Technische Universität München
Fakultät für Mathematik
Diskrete Mathematik & Kombinatorische Optimierung
Prof. Dr. Andreas Wiese
PD Dr. René Brandenberg, M.Sc. Stefan Kober

Laden Sie die Datei `P2.zip` von Moodle herunter und entpacken Sie diese. Dort finden Sie 3 Unterordner, die den 3 Aufgaben entsprechen. Wir gehen davon aus, dass Sie Julia, sowie eine geeignete Entwicklungsumgebung bereits installiert haben und Sie mit den Grundlagen von Julia und JuMP vertraut sind. Installieren Sie bitte vor Bearbeitung der Aufgaben die Julia-Pakete `Cairo`, `Fontconfig`, `Graphs`, `GraphPlot` und `NLPModels`.

Programmieraufgabe 1. Dualität

Gegeben sei das folgende lineare Programm:

$$\begin{aligned} &\min_{x\in\mathbb{R}^3} -2x_1 - x_2 - 2x_3 \\ \text{u.d.N.} \\ &x_1 + 2x_2 && \leq 6 \\ &x_1 + x_2 + x_3 && \leq 4 \\ &3x_1 + x_3 && \leq 6 \\ &x_3 && \leq 3 \\ &x_1, x_2, x_3 && \geq 0 \end{aligned}$$

a) Implementieren Sie das primale und das duale Programm in den vorgegebenen Dateien `primal.jl` und `dual.jl` und lösen Sie beide mit dem Solver GLPK.

   Was ist der Wert der primalen und der dualen Lösung?

b) Nutzen Sie das duale Programm in `dual.jl`, um eine Lösung für das primale Problem zu finden.

   Ist das Ergebnis gleich wie in `primal.jl`?

   Hinweis: Es ist in JuMP möglich, auch die duale Lösung eines gelösten linearen Optimierungsproblems auszulesen (recherchieren Sie das kurz).

c) Wie können Sie mit Hilfe des Simplex Tableaus auch selbst eine duale Lösung eines gelösten linearen Programms erhalten?

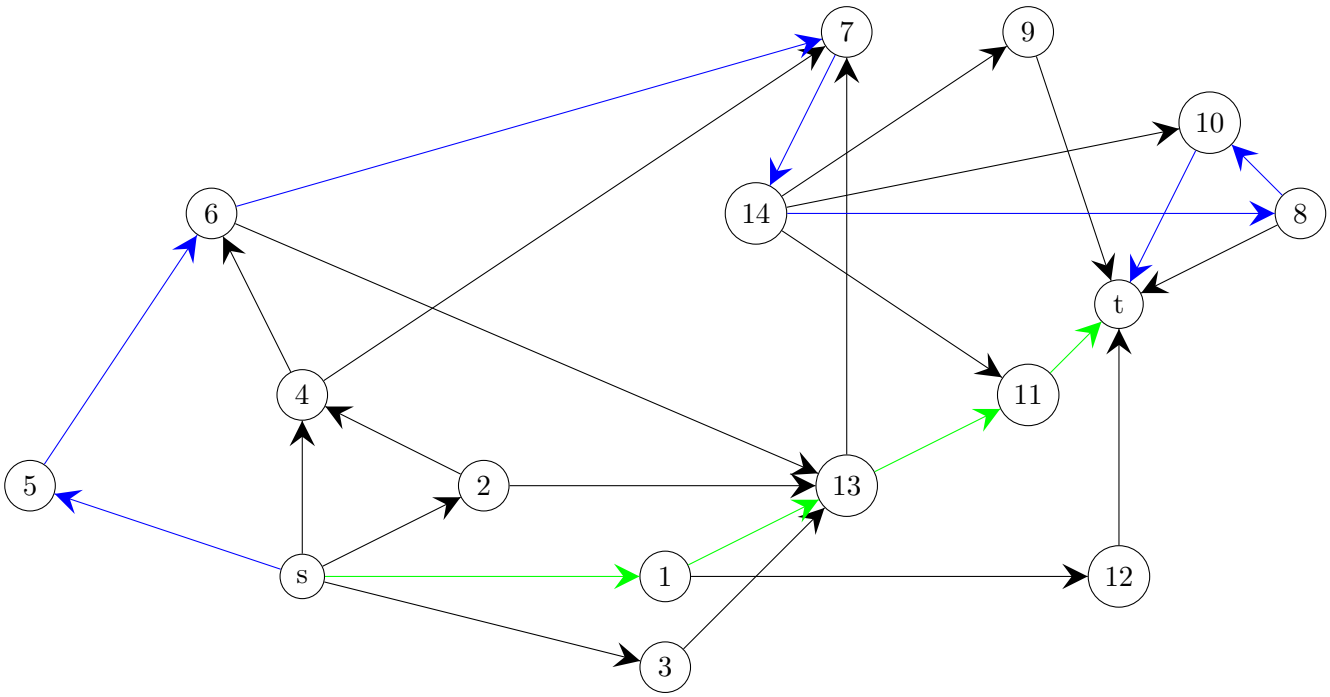
   Hinweis: Werfen Sie nochmal einen Blick auf Präsenzaufgabe 4.1. Wie wurde das Problem dort modifiziert?

Programmieraufgabe 2. Stadtplanung

Sie sind mit der Planung eines robusten Straßennetzes betraut. Das Ziel soll sein, möglichst viele ausfallsichere Wege zwischen einem Startpunkt s und einem Endpunkt t zu finden, damit auch im Falle eines Unfalls mit Vollsperrung noch genügend Alternativen vorhanden sind, um den Endpunkt zu erreichen.

Formal sei Ihnen dafür ein Netzwerk $G = (V, A)$ aus Knoten V und gerichteten Kanten $A \subseteq V \times V$ gegeben. Ihre Aufgabe ist es, zunächst eine möglichst große Menge an paarweise kantendisjunkten

Pfaden von s nach t in diesem Netzwerk zu finden. Ein Beispiel für zwei kantendisjunkte Pfade ist in der Skizze in grün und blau gegeben.



Dieses Problem lässt sich wie folgt modellieren:

$$\max \quad F \tag{1}$$

$$\text{u.d.N.} \quad \sum_{a \in \delta^+(s)} x_a - \sum_{a \in \delta^-(s)} x_a - F = 0$$

$$\sum_{a \in \delta^+(t)} x_a - \sum_{a \in \delta^-(t)} x_a + F = 0$$

$$\sum_{a \in \delta^+(v)} x_a - \sum_{a \in \delta^-(v)} x_a = 0 \qquad \forall v \in V \setminus \{s,t\}$$

$$x_a \in \{0,1\} \qquad \forall a \in A \tag{2}$$

Die Variablen $x_{i,j}$ für $(i,j) \in A$ geben dabei an, ob einer der kantendisjunkten Pfade die Kante (i,j) nutzt. Die in der Skizze dargestellte Lösung hat also formal folgende Struktur:

$$x_{s,1} = x_{1,13} = x_{13,11} = x_{11,t} = x_{s,5} = x_{5,6} = x_{6,7} = x_{7,14} = x_{14,8} = x_{8,10} = x_{10,t} = 1$$

$$x_{i,j} = 0 \qquad \text{sonst}$$

Die Zielfunktion 1 zählt hierbei die Anzahl der disjunkten Pfade. Die sonstigen linearen Nebenbedingungen stellen sicher, dass die Pfade alle von s bis t fortgesetzt werden. Die Bedingung 2 schränkt die Variablen auf 0 und 1 ein, damit die Entscheidungen wohldefiniert sind. In der Vorlesung *Diskrete Optimierung* werden Sie jedoch zeigen, dass in diesem speziellen Fall die Einschränkungen

$$x_{i,j} \leq 1 \qquad \forall (i,j) \in A \tag{3}$$

$$x_{i,j} \geq 0 \qquad \forall (i,j) \in A \tag{4}$$

auch ausreichen, und zu den selben Ergebnissen führen. Um die Theorie für lineare Programme verwenden zu können, ersetzen wir daher im Folgenden die Einschränkung 2 durch 3 und 4.

a) Vervollständigen und lösen Sie das in `failsafepaths.jl` anfgefangene Modell.

   Wie viele kantendisjunkte Pfade gibt es höchstens zwischen s und t?

b) Betrachten Sie das folgende lineare Programm. Überprüfen Sie, ob es das duale Programm zu obigem Modell ist.

   Wir nutzen dabei die Variablen y_v für alle $v \in V$ und z_a für alle $a \in A$.

$$\begin{aligned} \min \quad & \sum_{a \in A} z_a & & \\ \text{u.d.N.} \quad & y_v - y_w + z_{v,w} \geq 0 & & \forall (v,w) \in A \\ & -y_s + y_t = 1 & & \\ & z_a \geq 0 & & \forall a \in A \end{aligned}$$

   Die Werte der Variablen y_v teilen die Menge der Knoten in zwei Klassen, wie Sie in der Vorlesung gelernt haben. Wie können Sie mit eventuell auftretenden fraktionalen Werten umgehen? Wir konzentrieren uns nun auf die Variablen y_v und die sich ergebende Partition der Knotenmenge entsprechend der Zuweisungen. Lassen Sie die Optimierung erneut laufen und lesen Sie die entsprechende duale Lösung aus. Visualisieren Sie den Graphen mit der zugehörigen Partition (der Code dafür ist in `failsafepaths.jl` gegeben). Nutzen Sie diese Visualisierung, um eine Begründung zu finden, warum es nicht mehr kantendisjunkte Pfade zwischen s und t gibt als in der ersten Teilaufgabe berechnet.

c) Fügen Sie genau eine Kante zu dem Graphen hinzu (nicht einfach von s nach t), um die Anzahl an kantendisjunkten Wegen um eins zu erhöhen. Überprüfen Sie Ihre Lösung mithilfe des linearen Programms.

Programmieraufgabe 3. Lineare und quadratische Regression

Bei der linearen bzw. quadratischen Regression geht es darum, eine Menge von Punkten durch eine lineare bzw. quadratische Funktion zu approximieren, und dabei den Fehler zu minimieren. Formal sei Ihnen dabei für ein $n \in \mathbb{N}$ eine Menge an Punkten $\{(x_i, y_i)\}_{i \in \{1,\dots,n\}}$ gegeben. Gesucht ist nun eine lineare bzw. quadratische Funktion $f : \mathbb{R} \to \mathbb{R}$. Der Fehler in einem Punkt (x_i, y_i) ist als $(y_i - f(x_i))^2$ definiert.

Daraus lässt sich das Optimierungsproblem für die lineare Regression wie folgt formulieren:

$$\min \quad \sum_{i=1}^{n} (y_i - (a_0 + a_1 \cdot x_i))^2 \tag{5}$$

Das in 5 definierte Problem ist ein zwei-dimensionales, unrestringiertes Optimierungsproblem mit quadratischer Zielfunktion, welches Sie im Folgenden in Julia implementieren und mit der Gradientenmethode lösen sollen.

a) Machen Sie sich zunächst mit dem Code in `gradmethod.jl` und `armijo.jl` vertraut. Sind die Gradientenmethode und die Armijo-Regel so implementiert, wie Sie dies in der Vorlesung gelernt haben?

b) Vervollständigen Sie den Gradienten der Zielfunktion 5 in der Datei `linreg.jl`. Führen Sie die Datei `main.jl` aus und überprüfen sie mithilfe der Ausgabe, des Kosten-Plots und der Plots im Unterordner `plots/`, was dieser gemacht hat.

c) Stellen Sie das Parameter `plot_interval` in der Funktion `intermediate_callback` auf 100, um zu verhindern, dass im Folgenden zu viele Bilder geplottet werden.

   Im oberen Bereich der Datei `main.jl` finden Sie einen weiteren Datensatz `input_x`, `input_y`. Dieser lässt sich nicht gut durch eine lineare Funktion approximieren. Erstellen Sie ein Modell für quadratische Regression und implementieren Sie die Zielfunktion und den Gradienten in der vorbereiteten Datei `quadreg.jl`.

   Wenden Sie beide Algorithmen auf die beiden Datensätze an und testen Sie dabei unterschiedliche Startpunkte `xs`. Was fällt Ihnen dabei auf?