

1 Bewegungsplanung bei unvollständiger Information

1.1 Ausweg aus einem Labyrinth

1.1.1 Pledge-Strategie

Input: polygonales Labyrinth L, Roboter R, Drehwinkel $\varphi \in \mathbb{R}$
Output: Ausweg aus Labyrinth falls möglich, ansonsten Endlosschleife

- While $R \in L$
 - gehe vorwärts, bis $R \notin L$ oder Wandkontakt
 - gehe links der Wand, bis $R \notin L$ oder $\varphi = 0$

1.2 Zum Ziel in unbekannter Umgebung

1.2.1 Wanze (Bug)

Input:

- P_1, \dots, P_n disj. einf. zsh. endl. poly. Gebiete aus \mathbb{R}^2
- $\mathbf{s}, \mathbf{z} \in \mathbb{R}^2 \setminus \bigcup_{i=1}^n P_i$
- R Roboter mit Position \mathbf{r}

Output:

- While $\mathbf{r} \neq \mathbf{z}$
 - laufe in Richtung \mathbf{z} bis $\mathbf{r} = \mathbf{z}$ oder $\exists i : \mathbf{r} \in P_i$
 - If $\mathbf{r} \neq \mathbf{z}$
 - umlaufe P_i und suche ein $\mathbf{q} \in \arg \min_{\mathbf{x} \in P_i} ||\mathbf{x} - \mathbf{z}||_2$
 - gehe zu \mathbf{q}

terminiert.
Universales Steuerwort: Führt für alle Startpunkte zum geg. Ziel. (ungültige Befehle werden ignoriert)

1.3 Behälterproblem (bin packing)

Maximale Füllmenge h , verteile Zahlenmenge auf möglichst wenige Behälter. NP-hart.

First fit

- $B_1, \dots, B_m \leftarrow \emptyset$
- For $i = 1, \dots, m$
 - Bestimme kleinstes j mit $b_i + \sum_{b \in B_j} b \leq h$
- Füge b_i zu B_j hinzu

ist 2-kompetitiv.
Algorithmus A ist **c-kompetitiv** falls $k_A \leq a + ck_{min}$ für alle Eingaben

Türsuche

- Wähle Erkundungstiefen $f_i > 0$ für $i \in \mathbb{N}$
- For $i := 1$ to ∞ (stoppe, wenn Tür gefunden)
 - gehe f_i Meter die Wand entlang und zurück
 - wechsele Laufrichtung

$d := \text{dist}(\mathbf{s}, \text{Tür}) = f_n + \varepsilon \in (f_n, f_{n+1}]$
Legt $L = 2 \sum_{i=0}^n f_i + d$ zurück (oder $n+1$)
 $L \in \Theta(n^2) = \Theta(d^2)$
Bestmöglich: 9-kompetetitiv (z.B. für $f_i = 2^i$)

1.4 Sternsuche

Gleich Türsuche, nur mit mehr als zwei Wänden (Halbgeraden).
Bestmöglich: Für $f_i = (\frac{m}{m-1})^i$ c-kompetitiv mit $c := 2m(\frac{m}{m-1})^{m-1} + 1 < 2me + 1$

1.5 Suche in Polygonen

Roboter R sucht Weg in polygonalem Gebiet P mit n Ecken von s nach z.
Weglängen: gefunden: l , kürzest: d
Strategie existiert mit $\frac{l}{d} \in O(n)$
Baum der kürzesten Wege (BkW) (Blätter sind Polygonecken)

2 Konvexe Hüllen

2.1 Dualität

$\mathbf{x} := [1 \ \bar{\mathbf{x}}]^t, \bar{\mathbf{x}} \in \mathbb{R}^d$ bilden *affinen Raum* A^d .
 $\mathbf{u}^t \mathbf{x} := [u_0 \ u_1 \ \dots \ u_d] \cdot [1 \ x_1 \ \dots \ x_d]^t \geq 0$
 \mathbf{u} bezeichnet Halbraumvektor und \mathbf{x} einen seiner Punkte
Nur betrachtet mit $(1 \ 0 \ \dots \ 0)^t$ im Inneren, d.h. $u_0 > 0$, normiert $u_0 = 1$.
 \mathbf{u}^* ist *dual* zu \mathbf{u} und bezeichnet den Halbraum.
 $\mathbf{x} \in \mathbf{u}^* \Leftrightarrow \mathbf{u} \in \mathbf{x}^*$ (Dualität)

2.2 Konvexe Mengen

Verbindungsstrecke
 $\mathbf{x} := \mathbf{a}(1-t) + \mathbf{b}t, \quad t \in [0, 1]$ wird genannt **ab**.
 $M \subset A$ ist *konvex* wenn sie zu je zwei ihrer Punkte auch die Verbingungsstrecke enthält.
Konvexe Hülle $[M]$ von M ist Schnitt aller konvexen Obermengen.
Ist $M \subset A$ bilden alle Halbräume, die M enthalten, eine konvexe Menge im Dualraum.
Ist $M^* \subset A^*$ eine Halbraummenge, bilden alle Punkte, die in allen $m^* \in M^*$ enthalten sind, eine konvexe Menge im Primalraum A .

2.3 Konvexe Polyeder P

ist Schnitt endlich vieler Halbräume.
Rand ∂P ; Facetten darauf.
Jede Facette liegt auf Rand eines Halbraums (FHR)
 P ist konvexe Hülle seiner Eckenmenge
Ist P ein konvexes Polyeder mit den Ecken $\mathbf{p}_1, \dots, \mathbf{p}_e$ und den FHRen $\mathbf{u}_1^*, \dots, \mathbf{u}_f^*$, hat die Menge $U^* := \{\mathbf{u}^* | \mathbf{u}^* \supset P\} \subset A^*$ die Ecken $\mathbf{u}_1^*, \dots, \mathbf{u}_f^*$ und die FHRe $\mathbf{p}_1, \dots, \mathbf{p}_e$.
Dual ausgedrückt heißt das, dass die Menge $U := \{\mathbf{u} | \mathbf{u} \supset P\} \subset A$ die Ecken \mathbf{u}_i und die FHRe \mathbf{p}_i^* hat.
Polyeder P und $U \subset A$ heißen dual zueinander.

2.4 Euler: Knoten, Kanten, Facetten

v Knoten, e Kanten, f Seiten
Eulers Formel: $v - e + f = 2$

2.5 Datenstruktur für Netze

Für jede Ecke \mathbf{p} :

- Koordinaten von \mathbf{p}
- Liste von Zeigerpaaren:
 - die ersten Zeiger im Gegenuhrzeigersinn auf alle Nachbarn von \mathbf{p}
- Sind $\mathbf{p}, \mathbf{q}, \mathbf{r}$ im GUS geordnete Nachbarn einer Facette und weist der 1. Zeiger eines Paares auf \mathbf{q} , zeigt der 2. Zeiger indirekt auf \mathbf{r} . Er weist auf das Zeigerpaar von \mathbf{q}

2.6 Konvexe Hülle

Input: $P := (\mathbf{p}_1, \dots, \mathbf{p}_n) \subset A^3$
Output: $[P]$

1. Verschiebe P sodass Ursprung in P liegt
2. $U_4 \leftarrow \mathbf{p}_1^* \cap \dots \cap \mathbf{p}_4^*$
3. For $i = 5, \dots, n$
 - (falls $U_4 \subset \mathbf{p}_i^*$, markiere \mathbf{p}_i als gelöscht
 - sonst verknüpfe \mathbf{p}_i bidirektional mit einem Knoten von $U_4 \notin \mathbf{p}_i^*$
4. For $i = 5, \dots, n$
 - $U_i \leftarrow U_{i-1} \cap \mathbf{p}_i^*$
 - ...zeug
5. Dualisiere, verschiebe und gib $\bigcap_{\mathbf{u} \in U} \mathbf{u}^* - \mathbf{v}$ aus

3 Distanzprobleme

3.1 Voronoi-Gebiet

eines der Punkte \mathbf{p}_i ist $V_i = \{\mathbf{x} \in \mathbb{R}^2 | \forall j = 1, \dots, n : ||\mathbf{x} - \mathbf{p}_i||_2 \leq ||\mathbf{x} - \mathbf{p}_j||_2\}$
 V_i ist konvex da Schnitt der Halbebenen.
Voronoi-Kreis (Punkte des Schnitts von drei Voronoi-Gebieten) ist *leer*.

3.2 Delaunay-Triangulierung

Delaunay-Triangulierung $D(P)$ einer Punktmenge P hat Kantenmenge $\{\mathbf{p}_i \mathbf{p}_j | V_i \cap V_j \text{ ist Kante des Voronoi-Diagramms } V(P)\}$.
Ist der zu $V(P)$ duale Graph. Die Gebiete von $D(P)$ sind disjunkte Dreiecke und zerlegen die konvexe Hülle $[P]$

3.2.1 Eigenschaften

Umkreise der Dreiecke sind leer
Paraboloid-Eigenschaft:
Sei $Z(x, y) = x^2 + y^2$.
Projiziert man den unteren Teil der konvexen Hülle $\{ \{ \binom{\mathbf{p}_i}{Z(\mathbf{p}_i)} \} | i = 1, \dots, n \}$ orthogonal auf die xy-Ebene, erhält man $D(P)$
 $D(P)$ kann mit *Konvexe Hülle* und mittlerem Aufwand $O(n \log n)$ berechnet werden
Kanten einer Triangulierung von Q sind konvex (Tal) oder konkav (Berg), ersetze sukzessiv in konkave durch konvexe Kanten
Winkeleneigenschaft: Der kleinste Winkel in jedem Viereck ist größer bei DT als bei jeder anderen Triangulierung
jeder Punkt \mathbf{p}_i ist mit nächstem Nachbarn durch Kante in $D(P)$ verbunden \rightarrow nächste Nachbarn aller p_i können in $O(n)$ bestimmt werden
minimale Spannbäume von P liegen auf $D(P)$ (findbar mit Kruskal (greedy))
Rundweg um minimalen Spannbaum ist 2-kompetitiv zu kürzestem Rundweg.

4 Stationäre Unterteilung für Kurven

4.1 Kardinale Splines

$N^0(u) := \begin{cases} 1, & u \in [0, 1) \\ 0, & \text{sonst} \end{cases}$
 $N^n(u) := \int_{u-1}^u N^{n-1}(t) dt$
 $N^n(u) \begin{cases} = 0, & u \notin [0, n+1) \\ > 0, & u \in (0, n+1) \end{cases}$

4.2 Symbole

Dopplung: $\alpha_0(z) = 1 + z$
Mittelung: $\mu(z) = (1 + z)/2$
Lane-Riesenfeld-Algorithmus:
 $\alpha_n(z) = \frac{(1+z)^{n+1}}{2^n}$, Differenz:
 $\beta(z) = \alpha_{n-1}(z)/2$
Chaikin: $\alpha_1(z) = \frac{1}{2}(1 + z)^2$
Unterteilungsgleichung:
 $\alpha(z)c(z^2) = b(z)$
Differenzenschema zu einem $\alpha(z)$:
 $\beta(z) = \frac{\alpha(z)}{1+z}$ (Polynomdivision).
Existiert nur wenn $\alpha(z)$ den Faktor $(1+z)$ hat, bzw. wenn $\alpha(-1) = \sum_{j \in \mathbb{Z}} \alpha_{2j} - \sum_{j \in \mathbb{Z}} \alpha_{2j+1} = 0$
Für konvergentes $\alpha(z)$ gilt $\sum_{j \in \mathbb{Z}} \alpha_{2j} = \sum_{j \in \mathbb{Z}} \alpha_{2j+1} = 1$
Ableitungsschema: $2\alpha(z)/(1 + z)$
Existiert das r-te Ableitungsschema von α und ist konvergent, konvergieren alle durch α erzeugten Folgen $(c^m)_{m \in \mathbb{N}}$ gegen r-mal stetig differenzierbare Funktionen.
Unterteilungsschema konvergent \leftrightarrow Differenzenschema Nullschema
konvergent: für jede Maske ist die Summe der Gewichte 1

5 Unterteilung für Flächen

Matrix $C = \mathbf{c}_{Z^2}$ hat das Symbol $\mathbf{c}(\mathbf{x}) := \mathbf{c}(x, y)$
 $:= \sum_{i \in \mathbb{Z}} \sum_{j \in \mathbb{Z}} \mathbf{c}_{ij} x^i y^j$
 $=: \sum_{i \in \mathbb{Z}^2} \mathbf{c}_i \mathbf{x}^i$
Seien U, V
Unterteilungsalgorithmen mit Symbol $\alpha(x), \beta(x)$
Das Unterteilte Netz $B := \mathbf{b}_{Z^2} := UC V^t$ hat das Symbol $\mathbf{b}(x, y) := \alpha(x)\mathbf{c}(x^2, y^2)\beta(y)$
 $\gamma(x, y) := \alpha(x)\beta(y)$ ist das Symbol des *Tepus* (U, V) mit der Unterteilungsgleichung $\mathbf{b}(\mathbf{x}) = \gamma(\mathbf{x})\mathbf{c}(\mathbf{x}^2) \quad \mathbf{b}_i = \sum_{\mathbf{k} \in \mathbb{Z}^2} \gamma_{i-2\mathbf{k}} \mathbf{c}_{\mathbf{k}}$
 $\mathbf{x}^2 = (x^2, y^2)!$
Verfeinerungsschema (U_1, U_1) :
 $\gamma(x, y) :=$

$$\frac{1}{4} [1 \ x \ x^2] \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \cdot [1 \ 2 \ 1] \begin{bmatrix} 1 \\ y \\ y^2 \end{bmatrix}$$

5.1 Wavelets 1D

Grundfunktionen
 $B_i^k := N_i^0(2^k u)$
Wavelets $W_i^k := B_{2i}^{k+1} - B_{2i}^{k+1}$
geg: $s(u) = \sum_{i=0}^{2^m-1} c_i^m N_i^0(2^m u)$
oder $s = \sum_{i=0}^{2^{m-1}-1} (c_i^{m-1} B_i^{m-1} + d_i^{m-1} W_i^{m-1})$
Zerlegung

- For $k = m - 1, \dots, 0$
 - For $i = 0, \dots, 2^k - 1$
 - $c_i^k = 0.5(c_{2i}^{k+1} + c_{2i+1}^{k+1})$
 - $d_i^k = 0.5(c_{2i}^{k+1} - c_{2i+1}^{k+1})$

Ausgabe: $s = c_0^0 B_0^0 + \sum_{i=0}^{2^0-1} d_i^0 W_i^0 + \dots + \sum_{i=0}^{2^{m-1}-1} d_i^{m-1} W_i^{m-1}$

Rekonstruktion

- For $k = 0 \dots m-1$
 - For $i = 0 \dots 2^k - 1$
 - $c_{2i}^{k+1} = c_i^k + d_i^k$
 - $c_{2i+1}^{k+1} = c_i^k - d_i^k$

5.2 Wavelets 2D

s(x,y) = \sum_{i,j=0}^{2^m-1} c_{ij}^m B_i^m(x) B_j^m(y)

Zerlegung^2 (Spalte erster Index!)

- Für k = m-1...0
 - Für i,j = 0...2^k - 1
 - c_{ij}^k = 0.25(c_{2i,2j}^{k+1} + c_{2i+1,2j}^{k+1} + c_{2i,2j+1}^{k+1} + c_{2i+1,2j+1}^{k+1})
 - d_{ij}^k = 0.25(+ - + -)
 - e_{ij}^k = 0.25(+ + - -)
 - f_{ij}^k = 0.25(+ - - +)

Beachte auch: in der nächsten Matrix sind die c_{ij} nur in den 4er Feldern jeweils links oben! Rekonstruktion^2 analog zu Zerlegung^2, jedoch mit Faktor 4 statt 0.25 und c, d, e, f, ergeben jeweils (2i,2j), (2i+1,2j) usw.

6 Flussmaximierung

Flussnetzwerk F := (G = (V, E), q \in V, s \in V, k : V^2 \to \mathbb{R}_{\geq 0}) Graph zusammenhängend (für jeden Knoten ex. Weg von q zu s), |E| \geq |V| - 1 Fluss f : V^2 \to \mathbb{R} mit f \leq k \forall x,y \in V : f(x,y) = -f(y,x) \forall x \in V \setminus \{q,s\} : \sum f(x,V) := \sum_{y \in V} f(x,y) = 0 Residualgraph G_f := (V, E_f := \{e \in V^2 | f(e) < k(e)\}) Residualnetz F_f := (G_f, q, s, k_f := k - f)

6.1 Methoden

6.1.1 Ford-Fulkerson (naiv)

solange es einen Weg q \rightsquigarrow s in G_f gibt, erhöhe f maximal über diesen Weg. (Nur für k \in \mathbb{Q})

6.1.2 Edmonds-Karp

=FF, erhöhen immer längs eines kürzesten Pfades in G_f. (für bel. k \in \mathbb{R})

6.1.3 Präfluss-Pusch

Präfluss-Eigenschaft Fluss mit Rein-Raus >= 0 Höhenfunktion h(q) = |V|, h(s) = 0, \forall (x,y) \in E_f : h(x) - h(y) <= 1 Push(x,y) schiebe maximal Mögliches (ü und k beachten!) über Kante Pushbar(x,y) x \in V \setminus \{q,s\} und h(x) - h(y) = 1 und \ddot{u}(x) > 0 und (x,y) \in E_f Lift(x) h(x) \leftarrow 1 + \min_{(x,y) \in E_f} h(y) Liftbar(x) x \in V \setminus \{q,s\} und \ddot{u}(x) > 0 und h(x) \leq \min_{(x,y) \in E_f} h(y)

Präfluss-Push:

- h(x) \leftarrow \text{if } x = q \text{ then } |V| \text{ else } 0
- f(x,y) \leftarrow \text{if } x = q \text{ then } k(x,y) \text{ else } 0

6.1.4 An-Die-Spitze

- Leere(x)
 - while \ddot{u}(x) > 0
 - if i_x \le Grad(x)
 - if pushbar(x, n_x(i_x)) : push(x, n_x(i_x))
 - sonst: i_x += 1
 - else
 - Lift(x), i_x \leftarrow 1

L ist Liste aller x \in V \setminus \{q,s\} mit x vor y falls pushbar(x,y) n_x(i) (1 \le i \le Grad(x)) sind Nachbarn von x (auch Gegenrichtung) i_x ist Zähler (alle n_x(i) mit i \le i_x nicht pushbar)

- An die Spitze
 - Initialisiere f und h wie bei Präfluss-Push
 - \forall x \in V : i_x \leftarrow 1
 - Generiere L
 - x \leftarrow Kopf(L)
 - while x \neq NIL
 - Leere(x)
 - Falls h_{alt} < h(x), setze x an Spitze von L
 - x \leftarrow Nachfolger von x in L

7 Zuordnungsprobleme

7.1 Paaren in allgemeinen Graphen

Alternierender Weg ist maximal, wenn er nicht Teil eines längeren alternierenden Weges ist. \to Maximale Paarung kann durch sukzessive Vergrößerung gefunden werden

7.2 Berechnung vergrößernder Wege

- Vergrößernder Weg
 - Input: G und P, Output: Vergrößernder Weg für P
 - h(x) \leftarrow 0 wenn x frei, -1 wenn x gebunden
 - Solange kein vergrößernder Pfad gefunden und gibt ununtersuchte Kante \langle x,y \rangle mit h(x) \in 2\mathbb{N}_0
 - if h(y) = -1
 - v(y) \leftarrow x, v(p(y)) \leftarrow y, h(y) \leftarrow h(x) + 1, h(p(y)) \leftarrow h(y) + 1
 - if y = v^i(x) und i \in 2\mathbb{N}_0 schrumpfe die Blüte x - v(x) - v^2(x) - \dots - y - x
 - if h(y) \in 2\mathbb{N} und w_x := v^{h(x)}(x) \neq w_y := v^{h(y)}(y), ist ein vergrößernder Pfad w_x \rightsquigarrow w_y über \langle x,y \rangle gefunden

7.3 Maximal gewichtete Paarungen

Berechnung möglich in O(|V|^3) bzw. O(|V| \cdot |E| \log |V|)

8 Minimale Schnitte

Sei

- \bar{G} := (V, \bar{E}), \bar{E} := \{ \langle x,y \rangle | \langle y,x \rangle \in \langle x,y \rangle \in E \}
- k : V^2 \to \mathbb{R}_{\geq 0}, k(x,y) := \text{if } \langle x,y \rangle \in \bar{E} \text{ then } \gamma(\langle x,y \rangle) \text{ else } 0
- x, z \in V beliebig

Berechne maximalen Fluss \to A := \{y | \exists \text{ Pfad } x \rightsquigarrow y \text{ in } \bar{G}_f\} und B := V \setminus A bilden minimalen xz-Schnitt (x \in A, z \in B) Gewicht des Schnitts = Wert des Flusses kleinsten xz-Schnitt in G lässt sich mit Flussmaximierung in O(|V|^4) berechnen (es existieren Algorithmen in O(|V|^2 \log |V| + |V||E|))

8.1 Zufällige Kontraktion

ggf. todo Monte-Carlo-Algorithmus = stochastischer Algorithmus, kann falsche Ergebnisse liefern Las-Vegas-Algorithmus = stoch. Algo., immer richtig

IV Optimierungsalgorithmen

9 Kleinste Kugeln

Für jede Punktmenge P ist die kleinste Kugel K(P) \supset P eindeutig.

9.1 Algorithmus von Welzl

- K(P, R) ist Kugel die P enthält und R auf der Oberfläche hat
- Welzl
 - Input: P, R \subset \mathbb{R}^d, K(P, R) exist., P,R endlich
 - if P = \emptyset \text{ or } |R| = d + 1
 - C \leftarrow K(R)
 - else wähle p \in P zufällig
 - C \leftarrow Welzl(P \setminus \{p\}, R)
 - if p \notin C
 - C \leftarrow Welzl(P \setminus \{p\}, R \cup \{p\})
 - Gib C aus

10 Lineare

Programmierung

10.1 Lineare Programme

LP ist z(x) := zx = max!, Ax \ge a, wobei z, x \in \mathbb{R}^d, A \in \mathbb{R}^{n \times d}, a \in \mathbb{R}^n, und zx := z^t x d ist die Dimension des linearen Programms. Die Ungleichungen Ax \ge a repräsentieren den Schnitt S von n Halbräumen, der Simplex genannt wird. Die Punkte x \in S heißen zulässig. Die Ecken von S liegen je auf d Hyperebenen (d Gleichungen des Gleichungssystems).

- Simplexalgorithmus: Iterativ Ecken entlang gehen, bis z maximal.

10.2 Flussmaximierung als LP

maximiere Summe der ausgehenden Flüsse aus der Quelle. Gleichungen zur Flusserhaltung (je eingehende Kanten - ausgehende Kanten = 0 (\ge \text{ und } \le)) Gleichungen zur Kapazitätsbeschränkung (Fluss \ge 0 und (Kapazität - Fluss) \ge 0) f(a,b) = -f(b,a)

10.3 Kürzester Weg als LP

Suche Weg 1 \rightsquigarrow 2 \sum_{(i,j) \in E} x_{ij} \gamma_{ij} = \min! x_{ij} \ge 0, (i,j) \in E

\sum_j x_{ij} - \sum_j x_{ji} = \begin{cases} 1 & i = 1 \\ -1 & i = 2 \\ 0 & \text{sonst} \end{cases} (Ausgehende Kanten = Eingehende Kanten außer für i \neq 1, 2) negative Kreise \Rightarrow keine endliche Lösung. Erzwingbar durch x_{ij} \le 1, (i,j) \in E (?)

10.4 Maximusnorm

geg: r = A * a - c mit A Matrix wobei c konstanter Vektor und A Vektor aus Variablen. Dann LP mit y_0 = 1/r, y_1 = a_1/r, y_2 = a_2/r, ... y_0 = max! -c A c -A <= [1, 1, ..., 1]

10.5 Simplexalgorithmus

y(x) = Ax

\begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \dots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}

wobei n = d + 1 und x_n = 1

Hyperebenen H_i : y_i(x) = 0 Gegeben: A = [a_{ij}]_{i,j=1,1}^{m,n} Gesucht: B = [b_{ij}]_{i,j=1,1}^{m,n} r=Pivotzeile, s=Pivotspalte

Austausch

- b_{rs} \leftarrow \frac{1}{a_{rs}}
- b_{rj} \leftarrow -\frac{a_{rj}}{a_{rs}} (Pivotzeile, j \neq s)
- b_{is} \leftarrow \frac{a_{is}}{a_{rs}} (Pivotspalte, i \neq r)
- b_{ij} \leftarrow a_{ij} - \frac{a_{is} a_{rj}}{a_{rs}} (i \neq r, j \neq s)

10.6 Normalform

Jedes lin. Programm kann auf die Form zx = max! Ax \ge 0 mit x = [x_1 ... x_d 1]^t kann auf die Form [c^t c]y = max! y \ge 0 [Bb]y \ge 0 mit y := [y_1 ... y_d 1]^t gebracht werden. Notation:

y_{d+1} = \begin{matrix} x_0 \dots d & 1 \\ \vdots & \\ B & \mathbf{b} \\ \hline \mathbf{c}^t & c \end{matrix} \ge 0 = \max!

b \ge 0, sonst Simplex leer.

10.7 Simplexalgorithmus

- Simplex
 - Input: A Normalformmatrix eines lin. Progr. \bar{A} := \begin{bmatrix} A & \mathbf{a} \\ \mathbf{c}^t & c \end{bmatrix}
 - Solange ein c_s > 0
 - Falls alle a_{is} \ge 0: gib c \leftarrow \infty aus; Ende
 - sonst
 - bestimme r so, dass \frac{a_r}{a_{rs}} = \max_{a_{is} < 0} \frac{a_i}{a_{is}}
 - \bar{A} \leftarrow \text{Austausch}(\bar{A}, r, s)
 - Gib \bar{A} aus

Die Lösung ist dann, dass alle y_i die oben an der Tabelle stehen = 0 sind. (total) unimodular = quadratisch, det A \in \{-1, 0, 1\} (+alle quad. Untermatrizen) Util a \cdot b = |a||b| \cos \sphericalangle(a,b) \sum_{k=0}^n 2^k = 2^{n+1} - 1 Laufzeiten Kap.Name | Laufzeit

2.6	Konvexe Hülle	erw: O(n log n), max: O(n^2)
6	Ford-Fulkerson	O(E * W) (k Wert eines max. Flusses)
6	Edmonds-Karp	O(E ^2 * V)
6	Präfluss-Push	O(V ^2 * E)
6	An-Die-Spitze	O(V ^3)
7	Paare	O(E \cdot \min\{ L , R \})
7	Vergrößernder Weg	O(V \cdot E)
8.2	Schnitt	O(V ^2) gef. mit P = 1 - 1/e^2
8.3	Min Schnitt	O(V ^2 log V) richtig mit P \in \Theta(1/\log V)
9	Welzl	mittl: O(n)
10	Simplex	erw: O(n^2 d), max: \Omega(n^{d/2})
10	Ellipsoid	polyn.; in praxis langsamer als Simplex
10	Innere Punkte	polyn.; in praxis fast so gut wie Simplex
10,5	Seidel	O(d^3 d! + d n d!)