Operations Research

Primaler Simplex

Voraussetzung: Negativer Eintrag in z Zeile

1. Standardform → Maximieriungsfunktion, Zielfunktion nach 0 umstellen, <=
2. Pivotelement finden
    a. Pivotspalte: Für alle $\mathbf{z_j < 0}$ → kleinstes z finden (betragsmäßig größter Wert)
    b. Pivotzeile: Für alle $\mathbf{a_{ij} > 0}$ → **min** $\left\{\frac{b_i}{a_{ij}}\right\}$
3. Neues Tableau bestimmen

Dualer Simplex

Voraussetzung: Negativer Eintrag in b Spalte

1. Zeile: Wähle kleinsten $\mathbf{b_i}$ Wert
2. Spalte: Für alle $\mathbf{a_{ij} < 0}$ → **max** $\left\{\frac{z_j}{a_{ij}}\right\}$

Sonderfälle

- **Keine zulässige Lösung** unzulässig
    a. Dualer Simplex: Pivotzeile hat nur Elemente ≥ 0 (看行)
- **Zulässige Lösung**
    a. Unbeschränktheit (只有 zulässige Basislösung)
        i. Pivotspalte hat nur Elemente ≤ 0
    b. Redundanz
        i. Alle Werte (außer Einheitsvektor und b_i Spalte) ≤ 0 (看列)
- **Optimale Lösung**
    a. Primale Degeneration · Im Optimum schneiden sich n+1 Nebenbedingungen ($\mathbb{R}^n$) · Sonderfall der Redundanz
        i. Eine Basisvariable $b_i = 0$
    b. Duale Degeneration · kein Sonderfall der Redundanz ·
        i. Eine Nichtbasisvariable mit $z_j = 0$

Duales Problem

1. Kapazitätsbegrenzungen der Nebenbedingung kommen in die Zielfunktion
2. Spalte als neue Zeile schreiben

Maximierungsproblem	Minimierungsproblem
Zielfunktion: $\max F_{Max}(x)$	Zielfunktion: $\min F_{Min}(u)$
Nebenbedingungen: i-te NB: ≤ i-te NB: ≥ i-te NB: =	Variablen: $u_i \geq 0$ $u_i \leq 0$ $u_i \in \mathbb{R}$
Variablen: $x_j \geq 0$ $x_j \leq 0$ $x_j \in \mathbb{R}$	Nebenbedingungen: j-te NB: ≥ j-te NB: ≤ j-te NB: =

maX → x Variablen

miN → Nebenbedingung

primales Problem \ duales Problem	unbeschränkt	keine Lösung
unbeschränkt	✗	✔
keine Lösung	✔	✔

- Hat das primale Problem P eine **optimale Lösung x^***, so besitzt das zugehörige duale Problem D eine optimale Lösung u^* und es gilt $z_P(x^*) = z_D(u^*)$
  Ist P unbeschränkt, so besitzt D keine zulässige Lösung. Ist D unbeschränkt, so besitzt P keine zulässige Lösung. Achtung: Der Umkehrsatz ist notwendig, aber nicht hinreichend!
- Sei A ein Maximierungsproblem mit der **zulässigen Lösung $(x_1, \ldots, x_k)$** und sei B das duale (Minimierungs-)Problem von A mit der **zulässigen Lösung $(u_1, \ldots, u_n)$**.
  Dann gilt $z_p(x_1, \ldots, x_k) \leq z_D(u_1, \ldots, u_n)$. (Einschließungssatz / schwache Dualität)

Substitutionskoeffizienten

- Geben an, um wie viele Einheiten sich die Basisvariable zur Zeile i erhöht ($a_{ij} < 0$) bzw. verringert ($a_{ij} > 0$), wenn man die Nichtbasisvariable zur Spalte j um eine Einheit erhöht.

Schattenpreise

- Kostenmäßige Werte jeder Einheit der Mindestanforderungen (Kapazitätrestiktion)
- Erhöht (senkt) man die Anforderungen um eine Einheit, verschlechtert (verbessert) sich der Ziel- funktionswert um den angegebenen Wert.

Sensitivitätsanalyse

Voraussetzung: Keine Degeneration

1. Singuläre Sensitivitätsanalyse: eine Variable unter Beibehaltung der übrigen wird sinnvoll variiert

2. Multiple Sensitivitätsanalyse: Änderung mehrerer Variablen („Dreipunktschätzung")

Zielfunktionskoeffizient

- **Nichtbasisvariable**
  - $c_k^- = \infty$
  - $c_k^+ = c_k^*$
- **Basisvariable**
  - $c_k^- = \infty$, falls alle $a_{kj}^* \leq 0$ mit j $\neq$ k, sonst
  - $c_k^- = \min \frac{c_j^*}{a_{kj}^*}$ mit j $\neq$ k für positive a_{kj}^*
  - $c_k^+ = \infty$, falls alle $a_{kj}^* \geq 0$ mit j $\neq$ k, sonst
  - $c_k^+ = \min \frac{-c_j^*}{a_{ki}^*}$ mit j $\neq$ k für negative a_{kj}^*

Ressourcenbeschränkungen

- **Basisvariable**
  - $b_k^- = x_q$
  - $b_k^+ = \infty$
- **Nichtbasisvariable**
  - $b_k^- = \infty$, falls alle $a_{iq}^* \leq 0$, sonst
  - $b_k^- = \min \frac{b_i^*}{a_{iq}^*}$ für positive a_{iq}^*
  - $b_k^+ = \infty$, falls alle $a_{iq}^* \geq 0$, sonst
  - $b_k^+ = \min \frac{-b_i^*}{a_{iq}^*}$ für negative a_{iq}^*

Merke: Bei - → <= 0 und bei + → größer als 0 // mehr Ressourcen geht immer, weniger Ziel auch

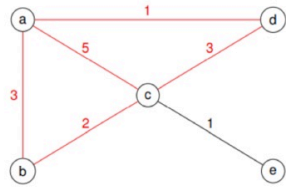
Graphentheorie

- Ein Graph ohne parallele Kanten und ohne Schlinge wird als **schlichter Graph** bezeichnet.
- Ein schlichter, gerichteter Graph mit endlicher Knotenmenge heißt **Digraph**.
- Ein Graph mit parallelen Kanten wird als **Multigraph** bezeichnet.
- Ein geschlossener Weg heißt **Zyklus**.

b. Nun stellen sie das spezielle Problem für den gegebenen Graphen auf.

$$\min z = 3x(a,b) + 5x(a,c) + 1x(a,d) + 2x(b,c) + 3x(c,d) + x(c,e)$$
$$s.t.\ \ x(a,b) + x(a,c) + x(a,d) + x(b,c) + x(c,d) + x(c,e) = 4$$
$$x(a,b) + x(a,c) + x(b,c) \leq 2$$
$$x(a,c) + x(a,d) + x(c,d) \leq 2$$
$$x(a,b) + x(a,c) + x(a,d) + x(b,c) + x(c,d) \leq 3$$
$$x(i,j) \in \{0,1\}$$

für minimale Spannbäume



TSP: 所有点走1次回来

ZF: $\min z = \sum_{(i,j)\in E} w(i,j) \cdot x(i,j)$

NB1: $s.t. \sum_{j=1}^{n} x(i,j) = 1$ für $i = 1,\dots,n$

NB2: $\sum_{i=1}^{n} x(i,j) = 1$ für $j = 1,\dots,n$

NB3: $\sum_{(i,j)\in E(S)} x(i,j) \leq |S| - 1$ für jede Teilmenge $S \subset V$

NBX: $w(i,j) \in \mathbb{R}$

Def. B.: $x(i,j) \in \{0,1\}$

$$\min z = 3 * x(a,b) + 5 * x(a,c) + 1 * x(a,d) + 2 * x(b,c) + 3 * x(c,d)$$
$$+3 * x(b,a) + 5 * x(c,a) + 1 * x(d,a) + 2 * x(c,b) + 3 * x(d,c)$$

Algorithmus

Funktionsweise

Knoten a: $x(a,b) + x(a,c) + x(a,d) = 1$

Knoten b: $x(b,a) + x(b,c) = 1$

Knoten a: $x(b,a) + x(c,a) + x(d,a) = 1$

Knoten b: $x(a,b) + x(c,b) = 1$

$x(a,b) + x(b,a) \leq 1$

$x(a,c) + x(c,a) \leq 1$

$x(a,b) + x(b,c) + x(c,a) \leq 2$

$x(a,c) + x(c,b) + x(b,a) \leq 2$

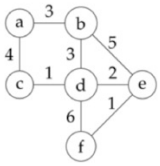
Kruskal	Kürzeste Kante gesamt finden, dann hocharbeiten
Greedy	Von Knoten ausgehend die kürzeste Kante markieren, keine Kreise
Dijkstra	Tabelle mit Gewicht und Vorgänger
Yen	Kürzesten Weg bestimmen und alternativen Pfad

- Ein Weg ist die Folge von gerichteten Kanten, jeweils vom Pfeilende zur Pfeilspitze. Ein geschlossener Weg heißt Zyklus.

Bellman-Ford

kürzeste Wege:

$$\min z = 3 \cdot (x(a,b) + x(b,a)) + 4 \cdot (x(a,c) + x(c,a))$$
$$+1 \cdot (x(c,d) + x(d,c)) + 3 \cdot (x(b,d) + x(d,b))$$
$$+5 \cdot (x(b,e) + x(e,b)) + 2 \cdot (x(d,e) + x(e,d))$$
$$+6 \cdot (x(d,f) + x(f,d)) + 1 \cdot (x(e,f) + x(f,e))$$



Start: a: $x(c,a) + x(b,a) - (x(a,b) + x(a,c)) = -1$

Knoten d: $x(c,d) + x(b,d) + x(f,d) + x(e,d) - (x(d,c) + x(d,b) + x(d,f) + x(d,e)) = 0$

End f: $x(d,f) + x(e,f) - (x(f,d) + x(f,e)) = 1$

- **Adjadenzmatrix**
  - 0 auf der Diagonalen
  - 1, falls verbunden
  - Für ungerichtete Graphen ist die Adjazenzmatrix symmetrisch.
- **Inzidenzmatrix**
  - 1, falls zwei Knoten verknüpft
  - 0, falls nicht verbunden
  - -1, falls Endpunkt
- **Bewertungsmatrix B**
  - ∞, falls nicht verbunden
  - Kantengewicht, falls verbunden
- **Multiplikation**
  - $U^{(2)} = U^{(1)} \otimes B = U^{(1)} \otimes U^{(1)}$
  - $U_{ab}^{(2)} = \min\{1.\text{Eintrag Zeile a} + 1.\text{Eintrag Spalte b};\ 2.\text{Eintrag a} + \text{b};\ \dots\}$

Branch-and-Bound Algorithmus

- Zerlegung des Optimierungsproblems in kleinere Teilprobleme (Branching)
- Entscheidung, welches Teilproblem weitergeführt oder durch anderes dominiert wird (Bounding)

Auswahlregel für Variable (Branching)

- **Zufallsauswahl**
- **Fraktionellste Variable (1/2-Regel)** Wähle diejenige Variable zum Einschränken, deren aktueller, nicht ganzzahliger Anteil näher an 1/2 liegt
- **Strong Branching** Wahl derjenigen Strukturvariable, die den Zielfunktionswert am meisten verändert, das heißt, den größten Zielfunktionskoeffizienten besitzt

Auswahlstrategie für Teilprobleme (Bounding)

- **Maximum Upper Bound (MUB)** Wähle Problem mit bestem Zielfunktionswert aus Liste (beachte die Optimierungsrichtung)
- **Tiefensuche** Wähle Problem aus Liste, welches als letztes eingefügt wurde (LIFO)
- **Breitensuche** Wähle Problem aus Liste, welches als erstes eingefügt wurde (FIFO)

Eliminierung von Teilproblemen (Ausloten)

- **Ganzzahligkeit** Teilproblem ist optimal ganzzahlig gelöst
- **Beschränkung** Zielfunktionswert ist schlechter als der eines bereits optimal gelösten ganzzahligen Teilproblems (wird dominiert)
- **Unzulässigkeit** Der zulässige Bereich ist leer

Gomory-Algorithmus

1. Beschneidung des Lösungsraumes durch weitere Schranken, sogenannte Schnittebenen
2. Schnittebenen sind zusätzliche Nebenbedingungen, die von allen zulässigen, ganzzahligen Lösungspunkten erfüllt werden
3. Momentan optimaler Punkt des linearen Problems wird „abgeschnitten"
4. Es werden solange weitere Schnittebenen hinzugefügt, bis eine zulässige Lösung erreicht ist

Gomory 算法 1. 通过进一步的边界切割解空间，即所谓的截面平面 2. 截面是所有允许的整数解点都满足的附加约束 3. 目前线性问题的最优点是"截断" 4.添加更多的切割平面，直到达到可接受的解决方案

Rucksackproblem

Nr.	Gegenstand	Kosten	Nutzen	Nutzen/Kosten	Rang
1	Ziegelsteine	6	1		
2	Zelt	7	3		
3	Flasche Bier	4	2		
4	Mückenschutz	2	5		
5	Grill	9	4		

1. Reihenfolge festlegen basierend auf Rang
2. Bruch auf 0 und auf 1 setzen und als 1.Rang festlegen
3. Restliche Werte berechnen

Binäre Variablen

$$\max z = 6x_1 + 4x_2 + 4x_3 - 200y_1 - 100y_2$$

Fixkosten

Wenn x_1 produziert wird, fallen y_1 Fixkosten an

$$x_1 \leq BIG \cdot y_1$$

Entweder-Oder

Wenn x_1 produziert wird, dann mindestens 10

$$x_1 \leq BIG \cdot y_1$$

$$10 - x_1 \leq BIG \cdot (1 - y_1)$$

Wenn-dann

Wenn x_2 und x_3 mehr als 20 sind, dann x_1 min 10

$$x_1 \leq BIG \cdot y_1$$

$$10 - x_1 \leq BIG \cdot q$$

$$q \begin{cases} 0, falls\ x_2 + x_3 \leq 20 \\ 1, falls\ x_2 + x_3 \geq 20 \end{cases}$$

Hinweis: BIG wird als hinreichend große Zahl definiert

Dynamische Optimierung

k = 3

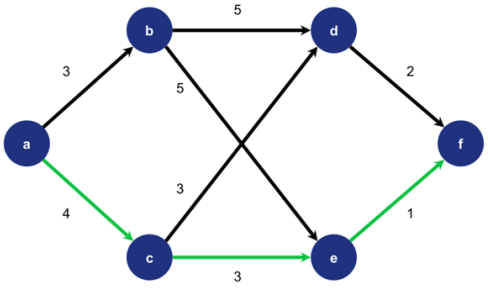
	f	x_3^*	$c_3^*(x)$
d	2	f	2
e	1	f	1

k = 2

	d	e	x_2^*	$c_2^*(x)$
b	7	6	e	6
c	5	4	e	4

k = 1

	b	c	x_1^*	$c_1^*(x)$
a	9	8	c	8



- Einteilung in Stufen, mit letzter Stufe beginnend nach vorne arbeiten
- x_k^* bezeichnet den „ausgewählten", effizienteren Knoten
- c_k^* bestimmt die bisherige Gesamtlänge

Effizienzanalyse

- Berechne Produktivität eines Unternehmens über $\frac{y_A}{x_A}$ mit Output y und Input x
- Die Produktivität des Unternehmens A relativ zu Unternehmen B wird als **Efficiency Ratio (E)** bezeichnet: $E = \frac{y_A}{x_A} / \frac{y_B}{x_B} = \frac{y_A}{y_B} / \frac{x_A}{x_B}$

Farrell-Effizienz

- Inputbasiert: $\boldsymbol{E} = \frac{\boldsymbol{x^*}}{\boldsymbol{x}}$ mit $0 < E \leq 1$ Outputbasiert: $\boldsymbol{F} = \frac{\boldsymbol{y^*}}{\boldsymbol{y}}$ mit $F \geq 1$

Dominanz

- Eine Input-Output-Kombination x^2, y^2 dominiert die Input-Output-Kombination x^1,y^1 ,wenn $x^2 \leq x^1$, $y^2 \geq y^1$ und $x^1,y^1 \neq x^2,y^2$

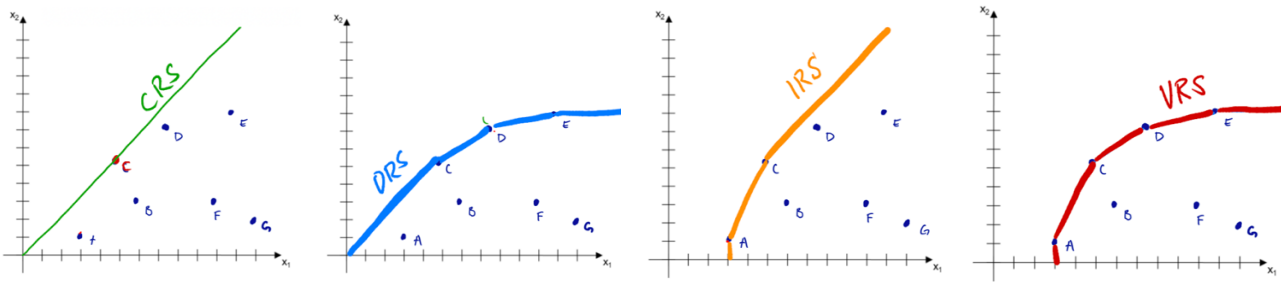
Effizienz nach Koopmann

- Eine Input-Output-Kombination ist effizient, wenn es von keiner weiteren dominiert wird

Technologieannahmen

- Free Disposability
    - Input: Es kann immer mehr Input verwendet werden für denselben Output
    - Output: Es kann immer weniger Output produziert werden mit demselben Input
- Konvexität: jede lineare Kombination von zwei Produktionsplänen ist ebenfalls möglich
- Additivität: Addition zwei Produktionsplänen ist möglich (Teil des Technologie-Sets)
- Skalierung: Vier verschiedene Arten (siehe unten)

Konstante Skalenerträge (CRS)	Fallende Skalenerträge (DRS)	Steigende Skalenerträge (IRS)	Variable Skalenerträge (DRS)



Julia

- **Wert definieren**

  Zahnpasta = ["Colgate Total", "Dentagard", "Advanced White"]

- **Wert deklarieren**

  # Rohstoffverbrauch in Gramm
  b = dict(
  "Colgate Total" => 50,
  "Dentagard" => 100,
  "Advanced White" => 80
)

- **Model deklarieren**

  past = Model(with_optimizer(Clp.with_optimizer)