

Kapitel 5

Weitere Themen

Im folgenden geben wir einen sehr kurzen Überblick über weitere interessante Themen der Optimierung, wobei wir vor allem auf weiterführende Literatur verweisen.

5.1 Nichtdifferenzierbare Optimierung

Als nichtdifferenzierbare Optimierung bezeichnet man üblicherweise Probleme in denen das Zielfunktional zwar Lipschitz-stetig, aber nicht differenzierbar ist. Dies ist typischerweise der Fall, wenn das Zielfunktional Anwendungen von Betragsfunktionen oder Maximumfunktionen enthält. In diesem Fall bedient man sich meist des *Subgradienten* $\partial f(x)$, der (für f konvex) definiert ist durch

$$\partial f(x) = \{ g \in \mathbb{R}^n \mid f(y) - f(x) \geq g^T(y - x), \forall y \in \mathbb{R}^n \}.$$

Z.b. gilt für $f(x) = |x|$, dass

$$\partial f(x) = \begin{cases} \{1\} & \text{für } x > 0 \\ [-1, 1] & \text{für } x = 0 \\ \{-1\} & \text{für } x < 0 \end{cases}.$$

Geometrisch gesehen enthält der Subgradient jene verallgemeinerten Tangenten, die zur Gänze unterhalb der Funktion liegen.

Zur Optimierung einer solchen Funktion kann man z.B. das Subgradientenverfahren anwenden, d.h. man bestimmt die Suchrichtung einfach in dem man *ein Element* des Subgradienten wählt. Solange man keinen stationären Punkt (d.h. ein Minimum im konvexen Fall) erreicht hat, enthält der Subgradient nie die 0 und man erreicht immer einen Abstieg im Zielfunktional. Für eine detaillierte Diskussion von Verfahren zur nichtdifferenzierbaren Optimierung sei auf [GeKa] verwiesen, sowie auf das einführende Kapitel in [Fl].

5.2 Lineare Optimierung

Das allgemeine lineare Optimierungsproblem hat die Form

$$c^T x \rightarrow \min_{x \in \mathbb{R}^N} \quad (5.1)$$

$$Ex = f \quad (5.2)$$

$$Cx \leq g, \quad (5.3)$$

mit einem Vektor c und Matrizen E und C . Da das Zielfunktional in jeder Richtung s mit $s^T c < 0$ fällt (d.h. in einem Halbraum), kann man einen Abstieg im Zulässigkeitsbereich so lange durchführen, bis man am Rand des Zulässigkeitsbereiches angelangt ist, und am Rand wieder so lange, bis man in einem Eckpunkt des Zulässigkeitsbereiches (der die Form eines Simplex hat) anlangt. Somit muss das Minimum eines linearen Optimierungsproblems immer in einem Eckpunkt liegen, und man kann einen Algorithmus darauf aufbauen, die Eckpunkte clever zu durchsuchen (Simplex-Algorithmus, cf. [GeKa, ChZa]).

Mit zunehmender Problemgröße und zunehmender Anzahl an Ungleichungsnebenbedingungen wächst die Anzahl der Eckpunkte des Simplex meist stark an, und das Simplex-Verfahren ist wenig effizient. Deshalb benutzt man gerne innere-Punkt-Methoden (siehe Barriere-Verfahren), die auf einfache nichtlineare, aber konvexe Probleme führen (cf. [GeKa]).

Lineare Optimierungsprobleme treten häufiger in ökonomischen Anwendungen (Kostenminimierung, optimale Produktionsplanung) als in Ingenieursanwendungen auf. Lineare Zielfunktionale mit nichtlinearen Nebenbedingungen findet man aber auch in der Mechanik und Elektrotechnik, z.B. bei der Gewichtsminimierung unter Spannungsnebenbedingungen. In der nichtlinearen Optimierung treten lineare Probleme auch als Teilprobleme in *SLP*-Algorithmen auf (sequentielle lineare Programmierung). Die Grundidee eines *SLP*-Verfahrens ist dieselbe wie bei *SQP*-Verfahren, das Zielfunktional wird aber nur linear statt quadratisch approximiert.

5.3 Diskrete Optimierung

Diskrete oder auch *kombinatorische Optimierung* nennt man die Minimierung von Funktionalen, die auf \mathbb{Z}^n definiert sind, d.h., die Variable können nur ganzzahlige Werte annehmen. Das klassische Beispiel der diskreten Optimierung ist das *Traveling-Salesman Problem*:

Ein Handelsreisender soll N Städte S_1, S_2, \dots, S_N bereisen, wobei die Reise von S_i nach S_j mit den Kosten C_{ij} verbunden ist. Die Route soll so gewählt werden, dass keine Stadt ausgelassen wird und die Kosten minimal sind.

Probleme dieser Art sieht man auch gerne als die Suche optimaler Wege in einem (durch die Kosten) bewerteten Graphen, die man für kleine Graphen z.B. durch den *Dijkstra-Algorithmus* und ähnliche Verfahren lösen kann, deren Grundidee im Prinzip ein schnelles

(rekursives) Ausprobieren von möglichen Wegen ist. Die meisten der üblicherweise auftretenden Probleme sind aber sehr schwierig zu lösen, und es existieren dafür keine Algorithmen mit polynomialen Aufwand in der Anzahl der Knoten (NP-hard). Deshalb wird die Laufzeit mit wachsender Knotenanzahl sehr schnell unrealistisch groß. Um schnellere Verfahren zu konstruieren, verwendet man sogenannte *Heuristiken*, d.h. man lässt zusätzliche Einsicht in das Problem einfließen, um noch schneller auszuprobieren. Sehr oft verwendet werden dabei sogenannte *Branch-and-Bound Methoden* (siehe etwa [Fl]).

Bei Optimierungsproblemen, in denen man Zielfunktional und Nebenbedingungen auf den reellen Zahlen definieren könnte, aber nur ganzzahlige Lösungen sucht, verwendet man gerne auch Algorithmen für kontinuierliche Probleme um ein reelles Minimum zu berechnen und betrachtet dann die nächstgelegenen ganzen Zahlen.

Für eine detaillierte Einführung in kombinatorische Optimierung verweisen wir auf [Fo, GrLoSch], eine kurze und leicht lesbare Einführung findet man auch in [Fl].

5.4 Globale Optimierung

Wir haben in den ersten Kapiteln gesehen, dass Algorithmen zur stetigen Optimierung immer nur lokal konvergieren, insbesondere kann man nicht die Konvergenz gegen ein globales Minimum erwarten. Um dies zu erreichen, verwendet man andere Ansätze, die man auch gerne mit den lokal konvergenten Verfahren kombiniert.

5.4.1 Stochastische Anfangswertbestimmung

Einer der einfachsten Ansätze zur globalen Optimierung ist die Bestimmung von verschiedenen Anfangswerten für lokal konvergente Verfahren durch einen Zufallszahlengenerator. Durch Erzeugung einer hinreichend grossen Anzahl von Anfangswerten wird man irgendwann nahe genug zum globalen Minimum kommen, um lokale Konvergenz dahin zu erhalten. Der Aufwand eines solchen Verfahrens ist aber nicht begrenzt, und es ist schwer abzuschätzen, wann man das globale Minimum erreicht hat.

5.4.2 Konvexe Relaxierung

Als konvexe Relaxierung bezeichnet man die Approximation eines Optimierungsproblems durch konvexe Probleme, deren Minima gegen ein globales Minimum des ursprünglichen Problems konvergieren. Falls man eine solche Relaxierung findet, kann man das globale Minimum effizient berechnen. Im allgemeinen hat man aber kaum Mittel zur Berechnung einer solchen Relaxierung zur Verfügung, dies funktioniert nur für spezielle Problemklassen gut, wie z.B. in der Variationsrechnung, wo man ein Integralfunktional der Form

$$f(u) = \int_{\Omega} h(x, u, \nabla u) dx$$

über Funktionen $u : \Omega \rightarrow \mathbb{R}$ minimieren will. In diesem Fall ist eine konvexe Relaxation gegeben durch

$$f^*(u) = \int_{\Omega} h^*(x, u, \nabla u) \, dx,$$

wobei h^* die konvexe Einhüllende von h ist. In manchen Fällen lässt sich zeigen, dass f und f^* dieselben Minima besitzen, d.h., man kann das ursprüngliche Problem einfach durch ein konvexes ersetzen.

5.4.3 Genetische Algorithmen

Genetische Algorithmen verwendet man allgemein zur Minimierung einer Funktion f über einem Gebiet $\Omega \subset \mathbb{R}^n$, sie basieren auf folgender Grundidee: man startet mit einer ursprünglichen Menge $P(0)$ (genannt *Anfangspopulation*), die Punkte in Ω enthält, und wertet die Funktionswerte dieser Punkte aus. Nun führt man iterativ folgende Prozedur aus:

- *Selektion*: in diesem Schritt bilden wir aus der Population $P(k)$ einen *Matingpool* $M(k) = \{m_1, \dots, m_N\}$ von derselben Grösse, wobei wir folgende Zufallswahl anwenden: der Punkt m_j ist gleich $x \in P(k)$ mit Wahrscheinlichkeit $p(x) = \frac{f(x)}{F_k}$, wobei $F_k = \sum_{\xi \in P(k)} f(\xi)$. Damit erhalten wir eine Liste von Elementen aus $P(k)$, deren Häufigkeit proportional zu ihrer *Fitness* ist, d.h. ein Element kommt umso öfter vor je niedriger der zugehörige Funktionswert ist.
- *Crossover*: In diesem Schritt wählt man jeweils zwei Elemente aus $M(k)$ (Eltern) und erzeugt daraus ein neues Element (Erbe).
- *Mutation*: In diesem Schritt werden die erzeugten Erben zufällig geändert und damit eine neue Population $P(k+1)$ erzeugt.

Zum Crossover und zur Mutation gibt es viele verschiedene Möglichkeiten. Eine der einfachsten besteht darin, alle Zahlen binär darzustellen und zufällig Ziffern zu verändern bzw. zwischen Eltern auszutauschen. Im Laufe des genetischen Algorithmus behält man natürlich immer das Element mit dem niedrigsten Funktionswert, um ein globales Minimum zu erreichen.

Genetische Algorithmen erfreuen sich zunehmender Beliebtheit unter Ingenieuren, vor allem wegen ihrer leichten Realisierbarkeit. In Bezug auf Laufzeit gilt aber dasselbe wie für stochastische Anfangswertbestimmung: die Algorithmen werden irgendwann konvergieren, der Aufwand bis dahin kann aber beliebig groß sein. Für detaillierte Darstellungen und Details verweisen wir auf [ChZa] und [He].

5.5 Web-Ressourcen

Im Internet findet man beinahe Unbegrenzte Information zum Thema Optimierung. Nützliche Wegweiser dafür sind:

- **Optimization and Operations Research Sites:**

http://www.rpi.edu/~mitchj/sites_or.html, umfangreiche Sammlung von Links zum Thema Optimierung.

- **Mathematical Programming Glossary:**

<http://carbon.cudenver.edu/~hgreenbe/glossary/>, ein Wörterbuch der mathematischen Optimierung.

- **Opt-Net:**

<http://optnet.itwm.fhg.de/opt-net/>, Newsgroup der DMV zum Thema Optimierung.

Software und Information dazu findet man am leichtesten auf folgenden Sites:

- **NEOS Guide Optimization Software:**

<http://www-fp.mcs.anl.gov/otc/Guide/SoftwareGuide/>, umfangreiche Sammlung von Software unterteilt nach Typ des Optimierungsproblems, auch mit Suchmaske.

- **Guide to Available Mathematical Software (GAMS):**

<http://gams.nist.gov/>, bietet auch eine Suchmaske nach Problemtyp (<http://gams.nist.gov/Problem.html>).

- **Mathtools.net:**

<http://www.mathtools.net/>, bietet Links zu Software in verschiedenen Programmiersprachen, unter anderem gibt es auch Seiten für Optimierung in MATLAB (<http://www.mathtools.net/MATLAB/Optimization/index.html>) und C++ (<http://www.mathtools.net/C++/Optimization/index.html>), Genetic Algorithms (http://www.mathtools.net/MATLAB/Genetic_algorithms/index.html), sowie Links zu Tutorials, Newsgroups, etc. (<http://www.mathtools.net/Learning/index.html>).

- **Mathworks:**

<http://www.mathworks.com/>, MATLAB und SIMULINK.

- **Iterative Methods for Optimization MATLAB Codes:**

http://www4.ncsu.edu/eos/users/c/ctkelley/www/matlab_darts.html, MATLAB Files zum Buch von Kelley [Ke].