Optimierungsprobleme

Definition 6.5. Ein **gerichteter Multigraph** $\Gamma = (V, E, f)$ besteht aus einer endlichen Menge $V \neq \emptyset$ von Knoten (vertex), einer endlichen Menge E von Kanten (edge) und einer Abbildung $f : E \to V \times V$.
Für jede Kante $e \in E$ definieren wir die **Quelle** (source) von e durch $s(e) := pr_1 \circ f(e)$ und das **Ziel** (target) von e durch $t(e) := pr_2 \circ f(e)$. Eine Kante e mit $s(e) = t(e)$ heißt **Schleife** (loop).
Die Kanten e und e' heißen **vielfach**, falls $f(e) = f(e')$. Für jeden Knoten v ist die Menge der **In-Kanten** (in-edges) $E_i(v) := \{e \in E | t(e) = v\}$ und die Menge der **Out-Kanten** (out-edges) $E_o(v) := \{e \in E | s(e) = v\}$ definiert. Der **In-Grad** (in degree) eines Knoten v ist $indeg(v) := |E_i(v)|$ und der **Out-Grad** (out-degree) ist $outdeg(v) := |E_o(v)|$.
Ein Knoten $v \in V$ mit $indeg(v) = 0$ heißt **(lokale) Quelle des Graphen**, ein v mit $outdeg(v) = 0$ heißt **(lokale) Senke des Graphen** Γ.

Definition 6.6. Sei $\Gamma = (V, E, f)$ ein gerichteter Multigraph. Ein **gerichteter Pfad** ist eine Sequenz von Kanten $e_1, \ldots, e_n$ mit $t(e_i) = s(e_{i+1}) \ \forall \ i = 1, \ldots, n-1$. Wir sagen er **führt von** $v \in V$ **nach** $v' \in V$, falls $v = s(e_1)$ und $v' = t(e_n)$ gelten. Ein gerichteter Pfad heißt **geschlossen** oder ein **Kreis** (Zyklus (cycle)), falls er von v nach v führt. Ein Multigraph Γ heißt **azyklisch** (acyclic) falls er keinen Kreis enthält.

Proposition 6.7. *Ein azyklischer gerichteter Multigraph Γ enthält mindestens eine Quelle und mindestens eine Senke.*

Beweis. Indirekt: Angenommen, es gibt keine Senke. Dann gilt $\forall \ v \ \in V : outdeg(v) > 0$. Sei $v_0 \in V$. Dann gibt es $v_1 \in V$ mit $v_0 \neq v_1$ und eine Kante $e_1 \in E$ mit $t(e_1) = v_1$ und $s(e_1) = v_0$. Induktiv definieren wir Knoten v_n und Kanten e_n so, dass jeweils $t(e_n) = v_n$ und $s(e_n) = v_{n-1}$ gelten. Weil V nur endlich viele Knoten enthält gibt es k, l mit $v_k = v_l$. Dann ist der gerichtete Pfad $e_k, \ldots, e_{l-1}$ ein Zyklus. Widerspruch.
Analog beweist man, dass eine Quelle existieren muss. □

Definition 6.8. Ein **gerichteter Multigraph mit geordneten Kanten** ist ein Multigraph Γ mit einer Totalordnung $\leq$ auf E. Für solch einen geordneten Multigraphen werden in natürlicher Weise auch die Mengen $E_i(v)$ und $E_o(v) \ \forall \ v \in V$ in natürlicher Weise geordnete Mengen durch die induzierte Totalordnung.

Wir stellen faktorable globale Optimierungsprobleme durch einen gerichteten azyklischen Multigraphen mit geordneten Kanten dar:

Beispiel 6.9.

$$\begin{aligned} &\min 3x^2 + 4xy + 2y \\ &s.t.\ e^{3x^2+2z} \leq 4 \\ &\quad z - 3xy = 0 \end{aligned}$$

Man kann dieses Problem nun wie in Abbildung 14 darstellen. Allerdings ist diese Darstellung nicht eindeutig und benötigt noch Zusätze.

Für die Darstellung eines globalen Optimierungsproblems sind noch drei Zusätze nötig:

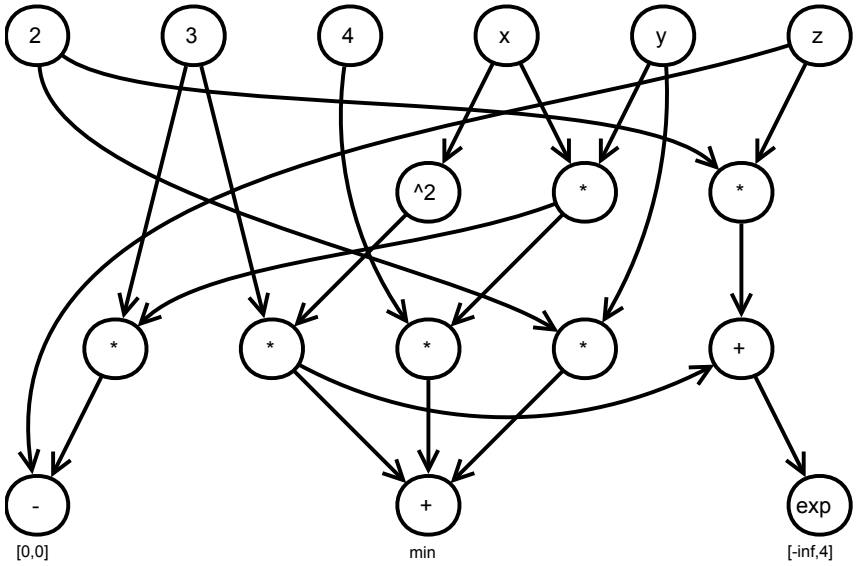


Abbildung 14: Dies ist eine mögliche (keineswegs eindeutige!) Darstellung mittels eines gerichteten Multigraphen. Allerdings enthält diese noch nicht alle Informationen um das Problem vollständig zu beschreiben.

(i) Eine endliche Menge $O = \{+, -, *, \div, \wedge, \exp, \sin, \dots\}$ von erlaubten elementaren Operationen zusammen mit einer Abbildung $o : V \to O \cup V_{\mathbb{N}} \cup \mathbb{R}$ wobei $V_{\mathbb{N}}$ die Menge der Variablen $\{v_1, v_2, \dots\}$ ist, zusammen mit den folgenden Verträglichkeitsbedingungen:

- $o(v) \in V_{\mathbb{N}} \cup \mathbb{R} \Rightarrow indeg(v) = 0$
- $o(v) \in \{-, \div, \wedge\} \Rightarrow indeg(v) = 2$
- $o(v) \in \{\exp, \sin, \cos, \log, \dots\} \Rightarrow indeg(v) = 1$
- $o(v) \notin V_{\mathbb{N}} \cup \mathbb{R} \Rightarrow indeg(v) > 0$

(ii) Eine Abbildung $b : V \to \mathcal{I}\mathbb{R}$, die die Schranken an die Knoten angibt (Im Graphen schreiben wir $b(v)$ unter (über) den Knoten falls $b(v) \neq [-\infty, \infty]$).

(iii) Eine Teilmenge $M \subseteq V$ der zu minimierenden Knoten. Ist $M = \emptyset$, dann ist das Problem ein CSP. Ist $|M| = 1$, dann liegt ein gewöhnliches Optimierungsproblem vor. Gilt nun $|M| \geq 2$, dann ist das Problem ein multiobjective-Problem.

Wir werden in Zukunft $(\Gamma, \geq, O, o, b, M)$ auch als Problem DAG bezeichnen.

Definition 6.10. Sei $\Gamma = (V, E, f)$ ein gerichteter azyklischer Multigraph. Wir sagen, dass $v \in V$ ein **Vater** (parent) von $v' \in V$ ist, falls es ein $e \in E$ gibt mit $s(e) = v$ und $t(e) = v'$. Dann heißt v' ein **Kind** (child) von v. v' heißt **Vorfahre** (ancestor) von v, falls ein gerichteter Pfad von v' nach v existiert und v heißt dann **Nachkomme** von v',

Proposition 6.11. *Für jeden gerichteten azyklischen Multigraphen $\Gamma = (V, E, f)$ existiert eine Totalordnung $\leq$ auf V mit der Eigenschaft: v ist Nachkomme von $v' \Rightarrow v < v'$.*

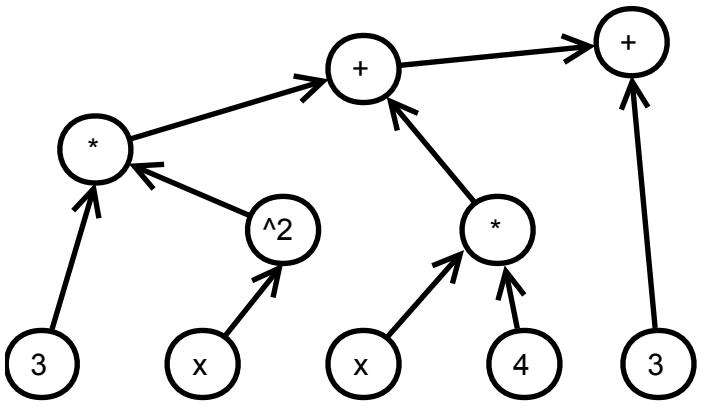
Beweis. Weil V endlich ist genügt es eine Bijektion $\varphi : V \to \{0, \dots, |V| - 1\}$ zu konstruieren mit der Eigenschaft, v ist Nachkomme von $v' \Rightarrow \varphi(v) < \varphi(v')$.
Dann gelte $v < v' :\Leftrightarrow \varphi(v) < \varphi(v')$. Beweis durch Algorithmus: □

```
function ORDER(&k, v)
    if v has been visited return; then
        for all children v' of v do
            order (k, v')
        end for
    end if

    φ(v) := k
    k = k + 1
    return
end function
k = 0
for all sinks v of Γ do
    order (k,v)
end for
```

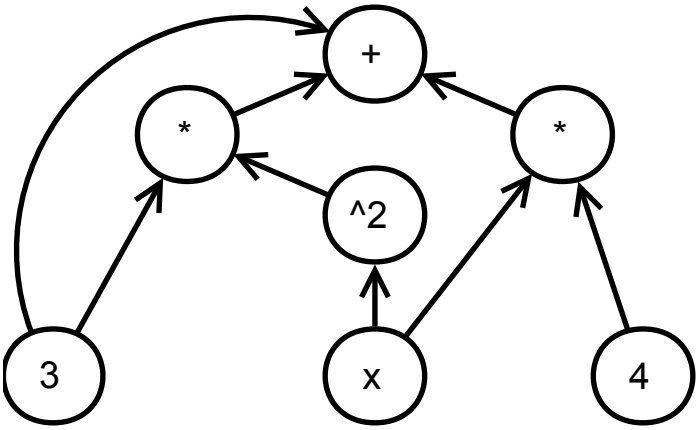
Bemerkung 6.12. Man kann den Algorithmus aus dem Beweis der vorhergehenden Proposition auch leicht modifizieren und damit einen Test programmieren, ob der gerichtete Multigraph azyklisch ist. Der Algorithmus sieht dann so wie der weiter unten angeführte Algorithmus aus.

```
function ORDER(&k, v)
    if visited(v)== 1 then
        error('nicht azyklisch')
    end if
    if visited(v)== 2 return; then
        visited(v)= 1
        for all children v' of v do
            order (k, v')
        end for
    end if

    φ(v) := k
    k = k + 1
    visited(v)= 2
    return
end function
k = 0
visited(v)= 0 ∀ v

for all sinks v of Γ do
    order (k,v)
end for
```

Beispiel 6.13. Wenn wir das Problem $3x^2 + 4x + 3$ betrachten, so lässt sich dieses äquivalent mit den folgenden zwei Graphen darstellen.



Dies ist die Darstellung des Problems als Baum mit 10 Knoten und 9 Kanten.



Das ist nun zum Vergleich zu obiger Abbildung eine Darstellung mit 7 Knoten und 8 Kanten. Diese Darstellung ist nun auch eine minimale Darstellung.

Der erste Graph ist offenbar „größer“ als der zweite Graph und ist ein Baum. Man kann die erste Abbildung „verkleinern“, indem man die Knoten 3,x und + „zusammenlegt“ und so zur zweiten Abbildung gelangt.
Der Vorteil der Baumstruktur liegt darin, dass der Baum im Gegensatz zum azyklischen gerichteten Multigraphen auch keinen Kreis enthält, wenn man die Richtung der Kanten weglässt.
Dies gibt nun Anlass zu folgender Definition:

Definition 6.14. Zwei Knoten v und v' eines DAG $\Gamma = (V, E, f, o, b, M, \leq)$ heißen **einfach äquivalent**, falls $o(v) = o(v')$ gilt und eine monoton wachsende, bijektive Abbildung $g : E_i(v) \to E_i(v')$ existiert mit $s(e) = s(g(e)) \; \forall \; e \in E_i(v)$. Ein DAG heißt **reduziert**, falls er keine einfach äquivalenten Knoten enthält.

Die Verwendung von Assoziativität und Kommutativität und deren Formulierung als Äquivalenz ist auch möglich-aber komplizierter. Sie benötigt Teilgraphen.

- „Kommutativ äquivalent“: $o(v) = o(v')$ ist kommutativ zum Beispiel mit $\in \{+, *\}$ und es existiert eine bijektive Abbildung $g : E_i(v) \to E_i(v')$ mit $s(e) = s(g(e)) \; \forall \; e \in E_i(v)$.
- „Assoziativ äquivalent“: Bezieht sich auf Teilgraphen, siehe dazu die Skizze in Abbildung 15.

Besitzt ein Graph Γ zwei einfach (kommutativ) äquivalente Knoten v,v', so lässt sich ein

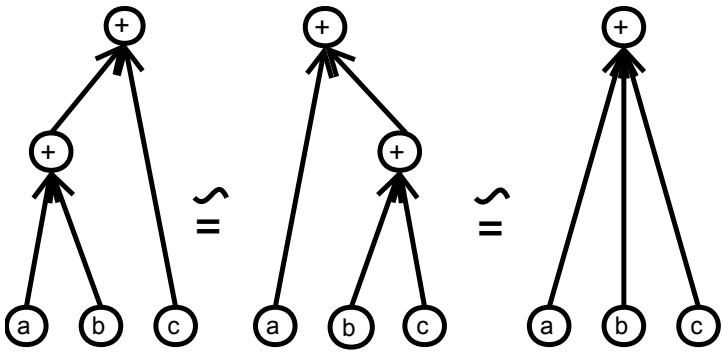


Abbildung 15: Eine Veranschaulichung bezüglich assoziativ äquivalent.

äquivalenter DAG Γ' erzeugen, der einen Knoten weniger enthält:

$$
\begin{aligned}
V' &:= V \setminus \{v'\} \\
E' &:= (E \setminus (E_i(v') \cup E_i(v))) \cup \{\overline{e}_1, \ldots, \overline{e}_{|E_o(v')|}\} \\
f' &: E' \to V' \times V' \\
&\quad e \mapsto \begin{cases} f(e) \text{ , falls } e \in E \\ (v, t(e_i)) \text{ , falls } e = \overline{e}_i \end{cases} \\
o' &: V' \to O \cup V_{\mathbb{N}} \cup \mathbb{R} = o|_{v'} \\
b' &: V' \to \mathcal{I}\mathbb{R} \\
&\quad \widetilde{v} \mapsto \begin{cases} b(\widetilde{v}) \text{ , falls } \widetilde{v} \neq v \\ b(v) \cap b(v') \text{ , falls } \widetilde{v} = v) \end{cases} \\
M' &= M \cap V'
\end{aligned}
$$

Daher gibt es zu jedem DAG einen äquivalenten reduzierten DAG. Daher lassen sich faktorable Optimierungsprobleme durch einen reduzierten DAG darstellen.

Beispiel 6.15. Betrachten wir nun folgendes Beispiel:

$$
\begin{aligned}
&\min\ (4x_1 - x_2x_3)(x_1x_2 + x_3) \\
&s.t.\ x_1^2 + x_2^2 + x_1x_2 + x_2x_3 + x_2 = 0 \\
&\qquad \exp(x_1x_2 + x_2x_3 + x_2 + \sqrt{x_3}) \in [-1, 1]
\end{aligned}
$$

Eine mögliche Darstellung dieses Problems als DAG ist Abbildung 16. Wie man an dieser Stelle bereits erkennt, ist das problem stark nicht linear und auch stark nicht quadratisch.

Nun können wir aber das obige Problem umformulieren und den DAG interpretieren durch Einführen von Variablen für „Zwischenergebnisse“:

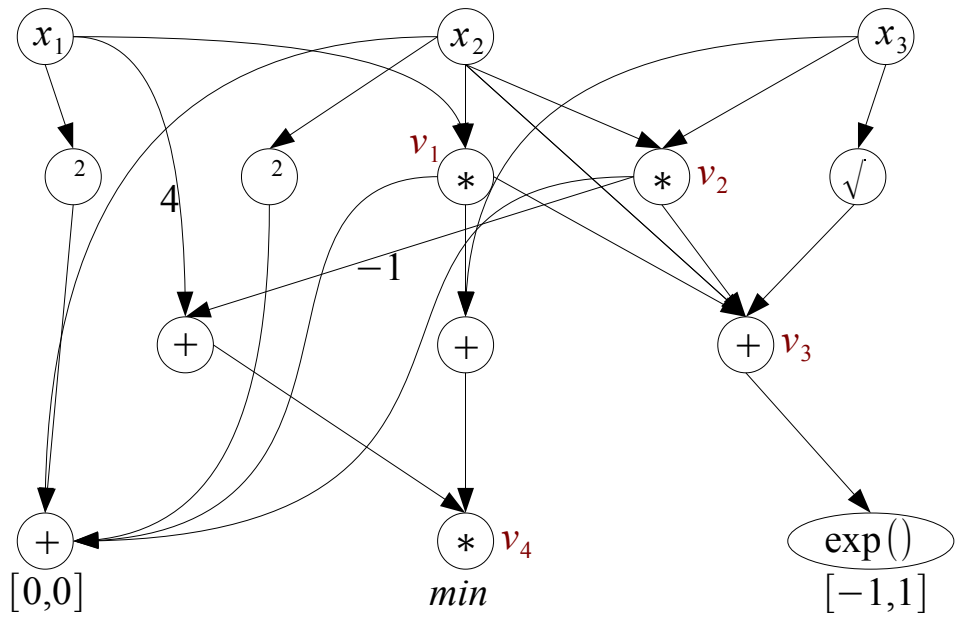


Abbildung 16: Der Graph zum vorhergehenden Beispiel 6.15 und die Stellen an denen die Variablen eingeführt werden.

$$
\begin{aligned}
\textit{linear } & \min\ v_4 \\
\textit{separabel quadratisch s.t. } & x_1^2 + x_2^2 + v_1 + v_2 + x_2 = 0 \\
\textit{nichtlinear eindimensional } & \exp(v_3) \in [-1, 1] \\
\textit{quadratisch } & v_4 = (4x_1 - v_2)(v_1 + x_3) \\
\textit{separabel } & v_3 = v_1 + v_2 + x_2 + \sqrt{x_3} \\
\textit{quadratisch } & v_2 = x_2 x_3 \\
\textit{quadratisch } & v_1 = x_1 x_2
\end{aligned}
$$

Dieses Optimierungsproblem ist äquivalent zum ursprünglichen. Es ist zwar größer, aber seine Struktur ist viel einfacher. Die neue Struktur muss nicht durch erstellen eines neuen Graphen erzeugt werden, sondern ist in natürlicher Weise bereits vorhanden.

Wir werden uns nun überlegen wie man an einzelnen Punkten den Wert der Funktion und des Gradienten der Funktion bestimmen kann sowie den Wertebereich der Funktion ermitteln kann. Dazu werden wir die Funktion $f(x_1, x_2, x_3,) = (4x_1 - x_2x_3)(x_1x_2 + x_3)$ betrachten.

6.2.1 Punktevaluation

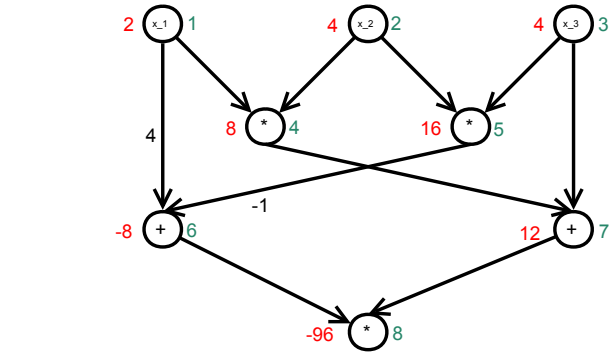
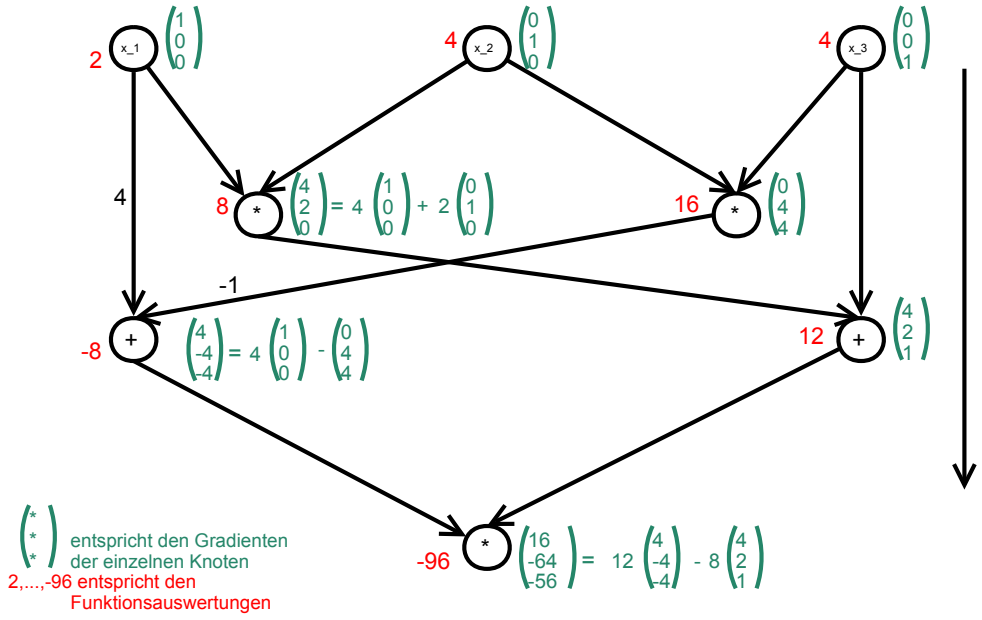


Abbildung 17: Veranschaulichung der Punktevaluation

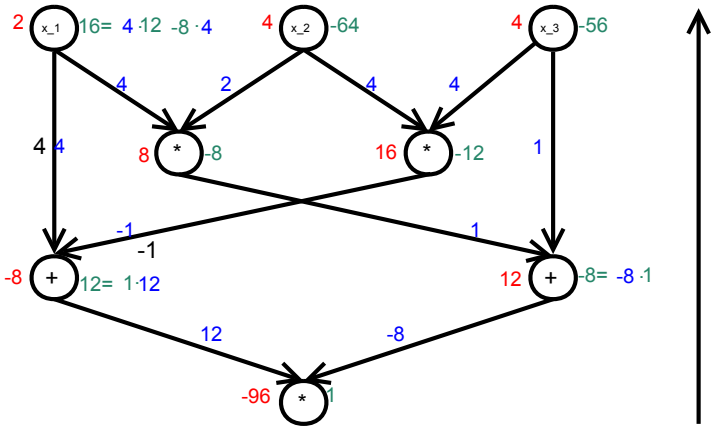
In dieser Abbildung nehmen wir an, dass wir bereits eine Ordnung festgelegt haben und wollen den Funktionswert von f bestimmen. Der Funktionswert ist also $f(2,4,4) = -96$. Man kann diese Form der Punktevaluation auch anhand Abbildung 17 darstellen. Diese Art der Punktauswertung ist linear in der Anzahl der Knoten und Kanten (dies folgt aus der Ordnung), also $\mathcal{O}(|V| + |E|)$.

6.2.2 Gradientenauswertung



Man nennt diesen Algorithmus auch **automatische Vorwärtsdifferentiation**. Automati-

sches Vorwärtsdifferenzieren: Hierbei nehmen wir an, dass wir bereits die Funktionsauswertung in den Knoten (und damit in dem Punkt indem wir den Gradienten berechnen wollen) berechnet haben und wollen den Gradienten von f ausrechnen. Der Aufwand ist linear in der Anzahl der Knoten, Kanten und der Dimension, also $\mathcal{O}(n(|V|+|E|))$.
Nun gibt es aber auch eine zweite Möglichkeit zu differenzieren-indem man die Kettenregel verwendet und von „unten nach oben“ rechnet, also **automatische Rückwärtsdifferentiation**: $\frac{\partial}{\partial x_l}(f \circ g)(x) = \sum_j (\frac{\partial}{\partial x_j} f)(g(x)) \frac{\partial g_j(x)}{\partial x_l}$.
Man erhält mit dem Algorithmus aus Abbildung 18 das gleiche Resultat wie beim autma-



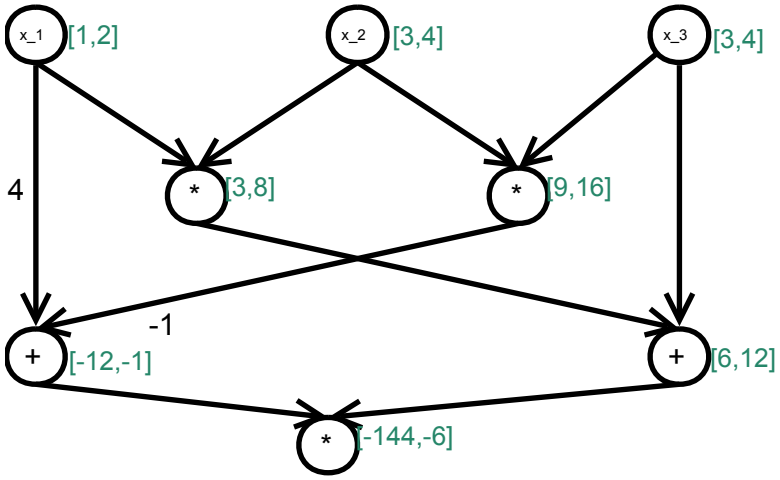
2,...,-96 entspricht den Funktionsauswertungen
1,...,-64 entspricht den Komponenten des Gradienten
-1,...,12 entspricht den inneren Ableitungen der Knoten

Abbildung 18: Automatisches Rückwärtsdifferenzieren: Hierbei nehmen wir an, dass wir die Funktionsauswertung schon kennen und den Gradienten an einer Stelle bestimmen wollen.

tischen Vorwärtsdifferenzieren, nämlich dass $\nabla f(2,4,4) = \begin{pmatrix} 16 \\ -64 \\ -56 \end{pmatrix}$. Allerdings beträgt der Aufwand beim autmatischen Rückwärtsdifferenzieren nur $\mathcal{O}(|V|+|E|)$. Man verwendet also um den Gradienten an einer Stelle von f auszuwerten für gewöhnlich nur die automatische Rückwärtsdifferentiation!

6.2.3 Abschätzungen des Wertebereichs 1

Gesucht ist in unserem Beispiel eine Abschätzung des Wertebereichs von f auf $[1,2] \times [3,4] \times [3,4]$. Eine Möglichkeit an eine Abschätzung zu gelanden ist die Folgende:



Mit dieser Methode erhält man als Abschätzung $[-144, -6] \supseteq f([1,2],[3,4],[3,4])$. Dies ist aber *nicht* die bestmögliche Abschätzung!

6.2.4 Abschätzungen des Wertebereichs 2

Für $f \in \mathcal{C}^1$ gilt: $f(x) = f(y) + f'(\xi)(x-y)$ (MWS) $\Rightarrow f(x) \in f(y) + f'(\overline{xy})(x-y)$. Sei nun $x \in \mathbf{x}$

$$\Rightarrow f(\mathbf{x}) \subseteq f(y) + f'(\mathbf{x})(\mathbf{x}-y) \text{ für } y \in \mathbf{x} \tag{6.1}$$

[1] Diese Art der Abschätzung lässt sich auch wie in Abbildung 19 darstellen. Aus der In-

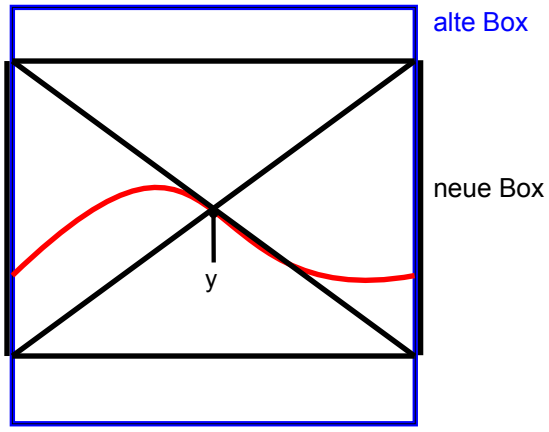


Abbildung 19: Man ermittelt die maximale Steigung der Kurve und schneidet anschließend mit dem Rand der Box. Die neuen kleineren Achsen stellen dann den neuen Wertebereich dar.

tervallanalysis weiß man, dass die Überschätzung in der Berechnung von $f(\mathbf{x})$ mit Hilfe der

[1] $f(\mathbf{x}) = \square \bigcup_{x \in \mathbf{x}} f(x) \subseteq \square \bigcup_{x \in \mathbf{x}} (f(y) + f'(\overline{xy})(x-y)) = f(y) + \square \bigcup_{x \in \mathbf{x}} f'(\overline{xy})(x-y) \subseteq$
$f(y) + \square \bigcup_{x \in \mathbf{x}} f'(\mathbf{x})(\mathbf{x}-y) = f(y) + f'(\mathbf{x})(\mathbf{x}-y)$

Intervallarithmetik (Abschätzung des Wertebereichs 1) $\mathcal{O}(rad(\mathbf{x}))$
$(rad(\mathbf{x}) = \frac{1}{2}(\overline{x} - \underline{x}))$ ist. Wenn wir in (6.1) das $f'(\mathbf{x})$ mit Intervallarithmetik bestimmen, dann ist die Überschätzung von $f'(\mathbf{x})$ von der Größenordnung $\mathcal{O}(rad(\mathbf{x}))$. Das Ergebnis in (6.1) hat also eine Überschätzung, die $\mathcal{O}(rad(\mathbf{x})^2)$ ist. (6.1) heißt Mittelwertform und y wird meist als $\check{x} := \frac{1}{2}(\overline{x} + \underline{x})$ gewählt.
Gesucht ist nun eine Wertebereichsabschätzung für ∇f auf $\mathbf{x} = [1,2] \times [3,4] \times [3,4]$. Auch hier gibt es wieder 2 Möglichkeiten zu einer Abschätzung zu gelangen: Zum einen die **Wertebereichsabschätzung vorwärts**, siehe Abbildung 20. Zum anderen die **Wertebereichs-**

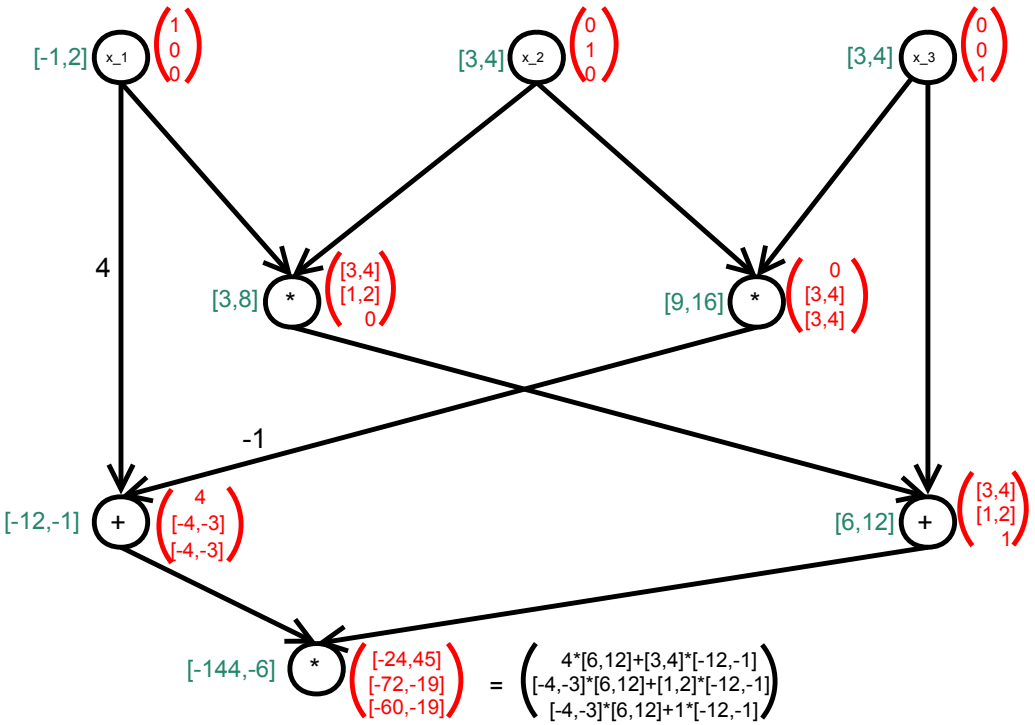


Abbildung 20: Wertebreichsabschätzung vorwärts- analog in der Berechnung zum Vorwärtsgerichteten Differenzieren

abschätzung rückwärts, siehe dazu Abbildung 21. Hier gilt in beiden Fällen:
$\nabla f([1,2] \times [3,4] \times [3,4]) \subseteq \begin{pmatrix} [-24,45] \\ [-72,-19] \\ [-60,-19] \end{pmatrix}$. Es ist Zufall, dass die beiden Gradientenabschätzungen in diesem Fall gleich sind! Üblicherweise ist die Vorwärtsauswertung besser, aber langsamer. Grund ist die Subdistributivität.

Was zur Rückwärts- beziehungsweise Vorwärtsauswertung des Intervallgradienten noch

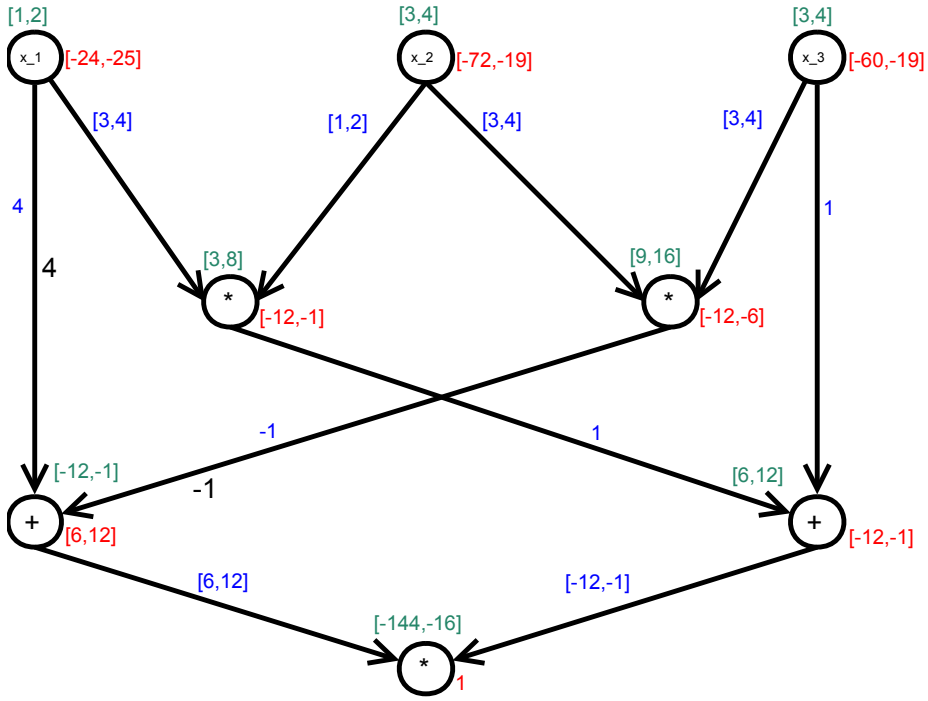


Abbildung 21: Wertebereichsabschätzung rückwärts-die Berechnung erfolgt analog zum Rückwärtsgerichteten Differenzieren.

fehlt, sind die partiellen Ableitungen der elementaren Funktionen:

$$
\begin{aligned}
f(x_1, x_2) &= \lambda x_1 + \mu x_2 \\
\nabla f(\mathbf{x}_1, \mathbf{x}_2) &= \begin{pmatrix} \lambda \\ \mu \end{pmatrix} \\
f(x_1, x_2) &= x_1 x_2 \\
\nabla f(\mathbf{x}_1, \mathbf{x}_2) &= \begin{pmatrix} \mathbf{x}_2 \\ \mathbf{x}_1 \end{pmatrix} \\
f(x_1, x_2) &= \frac{x_1}{x_2} \\
\nabla f(\mathbf{x}_1, \mathbf{x}_2) &= \begin{pmatrix} \frac{1}{\mathbf{x}_2} \\ -\frac{\mathbf{x}_1}{(\mathbf{x}_2)^2} \end{pmatrix}
\end{aligned}
$$

Diese sind exakt, da in jeder Komponente jede Variable nur einmal vorkommt. Nun folgen

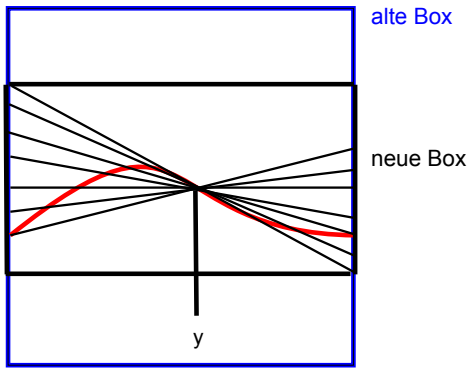
noch partielle Ableitungen die nicht mehr exakt sind:

$$\begin{aligned} f(x_1) &= \exp(x_1) \\ \nabla f(\mathbf{x}_1) &= f(\mathbf{x}_1) \\ f(x_1) &= \sqrt{x_1} \\ \nabla f(\mathbf{x}_1) &= -\frac{1}{2f(\mathbf{x}_1)} \\ f(x_1, x_2) &= x_1^{x_2} = e^{x_2 \ln(x_1)} \\ \nabla f(x_1, x_2) &= \begin{pmatrix} e^{x_2 \ln(x_1)} \frac{x_2}{x_1} \\ e^{x_2 \ln(x_1)} \ln(x_1) \end{pmatrix} \\ \Rightarrow \nabla f(\mathbf{x}_1, \mathbf{x}_2) &= \begin{pmatrix} f(\mathbf{x}_1, \mathbf{x}_2) \frac{\mathbf{x}_2}{\mathbf{x}_1} \\ f(\mathbf{x}_1, \mathbf{x}_2) \ln(\mathbf{x}_1) \end{pmatrix} \end{aligned}$$

Diese Einschließungen verursachen Überschätzung!.

6.2.5 Abschätzungen des Wertebereichs 3

Wir werden nun im Folgenden noch eine Methode kennenlernen, um die Abschätzungen der Wertebereiche von f und ∇f zu verbessern. Dazu verfolgen wir im Prinzip die selbe Idee wie bei Abbildung 19, aber wir werden nun nicht die Ableitung verwenden um den Bereich zu verkleinern sondern die Sekanten.



An dieser Stelle werden wir mit dem mehrdimensionalen Pendant der Dividierten Differenzen arbeiten, bekannt aus der Numerik, und diese mit $f[z, x]$ bezeichnen. Ist f Lipschitz-stetig, dann gibt es für je zwei x, z ein $f[z, x]$ mit $f(x) = f(z) + f[z, x](x - z)$ (im Mehrdimensionalen ist $f[z, x]$ ein Zeilenvektor). Unter dieser Annahme erhält man im Eindimensionalen:
$f[z, x] = \frac{f(x) - f(z)}{x - z}$ für $x \neq z$. Man bezeichnet die Dividierte Differenz auch als Slope.
Im Mehrdimensionalen ist $f[z, x]$ nicht eindeutig! Es folgt, dass

$$\begin{aligned} \forall x \in \mathbf{x} : f(x) &\in f(z) + f[z, \mathbf{x}](\mathbf{x} - z) \\ \Rightarrow f(\mathbf{x}) &\subseteq f(z) + f[z, \mathbf{x}](\mathbf{x} - z) \\ \Rightarrow f(\mathbf{x}) &\subseteq f(\mathbf{z}) + f[\mathbf{z}, \mathbf{x}](\mathbf{x} - \mathbf{z}) \end{aligned}$$

Wir werden nun die Struktur der Slopes und der Ableitungen von verketteten Funktionen untersuchen: Im Eindimensionalen gilt:

$$\begin{aligned}(f \circ g)(x) &= f(g(y) + g[y,x](x-y)) = \\ &= f(g(y)) + f[g(y), g(x)](g(y) + g[y,x](x-y) - g(y)) = \\ &= (f \circ g)(y) + f[g(y), g(x)]g[x,y](x-y)\end{aligned}$$

Wenden wir uns nun dem Mehrdimensionalen zu: Geht $g : \mathbb{R}^n \to \mathbb{R}^m$, dann ist $g[y,x] \in \mathbb{R}^m \times \mathbb{R}^n$.

$$\begin{aligned}(f \circ g)'(x) = f'(g(x))g'(x) &\Rightarrow (f \circ g)' = (f' \circ g)g' \\ (f \circ g)(x) &= (f \circ g)(y) + (f \circ g)[y,x](x-y) = \\ &= (f \circ g)(y) + f[g(y), g(x)]g[y,x](x-y)\end{aligned}$$

Daraus folgt, dass $(f \circ g)[y,x]$ gewählt werden kann als $f[g(y), g(x)]g[y,x]$ (man kann im Allgmeinen nicht $=$ wählen, da keine Eindeutigkeit gegeben ist im Mehrdimensionalen.)
Wir sehen also, dass Slopes und Ableitungen gleich strukturierte Kettenregeln haben. Also können sie im DAG auf gleiche Weise berechnet werden. Das heißt zuerst müssen wir die Slopes der elementaren Operationen bestimmen:

$$\begin{aligned}f(x_1, x_2) &= \lambda x_1 + \mu x_2 \\ f[(z_1, z_2), (x_1, x_2)] &= (\lambda, \mu) \\ f(x_1, x_2) &= x_1 x_2 \\ x_1 x_2 &= z_1 z_2 + (s_1, s_2) \begin{pmatrix} x_1 - z_1 \\ x_2 - z_2 \end{pmatrix} = z_1 z_2 + s_1(x_1 z_1) + s_2(x_2 - z_2) \\ x_1 x_2 - z_1 z_2 &= s_1 x_1 - s_1 z_1 + s_2 x_2 - s_2 z_2 \text{ (auflösen nach } s_1, s_2) \\ &\vdots \\ s_1 &= \lambda x_2 + (1-\lambda) z_2, \lambda \in \mathbb{R} \\ s_2 &= (1-\lambda) x_1 + \lambda z_1, \lambda \in \mathbb{R}\end{aligned}$$

An dieser Stelle ist natürlich die Frage für welche $\lambda \in \mathbb{R}$ die Einschließung für $f(\mathbf{x})$ bei z am besten ist? Es gilt (und das ist nicht trivial!): $f(\mathbf{x})$ hat die geringste Überschätzung, falls

$$\begin{aligned}\lambda &= \begin{cases} 0 \text{ für } rad(\mathbf{x}_1)|z_2| \leq rad(\mathbf{x}_2)|z_1| \\ 1 \text{ sonst} \end{cases} \\ f[(z_1, z_2), (x_1, x_2)] &= (z_2, x_1) \\ f(x_1, x_2) &= \frac{x_1}{x_2} \\ \frac{x_1}{x_2} &= \frac{z_1}{z_2} + \begin{pmatrix} \frac{\lambda}{z_2} + \frac{1-\lambda}{x_2} \\ -\frac{\lambda}{z_2}\frac{x_1}{x_2} - \frac{1-\lambda}{x_2}\frac{z_1}{z_2} \end{pmatrix}^T \begin{pmatrix} x_1 - z_1 \\ x_2 - z_2 \end{pmatrix}, \lambda \in \mathbb{R}\end{aligned}$$

Der günstigste Slope ergibt sich für $\lambda = 1$:

$$f[(z_1, z_2), (x_1, x_2)] = (\frac{1}{z_2}, -\frac{1}{z_2} f(x_1, x_2))$$

$$f(x) = \varphi(x), \varphi : \mathbb{R} \to \mathbb{R}, \varphi \in \mathcal{C}^1$$

$$\varphi[z, \mathbf{x}] \subseteq \varphi[\mathbf{x}, \mathbf{x}] \underbrace{=}_{\text{folgt aus dem Mittelwertsatz und } \varphi \in \mathcal{C}^1} \varphi'(\mathbf{x}), z \in \mathbf{x}$$

$$\exp[z, x] = \frac{e^x - e^z}{x - z} = e^z \frac{e^{x-z} - 1}{x - z} = e^z \exp_1(x - z)$$

$$\exp_1(t) = \frac{e^t - 1}{t} = \sum_{k=0}^{\infty} \frac{t^k}{(k+1)!}$$

Nun wollen wir uns im Folgenden wieder anhand unseres Beispiels ausrechnen, wie groß die Überschätzung durch Slopes im Vergleich zur unserer bisherigen Überschätzung ist:
$f(x_1, x_2, x_3) = (4x_1 + x_2 x_3)(x_1 + x_2 x_3)$ mit $f[(2, 4, 4), ([1, 2], [3, 4], [3, 4])]$:
Wir erhalten sowohl bei der Slopebrechnung rückwärts als auch bei der Slopeberechnung

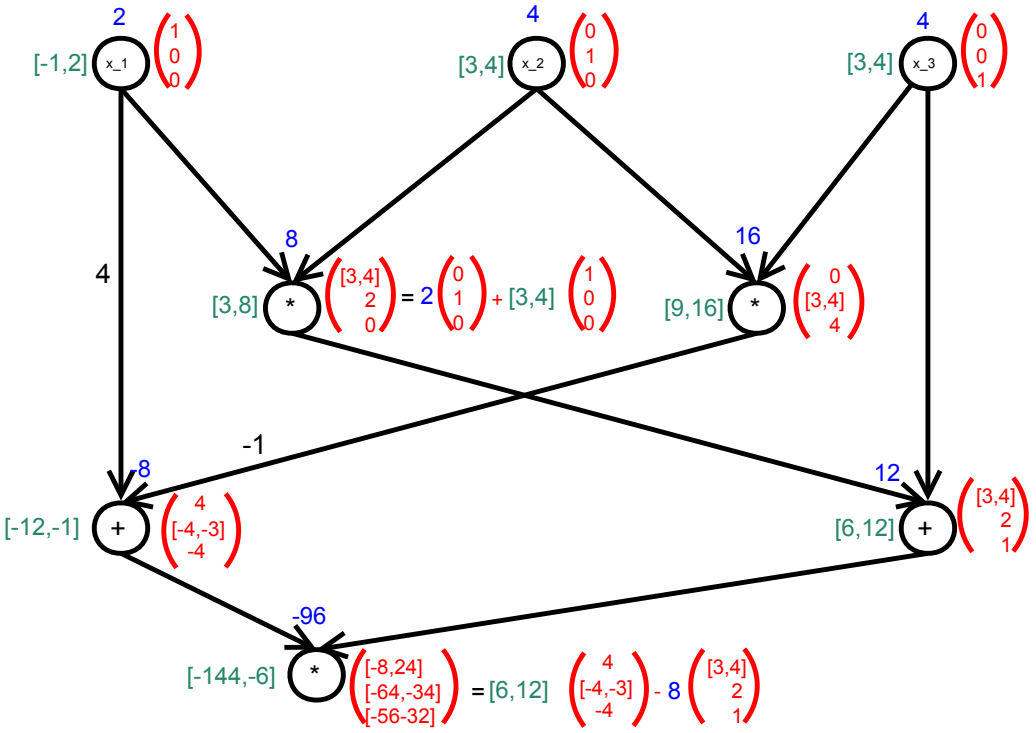


Abbildung 22: Slopeberechnung vorwärts, im Prinzip analog zum automatischen Vorwärtsdifferenzieren.

vorwärts die Überschätzung $\begin{pmatrix} [-8, 24] \\ [-64, -34] \\ [-56, -32] \end{pmatrix}$, zur Erinnerung: $f'(\mathbf{x}) = \begin{pmatrix} [-24, 45] \\ [-72, -19] \\ [-60, -19] \end{pmatrix}$. Slopes haben also meist eine geringere Überschätzung, bis zu 50 % in jeder Dimension, als $f'(\mathbf{x})$.

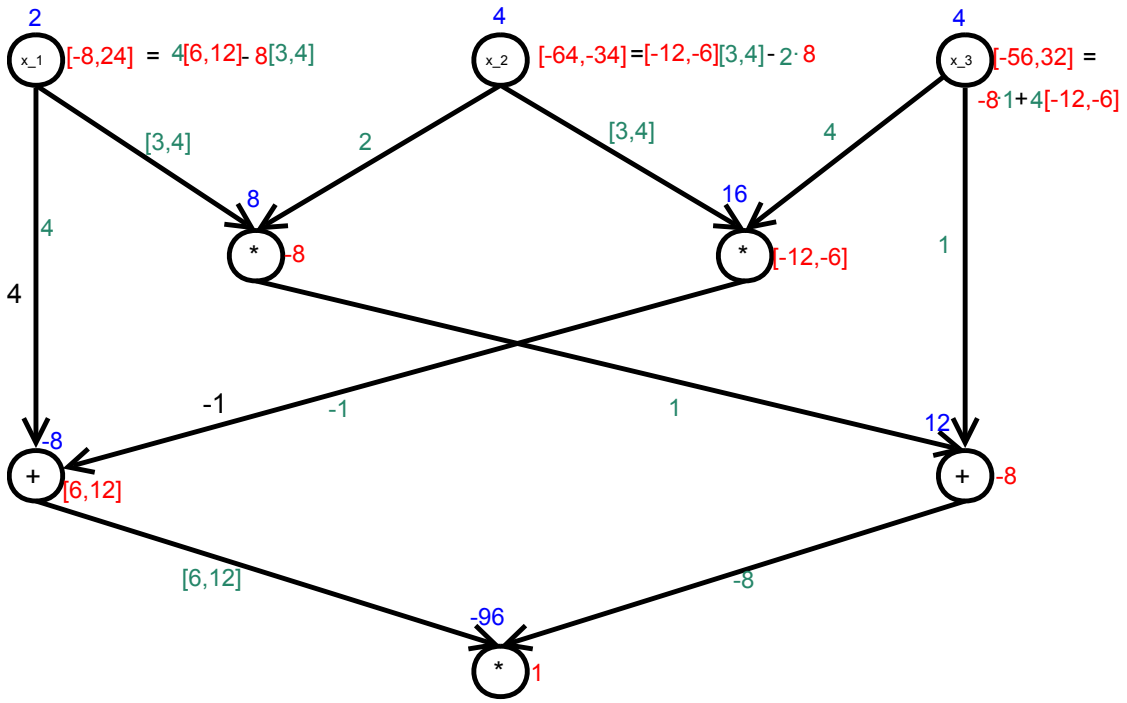


Abbildung 23: Slopeberechnung rückwärts, die Berechnung erfolgt analog zum rückwärtsgerichteten Differenzieren.

Beim Vorwärtsauswerten von Intervallableitung und Slope kann man sich zunutze machen, dass jeweils an den Knoten Ableitungen, beziehungsweise Slopes, bezüglich der Eingangskoordinaten berechnet werden. So gilt zum Beispiel für Knoten 4 (x_1, x_2):

$$
\begin{aligned}
& f_4(x_1, x_2, x_3) = x_1 x_2 \\
& f_4(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) \subseteq \mathbf{x}_1 \mathbf{x}_2 \\
& f_4(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) \subseteq z_1 z_2 + s_4(\mathbf{x} - z) = z_1 z_2 + (\mathbf{x}_2, z_1) \begin{pmatrix} \mathbf{x}_1 - z_1 \\ \mathbf{x}_2 - z_2 \end{pmatrix} = \\
& \qquad = 8 + ([3,4], 2) \begin{pmatrix} [-1,0] \\ [-1,0] \end{pmatrix} = 8 + [-4,0] + [-2,0] = [2,8] \\
\Rightarrow\ & f_4(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) \subseteq (\mathbf{x}_1 \mathbf{x}_2 \cap (z_1 z_2 + s_4(\mathbf{x} - z)))
\end{aligned}
$$

Die Überschätzung, die durch die Abschätzung mit Slopes entsteht, ist $\mathcal{O}(rad(\mathbf{x})^2)$. Die Durchschnittsbildung $f(\mathbf{x}) \cap (f(z) + f[z, \mathbf{x}](\mathbf{x} - z))$ beziehungsweise $f(\mathbf{x}) \cap (f(\check{x}) + f'(\mathbf{x})(\mathbf{x} - \check{x}))$ sollte man bei Vorwärtsauswertung an jedem Knoten durchführen.

6.3 Constraint Propagation

Betrachten wir wieder unser Beispiel aus dem vorigen Abschnitt, allerdings diesmal mit einer Nebenbedigung mehr:

$$\begin{aligned} \min\ & f(x_1, x_2, x_3) = (4x_1 - x_2x_3)(x_1 + x_2x_3) \\ s.t.\ & x \in [1,2] \times [3,4] \times [3,4] \\ & f(2,4,4) = -96 \end{aligned}$$

Für das globale Optimum x^* muss $f(x^*) \leq -96$ gelten. Das heißt wir können, ohne die Lösungsmenge zu verändern, die Nebenbedigung $f(x) \leq -96$ zu unserem Optimierungsproblem hinzufügen.
Die Constraint Propagation (CP) stammt aus dem Constraint Programming (Teilgebiet der kombinatorischen Optimierung) $x_1, \ldots, x_k \in \{0, 1\}$.

Beispiel 6.16. In diesem Beispiel suchen wir x_1, x_2, x_3 die Folgendes erfüllen:

$$\begin{aligned} (x_1 \wedge x_2) \vee x_3 &= 1 \\ x_2 \wedge x_3 &= 0 \end{aligned}$$

Nun gibt es mehrere Möglichkeiten um die richtigen Kombinationen zu finden:

(i) Wir untersuchen einfach alle möglichen Kombinationen:

Möglichkeit	x_1	x_2	x_3	$(x_1 \wedge x_2) \vee x_3$	$x_2 \wedge x_3$
1	0	0	0	0	0
2	0	0	1	1	0
3	0	1	0	0	0
4	0	1	1	1	1
5	1	0	0	0	0
6	1	0	1	1	0
7	1	1	0	1	0
8	1	1	1	1	1

Wie wir der obigen Tabelle entnehmen können, gibt es 3 mögliche Kombinationen: Möglichkeit 2,6,7.

(ii) Nun ist das Durchprobieren aller möglichen Kombinationen aber nicht der beste Weg, es bietet sich daher an folgende Methode zu probieren: Sei $x_1 = 0$.

$$\begin{aligned} x_1 = 0 \Rightarrow x_1 \wedge x_2 = 0 \Rightarrow (x_1 \wedge x_2) \vee x_3 = 1 \Leftrightarrow x_3 = 1 \\ \Rightarrow x_3 = 1 \Rightarrow (x_2 \wedge x_3 = 0 \Leftrightarrow x_2 = 0) \Rightarrow x_2 = 0 \end{aligned}$$

Es folgt also: $x_1 = 0 \Rightarrow x_2 = 0 \wedge x_3 = 1$.
Sei $x_1 = 1$.

$$\begin{aligned} x_1 = 1 \Rightarrow x_1 \wedge x_2 = x_2 \Rightarrow (x_1 \wedge x_2) \vee x_3 = x_2 \vee x_3 \\ \Rightarrow (x_2 \vee x_3 = 1) \wedge (x_2 \wedge x_3) = 0 \Rightarrow x_2 \neq x_3 \end{aligned}$$

Es folgt also: $x_1 = 1 \Rightarrow (x_2 = 0 \wedge x_3 = 1) \vee (x_2 = 1 \wedge x_3 = 0)$. Also sind alle möglichen Lösungen gegeben durch $(0,0,1), (1,0,1), (1,1,0)$.
Man kann sich diese Überlegung auch im folgenden Graphen und dem dazupassenden Suchbaum veranschaulichen:

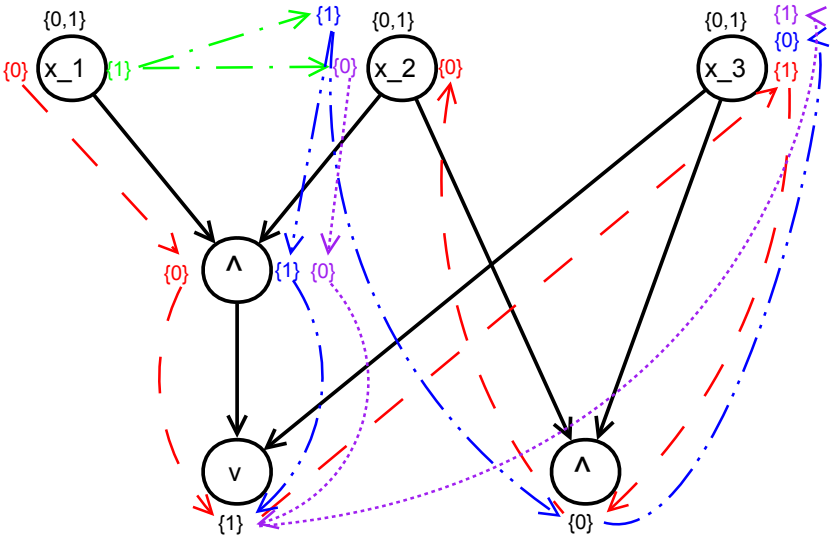


Abbildung 24: Darstellung als Graph.

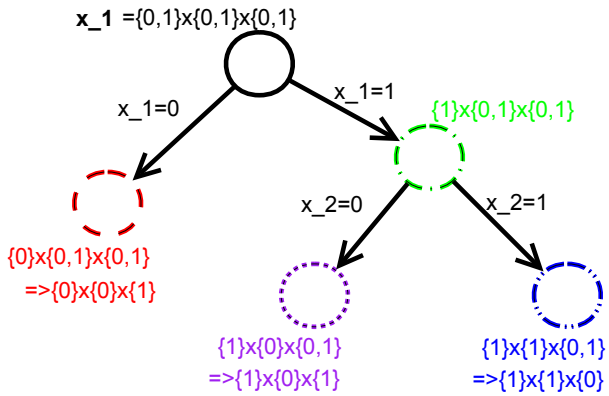


Abbildung 25: Der dazu passende Baum.

Wir verallgemeinern nun das Ziehen logischer Schlüsse auf unser Problem:

$\left.\begin{array}{l} x_1 = 0 \\ x_2 \in \{0,1\} \end{array}\right\} \Rightarrow x_1 \wedge x_2 = 0$ bezeichnet man als „Schluss in Pfeilrichtung“

$\left.\begin{array}{l} (x_1 \wedge x_2) = 0 \\ (x_1 \wedge x_2) \vee x_3 = 1 \end{array}\right\} \Rightarrow x_3 = 1$ bezeichnet man als „Schluss gegen die Pfeilrichtung“.

Man kann diesen Schluss auch äquivalent umformulieren zu

$$\left.\begin{array}{l} v = 0 \\ v \vee w = 1 \\ w \in \{0,1\} \end{array}\right\} \Rightarrow w = 1$$

Wenden wir uns nun einem allgemeineren Vorgehen zu-dem **Vorwärtsschritt**: Propagation in Pfeilrichtung. Hierbei nehmen wir an, dass wir bereits $\mathbf{f}$ kennen und diesen Bereich aber verkleinern wollen: also $f(x_1, \ldots, x_k)$ verkleinern wollen, $\mathbf{x}_1, \ldots, \mathbf{x}_j \to f(\mathbf{x}_1, \ldots, \mathbf{x}_k) \cap \mathbf{f}$.

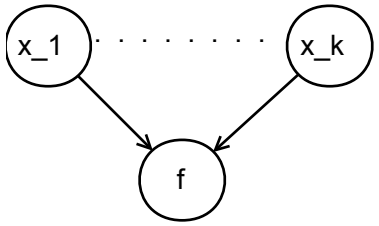


Abbildung 26: Veranschaulichung eines Vorwärtsschrittes.

Betrachten wir einen Rückwärtsschritt (diesmal allerdings nur mit 2 Knoten x und y):
Dieses mal kennen wir $f(x, y) \in \mathbf{f}$ mit $x \in \mathbf{x}, y \in \mathbf{y}$. Wir wollen hier den Bereich für x und y verkleinern!.
$\mathbf{x}^0_{neu} \supseteq \square \{x \in \mathbf{x} | f(x, y) \in \mathbf{f} \text{ für ein } y \in \mathbf{y}\}$, $\mathbf{y}^0_{neu} \supseteq \square \{y \in \mathbf{y} | f(x, y) \in \mathbf{f} \text{ für ein } x \in \mathbf{x}\}$.

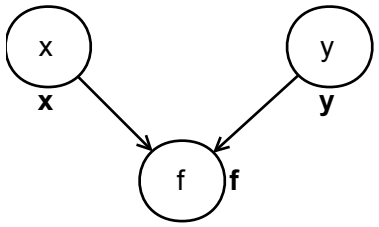


Abbildung 27: Veranschaulichung eines Rückwärtsschrittes.

$\mathbf{x}_{neu} = \mathbf{x} \cap \mathbf{x}^0_{neu}$, $\mathbf{y}_{neu} = \mathbf{y} \cap \mathbf{y}^0_{neu}$.

Vorwärts Evaluation (Forward Evaluation):
$FE(N, \mathbf{f}) = \mathbf{f} \cap f(\mathbf{x}_1, \ldots, \mathbf{x}_k)$ mit $N = f(x_1, \ldots, x_k), x_1 \in \mathbf{x}_1, \ldots, x_k \in \mathbf{x}_k$.
Rückwärtspropagation (Backward Propagation):
$BP(N, x_i) = \mathbf{x}_i \wedge \square \{x_i \in \mathbf{x}_i | f(y_1, \ldots, y_{i-1}, x_i, y_{i+1}, \ldots, y_k) \in \mathbf{f} \text{ für ein}$
$(y_1, \ldots, y_{i-1}, y_{i+1}, \ldots, y_k) \in \mathbf{x}_1 \times \cdots \times \mathbf{x}_{i-1} \times \mathbf{x}_{i+1} \times \cdots \times \mathbf{x}_k\}$.

Beispiel 6.17. Sei $f(x) = x^2, \mathbf{x} = [0, 2], \mathbf{f} = [1, 8]$. $FE(f, \mathbf{f}) = [1, 8] \cap [0, 2]^2 = [1, 4]$.

$$BP(f) = BP(f, x) = [0, 2] \cap \underbrace{\square \left\{ x \in [0, 2] | \underbrace{x^2 \in [1, 4]}_{[-2,-1] \cup [1,2]} \right\}}_{[1,2]} = [1, 2].$$

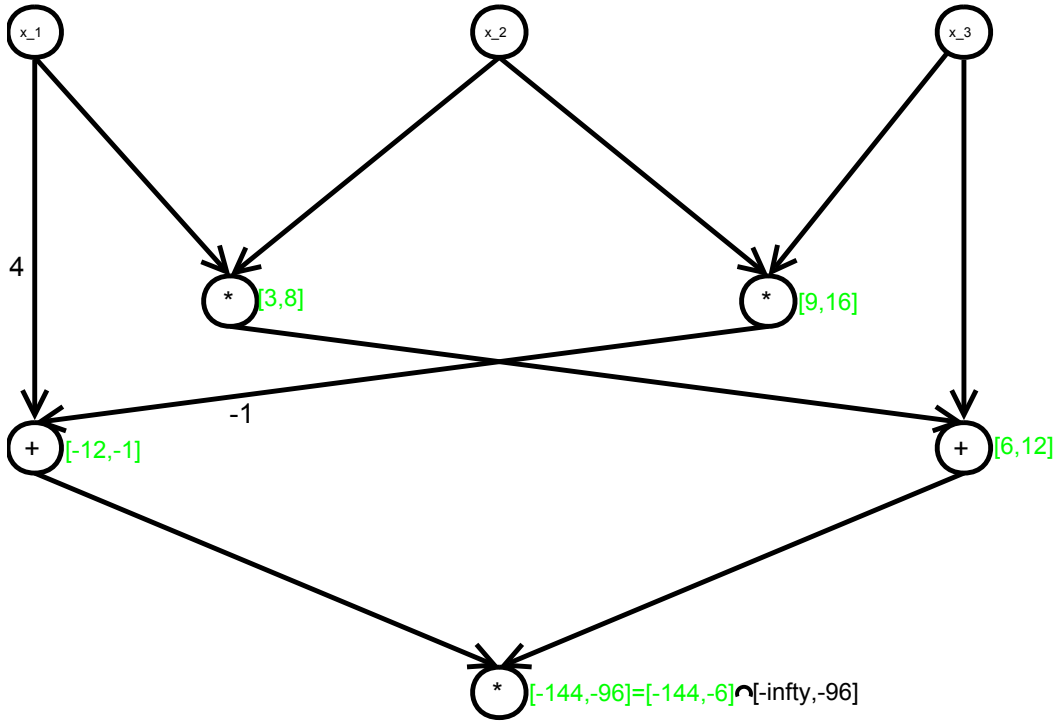
Definition 6.18. 2 Knoten bei denen Vorwärts Evaluation und Rückwärtspropagation keine Verbesserung mehr bringen bezeichnet man als **konsistent**.

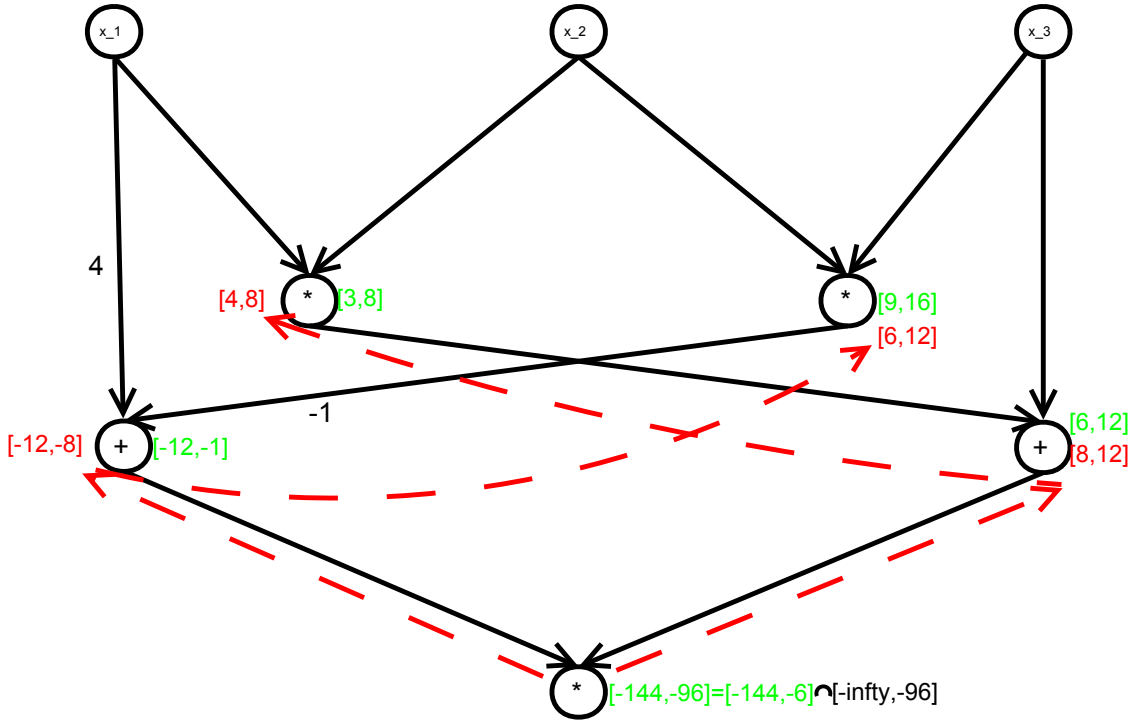
Ist $\varphi : \mathbb{R} \to \mathbb{R}$ gegeben, so benötigt man zur Implementation von BP eigentlich die „inverse Funktion“ $\varphi^{-1} : \mathcal{IR} \mapsto \mathcal{IR}$. Für uns interessanter ist aber eigentlich $\varphi^{-1} : \mathcal{IR} \to \bigcup \mathcal{IR}$ mit $\bigcup \mathcal{IR} \ldots$ Menge der abzählbaren Vereinigungen von Intervallen. Wollen wir uns nun im Folgenden überlegen wie wir Rückwärtspropagation ausrechnen können:

$$
\begin{aligned}
f(x,y) &= \lambda x + \mu y, \lambda x + \mu y \in \mathbf{f} \Rightarrow x \in \frac{1}{\lambda}(\mathbf{f} - \mu y), y \in \mathbf{y} \Rightarrow x \in \frac{1}{\lambda}(\mathbf{f} - \mu \mathbf{y}) \\
BP(f,x) &= \mathbf{x} \cap \frac{1}{\lambda}(\mathbf{f} - \mu \mathbf{y}) \\
f(x,y) &= xy \\
BP(f,x) &= \mathbf{x} \cap \frac{\mathbf{f}}{\mathbf{y}} \\
f(x,y) &= \frac{x}{y}, f = \frac{x}{y} \Leftrightarrow x = yf \\
BP(f,x) &= \mathbf{x} \cap \mathbf{fy} \\
BP(f,y) &= \mathbf{y} \cap \frac{\mathbf{x}}{\mathbf{f}} \\
f(x,y) &= x^y = e^{y \ln x}, f = e^{y \ln x} \Leftrightarrow \ln f = y \ln x \Leftrightarrow x = e^{\frac{\ln f}{y}} \\
BP(f,x) &= \mathbf{x} \cap e^{\frac{\ln \mathbf{f}}{\mathbf{y}}} \\
BP(f,y) &= \mathbf{y} \cap \frac{\ln \mathbf{f}}{\ln \mathbf{x}}
\end{aligned}
$$

Bei Ausdrücken der Form $\frac{\mathbf{f}}{\mathbf{y}}, \frac{\mathbf{x}}{\mathbf{f}}, \frac{\ln \mathbf{f}}{\ln \mathbf{x}}, \ldots$ gilt es zu beachten, dass man nicht nur ein Intervall erhalten kann sondern auch die Vereinigung von 2 Intervallen, zum Beispiel:
$\frac{[-1,2]}{[-1,1]} = [-\infty, -1] \cup [1, \infty]$.
Wie man nun Vorwärts Evaluation und Rückwärtspropagation konkret einsetzt wollen wir uns wieder anhand unseres Beispiels überlegen:

$$
\begin{aligned}
&\min f(x_1, x_2, x_3) \\
&s.t.\ f(x_1, x_2, x_3) \leq -96 \\
&\quad x_1 \in [1,2] \\
&\quad x_2, x_3 \in [3,4]
\end{aligned}
$$





Zu obiger Abbildung ist noch Folgendes zu bemerken:
$\frac{[-144,-96]}{[6,12]} = [-24,-8] \Rightarrow [-12,-1] \cap [-24,-8] = [-12,-8]$;
$\frac{[-144,-96]}{[-12,-8]} = [8,18] \Rightarrow [8,18] \cap [6,12] = [8,12]$. Nun ist dieses Vorgehen natürlich nicht nur

bei unserem Beispiel anwendbar sondern auch im Allgemeinen, wie der folgende Algorithmus zeigt:
$\mathcal{L}_f := \emptyset \ldots$ Liste, die die Knoten enthält, von denen aus man Einschließungen wahrscheinlich durch Vorwärts Evaluation verbessern kann.
$\mathcal{L}_b := \emptyset \ldots$ Liste, die die Knoten enthält, von denen aus man Einschließungen wahrscheinlich durch Rückwärtspropagation verbessern kann.
Numeriere die Ecken gemäß Proposition 6.11.
$V_{OC} = 0 \ldots$ Vektor, dessen Länge gleich der Anzahl der Knoten ist.
$V_{ch} = 0 \ldots$ Vektor, dessen Länge gleich der Anzahl der Knoten ist.

```
for each node c, die eine Nebenbedingung beschreibt do
    NodeOccurances(c,V_OC)
    ForwardEvaluation(c,V_ch, L_b)
end for
while (L_b ≠ ∅ oder L_f ≠ ∅) do
    N:=getNextNode(L_b, L_f)
    if (N war in L_b) then
        for (each child c of N) do
            BP(N,c)
            if c = ∅ then
                return „infeasible“
            end if
            if c hat sich genügend verkleinert seit der letzten Vorwärts Evaluation then
                for each parent P von c mit p ≠ N do
                    if V_OC(P) > 0 then
                        füge P zu L_f hinzu
                    end if
                end for
            end if
            if c hat sich durch das Rückwärtspropagieren genügend verkleinert then
                füge c zu L_k hinzu
            end if
        end for
    else[|| N war in L_f]
        FE(N,N)
        if N=∅ then
            return „infeasible“
        end if
        if N hat sich genug geändert then
            for each parent P of N do
                if V_OC(P) > 0 then
                    füge P zu L_f hinzu
                end if
            end for
        end if
        if N hat sich seit dem letzten BP genug geändert then
            füge N zu L_b hinzu
        end if
    end if
end while
```

---

function NodeOccurences(N,& V_{OC})
  $V_{OC}(N) += 1$;
  for each child c of N **do**
    NodeOccurences(c,V_{OC})
  end for
end function
function ForwardEvaluation(M,&V_{ch},&$\mathcal{L}_b$)
  if N ist lokale Quelle $\vee V_{ch}(N) == 1$ **then**
    return
  end if
  for each child c of N **do**
    ForwardEvaluation(c,$V_{ch}, \mathcal{L}_b$)
  end for
  FE(N,**N**)
  $V_{ch}(N) = 1$
  if N== $\emptyset$ **then**
    return „infeasible“
  end if
  if N hat sich genug geändert **then**
    füge N zu $\mathcal{L}_b$ hinzu
  end if
end function
füge N zu $\mathcal{L}_b$ hinzu: Das muss so gemacht werden, dass die Knotennummern in $\mathcal{L}_b$ absteigend und diejenigen in $\mathcal{L}_f$ aufsteigend sortiert sind.
function getNextNode(&$\mathcal{L}_b$,&$\mathcal{L}_f$)
  entscheide,ob aus $\mathcal{L}_b$ oder $\mathcal{L}_f$ gewählt werden soll $\rightarrow \mathcal{L}$ wähle N als das erste Element in der gewählten Liste $\mathcal{L}$. Entferne N aus $\mathcal{L}$.
  return N
end function
entscheide,ob aus $\mathcal{L}_b$ oder $\mathcal{L}_f$ gewählt werden soll: Propagationsstrategie:
Bei Tests am effizientesten (unter den einfachen Strategien!) war:
if $\mathcal{L}_b \neq \emptyset$ **then**
  $\mathcal{L} = \mathcal{L}_b$
else$\mathcal{L} = \mathcal{L}_f$
end if

6.4 Synergie von CP und Auswertung

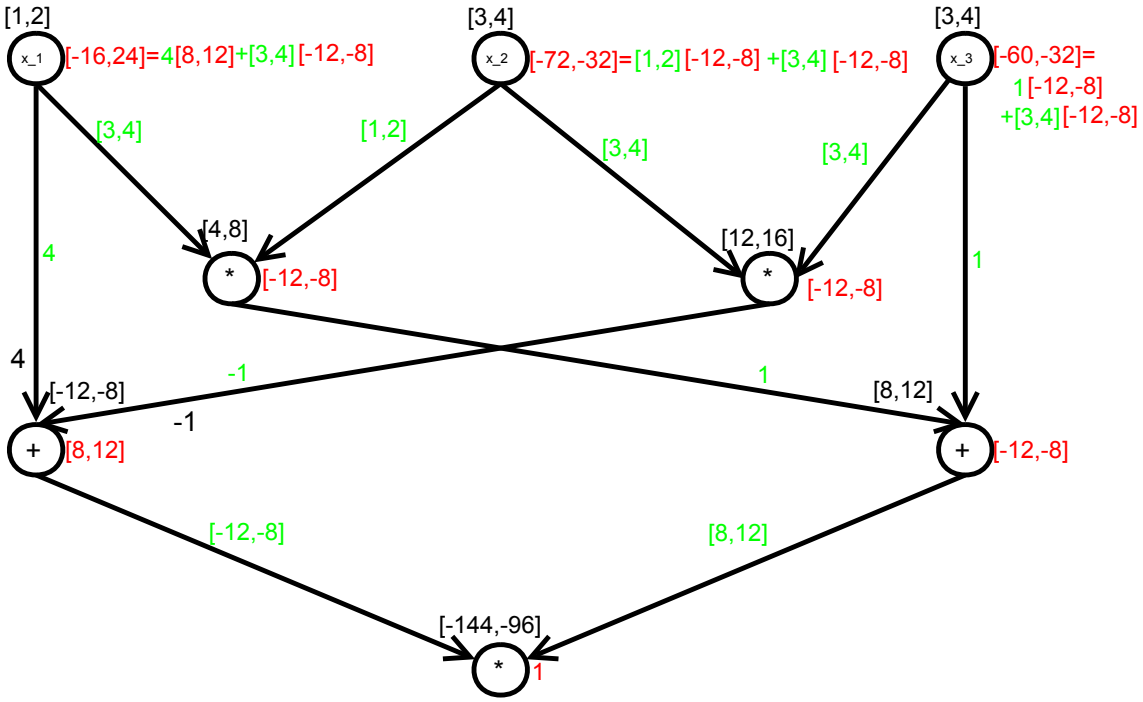


Abbildung 28: Veranschaulichung des Optimierungsproblems.

Das Optimierungsproblem

$$\begin{aligned} &\min f(x_1, x_2, x_3) \\ &s.t.\ x_1 \in [1, 2] \\ &\quad x_2, x_3 \in [3, 4] \end{aligned}$$

ist also demnach äquivalent zu

$$\begin{aligned} &\min f(x_1, x_2, x_3) \\ &s.t.\ x_1 \in [1, 2] \\ &\quad x_2, x_3 \in [3, 4] \\ &\quad f(x_1, x_2, x_3) \leq -96 \\ &\quad x_1 x_2 \in [4, 8] \\ &\quad x_2 x_3 \in [12, 16] \\ &\quad x_1 x_2 + x_3 \in [8, 12] \\ &\quad 4x_1 - x_2 x_3 \in [-12, -8] \end{aligned}$$

Wollen wir $f'(\mathbf{x} \cap \mathcal{F})$ bestimmen, dann gilt:
$(f \circ g)'(\mathbf{x} \cap \mathcal{F}) \subseteq f'(\underbrace{g(\mathbf{x} \cap \mathcal{F})}_{\subseteq \mathbf{b}}) g'(\mathbf{x} \cap \mathcal{F}) \subseteq f'(\mathbf{b}) g'(\mathbf{x} \cap \mathcal{F})$.

$$f'(\mathbf{x} \cap \mathcal{F}) \subseteq \begin{pmatrix} [-16, 24] \\ [-72, -32] \\ [-60, -32] \end{pmatrix} \subset \begin{pmatrix} [-24, 45] \\ [-72, -19] \\ [-60, -19] \end{pmatrix} = f'(\mathbf{x}).$$

$$\Rightarrow f\,[z, \mathbf{x} \cap \mathcal{F}] \subseteq \begin{pmatrix} [0, 24] \\ [-64, -40] \\ [-56, -40] \end{pmatrix} \subset \begin{pmatrix} [-8, 24] \\ [-64, -34] \\ [-56, -32] \end{pmatrix} = f[z, \mathbf{x}].$$

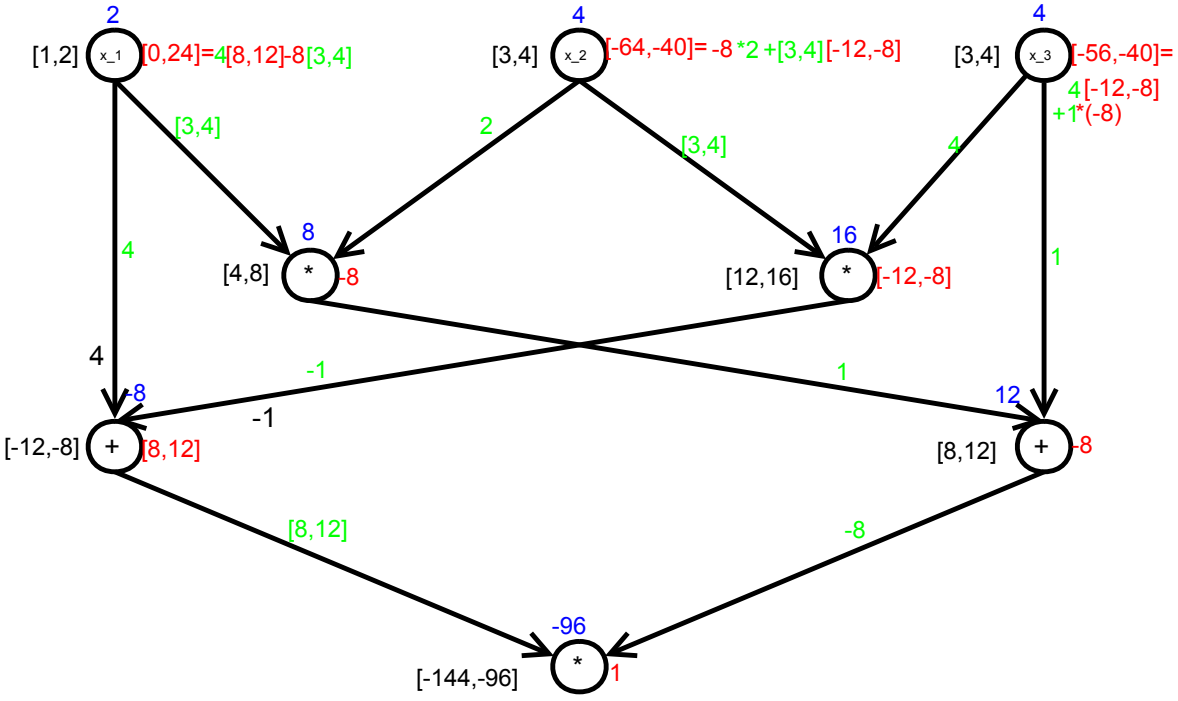


Abbildung 29: Veranschaulichung der Berechnung des Slopes.

6.5 Relaxationen

Betrachten wir wieder unser ursprüngliches Problem

$$\begin{aligned} &\min f(x) \\ &s.t.\ F(x) \in \mathbf{F} \\ &\quad x \in \mathbf{x} \end{aligned} \tag{6.2}$$

so ist eine Relaxation von (6.2) gegeben durch

$$\begin{aligned} &\min r(x) \\ &s.t.\ R(x) \in \mathbf{R} \\ &\quad x \in \mathbf{x} \\ &\quad \mathcal{R} = \{x \in \mathbf{x} | R(x) \in \mathbf{R}\} \supseteq \mathcal{F} \\ &\quad \forall\, x \in \mathcal{F} : r(x) \leq f(x) \end{aligned} \tag{6.3}$$

Ist $\hat{x}$ Lösung von (6.3) und x^* Lösung von (6.2), dann gilt: $r(\hat{x}) \leq f(x^*)$. Gilt $\hat{x} \in \mathcal{F}$ und $r(\hat{x}) = f(\hat{x}) \Rightarrow \hat{x} \in Glob(6.2)$. Diese Eigenschaften kann man auf zweierlei Arten verwenden, um $\mathbf{x}$ zu verkleinern:

- Man verwendet (6.3), um ein kleineres $\mathbf{x}_{neu}$ zu finden mit $\mathbf{x}_{neu} \supseteq \mathbf{R}$ und $\mathbf{x}_{neu} \subset \mathbf{x}$. $\mathbf{x}_{neu}$ kann man zum Beispiel mit linearer Optimierung berechnen.

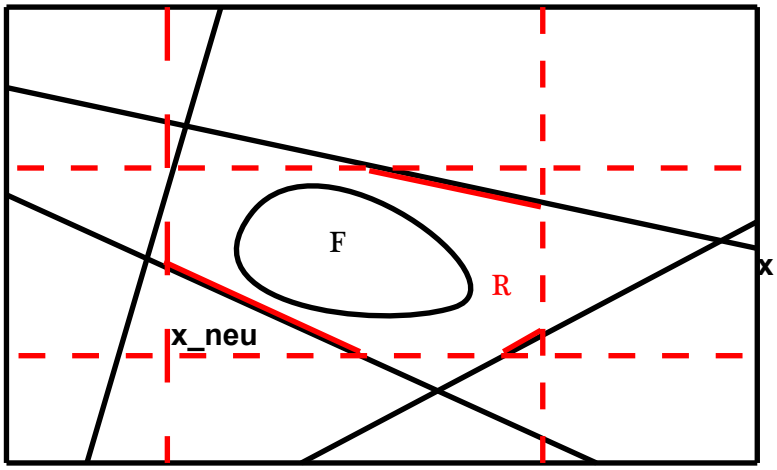


Abbildung 30: Eine Möglichkeit, x_{neu} zu berechnen.

- Man löst (6.3) und erhält $\hat{x}$. Dann überprüft man, ob $\hat{x} \in \mathcal{F}$ und $r(\hat{x}) = f(\hat{x})$ gilt. Wenn ja, dann ist (6.2) gelöst. Wenn nein, dann hat immerhin eine neue implizite Nebenbedingung $f(x) \geq r(\hat{x})$ gefunden, die man zum Beispiel in CP ausnutzen kann.

6.5.1 Lineare Relaxation

In diesem Abschnitt wollen wir Relaxationen mit Hilfe von Slopes erzeugen:

Proposition 6.19. *Sei $f : \mathbb{R}^n \to \mathbb{R}$ eine L^1-Funktion (Lipschitz-stetig). Ist $z \in \boldsymbol{x}$ und $f[z, \boldsymbol{x}] =: s$ ein Slope für f, dann ist die Funktion*

$$\underline{f}(x) := \underline{f} + \sum_{i=1}^{n} \overline{s}_i(\underline{x}_i - z_i) + \frac{\underline{s}_i(\overline{x}_i - z_i) - \overline{s}_i(\underline{x}_i - z_i)}{\overline{x}_i - \underline{x}_i}(x_i - \underline{x}_i)$$

eine lineare Funktion, die f auf $\boldsymbol{x}$ unterschätzt, das heißt $\underline{f}(x) \leq f(x)\ \forall\ x \in \boldsymbol{x}$. Die Funktion

$$\overline{f}(x) := \overline{f} + \sum_{i=1}^{n} \underline{s}_i(\underline{x}_i - z_i) + \frac{\overline{s}_i(\overline{x}_i - z_i) - \underline{s}_i(\underline{x}_i - z_i)}{\overline{x}_i - \underline{x}_i}(x_i - \underline{x}_i)$$

überschätzt f auf $\boldsymbol{x}$. Hier ist $\boldsymbol{f} := f([z, z])$.

Bemerkung 6.20. Für f linear gilt $\underline{f}(x) = f(x) = \overline{f}(x)$.

Beweis. Einfache Rechnung für $n = 1$. □

Wie verwendet man nun Proposition 6.19, um eine lineare Relaxation zu erhalten?

$$\begin{aligned} &\min f(x) \\ &s.t.\ F(x) \in \mathbf{F} \Leftrightarrow \forall\, j : F_j(x) \leq \overline{F} \wedge F_j(x) \geq \underline{F} \\ &\quad x \in \mathbf{x} \end{aligned} \tag{6.4}$$

$$\begin{aligned} &\min \underline{f}(x) \\ &s.t.\ \underline{F}(x) \leq \overline{F}\ \forall\, j \\ &\quad \overline{F}(x) \leq \underline{F}\ \forall\, j \\ &\quad x \in \mathbf{x} \end{aligned} \tag{6.5}$$

(6.5) hat dieselbe Dimension wie (6.4).

Beispiel 6.21. Betrachten wir wieder unser Problem

$$\begin{aligned} &\min f(x) = (4x_1 - x_2x_3)(x_1x_2 + x_3) \\ &s.t.\ f(x) \leq -96 \\ &\quad x \in \begin{pmatrix} [1,2] \\ [3,4] \\ [3,4] \end{pmatrix} \end{aligned}$$

Bei $z = (2,4,4)$ gilt $f(z) = -96, f[z, \mathbf{x}] = \begin{pmatrix} [0,24] \\ [-54,-48] \\ [-56,-32] \end{pmatrix}$.

$$\begin{aligned} \underline{f}(x) &= -96 + 24(1-2) + \frac{24}{1}(x_1 - 1) \\ &\quad + (-48)(3-4) + \frac{(-64)0 - 48}{1}(x_2 - 3) \\ &\quad + (-32)(3-4) + \frac{-32}{1}(x_3 - 3) \leq -96 \\ &\quad \Rightarrow 24(x_1 - 2) - 48(x_2 - 4) - 32(x_3 - 4) \leq 0 \end{aligned}$$

Mit Hilfe dieser Ungleichung können wir nun CP betreiben. Das Resultat ist, dass $x_2, x_3 \in [3,4,4]$. Nun können wir mit folgendem Problem arbeiten:

$$\begin{aligned} &\min \underline{f}(x) \\ &s.t.\ 24(x_1 - 2) - 48(x_2 - 4) - 32(x_3 - 4) \leq 0 \\ &\quad x \in \mathbf{x} \end{aligned}$$

Im Allgemeinen macht es durchaus Sinn, obiges Problem zusätzlich zu betrachten (in unserem Fall reicht aber schon CP aus).

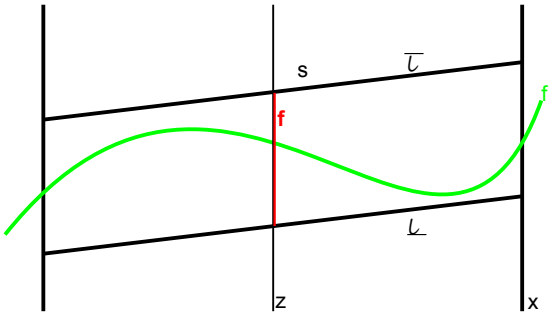
Eigentlich ist $24(x_1 - 2) - 48(x_2 - 4) - 32(x_3 - 4) \leq 0$ redudant, aber die Linearität ermöglicht es, die Ungleichung besser auszunutzen. Dies ist öfters der Fall. Deshalb werden globale Optimierungsprobleme öfters einfacher zu lösen, wenn man redundante Gleichungen und Ungleichungen hinzufügt.

6.5.2 Lineare Relaxationen durch Intervall-Taylor-Entwicklung bis zum Grad 1

Die Idee der wir nun nachgehen werden ist Folgende: Wir wissen, dass $f(x) \in \mathbf{f} + s(x-z)$ mit $s, z \in \mathbb{R}^n, \mathbf{f} \in \mathcal{IR}$ gilt. Wenn wir nun f bei z Taylorentwickeln so sieht dies wie folgt aus:

$$f(x) = f(z) + f'(z)(x-z) + R(z,x)(x-z)^2$$

Für $\mathbf{f}$ ergibt sich damit nun Folgendes: $\mathbf{f} = f(z) + R(z, \mathbf{x})(\mathbf{x} - z)^2$. Auch dieses s und f können auf dem DAG berechnet werden, analog zu Slopes. Hier ist es aber besonders günstig die Rechnung vorwärts zu führen.



$$\underline{l}(x) = \underline{f} + s(x-z), \overline{l}(x) = \overline{f} + s(x-z)$$
$$F_j(x) \in \mathbf{F}_j \Rightarrow \underline{l}(x) \leq \overline{F}_j, \overline{l}(x) \geq \underline{F}_j$$

Aus

$$\underline{f}_j + s(x-z) \leq \overline{F}_j \Rightarrow s(x-z) \leq \overline{F}_j - \underline{f}_j$$
$$\textit{und } \overline{f}_j + s(x-z) \geq \underline{F}_j \Rightarrow s(x-z) \geq \underline{F}_j - \overline{f}_j$$

folgt, dass $s(x-z) \in \mathbf{F}_j - \mathbf{f}_j$ liegt. Unser Originalproblem lässt sich nun äquivalent formulieren zu

$$\begin{aligned} &\min \underline{l}(x) \\ &s.t.\ s^{(j)}(x-z) \in \mathbf{F}_j - \mathbf{f}_j \ \forall j \\ &\quad x \in \mathbf{x} \end{aligned}$$

Dies hat dieselbe Dimension wie unser Originalproblem und dieselbe Anzahl an Nebenbedingungen. Die Nebenbedingungen aus **6.5.1** und **6.5.2** können auch direkt kombiniert werden.

6.5.3 Reformulierungs-Linearisierung

Die Idee hierzu stammt ursprünglich von Mc Cormick aus dem Jahr 1976. Idee: Jedes faktorable Optimierungsproblem lässt sich so umformulieren, dass die Zielfunktion linear ist und jede Nebenbedingung die Form

$$\begin{aligned} &z = \varphi(x) \textit{ oder} \\ &z = x \circ y, \circ \in \{+, *, \wedge\} \\ &z \in \mathbf{z} \end{aligned}$$

hat ($z = x - y \Rightarrow z + y = x, z = \frac{x}{y} \Rightarrow zy = x$; eigentlich ist es auch möglich $\hat{}$ loszuwerden: $z = x^y \Leftrightarrow z = e^{y \ln x} \Leftrightarrow z = e^a$ mit $a = yb, b = \ln x$). Lineare Relaxationen für $z = \varphi(x)$ kann mit Methoden aus Analysis 1 finden (Kurvendiskussion)-siehe dazu Abbildung (31).
Manchmal ist es günstiger, aus einer Gleichung $z = \varphi(x)$ mehrere Ungleichungen zu erzeugen:

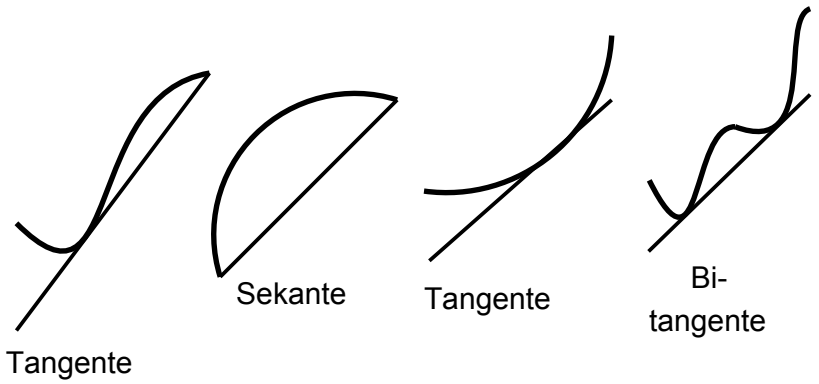
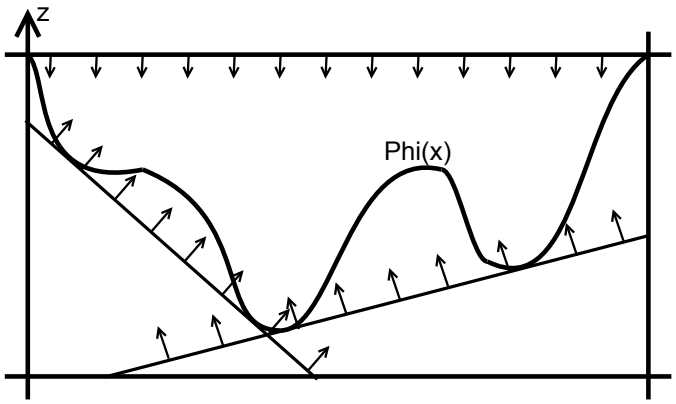


Abbildung 31: Lineare Relaxationen mittels Kurvendikussion und Tangenten, Sekanten und Bitangenten.



$$z = xy, x \in \mathbf{x}, y \in \mathbf{y}$$
$$x - \underline{x} \geq 0, y - \underline{y} \geq 0$$
$$\overline{x} - x \geq 0, \overline{y} - y \geq 0$$
$$0 \leq (x - \underline{x})(y - \underline{y}) = xy - \underline{x}y - x\underline{y} + \underline{x}\underline{y}$$
$$\Rightarrow \underline{x}y + x\underline{y} - \underline{x}\underline{y} \leq xy = z$$

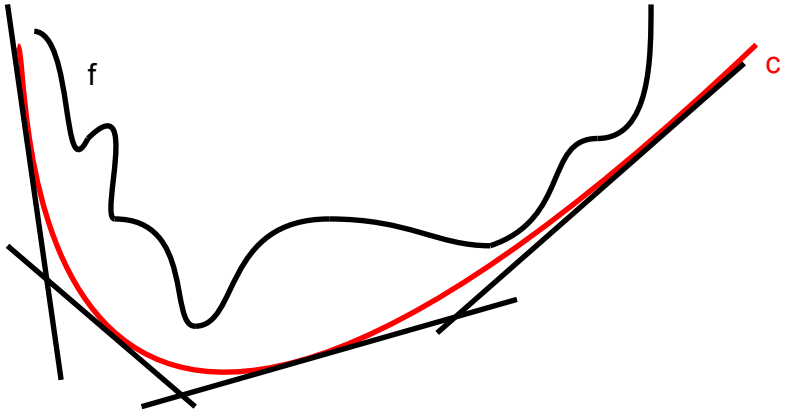
Analog kann man die übrigen 3 möglichen Produkte betrachten, die jeweils wieder zu einer linearen Ungleichung führen. 1983 wurde von Al Khayagal und Falk bewiesen, dass diese vier

Ungleichungen bestmöglich sind, das heißt jede andere gültige lineare Ungleichung, die $z = xy$ relaxiert ist eine Konsequenz aus diesen vier. Kennt man zusätzlich noch eine Einschließung $z \in \mathbf{z}$, dann gibt es noch weitere Ungleichungen.

Proposition 6.22. *Für $z = xy, x \in \boldsymbol{x}, y \in \boldsymbol{y}$ gilt:*

- $z \geq \underline{y}x + \underline{x}y - \underline{x}\,\underline{y}$
- $z \geq \overline{y}x + \overline{x}y - \overline{x}\,\overline{y}$
- $z \leq \underline{y}x + \overline{x}y - \overline{x}\,\underline{y}$
- $z \leq \overline{y}x + \underline{x}y - \underline{x}\,\overline{y}$

Eine weitere Möglichkeit, zu linearen Relaxationen zu kommen, ist die Funktion konvex zu unterschätzen. Diese sind nämlich durch Tangenten beziehungsweise Tangentialhyperebenen sehr leicht zu unterschätzen.
Die konvexe Funktion c kann durch lineare Unterschätzer beliebig gut approximiert werden.

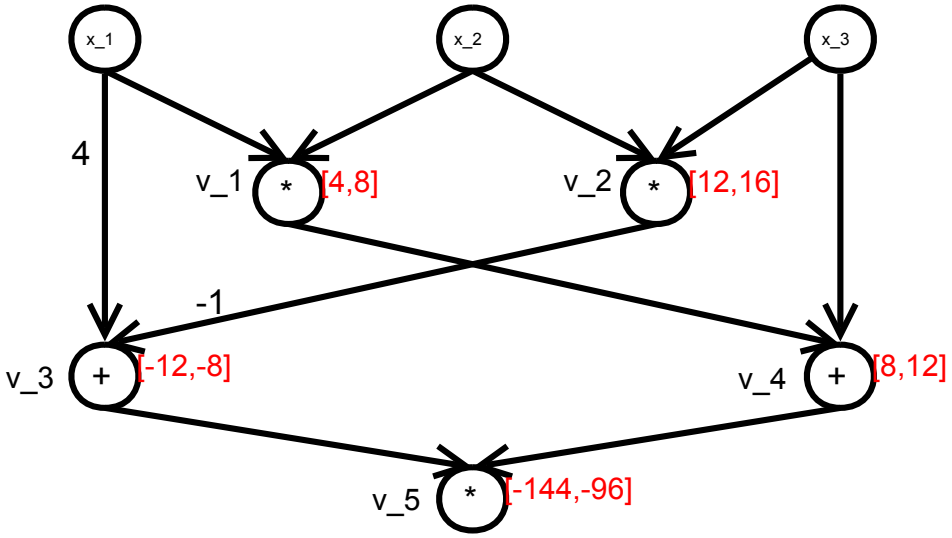


Reformulierungs-Linearisierung ist sehr effizient! Die besten globalen Optimierungspakete, darunter BARON und LINGO, verwenden den Hauptteil der Rechenzeit dafür, lineare Relaxationen zu Erzeugen und zu Lösen. Multilineare Funktionen $f : \mathbb{R} \times \cdots \times \mathbb{R} \to \mathbb{R}$ haben optimale konvexe Unterschätzer, die durch endlich viele (2^n) lineare Nebenbedingungen beschrieben werden können, zum Beispiel für $z = x_1 \cdots x_n$.

Proposition 6.23. *Gilt $x_i \in [0, 1], i = 1, \ldots, n$ dann ist für $z = x_1 \cdots x_n$*

$$z \geq 1 - n + \textstyle\sum_{k=1}^{n} x_k, 0 \leq z \leq x_i \leq 1 \; \forall \; i = 1, \ldots n$$

eine optimale lineare Relaxation.



Beispiel 6.24.

$$
\begin{aligned}
&\min\ v_5 \\
&s.t.\ v_5 = v_3 v_4 \\
&\quad v_4 = v_1 + x_3 \\
&\quad v_3 = 4x_1 - v_2 \\
&\quad v_2 = x_2 x_3 \\
&\quad v_1 = x_1 x_2 \\
&\quad x_1 \in [1,2] \\
&\quad x_2, x_3 \in [3,4]
\end{aligned}
$$

lässt mit Hilfe der Reformulierungs-Linearisierung so anschreiben:

$$
\begin{aligned}
&\min\ v_5\\
&s.t.\ v_5 \geq 8v_3 - 12v_4 + 96\\
&\quad v_5 \geq 12v_3 - 8v_4 + 96\\
&\quad v_5 \leq 8v_3 - 8v_4 + 64\\
&\quad v_5 \leq 12v_3 - 12v_4 + 144\\
&\quad v_4 = v_1 + v_3\\
&\quad v_3 = 4x_1 - v_2\\
&\quad v_1 \geq 3x_1 + x_2 - 3\\
&\quad v_1 \geq 4x_1 + 2x_2 - 8\\
&\quad v_1 \leq 4x_1 + x_2 - 4\\
&\quad v_1 \leq 3x_1 + 2x_2 - 6\\
&\quad v_2 \geq 3x_2 + 3x_3 - 9\\
&\quad v_2 \geq 4x_2 + 4x_3 - 16\\
&\quad v_2 \leq 3x_2 + 4x_3 - 12\\
&\quad v_2 \leq 4x_2 + 3x_3 - 12\\
&\quad x_1 \in [1,2], x_2, x_3 \in [3,4]\\
&\quad v_1 \in [4,8], v_2 \in [12,16]\\
&\quad v_3 \in [-12,-8], v_4 \in [8,12]\\
&\quad v_5 \in [-144,-96]
\end{aligned}
$$

Wie man anhand dieses Beispiels erkennen kann ist die Reformulierungs-Linearisierung meist hochdimensional-dafür aber dünn besetzt!

6.5.4 Semidefinite Relaxationen

Untersuchen wir vorerst einmal was wir unter einem semidefiniten Problem verstehen:

$$
\begin{aligned}
&\min\ f(x)\\
&s.t.\ x \in \mathcal{F}\\
&\quad f : V \to \mathbb{R}\\
&\quad V = \mathbb{R}^n, V = S(n) = \left\{A \in \mathbb{R}^{n\times n} | A^T = A\right\} = Sym(n)\\
&\quad V = Herm(n) = \left\{A \in \mathbb{C}^{n\times n} | A^* = A\right\}\\
&\text{verallgemeinert: } V = \mathbb{R}^n \times Sym(m_1) \times \cdots \times Sym(m_k) \qquad (6.6)\\
&\quad f \text{ ist linear auf } Sym(n), \text{ also :}\\
&\quad f(x) = tr(CX) \text{ für } C \in Sym(n)\\
&\quad \mathcal{F} \ldots \text{Polyeder in Sym(n), wobei die Ungleichungszeichen}\\
&\quad \leq \text{ durch } \preceq \text{ ersetzt werden}\\
&\quad \mathcal{F} = \{X \in Sym(n) | tr(A_i X) \leq b_i \text{ für } i = 1, \ldots, k,\ A_i \in Sym(n), X \succeq 0\}
\end{aligned}
$$

Für dieses Problem verwendbare Nebenbedingungen sind zum Beispiel:

$$\sum_{k=1}^{n} a_k X_k \succeq 0$$
$$||\sum a_k X_k|| \leq c$$
$$||Ax - b||_2 \leq \alpha x + \beta$$
$$||Ax - b||_2 \leq x_i x_j$$
$$A(x) = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \begin{pmatrix} x_1 & x_2 \end{pmatrix} = \begin{pmatrix} x_1^2 & x_1 x_2 \\ x_1 x_2 & x_2^2 \end{pmatrix} \succeq 0 \; \forall \, x_1, x_2$$

$A(x)$ hat Rang 1. Semidefinite Optimierungsprobleme lassen sich mit Inneren-Punkte-Verfahren in polynomialer Zeit lösen.

$$\begin{aligned} &\min f(x) \\ &s.t.\ F(x) \in \mathbf{F} \\ &\quad f, F \text{ polynomial, also haben die Gestalt} \\ &\quad g(x) = \sum_{\alpha \in \mathbb{N}^n} c_\alpha x^\alpha \end{aligned}$$

wobei α ein Multiindex ist mit $|\alpha| = \sum_{i=1}^{n} \alpha_i$, $x^\alpha = x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ Monome sind.

Beispiel 6.25.

$$f(x) = (4x_1x_2x_3)(x_1x_2 + x_3) = 4x_1^2x_2 - x_1x_2^2x_3 + 4x_1x_3 - x_2x_3^2 =$$
$$\begin{matrix} u_{11} = x_1^2 & u_{12} = x_1x_2 & u_{22} = x_2^2 \\ u_{13} = x_1x_3 & u_{23} = x_2x_3 & u_{33} = x_3^2 \end{matrix}$$
$$= 4u_{11}x_2 - u_{12}u_{23} + 4u_{13} - x_2u_{33}$$
$$\begin{pmatrix} u_{11} & u_{12} & u_{13} \\ u_{21} & u_{22} & u_{23} \\ u_{31} & u_{32} & u_{33} \end{pmatrix} \succeq 0$$

Nun werden wir das Vorgehen im obrigen Beispiel etwas systematischer untersuchen: Wähle eine Menge $\{\varphi_1, \ldots, \varphi_N\}$ von Basisfunktionen, definiere $M_{ik}(x) := \omega(x)\varphi_i(x)\varphi_k(x)$ mit $\omega(x) \geq 0$. $M(x)$ ist dann $\succeq 0$, weil $M(x) = \begin{pmatrix} \varphi_1(x) \\ \vdots \\ \varphi_N(x) \end{pmatrix} \begin{pmatrix} \varphi_1(x) & \cdots & \varphi_N(x) \end{pmatrix}$.
Die φ_i wählt man so, dass alle f, F_i als Linearkombinationen der φ_i geschrieben werden

können.

$$
\begin{aligned}
f(x) &= \sum_l c_l \varphi_l \\
F_i(x) &= \sum_l c_l^i \varphi_l,\ (\psi_l = \varphi_l(x), \psi_m = \varphi_k \varphi_l) \\
\min &\sum_l c_l \psi_l \\
s.t. &\sum_l c_l^i \psi_l \in \mathbf{F}_i \\
&M(\psi) \succeq 0
\end{aligned}
$$

wobei hier $M(\psi) \succeq 0$ die semidefinite Relaxation darstellt. $\varphi_i(x)\varphi_i(x) \geq 0$, generell: Sogenannte SOS (sum of squares) Polynome sind immer ≥ 0. Jedes SOS-Polynom $g(x) \geq 0$ ergibt eine lineare Nebenbedingung in den ψ. Außerdem gelten zum Beispiel:

$$
\left.
\begin{array}{l}
F_i(x) - \underline{F}_i \geq 0 \\
F_j(x) - \underline{F}_j \geq 0 \\
\overline{F}_i - F_i(x) \geq 0 \\
\overline{F}_j - F_j(x) \geq 0
\end{array}
\right\} \Rightarrow (F_i(x) - \underline{F}_i)(F_j(x) - \underline{F}_j) \geq 0
$$

Auf diesem Weg erhält man neben der obigen Bedingung noch 3 weitere. Jedes solche Produkt kann in ψ ausgedrückt werden.

$$
\begin{aligned}
f(x) &= (4x_1x_2x_3)(x_1x_2 + x_3) = 4x_1^2x_2 + 4x_1x_3 - x_1x_2^2x_3 - x_2x_3^2 \\
\varphi &: \left\{1, x_1, x_2, x_3, x_1^2, x_2^2, x_3^2, x_1x_2, x_1x_3, x_2x_3\right\} \\
M &= \begin{pmatrix}
1 & x_1 & x_2 & x_3 & x_1^2 & x_2^2 & x_3^2 & x_1x_2 & x_1x_3 & x_2x_3 \\
x_1 & x_1^2 & x_1x_2 & x_1x_3 & x_1^3 & x_1x_2^2 & x_1x_3^2 & x_1^2x_2 & x_1^2x_3 & x_1x_2x_3 \\
 & & & & \vdots & & & & & \\
x_1^2 & x_1^3 & x_1^2x_2 & x_1^2x_3 & x_1^4 & x_1^2x_2^2 & x_1^2x_3^2 & x_1^3x_2 & x_1^3x_3 & x_1^2x_2x_3 \\
 & & & & \vdots & & & & &
\end{pmatrix}
\end{aligned}
$$

$\psi_{i,j} = M_{i,j}$ für $j \geq i$. Doppelte werden „gestrichen“, also zum Beispiel $\psi_{2,4} = \psi_{1,9}, \psi_{2,3} = \psi_{1,8}$.

$$
\begin{aligned}
f &= 4\psi_{2,8} + 4\psi_{1,9} - \psi_{8,10} + \psi_{4,10} \\
M &= \begin{pmatrix}
1 & x_1 & x_2 & x_3 & \psi_{1,5} & \psi_{1,6} & \psi_{1,7} & \psi_{1,8} & \psi_{1,9} & \psi_{1,10} \\
x_1 & \psi_{1,5} & \psi_{1,8} & \psi_{1,9} & \psi_{2,5} & \psi_{2,6} & \psi_{2,7} & \psi_{2,8} & \psi_{2,9} & \psi_{2,10} \\
x_2 & \psi_{1,8} & \psi_{1,6} & \psi_{1,10} & \psi_{2,8} & \psi_{3,6} & \psi_{2,6} & \cdots & & \\
 & & & & \vdots & & & & &
\end{pmatrix}
\end{aligned}
$$

mit

$$\begin{aligned} x_1 &\in [1,2] \\ x_2, x_3 &\in [2,3] \\ x_1^2 + x_2^2 &\geq 0 \\ x_1^2 \geq 0 &\Rightarrow \psi_{1,5} \geq 0 \\ \psi_{1,6} &\geq 0 \\ &\vdots \\ M &\succeq 0 \end{aligned}$$

Es sind also relativ viele der Variablen gut einschränkbar. Mann kann auch noch zusätzlich viele andere Ungleichungen wie zum Beispiel die folgenden Ungleichungen verwenden:

$$(x_1 \mp x_2)^2 \geq 0 \Rightarrow x_1^2 \mp 2x_1x_2 + x_2^2 \geq 0 \Rightarrow \psi_{1,5}^2 \mp 2\psi_{1,8} + \psi_{1,6} \geq 0$$

Es bieten sich außerdem noch SOS Polynome an um die Anzahl der Nebenbedingungen zu erhöhen:

$$(x_2^2 - x_3)^2 + (x_1^2 - 5x_3)^2 \geq 0 \Leftrightarrow x_2^4 - 2x_2^2x_3 + x_3^2 + x_1^4 - 10x_1^2x_3 + 25x_3^2 \geq 0$$

Auf diese Weise können wir beliebig viele lineare Nebenbedingungen hinzufügen. [2]

6.5.5 Konvexe Relaxationen

Wir werden uns in diesem Abschnitt mit folgendem Problem befassen

$$\begin{aligned} &\min f(x) \\ &s.t.\ F(x) \in \mathbf{F} \\ &\quad x \in \mathbf{x} \\ &\quad \mathcal{F} = \{x \in \mathbf{x} | F(x) \in \mathbf{F}\} \end{aligned}$$

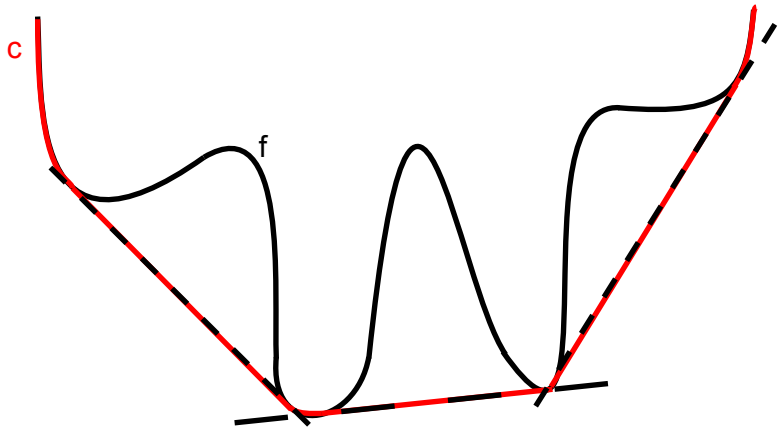
Die beste konvexe Relaxation für dieses Problem ist

$$\begin{aligned} &\min c(x) \\ &s.t.\ x \in \mathbf{C} \\ &\text{mit } \mathbf{C} = ch(\mathcal{F}) \\ &\quad c(x) \text{ist die konvexe Hülle von } f, \text{das heißt die} \\ &\quad \text{größte konvexe Funktion, die } f \text{ unterschätzt.} \end{aligned}$$

Betrachtet man die Menge $epif = \{(x,t) | f(x) \leq t\}$ den Epigraphen von f, dann gilt: f ist konvex $\Leftrightarrow$ $epif$ ist konvex. $ch(epif) =: \mathbf{G}$. Ist f nach unten beschränkt, dann ist $\partial\mathbf{G}$ der Graph einer Funktion: c (unter einfach Annahmen an f).

Bie Bestimmung von $\mathbf{C}$ ist im allgemeinen Fall leider NP-hart. Das Berechnen einer allgemeinen konvexen Relaxation durch Propagation im DAG ist nicht (leicht) möglich, da

[2] Man sollte so viele Nebenbedingungen wie Variable hinzufügen damit das Problem vernünftig eingeschränkt wird. Je mehr man nimmt desto genauer kommt man an das Originalproblem heran.



zum Beispiel das Produkt zweier konvexer Funktionen im Allgemeinen nicht konvex ist (Natürlich könnte man eine „Reformulierungs-Konvexifizierung“ analog zur Reformulierungs-Linearisierung durchführen. Allerdings ist es für allgemeine konvexe Probleme ein Problem, wenn die Dimension stark ansteigt.)
Ist $f \in \mathcal{C}^2$ und gilt $Hf \succ 0$ in $\mathbf{x}$, dann ist f strikt konvex in $\mathbf{x}$. Unter anderem ist das dann der Fall, wenn es $D = diag(d)$ mit $d > 0$ gibt und $Hf(x) \succ D \; \forall \; x \in \mathbf{x}$. Wichtig:

$$\underbrace{\begin{pmatrix} 1 & 4 \\ 4 & 1 \end{pmatrix}}_{\text{indefinit}} > \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}}_{\text{positiv definit}} \text{ aber } \begin{pmatrix} 1 & 4 \\ 4 & 1 \end{pmatrix} \not\succ \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Bei der Berechnung einer konvexen Relaxation muss man im ersten Schritt möglichst all Nebenbedingungen (Funktionen) finden, die bereits konvex sind, um das Problem nicht unnötig zu relaxieren. Die Erkennung, ob eine Funktion g konvex ist, ist nicht trivial. Man kann im DAG die Regeln zur Konvexitätserkennung propagieren:
Ist $f : \mathbb{R} \to \mathbb{R}$ konvex in $\mathbf{x}$, dann gilt:

(i) Ist g linear mit Werten in $\mathbf{x}$, dann ist $f \circ g$ konvex.

(ii) Ist g konvex mit Werten in $\mathbf{x}$ und ist f auf $\mathbf{x}$ monoton wachsend, dann ist $f \circ g$ konvex.

(iii) Ist g konkav mit Werten in $\mathbf{x}$ und ist f monoton fallens auf $\mathbf{x}$, dann ist $f \circ g$ konvex.

Ist $f : \mathbb{R} \to \mathbb{R}$ konkav in $\mathbf{f}$, dann gilt:

(i) Ist g konkav mit Werten in $\mathbf{x}$ und ist f auf $\mathbf{f}$ monoton wachsend, dann ist $f \circ g$ konkav.

(ii) Ist g konkav mit Werten in $\mathbf{x}$ und ist f auf $\mathbf{f}$ monoton fallend, dann ist $f \circ g$ konkav.

Wenn wir nun eine Verallgemeinerung für mehrdimensionale Funktionen betrachten so gibt es hier folgende Regeln für die Erkennung der Konvexität: Seien $h : \mathbb{R}^k \to \mathbb{R}, g : \mathbb{R}^n \to \mathbb{R}^k$, $f := h \circ g : \mathbb{R}^n \to \mathbb{R}$ und $\widetilde{h}$ die erweiterte Funktion von h:

$$\widetilde{h}(x) := \begin{cases} h(x) \text{ falls } x \in \text{ dom } h \\ +\infty \text{ sonst} \end{cases} \quad \text{für } h \text{ konvex.}$$

$$\widetilde{h}(x) := \begin{cases} h(x) \text{ falls } x \in \text{ dom } h \\ -\infty \text{ sonst} \end{cases} \quad \text{für } h \text{ konkav.}$$

Für $n > 1$ allgemein und $k = 1$ gilt:
f ist konvex, falls

(i) h konvex, $\widetilde{h}$ monoton wachsend und g konvex sind.

(ii) h konvex, $\widetilde{h}$ monoton fallend und g konkav sind.

f ist konkav, falls

(i) h konkav, $\widetilde{h}$ monoton wachsend und g konkav sind.

(ii) h konkav, $\widetilde{h}$ monoton fallend und g konvex sind.

Beispiel 6.26. Einfach Beispiele:

(i) g konvex $\Rightarrow e^g$ ist konvex.

(ii) g konkav, $g > 0 \Rightarrow \log(g)$ ist konkav.

(iii) g konkav, $g > 0 \Rightarrow \frac{1}{g}$ ist konvex.

(iv) g konvex, $g \geq 0$, $p \geq 1 \Rightarrow g^p$ ist konvex.

(v) g konvex, dann ist $-\log(-g)$ konvex auf $\{x | g(x) \leq 0\}$

Für $n > 1$ und $k > 1$ gilt:
f ist konvex, falls h konvex ist:

(i) $\widetilde{h}$ ist monoton wachsend in Argument i und g_i konvex ist $\forall\, i$.

(ii) $\widetilde{h}$ ist monoton fallend in Argument i und g_i konkav ist $\forall\, i$.

f ist ???**konkav**???, falls h konkav ist:

(i) $\widetilde{h}$ ist monoton wachsend in Argument i und g_i konkav ist $\forall\, i$.

(ii) $\widetilde{h}$ ist monoton fallend in Argument i und g_i konvex ist $\forall\, i$.

Beispiel 6.27. Seien g_i konvex auf $\mathbb{R}^n$. Dann ist die punktweise Summe der r größten g_i konvex. Für konvexe g_i ist $h(z) := \log(\sum_{i=1}^{k} e^{g_i})$ konvex. Für $p \geq 1$ ist $\sqrt[p]{\sum_{i=1}^{k} g_i(x)^p}$ konvex, falls die g_i konvex und $g_i \geq 0$ sind.
Sind die g_i konkav und $g_i \geq 0$, dann ist das geometrische Mittel der g_i konkav.

Eine andere Möglichkeit, die Konvexität einer Funktion f zu zeigen, ist die postiv Definitheit ihrer Hesse-Matrix zu beweisen. Nach dem Satz von Taylor gilt:

$$f(x) = f(z) + f'(z)(x - z) + \frac{1}{2} Hf(\xi)(x - z, x - z)$$

$$f \text{ konvex in } \mathbf{x} \Leftarrow Hf(\xi) \succ 0 \; \forall\, \xi \in \mathbf{x}$$

$$\mathbf{H} \supseteq Hf(\mathbf{x})$$

Wie können wir verifizieren, ob jede symmetrische Matrix $H \in \mathbf{H}$ positiv definit ist? Uns interessieren an dieser Stelle aber auch nur wirklich symmetrische Matrizen und nicht Matrizen wie die folgende:

$$\begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} \in \begin{pmatrix} [0,1] & [0,1] \\ [0,1] & [0,1] \end{pmatrix}$$

A ist positiv definit, wenn

- $A = LL^T$ für eine reguläre untere Dreiecksmatrix L gilt.
- $A = BB^T$ für eine reguläre Matrix B gilt.
- alle Eigenwerte von A positiv sind.
- $x^T Ax > 0 \; \forall \; x \neq 0 \Leftrightarrow x^T Ax > 0 \; \forall \; ||x|| = 1$ gilt.
- A diagonaldominant ist.
- $A = B^T B$ für $B \in \mathbb{R}^{m \times n}$ mit $rgB = m \geq n$ gilt.
- $B^{-1}AB^{-T} \succ 0$ für ein $B : (x^T B^{-1})A(B^{-T}x) \succ 0$ falls $x \neq 0$ ist $\Rightarrow y := B^{-T}x \Rightarrow y^T Ay \succ 0$ falls $y \neq 0$, weil B invertierbar ist.

Bevor wir das nächste Kriterium zur Überprüfung auf positive Definitheit anführen können benötigen wir noch eine Definition:

Definition 6.28. Wir definieren die **Magnitude** einer Box $\mathbf{x}$ als: $|\mathbf{x}| := \max(|\underline{x}|, |\overline{x}|)$.
Wir definieren die **Mignitude** einer Box $\mathbf{x}$ als: $< \mathbf{x} >:= \begin{cases} \min(|\underline{x}|, |\overline{x}|) \text{ falls } 0 \notin \mathbf{x} \\ 0 \text{ sonst} \end{cases}$

Man kann auch den Satz von Gerschgorin für Intervallmatrizen verwenden: Falls $\underline{H_{ii}} > \sum_{j \neq i} |\mathbf{H}_{ij}| \; \forall \; i$ gilt dann ist jede Matrix $H \in \mathbf{H}$ positiv definit.
Bestimme $\check{H} = LL^T$ für die Mittelpunktsmatrix $\check{H}$. Geht das schief, dann ist $\mathbf{H} \not\succ 0$. Sonst berechne $A = L^{-1}\mathbf{H}L^{-T}$ (das kann man im Prinzip durch den DAG propagieren). Das heißt $A \supseteq L^{-1}Hf(\xi)L^{-T} \; \forall \; \xi \in \mathbf{x}$. Dann verwendet man den Satz von Gerschgorin für A. Alternativ zeigt man, dass A eine H-Matrix ist (für genauere Informationen zur konkreten Vorgehensweise siehe VO Intervallanalysis). Alternativ zu $\check{H}$ kann man auch $Hf(\check{x})$ berechnen (das ist dann aber nur eine Punktauswertung).

Wie bestimmt man nach Überprüfung der Konvexität eine konvexe Relaxation von nichtkonvexen f?

Die 1. Variante stammt von Floudas und heißt: αBB.
In der ursprünglichen Version von αBB wird jede Funktion und jede Nebenbedingung als Differenz konvexer Funktionen geschrieben:

$$\begin{aligned} f(x) = & f_c(x) - f_v(x) + f_l(x) \\ & f_c, f_v \text{ konvex} \\ & f_l \text{ linear} \end{aligned}$$

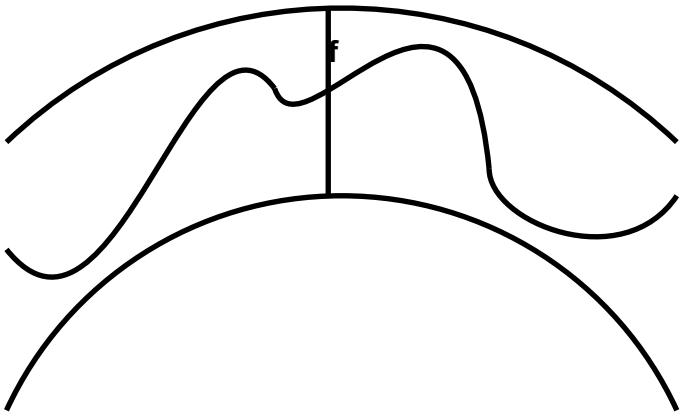
Da das aber nicht immer geht und zudem auch nicht eindeutig ist hat man diese Version weiterentwickelt und die neueste Version fügt noch einen Term hinzu:

$$f(x) = f_c(x) - f_v(x) + f_l(x) + f_r(x)$$
$$f_r(x) \text{ allgemein nichtlinear}$$

f_c ist schon konvex, $-f_v$ kann man als konkave Funktion linear unterschätzen und f_l ist linear. $f_{r,rel}(x) = f_r(x) + \frac{1}{2} \underbrace{(x-\underline{x})^T}_{\geq 0} \underbrace{D}_{\geq 0} \underbrace{(x-\underline{x})}_{\leq 0}, D = diag(d), d \geq 0$. Ist D groß genug, dann ist $f_{r,rel}$ konvex wegen $Hf_{r,rel}(x) = Hf_r(x) + D$.
Wie wählt man D? αBB wählt $D = \alpha I$ mit

$$\alpha \geq |\lambda_n(Hf_r(x))| + \epsilon \; \forall \; x \in \mathbf{x} \text{ falls}$$
$$\lambda_n(Hf_r(x)) < 0 \text{ für ein } x \in \mathbf{x}.$$

Am günstigen wäre es $\inf \bigcup_{x \in \mathbf{x}} \sigma(Hf_r(x))$ (mit $\sigma \ldots$ Spektrum) zu berechnen. Etwas schlechter ist $\inf_{H \in (\mathbf{H} \cap Sym(n))} \lambda_{\min}(H)$ für eine Intervall-Hesse-Matrix $\mathbf{H}$ von f_r zu berechnen.
Eine praktikablere Möglichkeit ist zum Beispiel $\check{H} = LL^T - E$ mit E diagonal und > 0. Berechne $L^{-1}(\mathbf{H} + E)L^{-T}$ und bestimme β so, dass $L^{-1}(\mathbf{H} + E)L^{-T} + \beta I \succ 0$.
$L^{-1}(\mathbf{H} + E)L^{-T} + \beta I = L^{-1}(\mathbf{H} + \underbrace{E + \beta LL^T}_{\leq \alpha I})L^{-T}$ und
$f_{r,rel}(x) = f_r(x) + \frac{1}{2}(x-\underline{x})^T(E + \beta LL^T)(x-\overline{x})^T$.
Eine Alternative um f_r zu bestimmen ist, Taylor-Formen 2. Ordnung zu Hilfe zu nehmen:
$f(x) \in \mathbf{f} + g^T(x-z) + \frac{1}{2}(x-z)^T G(x-z)$ mit G, g dünn und G symmetrisch $\forall \; x \in \mathbf{x}$.



$$\underline{f} + g^T(x-z) + (x-z)^T G(x-z) \leq f(x) \; \forall \; x \in \mathbf{f}, G = LL^T - E$$
$$\Rightarrow (x-z)^T G(x-z) = ||L^T(x-z)||_2^2 - \sum_i E_{ii}(x_i - z_i)^2 - (x_i - z_i)^2 \text{ für } x \in \mathbf{x}\text{-siehe dazu Abbildung (32).}$$

$$l_i(x) = -(\underline{x}_i - z_i)^2 - \frac{(\overline{x}_i - z_i)^2 - (\underline{x}_i - z_i)^2}{\overline{x}_i - \underline{x}_i}(x_i - \underline{x}_i)$$
$$f(x) \geq \underline{f} g^T(x-z) + \frac{1}{2}||L^T(x-z)||_2^2 - \sum_i E_{ii} l_i(x).$$

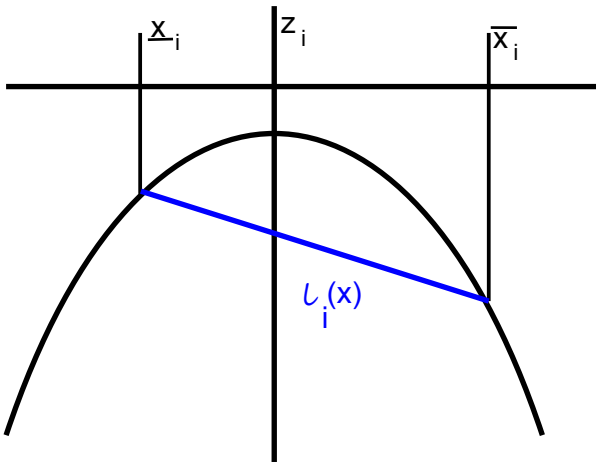


Abbildung 32: Veranschaulichung der Verwendung der $l_i(x)$.

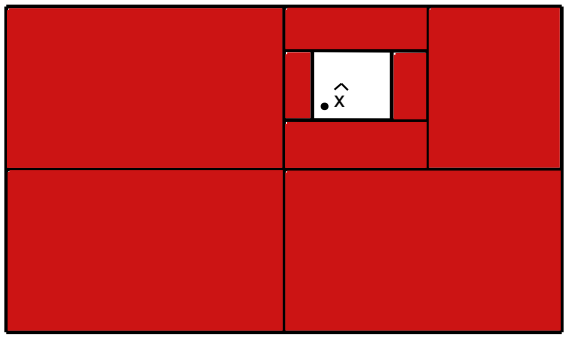
Welche Art der Unterschätzung ist die günstigste?

- Konstante Funktionen: $f(x) \in \mathbf{f} \Rightarrow \underline{f} \leq f(x)\ \forall\ x \in \mathbf{x}$. Überschätzung: $\mathcal{O}(rad(\mathbf{x}))$.
- Lineare Funktionen: Zum Beispiel $f(x) \in f(x) + s^T(x - z)$. Überschätzung: $\mathcal{O}(rad(\mathbf{x})^2)$.
- Konvexe oder quadratische Funktionen. Überschätzung: $\mathcal{O}(rad(\mathbf{x})^3)$.

Also scheint für größere Boxen die konstante Einschließung, für „mittelgroße“ die lineare Unterschätzung und für kleine Boxen die quadaratische beziehungsweise konvexe Unterschätzung die beste zu sein.

7 Der Cluster-Effekt

Ist $\hat{x}$ ein globales Minimum, so sind bei $\hat{x}$ die Karush-John Bedingungen erfüllt ???**in den Anhang**???, die ein quadratisches nichtlineares Gleichungssystem ergeben. Was tut ein Branch and Bound beziehungsweise Branch and Reduce Verfahren?
Die roten Boxen werden nach dem Teilen sukzessive ausgeschlossen. Es wird immer schwie-



riger, Boxen auszuschließen, jeh näher die Box bei $\hat{x}$ liegt. Betrachten wir das Problem

$$\min f(x)$$
$$s.t.\ x \in \mathbf{x}$$

und sei $\hat{x}$ das globale Minimum. Ist $f \in \mathcal{C}^3$ dann gilt:
$f(x) = f(\hat{x}) + \nabla f(\hat{x})(x-\hat{x}) + \frac{1}{2}(x-\hat{x})^T Hf(\hat{x})(x-\hat{x}) + \mathcal{O}(||x-\hat{x}||^3)$. Es sei $\hat{x} \in \mathbf{x}^\circ$
$\Rightarrow \nabla f(\hat{x}) = 0 \Rightarrow f(x) = f(\hat{x}) + \frac{1}{2}(x-\hat{x})^T G(x-\hat{x}) + \mathcal{O}(||x-\hat{x}||^3)$. Nehmen wir an, wir könnten auf Boxen vom Radius ε die Funktion f mit einer Genauigkeit $\Delta = K\varepsilon^{s+1}$ (mit $\varepsilon \leq 2, s \in \{0,1,2\}, K \in \mathbb{R}^+$) abschätzen. In diesem Fall können wir die Box $\mathbf{y}$ nicht ausschließen, falls $\forall\, x \in \mathbf{y} : \frac{1}{2}(x-\hat{x})^T G(x-\hat{x}) + \mathcal{O}(||x-\hat{x}||^3) \leq \Delta$ ist.
$\frac{1}{2}(x-\hat{x})^T G(x-\hat{x}) \leq 0$ mit $G \succ 0 \Rightarrow G = Q^T \Lambda Q$ mit Λ diagonal und > 0, $Qz := x - \hat{x}$,
$y := \Lambda^{\frac{1}{2}} z \Rightarrow z^T \Lambda z \leq 2\Delta \Rightarrow ||y||_2^2 \leq 2\Delta \Rightarrow ||y|| \leq \sqrt{2\Delta}$. Alle Boxen, die $||y|| \leq \sqrt{2\Delta}$ erfüllen, können vom Algorithmus nicht ausgeschlossen werden. Wir schätzen die Anzahl der Boxen vom Radius ε ab, die benötigt werden, um dieses Gebiet zu überdecken:

$$Vol\ y = \frac{\pi^{\frac{n}{2}}}{\Gamma(\frac{n}{2}+1)} \sqrt{(2\Delta)^n}$$
$$Vol\ z = \frac{\pi^{\frac{n}{2}}}{\Gamma(\frac{n}{2}+1)} \sqrt{\frac{(2\Delta)^n}{\det(G)}} = Vol\ x$$

Eine Box vom Radius ε hat höchstens Volumen $(2\varepsilon)^n$, also benötigt man mindestens

$$\frac{(\frac{\pi}{2})^{\frac{n}{2}}}{\Gamma(\frac{n}{2}+1)} \frac{1}{\varepsilon^n} \sqrt{\frac{\Delta^n}{\det(G)}}$$

Boxen, um das Volumen zu überdecken, und alle diese Boxen können nicht ausgeschlossen werden. Setzen wir für Δ ein, so erhalten wir für die Anzahl der nicht ausschließbaren Boxen

$$N \approx C\varepsilon^{\frac{n(s-1)}{2}} \frac{1}{\sqrt{\det(G)}} = \begin{cases} C\varepsilon^{-\frac{n}{2}} (\det(G))^{-\frac{1}{2}} & \text{für } s = 0 \\ C(\det(G))^{-\frac{1}{2}} & \text{für } s = 1 \\ C\varepsilon^{\frac{n}{2}} (\det(G))^{-\frac{1}{2}} & \text{für } s = 2 \end{cases}$$

Für $\varepsilon \to 0$ geht für $s = 0$ die Anzahl der Boxen $\to \infty$, die rund um $\hat{x}$ nicht ausgeschlossen werden können (Dies ist der sogenannte Cluster-Effekt). Das ist eine Methode 0. Ordnung.
Für $s = 1$ bleibt die Anzahl der Boxen rund um $\hat{x}$ im Wesentlichen konstant und hängt von der Determinante von G bei $\hat{x}$ ab. Das ist eine Methode 1. Ordnung.
Erst ab $s = 2$ ist damit zu rechnen, dass für ε klein genug alle Boxen rund um $\hat{x}$ ausgeschlossen werden können. Das ist eine Methode 2. Ordnung. Wir benötigen also Methoden 2. Ordnung, um den Cluster-Effekt völlständig zu vermeiden (wenigstens in nicht-degenerierten Fällen: Gilt nämlich $\det(G) = 0$, dann sind die Abschätzungen sinnlos). Nachteile der Methoden 2. Ordnung:

- Aufwand ist groß-bis $\mathcal{O}(n^4)$.
- Auf großen Boxen wird die Überschätzung enorm groß.

7.1 Back-Boxing

Ist x^* der bislang beste gefundene Punkt, dann tritt der Cluster-Effekt auch in einer Umgebung von x^* auf. Die Idee von Back-Boxing ist, das gleich von vornherein zu vermeiden. Um Degeneration zu vermeiden, seien die hinreichenden Optimalitätsbedingungen 2. Ordnung ???**in den Anhang**??? bei x^* erfüllt. Back-Boxing wählt eine Box $\mathbf{z}_0 \supseteq x^*$ und versucht, mit Methoden 2. Ordnung in einem ersten Schritt zu beweisen, dass x^* das globale Minimum von

$$\begin{aligned} &\min f(x) \\ &s.t.\ x \in \mathbf{z}_0 \end{aligned}$$

ist (Wie? Mehr dazu später!). Meist wählt man $\mathbf{z}_0$ so, dass $x^* = \hat{z}_0$ und $rad(\mathbf{z}_0) = \sqrt{eps}$. Danach wählt man iterativ Boxen $\mathbf{z}_n$ so, dass $\mathbf{z}_n \supseteq \mathbf{z}_{n-1}$ gilt und mit Methoden 2. Ordnung ein Reduktionsschritt $\mathbf{z}_n \to \mathbf{z}_n^* \subseteq \mathbf{z}_{n-1}$ möglich ist. Meist ist $\mathbf{z}_n$ die Box mit doppeltem Radius von $\mathbf{z}_{n-1}$. Das ist dann ein Beweis, dass $\mathbf{z}_n$ auf $\mathbf{z}_0$ und eventuell auch auf x^* reduzierbar ist. Die Iteration bricht ab, wenn kein $\mathbf{z}_k \supseteq \mathbf{z}_{k-1}$ gefunden werden kann, oder falls $rad(\mathbf{z}_{k-1}) > \delta$, wobei δ eine Grenze ist, ab der Methoden 1. Ordnung wahrscheinlich funktionieren.

7.2 Ausschlussgebiete

Sei $G : \mathbb{R}^n \to \mathbb{R}^n$ gegeben und $\hat{x}$ eine Nullstelle von G (so kann zum Beispiel G die Karush-John-Bedingungen beschreiben). Dann gilt (für $G \in \mathcal{C}^2$):
$G(x) = G(\hat{x}) + JG(\hat{x})(x - \hat{x}) + \frac{1}{2}HG(\hat{x})(x - \hat{x}, x - \hat{x}) + \mathcal{O}(||x - \hat{x}||^3)$. Betrachten wir nun unser Problem

$$\begin{aligned} &\min f(x) \\ &s.t.\ F(x) \in \mathbf{F} \end{aligned}$$

Die Karush-John-Bedingungen führen auf ein nichtlineares Gleichungssystem

$$\begin{aligned} G(x) &= 0 \\ G : \mathbb{R}^n &\to \mathbb{R}^n \end{aligned}$$

wobei hier G im Allgemeinen $\mathcal{C}^2$ ist. Nun man kann analog zur Taylor-Entwicklung mit Slopes vorgehen: $G(x) = G(z) + G[z, x](x - z)$. An dieser Stelle kann man $G[z, x]$ $\mathcal{C}^1$ wählen.

$G[z,x] = G[z,z'] + \sum_{k=1}^{n} \underbrace{G_k[z,z',x]}_{Slope2.Ordnung} (x_k - z'_k)$. Man ermittelt also einen „Slope vom Slope".
Im Spezialfall $z = z'$ gilt:
$G(x) = G(z) + (G'(z) + \sum_k (x_k - z_k) G_k[z,z,x])(x - z)$.

Beispiel 7.1. $G(x) := \begin{pmatrix} x_1^2 + x_2^2 - 25 \\ x_1 x_2 - 12 \end{pmatrix}$. Nehmen wir in diesem Fall das Zentrum
$z = \begin{pmatrix} 3 \\ 4 \end{pmatrix} \Rightarrow G(z) = 0$.

$$G'(x) = \begin{pmatrix} 2x_1 & 2x_2 \\ x_2 & x_1 \end{pmatrix} \Rightarrow G'(z) = \begin{pmatrix} 6 & 8 \\ 4 & 3 \end{pmatrix}$$
$$G[z,x] = \begin{pmatrix} x_1 + 3 & x_2 + 4 \\ x_2 & 3 \end{pmatrix}$$

Wenn man nun $G[z,x]$ zuerst nach x_1 $(= G_1)$ und dann nach x_2 ableitet so erhält man

$$G_1[z,z,x] = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \text{ und } G_2[z,z,x] = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Satz 7.2. *Sei $z \in \boldsymbol{z} \subseteq \boldsymbol{x}$. Gibt es $C \in \mathbb{R}^{n \times n}$ mit der Eigenschaft, dass der Krawczyk-Operator $K(\boldsymbol{z}, \boldsymbol{x}) := z - CG(x)(CG[\boldsymbol{z}, \boldsymbol{x}] - \mathcal{I})(\boldsymbol{x} - z)$ die Eigenschaft $K(\boldsymbol{z}, \boldsymbol{x}) \subseteq \boldsymbol{x}$ hat, dann enthält $\boldsymbol{x}$ eine Nullstelle von G. Gilt weiters, dass $K(\boldsymbol{x}, \boldsymbol{x}) \subseteq \boldsymbol{x}^\circ$, so enthält $\boldsymbol{x}$ genau eine Nullstelle von G.*
Hat G in $\boldsymbol{x}$ eine Nullstelle, dann auch in $K(\boldsymbol{z}, \boldsymbol{x})$. Gilt $K(\boldsymbol{z}, \boldsymbol{x}) \cap \boldsymbol{x} = \emptyset$, dann enthält $\boldsymbol{x}$ keine Nullstelle von G.

Beweis. Anwendung des Brouwerschen Fixpunktsatzes. □

Wollen nun versuchen, die Intervallmatrix „loszuwerden" (vergleiche mit der „Taylor-Entwicklung"):

$$\begin{aligned} K(\mathbf{z}, \mathbf{x}) &= z - CG(x) - (C(G'(z) + \sum_k (x_k - z_k) G_k[\mathbf{z}, \mathbf{z}, \mathbf{x}]) - \mathcal{I})(\mathbf{x} - z) = \\ &= z - CG(x) - (CG'(\mathbf{z}) + \sum_k (x_k - z_k) CG_k[\mathbf{z}, \mathbf{z}, \mathbf{x}] - \mathcal{I})(\mathbf{x} - z) = \\ &\text{also } C \approx (G'(\mathbf{z}))^{-1} \text{ und } z \text{ so, dass } G(z) \approx 0 \\ &= z - CG(x) - (CG'(\mathbf{z}) - \mathcal{I})(\mathbf{x} - z) + \sum_k (\mathbf{x}_k - \mathbf{z}_k) CG_k[\mathbf{z}, \mathbf{z}, \mathbf{x}](\mathbf{x} - z) \end{aligned}$$

Wir suchen $\overline{b}, \underline{b}, B_0, B_0', B_k$ mit

$$\begin{aligned} \overline{b} &\geq |CG(z)| \geq \underline{b} \\ B_0 &\geq |CG'(\mathbf{z} - \mathcal{I})| \\ B_0' &\geq |CG'(\mathbf{z})| \\ B_k &\geq |CG_k[\mathbf{z}, \mathbf{z}, \mathbf{x}]| \ \forall\ x \in \mathbf{x}, \forall\ k \end{aligned}$$

Proposition 7.3. *Für jede Nullstelle $x \in \boldsymbol{x}$ von G erfüllt die Abweichung $s := (x - z)$ die Ungleichung $0 \leq s \leq (B_0 + \sum_k B_k(x))s + \overline{b}$.*

Beweis. $G[z,x](x-z) = G(x) - G(z) = -G(z)$

$$\begin{aligned}
\Rightarrow -(x-z) &= -(x-z) + C(G[z,x](x-z) + G(z) + G'(z)(x-z) - G'(z)(x-z)) = \\
&= C(G[z,x] - G'(z))(x-z) + CG(x) + (CG'(x) - I)(x-z) = \\
&= (CG'(z) - I + \sum_k (x_k - z_k) CG_k[z,z,x])(x-z) + CG(z) \\
\Rightarrow s = |x-z| &\leq (|CG'(z) - I| + \sum s_k |CG_k[z,z,x]|) .... term fehlt hier! \\
\Rightarrow s &\leq (B_0 + \sum_k s_k B_k(x))s + \bar{b}
\end{aligned}$$

□

Satz 7.4. *Sei* $0 < u \in \mathbb{R}^n$ *so gewählt, dass*

$$(B_0 + \sum_k a_k \overline{B}_k)u + \bar{b} \leq u \tag{7.1}$$

mit $B_k(x) \leq \overline{B}_k \ \forall \ x \in M_u$, *wobei* $M_u := \{x \in X || x - z| \leq u\}$, *dann hat* G *eine Nullstelle* $x \in M_u$.

Beweis. Für x definieren wir $K(x) := x - CG(x)$. Sei $x \in M_u$.

$$\begin{aligned}
K(x) &= x - CG(x) = z - CG(z) - (CG[z,x] - \mathcal{I})(x-z) = \\
&= z - CG(z) - (C'G(z) - \mathcal{I} + \sum_k (x_k - z_k)(CG_k[z,z,x])(x-z) \\
\Rightarrow |K(x) - z| &= | - CG(z) - (CG'(z) - \mathcal{I} + \sum_k (x_k - z_k) CG_k[z,z,x])(x-z)| \leq \\
&\leq |CG(z)| + (|CG'(z) - \mathcal{I}| + \sum_k |x-z|_k |CG_k[z,z,x]|)|x-z| \leq \\
&\leq \bar{b} + (B_0 + \sum_k u_k \overline{B}_k)u
\end{aligned}$$

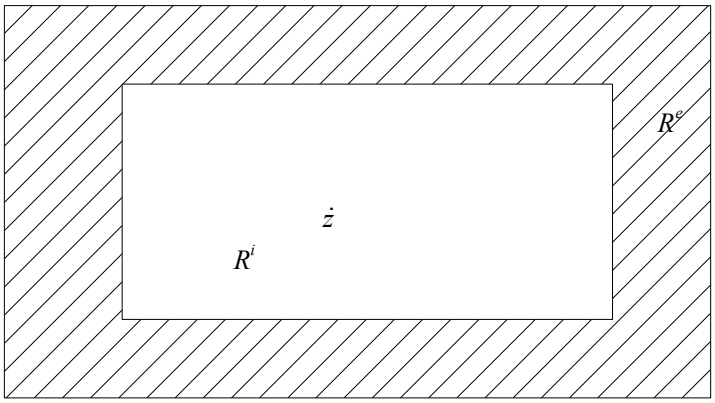
Gilt nun (7.1), so folgt $K(x) \leq u$. Weil das für alle x, z gilt, folgt aus Theorem 7.2, dass eine Nullstelle in M_u existiert. □

Es gilt insbesondere, dass $B_0 u \leq u \Rightarrow \rho(B_o) \leq 1$ mit ρ der Spektralradius. Daher muss C eine gute Approximation von $G'(z)^{-1}$ sein.

Satz 7.5. *Sei* S *eine Menge, die* z *enthält, und es gelte* $B_k(x) \leq \overline{B}_k \ \forall \ x \in S$. *Für beliebiges* $0 < v \in \mathbb{R}^n$ *setzen wir* $w := (\mathcal{I} - B_0)v, a := \sum_k \overline{B}_k v$.
Gilt nun $D_j := w_j^2 - 4a_j\bar{b}_j > 0 \ \forall \ j = 1, \ldots, n$ *dann definieren wir*

$$\begin{array}{cccc}
\lambda_j^e := \frac{w_j + \sqrt{D_j}}{2a_j} & , & \lambda_j^i := \frac{\bar{b}_j}{a_j \lambda_j^e} \\
\lambda^e := \min_j \lambda_j^e & , & \lambda^i := \max_j \lambda_j^i
\end{array}$$

Ist dann $\lambda^e > \lambda^i$, *so gibt es mindestens eine Nullstelle* x^* *von* G *in* $R^i := S \cup [z - \lambda^i v, z + \lambda^i v]$. *Alle Nullstellen von* G, *die in* R^{e° *liegen, wobei* $R^e := S \cup [z - \lambda^e v, z + \lambda^e v]$ *ist, liegen in* R^i.



Nullstellen können also nur außerhalb R^{e° oder in R^i liegen, in $R^{e^\circ} \setminus R^i$ gibt eine keine Nullstellen. Man bezeichnet R^e auch als Ausschlussregion (exclusion region) und R^i als Einschlussregion (inclusion region).

Beweis. Sei $v \in \mathbb{R}^n$ und $v > 0$. Wir setzen $u := \lambda v$. Wir untersuchen, wann (7.1) gilt. Das ist dann der Fall, wenn

$$\begin{aligned}
\lambda v &\geq (B_0 + \sum_k \lambda v_k \overline{B}_k)\lambda v + \overline{b} \\
&= \overline{b} + \lambda B_k v + \lambda^2 \sum_k v_k \overline{B}_k v \\
&= \overline{b} + \lambda(v - w) + \lambda^2 a \\
&\Leftrightarrow \lambda^2 a - \lambda w + \overline{b} \leq 0 \\
&\Leftrightarrow \lambda^2 a_j - \lambda w_j + \overline{b}_j \leq 0 \; \forall j = 1, \ldots, n \\
\text{Für } j \text{ beliebig ist die letzte Ungleichung erfüllt, wenn } &\lambda_j^i \leq \lambda \leq \lambda_j^e \\
&\Leftrightarrow \lambda^i \leq \lambda \leq \lambda^e
\end{aligned}$$

Angenommen $x \in R^{e^\circ} \setminus R^i$ und $G(x) = 0$. Sei λ minimal mit $|x - z| \leq \lambda v$. Nach Voraussetzung gilt $\lambda^i < \lambda < \lambda^r$. Weil $G(x) = 0$ ist folgt $K(x) = x$. Wegen Theorem 7.4 und der Dreiecksungleichung folgt

$$\begin{aligned}
|x - z| &\leq |CG(z)| + (|CG'(z) - \mathcal{I}| + \sum_k |x - z|_k |CG_k[z, z, x]|)|x - z| \leq \\
&\leq \overline{b} + \lambda B_0 v + \lambda^2 \sum_k \overline{B}_k v = \lambda v + (\underbrace{\overline{b} - \lambda w + \lambda^2 a}_{<0}) \qquad (7.2)
\end{aligned}$$

Dies ist aber ein Widerspruch zur Minimalität von λ. □

Auf R^i kann man dann Iterationen von Theorem 7.2 anwenden, entweder mit $\mathbf{x} = \mathbf{z} \subseteq R^i$ um die Eindeutigkeit zu zeigen, oder um R^i zu verkleinern unter Ausnutzung der quadratischen Konvergenz.

Satz 7.6. *Sei z eine approximative Nullstelle von G, und sei B eine Matrix mit*

$$|CG[z, \boldsymbol{x}] - \mathcal{I}| + \sum |\boldsymbol{x} - z|_k |CG_k[\boldsymbol{x}, z, \boldsymbol{x}| \leq B$$

Gilt $||B|| < 1$ für irgendeine Operatornorm, dann enthält $\boldsymbol{x}$ höchstens eine Nullstelle von G.

Beweis. Seien x und x' zwei Nullstellen von G.

$$\begin{aligned} 0 &= G(x) - G(x') = G[x, x'](x - x') \\ &= (G[x, z] + \sum (x' - z)_k G_k[x, z, x'])(x' - x) \end{aligned}$$

Sei C beliebig, dann gilt

$$\begin{aligned} x - x' &= (CG[x, z] - \mathcal{I} + \sum (x' - z)_k CG_k[x, z, x'])(x' - x) \\ |x - x'| &\leq (|CG[\mathbf{x}, z] - \mathcal{I}| + \sum |\mathbf{x} - z|_k |CG[\mathbf{x}, z, \mathbf{x}]|)|x' - x| \leq B|x' - x| \\ \Rightarrow ||x - x'|| &\leq ||B||||x' - x|| < ||x' - x|| \Rightarrow x = x' \end{aligned}$$

□

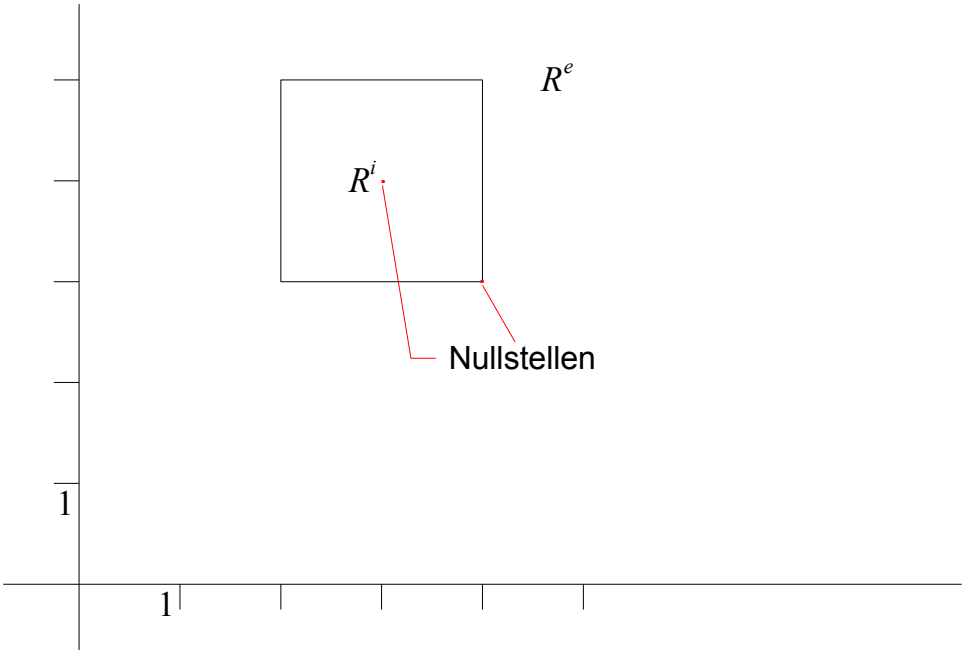
Beispiel 7.7. Fortsetzung von Beispiel 7.1. Die Nullstellen von G sind $\pm \begin{pmatrix} 3 \\ 4 \end{pmatrix}, \pm \begin{pmatrix} 4 \\ 3 \end{pmatrix}$,

$$\begin{aligned} z &= \begin{pmatrix} 3 \\ 4 \end{pmatrix} \\ C = G'(z)^{-1} &= \frac{1}{14} \begin{pmatrix} -3 & 8 \\ 4 & -6 \end{pmatrix} \\ G(z) = 0 &= \overline{b} \\ B_0 &= |CG'(z) - \mathcal{I}| = 0 \\ \overline{B}_1 &= |CG_1[z, z, x]| = \frac{1}{14} \begin{pmatrix} 3 & 0 \\ 4 & 0 \end{pmatrix} \\ \overline{B}_2 &= |CG_2[z, z, x]| = \frac{1}{14} \begin{pmatrix} 8 & 3 \\ 6 & 4 \end{pmatrix} \\ v &= \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} > 0, w_j = v_j, D_j = v_j^2 > 0 \\ a_1 &= \frac{1}{14}(3v_1^2 + 8v_1 v_2 + 3v_2^2) \\ a_2 &= \frac{1}{14}(4v_1^2 + 6v_1 v_2 + 4v_2^2) \end{aligned}$$

Sei nun $v = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \Rightarrow w = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, D = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, a = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$.

$$\left.\begin{array}{rcl} \lambda_j^e = 1 & \Rightarrow & \lambda^e = 1 \\ \lambda_j^i = 0 & \Rightarrow & \lambda^i = 0 \end{array}\right\} \Rightarrow \lambda^e > \lambda^i$$

$R^e = [2,4] \times [3,5]$
$R^i = \left\{ \begin{pmatrix} 3 \\ 4 \end{pmatrix} \right\}$



8 Polynomiale Probleme und reelle algebraische Geometrie

8.1 Algebraische Grundlagen

In dieser Vorlesung ist $0 \in \mathbb{N}$. $\mathbb{R}[x]$ sei der Ring der Polynome in $x = (x_1, \ldots, x_n)$, $\mathbb{R}[x]$ ist eine Algebra mit 1. Sei $\{R_i\}$ eine endliche Familie von Polynomen. $R = \begin{pmatrix} R_1 \\ \vdots \\ R_k \end{pmatrix}$.

Das von R erzeugte Ideal in $\mathbb{R}[x]$ ist

$$I\langle R\rangle = I\langle R_1, \ldots, R_k\rangle := \left\{\sum_{i=1}^{k} a_i R_i | a_i \in \mathbb{R}[x]\right\}$$

Das von R erzeugte multiplikative Monoid ist die Halbgruppe

$$M\langle R\rangle = M\langle R_1, \ldots, R_k\rangle := \left\{\prod_{i=1}^{k} R_i^{h_i} | h_i \in \mathbb{N}\right\}$$

Ein polynomialer Kegel C ist eine Teilmenge von $\mathbb{R}[x]$ mit

- $r, s \in C \Rightarrow r + s \in C, r \cdot s \in C$
- $a \in \mathbb{R}[x] \Rightarrow a^2 \in C$

Der kleinste polynomiale Kegel ist die Menge SOS aller Polynome, die als Summe von Quadraten darstellbar sind. Der polynomiale Kegel $C\langle R\rangle$, der von R erzeugt wird, ist der kleinste polynomiale Kegel, der alle R_i enthält.

$$C\langle R\rangle = C\langle R_1, \ldots, R_k\rangle := \left\{y_0 + Y^T R_S | y_0 \in SOS, Y \in SOS^{2^k}\right\}$$

wobei R_S jener Vektor sei, der die 2^k Polynome des quadratfreien Teils

$$S\langle R\rangle = S\langle R_1, \ldots, R_k\rangle := \left\{\prod_{i=1}^{k} R_i^{e_i} | e_i \in \{0, 1\}\right\}$$

von $M\langle R\rangle$ enthält.

8.2 Polynomiale Transpositionssätze

Satz 8.1. *Polynomialer Transpositionssatz 1:*
Seien P, Q, R Vektoren von Polynomen. Dann gilt genau eine der folgenden Aussagen:

(i) $\exists\, x \in \mathbb{R}^n : P(x) \geq 0, Q(x) = 0, R_i(x) \neq 0$ *für* $i = 1, \ldots, k$.

(ii) $\exists\, f \in C\langle P\rangle, g \in I\langle Q\rangle, h \in M\left\langle R_1^2, \ldots, R_k^2\right\rangle$ *mit* $f + g + h = 0$.

Beweis. (i) und (ii) schließen einander aus:
Gilt (i), dann haben wir für $f \in C\langle P\rangle$, dass $f(x) \geq 0$ gilt. Weiters gilt $g(x) = 0$ für $g \in I\langle Q\rangle$ und $h(x) > 0$ für $h \in M\left\langle R_1^2, \ldots, R_k^2\right\rangle$. Daher gilt:

$$f(x) + g(x) + h(x) > 0$$

$\Rightarrow$ (ii) gilt nicht.
Gilt (i) nicht, dann gilt (ii) nicht: Das ist nicht trivial und folgt aus dem schwachen Positivstellensatz für Polynome (Für den Beweis dazu siehe: „Positive Polynomials“, Bochnak, Theorem 4.4.2). □

Satz 8.2. *Polynomialer Transpositionssatz 2:*
Seien P, Q, R Vektoren von Polynomen. Dann gilt genau eine der folgenden Aussagen:

(i) $\exists\, x \in \mathbb{R}^n : P(x) \geq 0, Q(x) = 0, R(x) > 0.$

(ii) $\exists\, f \in C\langle P, R\rangle, g \in I\langle Q\rangle, h \in M\langle R\rangle$ *mit* $f + g + h = 0.$

Beweis. Gilt (i) so folgt:

$$\begin{aligned} &\forall\, f \in C\langle P, R\rangle : f(x) \geq 0 \\ &\forall\, g \in I\langle Q\rangle : g(x) = 0 \\ &\forall\, h \in M\langle R\rangle : h(x) > 0 \\ \Rightarrow &f(x) + g(x) + h(x) > 0 \Rightarrow \text{ (ii) ist falsch.} \end{aligned}$$

Ist (i) nicht erfüllbar, dann gibt es auch kein $x \in \mathbb{R}^n$ mit $(P(x), R(x)) \geq 0, Q(x) = 0, R_i(x) \neq 0$ für $i = 1, \ldots, k$. Nach Theorem 8.1 gilt $\exists\, f \in C\langle P, R\rangle, g \in I\langle Q\rangle, h \in M\left\langle R_1^2, \ldots, R_k^2\right\rangle$ mit $f + g + h = 0$, und außerdem ist $M\left\langle R_1^2, \ldots R_k^2\right\rangle \subseteq M\langle R\rangle \Rightarrow$ (ii) gilt. □

Satz 8.3. *Polynomialer Transpositionssatz 3:*
Seien P, Q, R und $S_1, \ldots S_m$ Vektoren von Polynomen. Dann gilt genau eine der folgenden Aussagen:

(i) $\exists\, x \in \mathbb{R}^n : P(x) \geq 0, Q(x) = 0, R(x) > 0, S_i(x) \neq 0$ *für* $i = 1, \ldots, m.$

(ii) $\exists\, f \in C\langle P, R\rangle, g \in I\langle Q\rangle, h \in M\left\langle R, S_1^T S_1, \ldots, S_m^T S_m\right\rangle$ *mit* $f + g + h = 0.$

Beweis. Gilt (i), dann haben wir:

$$\begin{aligned} &\forall\, f \in C\langle P, R\rangle : f(x) \geq 0 \\ &\forall\, g \in I\langle Q\rangle : g(x) = 0 \\ &\forall\, h \in M\left\langle R_1, S_1^T S_1, \ldots, S_m^T S_m\right\rangle : h(x) > 0 \\ \Rightarrow &f(x) + g(x) + h(x) > 0 \Rightarrow \text{ (ii) gilt nicht.} \end{aligned}$$

Wenn (i) nicht gilt, dann gibt es kein $x \in \mathbb{R}^n$ mit $P(x) \geq 0, Q(x) = 0,$
$(R(x), S_1^T S_1(x), \ldots, S{,}m^T S_m(x)) > 0$. Wegen Theorem 8.2 gibt es daher
$f \in C\left\langle P, R, S_1^T S_1, \ldots, S_m^T S_m\right\rangle, g \in I\langle Q\rangle, h \in M\left\langle R, S_1^T S_1, \ldots, S_m^T S_m\right\rangle$ mit $f + g + h = 0$
und $S_j^T S_j \in SOS$, also gilt $C\left\langle P, R, S_1^T, \ldots, S_m^T S_m\right\rangle \subseteq C\langle P, R\rangle$. □

Theorem 8.3 ist nicht spezieller als Theorem 8.1, weil für $R = (), S_j = (R_j)$ folgt aus 8.3 sofort 8.1.

8.3 Optimalitätsbedingungen für polynomiale Probleme

Definition 8.4. Ein Punkt $\hat{x}$ heißt schwach Pareto-optimal für das Optimierungsproblem

$$\begin{aligned} &\min f(x) \\ &s.t.\ x \in \mathcal{F} \end{aligned}$$

mit $f : \mathbb{R}^n \to \mathbb{R}^k$, falls kein $y \in \mathcal{F}$ existiert mit $f(y) < f(\hat{x})$.

Satz 8.5. *Optimalitätsbedingungen für globale Optimalität in polynomialen Problemen: Sei $\hat{x}$ ein zulässiger Punkt des polynomialen Pareto-Optimierungsproblems*

$$\begin{aligned} \min\ & f(x) \\ s.t.\ & C(x) \geq 0 \\ & F(x) = 0 \end{aligned}$$

mit $f \in \mathbb{R}[x]^k, C \in \mathbb{R}[x]^m, F \in \mathbb{R}[x]^r, x \in \mathbb{R}^n$. Wir definieren $B(x) := \begin{pmatrix} C(x) \\ f(\hat{x}) - f(x) \end{pmatrix}$, $G(x) := f(\hat{x}) - f(x)$. Dann sind äquivalent:

(i) *$\hat{x}$ ist ein schwaches Pareto-Minimum.*

(ii) *$\exists\, y_0 \in SOS, Y \in SOS^{2^{m+k}}, Z \in \mathbb{R}[x]^r$ und einen Multiindex $\alpha \in \mathbb{N}^k$ mit $|\alpha| > 0$, sodass*

$$G(x)^\alpha + y_0(x) + Y(x)^T B_S(x) + Z(x)^T F(x) \equiv 0 \tag{8.1}$$

Weiters erfüllt jede Lösung von (8.1)

$$\left.\begin{array}{c} y_0(\hat{x}) = 0 \\ \inf\{Y(\hat{x}), B_S(\hat{x})\} = 0 \\ F(\hat{x}) = 0 \end{array}\right\} \text{ Komplementaritätsbedingungen}$$

$\delta_{|\alpha|,1} f_i'(\hat{x})^T = B_S'(\hat{x})^T Y(\hat{x}) + F'(\hat{x})^T Z(\hat{x})$ (Karush-John-Bedingungen) wobei i so gewählt ist, dass $\alpha_i = 1$, falls $|\alpha| = 1$.

Beweis. $\hat{x}$ ist schwach Pareto-optimal genau dann, wenn die Bedingungen $C(x) \geq 0$, $F(x) = 0, G(x) > 0$ inkonsistent sind. Nach Theorem 8.2 folgt: $\exists\, q \in C\langle B\rangle, \widetilde{r} \in I\langle F\rangle$, $s \in M\langle G\rangle$ mit $q + \widetilde{r} + s = 0$.
Es gilt:

- $q = y_0 + Y^T B_S$ für ein $y_0 \in SOS, Y \in SOS^{2^{m+k}}$
- $\widetilde{r} = Z^T Y$ für ein $Z \in \mathbb{R}[x]^r$
- $s = G^\alpha$ für ein $\alpha \in \mathbb{N}^k$.

$\Rightarrow G(x)^\alpha + y_0(x) + Y(x)^T B_S(x) + Z(x)^T F(x) \equiv 0$, also gilt (8.1).
$\hat{x}$ ist zulässig, somit folgt $F(\hat{x}) = 0, B(\hat{x}) \geq 0, G(\hat{x}) = 0$. Setzen wir $\hat{x}$ in (8.1) ein, dann erhalten wir $\delta_{|\alpha|,0} \leq \delta_{|\alpha|,0} + \underbrace{y_0(\hat{x})}_{\geq 0} + \underbrace{\underbrace{Y(\hat{x})^T}_{\geq 0}\underbrace{B_S(\hat{x})}_{\geq 0}}_{\geq 0} + \underbrace{Z(\hat{x})^T \underbrace{F(\hat{x})}_{=0}}_{=0} = 0$

$$\begin{array}{rl} & \delta_{|\alpha|,0} = 0 \\ \Rightarrow & y_0(\hat{x}) = 0 \\ & Y(\hat{x})^T B_S(\hat{x}) = 0 \quad \Rightarrow \inf(Y(\hat{x}), B_S(\hat{x})) = 0 \end{array}$$

Es bleiben die Karush-John-Bedingungen zu zeigen: Wir differenzieren (8.1).
$0 = \sum_{i=1}^k \alpha_i G^{\alpha - e_i}(\hat{x}) G_i'(\hat{x}) + \underbrace{y_0'(\hat{x})}_{=0} + Y'(\hat{x})^T B_S(\hat{x}) Y(\hat{x}) + \underbrace{Z'(\hat{x}) \underbrace{F(\hat{x})}_{=0}}_{=0} + F'(\hat{x})^T Z(\hat{x})$

$y_0'(\hat{x}) = 0$, weil $y_0(\hat{x}) = 0$, also hat y_0 an $\hat{x}$ (weil es ja ein SOS-Polynom ist) eine doppelte Nullstelle. $G(\hat{x}) = 0 \Rightarrow G(\hat{x})^\beta = \delta_{|\beta|,0}$.
$\Rightarrow \sum_{i=1}^k \alpha_i G^{\alpha - e_i}(\hat{x}) G_i'(\hat{x}) = \sum_{i=1}^k \alpha_i \delta_{|\alpha - e_i|,0} G_i'(\hat{x}) \neq 0 \Leftrightarrow \alpha = e_j$ für ein $j \Leftrightarrow |\alpha| = 1$.

$$\sum_{i=1}^{k} \alpha_i G^{\alpha-e_i}(\hat{x}) G_i'(\hat{x}) = \begin{cases} G_j'(\hat{x}) = -f_j'(\hat{x}) & \text{für } |\alpha| = 1 \\ 0 & \text{für } |\alpha| > 1 \end{cases}$$

$\Rightarrow 0 = -\delta_{|\alpha|,1} f_j'(\hat{x}) + Y'(\hat{x})^T B_S(\hat{x}) + B_S'(\hat{x})^T Y(\hat{x}) + F'(\hat{x})^T Z(\hat{x})$. Wegen $\inf\{Y(\hat{x}), B_S(\hat{x})\} = 0$ gilt: $B_S(\hat{x})_+ \neq 0 \Rightarrow Y(\hat{x})_+ = 0 \Rightarrow Y'(\hat{x})_+ = 0 \Rightarrow Y'(\hat{x})^T B_S(\hat{x}) = 0$. Daraus folgt:
$\delta_{|\alpha|,1} f_j'(\hat{x}) = B_S'(\hat{x})^T Y(\hat{x}) + F'(\hat{x})^T Z(\hat{x})$. □

(8.1) ist ein polynomiales Zertifikat für (globale) Pareto-Optimalität von $\hat{x}$. Algorithmisch bestimmt man zunächst einen Kandidaten $\hat{x}$, und danach versucht man, Gleichung (8.1) sequentiell mit steigendem Totalgrad zu lösen.
Alle Polynomkoeffizienten, die gesucht werden (die y_0, Y) treten in (8.1) linear auf. Die SOS-Bedingungen sind semidefinit. Jeder einzelne Versuch, (8.1) zu lösen, führt zu einem semidefiniten CSP. Der Totalgrad kann sehr groß werden für das Zertifikat. Dort steckt der potentiell exponentielle Aufwand.

Satz 8.6. *Kuhn-Tucker-Bedingungen:*
Ist $\hat{x}$ ein globales Optimum des polynomialen Optimierungsproblems

$$\begin{aligned} &\min f(x) \\ &s.t.\ C(x) \geq 0 \\ &\quad F(x) = 0 \end{aligned}$$

mit $k = 1$ und gilt $G(x)^\alpha + y_0(x) + Y(x)^T B_S(x) + Z(x)^T F(x) \equiv 0$ mit $\alpha = (1)$, dann gibt es $0 \leq y \in \mathbb{R}^m, z \in \mathbb{R}^r$ mit

$$\nabla f(\hat{x}) = C'(\hat{x})^T y + F'(\hat{x})^T z \tag{8.2}$$

und $\inf\{y, C(\hat{x})\} = 0$.

Beweis. Wegen den Karush-John-Bedingungen aus Theorem 8.5 wissen wir
$\nabla f(\hat{x}) = B_S'(\hat{x})^T Y(\hat{x}) + F'(\hat{x})^T Z(\hat{x})$. Für

$$a(x) = \begin{pmatrix} a_1(x) \\ a_2(x) \\ a_3(x) \end{pmatrix} \text{ ist } a_S(x) = \begin{pmatrix} 1 \\ a_1(x) \\ a_2(x) \\ a_3(x) \\ a_1(x)a_2(x) \\ a_1(x)a_3(x) \\ a_2(x)a_3(x) \\ a_1(x)a_2(x)a_3(x) \end{pmatrix}, \; B(x) = \begin{pmatrix} C(x) \\ G(x) \end{pmatrix}.$$

$$b(x) = \begin{pmatrix} a_1(x) \\ a_2(x) \end{pmatrix} \Rightarrow b_S(x) = \begin{pmatrix} 1 \\ a_1(x) \\ a_2(x) \\ a_1(x)a_2(x) \end{pmatrix} \Rightarrow a_S(x) = \begin{pmatrix} b_S(x) \\ a_3(x) \cdot b_S(x) \end{pmatrix}$$

$$\Rightarrow B_S(x) = \begin{pmatrix} C_S(x) \\ G(x)C_S(x) \end{pmatrix}, B_S'(\hat{x}) = \begin{pmatrix} C_S'(\hat{x}) \\ G'(\hat{x})C_S(\hat{x}) + \underbrace{G(\hat{x})}_{=0} C_S'(\hat{x}) \end{pmatrix}$$

$$\nabla f(\hat{x}) = C_S'(\hat{x})^T Y^{(1)}(\hat{x}) + (\underbrace{\underbrace{(G'(\hat{x})}_{=-\nabla f(\hat{x})} \underbrace{C_S(\hat{x}))^T}_{\geq 0} \underbrace{Y^2(\hat{x})}_{\geq 0}}_{\gamma - 1 :=} + F'(\hat{x})^T Z(\hat{x})$$

$\gamma \nabla f(\hat{x}) = \sum_{\beta \in \{0,1\}^m} \sum_{i=1, \beta_i = 1}^m C_i'(\hat{x})^T C^{\beta - e_i}(\hat{x}) Y_\beta^{(1)}(\hat{x}) + F'(\hat{x})^T Z(\hat{x}), C_S(\hat{x}) = (C^p(\hat{x}))_{\beta \in \{0,1\}^m}$,
$y_i := \frac{1}{\gamma} \sum_{\beta \in \{0,1\}^m, \beta_i = 1} C^{\beta - e_i}(\hat{x}) Y_\beta^{(1)}(\hat{x}) \geq 0, z := \frac{1}{\gamma} z(\hat{x}) \Rightarrow \nabla f(\hat{x}) = C'(\hat{x})^T y + F'(\hat{x}) z$, also
(8.2). $C_i(\hat{x}) y_i = \frac{1}{\gamma} \sum_{\beta \in \{0,1\}^m, \beta_i = 1} \underbrace{C^\beta(\hat{x}) Y_\beta^{(1)}(\hat{x})}_{=0 \text{ wegen } \inf\{B_S(\hat{x}), Y(\hat{x})\} = 0} = 0$ □

9 MIP-Relaxationen und semilineare Relaxationen

MIP steht für Mixed Integer Programming und inkludiert aber auch immer linear, also Mixed Integer (linear) Programming.

Definition 9.1. Eine Nebenbedingung heißt **semilinear**, falls sie in jeder beschränkten Box äquivalent ist zu einer endlichen Liste von linearen Bedingungen und Ganzzahligkeitsbedingungen. Ein Optimierungproblem heißt **semilinear**, wenn die Zielfunktion linear und alle Nebenbedingungen semilinear sind.
Offensichtlich ist jedes semilineare Optimierungsproblem äquivalent zu einem MIP.

9.1 Seminlineare Nebenbedingungen

Hier eine (keineswegs vollständige) Liste an semilinearen Nebenbedingungen:

- Jede Schranken-Nebenbedingung ist semilinear.
- Jede lineare Nebenbedingung ist semilinear.
- Jede Ganzzahligkeits-Nebenbedingung ist semilinear.
- Jede binäre Nebenbedingung $z \in \{0,1\}$ ist semilinear: $z \in [0,1], z \in \mathbb{Z}$.
- Eine Liste von Variablen x_k mit der Nebenbedingung $\sum_{k\in K} x_k = 1, x_k \in \{0,1\}, k \in K$ heißt binäre speziell geordnete Menge (BSOS-binary special ordered set). Daher ist die Nebenbedingung „x_k is BSOS“ semilinear.
- Weiters ist die Formulierung $x = e_k$ für ein $k \in K$ ebenfalls äquivalent und somit semilinear.

Satz 9.2. *Sei $C_0 \subseteq \mathbb{R}^n$ und seien $F_k : C_0 \to \mathbb{R}^{m_k} (k = 1, \ldots, d)$ Funktionen mit*

$$F_k(x) \geq \underline{F}_k \forall\, x \in C_0 \tag{9.1}$$

Dann gibt es ein $x \in C_0$ mit

$$F_1(x) \geq 0 \vee \cdots \vee F_d(x) \geq 0 \tag{9.2}$$

genau dann, wenn $\exists z \in \mathbb{R}^d$ und $x \in C_0$ mit z ist BSOS und

$$F_k(x) \geq \underline{F}_k(1 - z_k) \forall\, k = 1, \ldots, d \tag{9.3}$$

Das heißt sind alle F_k linear, so ist die Nebenbedingung (9.2) semilinear.

Beweis. Angenommen es gilt (9.2). Dann gibt es k mit $F_k(x) \geq 0$. Jedenfalls gilt (9.1) für alle $\ell : F_\ell(x) \geq \underline{F}_\ell$. Sei nun $z = e_k$. Dann ist $\underline{F}_\ell(1 - z_\ell) = \begin{cases} \underline{F}_\ell & \text{für } \ell \neq k \\ 0 & \text{für } k = l \end{cases}$
Also ist (9.3) erfüllt.
Umgekehrt: Ist z BSOS, so ist $z = e_k$ für ein k. Daher ist wegen (9.3) für ein k :
$F_k(x) \geq \underline{F}_k(1 - z_k) = 0$. □

Sind alle F_k linear, so heißt (9.2) **lineare disjunktive Nebenbedingung**, und der Satz beweist, dass jede lineare disjunktive Nebenbedingung semilinear ist.
Allgemeiner sind natürlich auch lineare disjunkte Nebenbedingungen der Form
$A_1x \in \mathbf{b}_1 \vee A_2x \in \mathbf{b}_2 \vee \cdots \vee A_dx \in \mathbf{b}_d$ semilinear, äquivalent kann man das umschreiben zu
$A_jx \in \mathbf{b}_j$ für ein $j \in \{1, \ldots, d\}$.
Halbstetige Variable sind Variablen, die folgende Nebenbedingung erfüllen: $x = 0 \vee x \in \mathbf{a}$. (Als Beispiel hierfür kann man angeben, dass solche Variablen eingesetzt werden wenn es um Kraftwerke geht, die entweder mit einem gewissen Minimum betrieben werden müssen oder ausgeschaltet werden müssen). Das ist eine semilineare Nebenbedingung.
Semiganzzahlige Variablen erfüllen die semilineare Nebenbedigung $(x = 0 \vee x \in \mathbf{a}) \wedge x \in \mathbb{Z}$.
Eine **numerisch speziell geordnete Menge** (NSOS) ist ein Vektor $y \in \mathbb{R}^d$ von Variablen mit $y \geq 0, \sum_{k=1}^{d} y_k = 1$, höchstens zwei $y_i \neq 0$ und sind zwei $y_i \neq 0$, so müssen sie aufeinander folgende Indizes besitzen. y ist NSOS ist semilinear, weil es äquivalent ist zu:

$$(y \geq 0, \textstyle\sum_{k=1}^{d} y_k = 1) \wedge \begin{cases} (y_k + y_{k+1} = 1 & \text{für ein } k \in \{1, \ldots, d-1\}) \vee \\ (y_j = 1 & \text{für ein } j \in \{1, \ldots, d\}) \end{cases}$$

Ausschlussbedingungen der Form $x \notin \mathbf{y}^\circ$ sind semilinear, weil sie äquivalent beschrieben werden können als $x_i \leq \underline{y}_i \vee x_i \geq \overline{y}_i$ für ein $i \in K$.
Alle **logischen Ausdrücke** in binären Variablen sind seminlinear umformulierbar:

$$\begin{aligned} x_1 \vee \cdots \vee x_k &\Leftrightarrow \sum_{i=1}^{k} x_i \geq 1 \\ x_1 \wedge \cdots \wedge x_k &\Leftrightarrow x = e = (1, \ldots, 1) \\ \neg x &\Leftrightarrow x = 0 \\ (x_1 \Leftrightarrow x_2) &\Leftrightarrow x_1 = x_2 \\ (x_1 \Rightarrow x_2) &\Leftrightarrow x_1 \leq x_2 \\ ((x_1 \vee \cdots \vee x_k) \Rightarrow (x_{k+1} \vee \cdots \vee x_\ell)) &\Leftrightarrow x_d \leq x_{k+1} + \ldots x_\ell \text{ für } d = 1, \ldots, k \end{aligned}$$

Mit Hilfe der disjunktiven oder der konjunktiven Normalform kann man jeden logischen Ausdruck semilinear formulieren.
Konditionale lineare Nebenbedingungen sind semilinear:
$Ax \in \mathbf{a}$, falls $Bx < b \Leftrightarrow Ax \in \mathbf{a} \vee (Bx)_1 \geq b_1 \vee \cdots \vee (Bx)_d \geq b_d$. Nebenbedingungen der Form $a^Tx \leq \min_{i=1,\ldots,d}(Ax - b)_i \Leftrightarrow a^Tx \leq (Ax - b)_i \; \forall \; i = 1, \ldots, d$ sind daher semilinear.
Auch die umgekehrte Nebenbedingung $a^Tx \geq \min_{i=1,\ldots,d}(Ax - b)_i \Leftrightarrow a^Tx \geq (Ax - b)_i$ für ein $i \in \{1, \ldots, d\}$ ist semilinear. Analoges gilt für max.
Daher sind **lineare Komplementaritätsbedingungen** semilinear:

$$\begin{aligned} &\inf\,(Ax - b, Cx - d) = 0 \text{ und auch} \\ &a^Tx - \alpha \leq |b^Tx - \beta| \\ &a^Tx - \alpha \geq |b^Tx - \beta| \\ &a^Tx - \alpha = |b^Tx - \beta| \end{aligned}$$

Sei φ **stückweise linear** mit endlichen vielen Unstetigkeitsstellen in der Ableitung. Die Nebenbedingung

$$a^Tx \leq \varphi(x_i) \tag{9.4}$$

ist semilinear: Seien $\xi_0 < \xi_1 < \cdots < \xi_d$ die Unstetigkeitsstellen der Ableitung von φ. Das heißt φ ist auf jedem $[\xi_i, \xi_{i+1}]$ linear. Für $\xi \in [\xi_k, \xi_{k+1}]$ gilt

$$\begin{aligned}\varphi(\xi) &= \varphi_k + \varphi'_k(\xi - \xi_k)\\ \varphi_k &= \varphi(\xi_k)\\ \varphi'_k &= \frac{\varphi(\xi_k) - \varphi(\xi_{k+1})}{\xi_k - \xi_{k+1}}\end{aligned}$$

(9.4) ist äquivalent zu $x_i \in [\xi_k, \xi_{k+1}] \wedge a^T x \leq \varphi_k + \varphi'_k(x_i - \xi_k)$ für ein k, beziehungsweise:

$$x_i = \textstyle\sum_{k=0}^{d-1} \xi_k y_k, y \text{ ist NSOS} \Rightarrow \xi(x_i) = \sum_{k=0}^{d-1} y_k \varphi_k \geq a^T x$$

In (9.4) kann man $\geq$ auch durch $\leq$ beziehungsweise $=$ ersetzen. Daher ist

$$f(x) \in \mathbf{a} \text{ mit } f(x) = \sum_{i=1}^{N} \varphi_i(x_i)$$

mit φ_i stückweise linear, semilinear.
Die meisten typischen Nebenbedingungen der kombinatorischen Optimierung sind semilineare:

(i) **all different**: $x_k \in \mathbb{Z}^{|K|}$, alle Einträge verschieden, $x_k \in \mathbb{Z}$ für $k \in K$,
$|x_j - x_k| \geq 1 \; \forall \, k, j \in K, k \neq j$ ist seminlinear.

(ii) **Kardinalitätsbedingung**: Die Anzahl der x_k, die $\neq 0$ sind, ist s.
$\exists \, z$ mit $\begin{cases} z_k = 1 & \text{falls } x_k > 0 \\ z_k = 0 & \text{falls } x_k < 1 \end{cases}$, $\sum z_k = s$ ist also seminlinear.

9.2 Semilineare Relaxationen

In diesem Abschnitt werden wir Probleme folgender Natur betrachten:

$$\begin{aligned}&\min f(x)\\ &s.t.\ F(x) \in \mathbf{F}\\ &\quad x_I \in \mathbb{Z}^{|I|}\end{aligned} \tag{9.5}$$

wobei $f(x)$ faktorabel sein soll, ist ein nichtlineares, gemischt-ganzzahliges, globales Optimierungsproblem. Zunächst transformieren wir (9.5) auf eine Gestalt, in der alle Nebenbedingungen entweder semilinear, quadratisch oder separabel sind.
Das kann man erreichen, indem man den DAG uminterpretiert-siehe dazu Beispiel 6.15. Für

$$\min \; f(x) = (4x_1 - x_2x_3)(x_1x_2 + x_3)$$

sieht das so aus:

$$\begin{aligned}&\min v_3\\ &s.t.\ v_1 = 4x_1 - x_2x_3\\ &\quad v_2 = x_3 + x_1x_2\\ &\quad v_3 = v_1v_2\end{aligned}$$

Außerdem ersetzen wir Quotienten und allgemeine Exponenten wie folgt:

- $z = \frac{x}{y} \Leftrightarrow yz = x \wedge y \neq 0$
- $z = x^y \Leftrightarrow e^{y \ln x} = z \Leftrightarrow \begin{cases} z = e^g \\ g = yh \\ h = \ln x, x > 0 \end{cases}$

Funktionen der Form $e^{x+3x^2 \ln x}$ werden nicht weiter zerlegt, da sie nur von einer Variable abhängen. Um eine semilineare Relaxation zu bestimmen, bleiben noch zwei Probleme zu lösen:

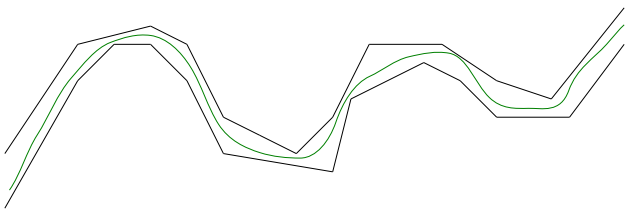
- Semilineare Relaxationen für eine separable Nebenbedingung:

$$\sum_{i \in K} \varphi_i(x_i) \begin{cases} \leq \alpha \\ \geq \alpha \\ = \alpha \end{cases}$$

- Semilineare Relaxationen für eine quadaratische Nebenbedingung:

$$x^T Q x + c^T x \in \mathbf{d}$$

9.2.1 Semilineare Relaxationen eindimensionaler Funktionen



Wir wissen bereits, dass Nebenbedingungen mit stückweise linearen Funktionen semilinear sind. Wir können also das Problem dadurch lösen, dass wir stückweise lineare Unter- und Überschätzung von eindimensionalen Funktionen $\varphi(x)$ bestimmen. Im Folgenden werden nun zwei Möglichkeiten genauer erläutert:

Erste Möglichkeit:
Sei $\varphi : \mathbb{R} \to \mathbb{R}$ faktorabel. Dann gilt

$$\varphi(a) + \int_a^y \varphi(x)' dx = \varphi(y) \; \forall \; a, y \tag{9.6}$$

Nun gehen wir folgender Idee nach: Wir unterteilen $[a,b]$ in Stücke $[\xi_i, \xi_{i+1}]$ mit $a = \xi_0 < \xi_1 < \cdots < \xi_N = b$ und bestimmen $\varphi'([\xi_i, \xi_{i+1}]) =: \mathbf{f}_i \; \forall \; i = 0, \ldots, N-1$.
$\Rightarrow \; \forall \; y, c \in [a,b]$:

$$\varphi(y) \in \varphi([c,c]) + \int_c^y \varphi'(x)dx \subseteq \varphi([c,c]) + \int_c^y \mathbf{f}(x)dx =$$
$$= \varphi([c,c]) + [\int_c^y \underline{f}(x)dx, \int_c^y \overline{f}(x)dx] =: \mathbf{g}(x)$$

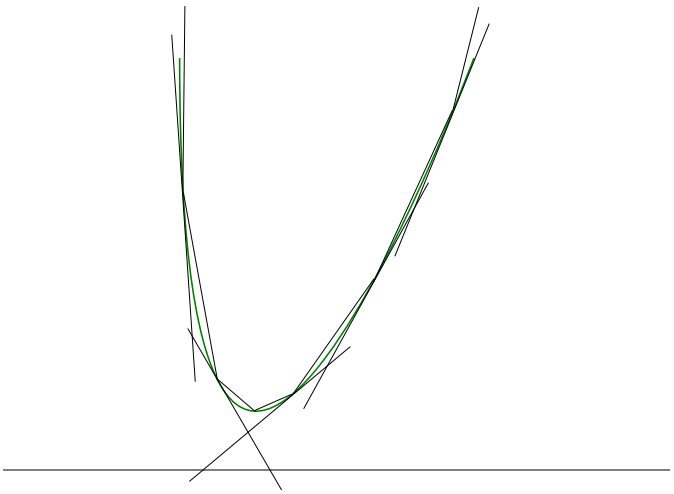
wobei $\mathbf{f}(x) := \mathbf{f}_i$ für $x \in [\xi_i, \xi_{i+1})$. Die oberen und unteren Schranken von $\mathbf{g}$ sind stückweise linear, weil $\mathbf{f}$ stückweise konstant ist. Falls $|\underline{f}(x) - \varphi'(x)| = \mathcal{O}(\varepsilon^s)$ für $\varepsilon = |\xi_i - \xi_{i+1}|$ gilt:

$$|\underline{g}(y) - \varphi(y)| \leq \int_c^y |\underline{f}(x) - \varphi'(x)|dx \leq N\varepsilon^s|y-c| \leq N\varepsilon^s|[a,b]-c| = |a-b|\varepsilon^{s-1}(|[a,b]-c|)$$

Das letzte $=$ in dieser Kette hält, da das Minimum für $c = \frac{a+b}{2}$ angenommen wird. Die Überschätzung kann beliebig klein gemacht werden, wenn $s \geq 2$ ist, also falls zum Beispiel φ' mit der Mittelpunktsregel ausgewertet wird.
Wie man das eben beschrieben in der Praxis umsetzen kann zeigt der folgende Algorithmus:

Setze $c = \frac{a+b}{2}$
$M := \emptyset$
$L := \{[a,c], [c,b]\}$
while $L \neq \emptyset$ **do**
  wähle $\mathbf{x} \in L$
  $L = L \setminus \{\mathbf{x}\}$
  bestimme $\mathbf{d} = f'(\mathbf{x})$
  if $\overline{d} - \underline{d} > \varepsilon$ **then**
    Teile $\mathbf{x} = \mathbf{x}_1 \cup \mathbf{x}_2$
    $L = L \cup \{\mathbf{x}_1, \mathbf{x}_2\}$
  else
    $\underline{x}, \overline{x}$ seien Stützpunkte von $\mathbf{f}$
    und $\mathbf{f}$=$\mathbf{d}$ auf $\mathbf{x}$, $M = M \cup \{(\underline{x}, \mathbf{d})\}$
  end if
end while
return M

Zweite Möglichkeit: Automatische Kurvendiskussion: Bestimme alle Maxima und Minima von φ'' (das heißt alle Nullstellen von φ'''). Ausgenommen in degenerierten Bereichen liegt zwischen einem Maximum und dem darauf folgenden Minimum höchstens eine Nullstelle von φ''.
Zwischen zwei Nullstellen von φ'' gibt es dann höchstens eine Nullstelle von φ'. Die Nullstellen von φ'' und φ' kann man dann mit Intervall-Newton bestimmen. Möglich ist auch der Satz von Miranda. Danach kennt man alle Extrema und Wendepunkte von φ.
Die Wendepunkte liefern die Konvexitätsbereiche von φ. Die Wendepunkte seien Stützstellen der stückweise linearen Funktion.



Mit Sekanten und Tangenten kann man φ beliebig genau stückweise linear aproximieren. φ''' berechnet man zum Beispiel mit Differentialzahlen 3. Ordnung.

9.2.2 Semilineare Relaxationen qudaratischer Terme

Wir werden diesen Abschnitt in zwei Teile gliedern: Zuerst behandeln wir den eindimensionalen Fall und danach den mehrdimensionalen:

- Wir untersuchen den Fall $z = xy$. Sei

$$\begin{aligned} u &= \alpha x + \beta y \\ v &= \alpha x - \beta y \\ \Rightarrow u^2 &= \alpha^2 x^2 + 2\alpha\beta xy + \beta^2 x^2 \\ v^2 &= \alpha^2 x^2 - 2\alpha\beta xy + \beta^2 y^2 \\ u^2 - v^2 &= 4\alpha\beta xy = 4\alpha\beta z \;\forall\; \alpha, \beta \neq 0 \end{aligned}$$

  wobei $u = \alpha x \pm \beta y$ linear sind und $u^2 - v^2 = 4\alpha\beta xy = 4\alpha\beta z$ separabel.

- Betrachten wir nun den allgemeinen Fall $x^T Hx$: Sei unser Problem gegeben durch

$$F_i(x) = \underbrace{\sum_{j=1}^{n} \varphi_{i,j}(x_j)}_{separabel} + \underbrace{x^T H_i x + c_i^T x + d_i}_{quadratisch} \in \mathbf{F}_i$$

Ohne Beschränkung der Allgemeinheit ist H_i symmetrisch. Sonst ersetzt man nämlich H_i durch $\frac{1}{2}(H_i + H_i^T)$. Dann kann man eine Eigenwertzerlegung durchführen:

$$H_i = Q_i^T \Lambda_i Q_i \Rightarrow x^T H_i x + c_i^T x = x^T Q_i^T \Lambda_i Q_i x + c_i^T Q_i^T Q_i x$$

Seien $\underbrace{y_i := Q_i x}_{linear}, \gamma_i := Q_i c_i$

$$\Rightarrow x^T H_x + c^T x = y_i^T \Lambda_i y_i + \gamma_i^T \gamma_i = \underbrace{\sum_{j=1}^{n} \Lambda_{i,jj} y_{i,j}^2 + \gamma_{i,j} y_{i,j}}_{separabel}$$

$$\Rightarrow F_i(x) = \underbrace{\sum_{j=1}^{n} \varphi_{i,j}(x_j) + \sum_{j=1}^{n} \Lambda_{i,jj} y_{i,j}^2 + \gamma_{i,j} y_{i,j}}_{separabel} \in \mathbf{F}_i - d_i$$

Die Schwierigkeit bei diesem Trick ist es, die Rundungsfehler zu kontrollieren:
$Q_i^T \Lambda_i Q_i = H_i$ wird mittels numerischer linearer Algebra berechnet und ist daher rundungsfehlerbehaftet. Es ist sehr schwierig **dünne** oder **fast-dünne** Matrizen,$\mathbf{Q}_i, \mathbf{\Lambda}_i$ zu finden, die $\mathbf{Q}_i^T \mathbf{\Lambda}_i \mathbf{Q}_i \ni H_i$ erfüllen und wo die Elemente von $\mathbf{Q}_i$ jeweils orthogonale Matrizen sind. Daher kann man diese Umformulierung zwar im Rahmen eines vollständigen Algorithmus verwenden, nicht jedoch in einem rigorosen.

Zusammenfassung:
Semilineare Relaxationen mit gemischt ganzzahligen Problemen erhält man folgendermaßen:

$$\begin{aligned} &\min f(x) \\ &s.t.\ F(x) \in \mathbf{F} \\ &\quad x_I \in \mathbb{Z}^{|I|} \end{aligned}$$

Das kann man umformen zu

$$\begin{aligned} &\min x_0 \\ &s.t.\ F(x) \in \mathbf{F} \\ &\quad f(x) - x_0 \leq 0 \\ &\quad x_I \in \mathbb{Z}^{|I|} \end{aligned}$$

Nun bringen wir dieses Problem auf separable und quadratische Gestalt

$$\begin{aligned} &\min \hat{x}_0 \\ &s.t.\ F(\hat{x}) \in \hat{\mathbf{F}} \\ &\quad \hat{x}_j \in \mathbb{Z}^{|J|} \end{aligned}$$

Bis zu diesem Zeitpunkt sind die Optimierungsprobleme noch äquivalent. Dann muss man alle jene Nebenbedingungen semilinear machen, die nicht schon semilinear sind.

(i) $\hat{F}_i(x) \leq \overline{F}_i$

(ii) $\hat{F}_i(x) \geq \underline{F}_i$

(iii) $\hat{F}_i(x) \in \left[\underline{F}_i, \overline{F}_i\right]$

mit $F_i(x) = \sum_{j=1}^{n} \varphi_{i,j}(x_j) + c_i^T \hat{x} + \gamma_i$.

(i) $\sum_{j=1}^{n} z_{i,j} + c_i^T \hat{x}\gamma_i \leq \overline{F}_i$.
$\forall\, j : z_{i,j} \geq \hat{\varphi}_{i,j}(\hat{x}_j)$, wobei $\hat{\varphi}_{i,j}$ ein stückweise linearer Unterschätzer von $\varphi_{i,j}$ ist.

(ii) $\sum_{j=1}^{n} z_{i,j} + c_i^T \hat{x}\gamma_i \geq \underline{F}_i$.
$\forall\, j : z_{i,j} \leq \hat{\varphi_{i,j}}(\hat{x}_j)$, wobei $\hat{\varphi_{i,j}}$ ein stückweise linearer Überschätzer von $\varphi_{i,j}$ ist.

(iii) Zerlegen in zwei Nebenbedigungen vom Typ (i) und (ii).

Semilineare Relaxationen führen auf MIPs, die sich auch in höhreren Dimensionen noch effizient lösen lassen, besonders, wenn die ganzzahligen Variablen SOS oder semi-continuous sind. Die semilinearen Relaxationen fangen die nicht-konvexe Struktur der Optimierungsprobleme ein. MIPs lassen sich leicht weiter relaxieren durch Weglassen einzelner oder aller Ganzzahligkeitsbedingungen. Die Methode oder Teile davon werden auch oft schon in der Modellierung verwendet, um eigentlich nichtlineare Probleme linear zu modellieren.

A Grundidee der automatischen Hessematrixbestimmung

Im Folgenden wird nun noch kurz die Idee gebracht mit der die Hessematrix im DAG propagiert werden kann: Zur Berechnung von Hf erzeugt man einen Algorithmus, der $Hf(x) \cdot v$ für x, v beliebig berechnet:

$$Hf(x) \cdot v = (D\nabla f(x)) \cdot v = \nabla_x(\nabla f(x)^T \cdot v) = \nabla_x(\nabla f(x)^T \cdot v)$$

Man berechnet im letzten Ausdruck $\nabla f(x)^T$ durch Differentialzahlen (f, f') mit Konstanten $(c, 0)$ und Variablen (x_i, v_i). Die Differentialzahl, die das Endergebnis repräsentiert: (g, g') enthält die Info: $(f(x), \nabla f(x)^T \cdot v)$.
$(f(x), \nabla f(x)^T \cdot v)$ wird dann rückwärts differenziert und ergibt: $\nabla f(x); Hf(x) \cdot v$.

Literatur

[1] Hansen, ER. *Global optimization using interval analysis: the one-dimensional case.* Journal of Optimization Theory and Applications 29-3 (1979), 331-344, Springer.

[2] Kearfott, R.B. *Rigorous global search: continuous problems.* (1996), Kluwer Academic Publishers.

[3] C.A. Floudas and P.M. Pardalos. *State of the Art in Global Optimization*, Kluwer Academic Press, (1996), in press.

[4] Neumaier, A. *Complete search in continuous global optimization and constraint satisfaction* Acta Numerica 13 (2004), 271–369, Cambridge Univ Press.