2. Übung

Die Methode des kritischen Pfades (engl. critical path method) ist ein Verfahren zur Projektplanung, welches in den 1950er-Jahren entwickelt wurde. Unter anderem kann es verwendet werden, um diejenigen Aufgaben zu identifizieren, die zu einer Verzögerung der Gesamtdauer führen können. Während man den kritischen Pfad bei kleinen Projekten noch leicht von Hand bestimmen kann, so ist man bei Projekten mit einer Vielzahl von Aufgaben und Abhängigkeiten auf ein automatisiertes Verfahren angewiesen.

Aufgabe	Dauer	Vorgänger
A	1	-
B	3	A
C	2	A
D	4	A

Aufgabe	Dauer	Vorgänger
E	4	B, C
F	2	C, D
G	3	E
H	4	E, F

Aufgabe	Dauer	Vorgänger
I	1	F
J	2	G, H
K	3	H, I
L	2	J, K

Tabelle 1: Übersicht über die einzelnen Aufgaben im Projekt, deren Dauer und Abhängigkeiten.

Aufgabe 2.1

In Tabelle 1 sind die einzelnen Aufgaben mit ihren geplanten Zeitdauern für ein fiktives Projekt aufgelistet. Jede Aufgabe kann dabei mehrere Vorgänger haben, die zwingend abgeschlossen sein müssen, bevor die neue Aufgabe begonnen werden kann. Stellen Sie die Abhängigkeiten in Form eines Graphen dar, wobei jede Aufgabe einem Knoten, jede Dauer einem Knotengewicht und jede Abhängigkeit einer Kante entspricht. Das Projekt beginnt mit Aufgabe A und endet nach Abschluss von Aufgabe L.

Aufgabe 2.2

Im Folgenden soll der kritische Pfad für dieses Projekt mit Hilfe der linearen Programmierung bestimmt werden. Gegeben sind die Aufgaben $\mathcal{A} = \{\mathrm{A}, \mathrm{B}, \ldots, \mathrm{L}\}$, die Zeitdauern d_i für $i \in \mathcal{A}$ sowie die Vorgänger $\mathcal{P} = \{(i,j) \,|\, i \text{ ist Vorgänger von } j\}$. Als Optimierungsvariablen dienen die Gesamtdauer T des Projekts sowie die Anfangszeiten s_i jeder Aufgabe. Formulieren Sie ein geeignetes Optimierungsproblem mit Kostenfunktion und Beschränkungen um als Lösung die minimale Projektlaufzeit zu erhalten.

Aufgabe 2.3

Überführen Sie das Optimierungsproblem in die Standardform eines linearen Programms

$$\min_{\boldsymbol{x} \in \mathbb{R}^n} \quad \boldsymbol{c}^\mathrm{T} \boldsymbol{x} \tag{1a}$$

$$\text{u.B.v.} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \quad \boldsymbol{A} \in \mathbb{R}^{p \times n}, \quad \boldsymbol{b} \in \mathbb{R}^p \tag{1b}$$

$$\boldsymbol{x} \geq \boldsymbol{0}, \tag{1c}$$

indem Sie der Vorgehensweise im Skript folgen. Ergänzen Sie die MATLAB-Funktion `cpm_problem`, um die Matrix $\boldsymbol{A}$ sowie die Vektoren $\boldsymbol{b}$ und $\boldsymbol{c}$ für gegebene Projektdaten zu erstellen.

Aufgabe 2.4

Ein bekanntes Verfahren zur numerischen Lösung von linearen Optimierungsproblemen ist der Simplex-Algorithmus. Ergänzen Sie die MATLAB-Funktion `simplex`, um dieses Verfahren zu implementieren. Dabei soll in der Initialisierungsphase automatisch eine zulässige Basislösung bestimmt oder alternativ die Nichtlösbarkeit des Problems festgestellt werden. Bestimmen Sie anschließend die Lösung für das oben formulierte Optimierungsproblem. Tragen Sie die Anfangszeiten s_i in den Graphen ein und folgern Sie daraus, welche Aufgaben kritisch für die Projektlaufzeit sind.

Als Alternative zum Simplex-Algorithmus kommen vor allem Interior Point-Verfahren zum Einsatz. Die Grundidee besteht darin, die Komplementaritätsbedingung durch einen Parameter τ aufzuweichen und das resultierende nichtlineare Gleichungssystem iterativ mit $\tau \to 0$ zu lösen. Im Gegensatz zu Barrieremethoden, welche die Ungleichungsbeschränkungen mittels logarithmischer Strafterme zur Kostenfunktion addieren, sind Interior Point-Verfahren häufig numerisch besser konditioniert.

Aufgabe 2.5

Ergänzen Sie die MATLAB-Funktion `interiorpoint`, welche ein Interior Point-Verfahren zur Lösung von linearen Optimierungsproblemen implementiert. Beachten Sie, dass die Bedingungen $\boldsymbol{x}^k > \mathbf{0}$ und $\boldsymbol{\mu}^k > \mathbf{0}$ in jeder Iteration k eingehalten werden müssen. Implementieren Sie dazu eine Liniensuche mittels Backtracking. Untersuchen Sie den Einfluss des Reduktionsfaktors σ^k auf die Konvergenz und die Rechenzeit des Interior Point-Verfahrens bei der Lösung des oben formulierten Optimierungsproblems.

Aufgabe 2.6

Das Simplex- und das Interior Point-Verfahren weisen unterschiedliches Verhalten bezüglich der Skalierbarkeit der Problemgröße auf, d.h. der Dimension n der Optimierungsvariablen $\boldsymbol{x} \in \mathbb{R}^n$. Mit der MATLAB-Funktion `random_LP` können zufällige lineare Optimierungsprobleme generiert werden. Nutzen Sie diese Funktion, um die Rechenzeit der beiden Algorithmen in Abhängigkeit der Problemgröße zu untersuchen.