

# Graphentheorie

Graph:  $G = (V, E)$  Baum:  $|E| = |V| - 1$

**Spannbaum** von  $G$ : Teilgraph von  $G$  und Baum der alle Knoten von  $G$  enthält, not unique.

**Bipartit**:  $G = (V_1 \cup V_2, E)$ ,  $G$  ist bipartit  $\Leftrightarrow G$  enthält keinen Kreis ungerader Länge

**Isomorph**:  $G$  ist isomorph  $\rightarrow$  Gradfolge und  $|V|$  sowie  $|E|$  sind identisch

**Paarung**  $P$  von  $G$ :  $P$  hat keine gemeinsamen Endpunkte. —  $P \subset E$  es gibt nur Paare

**Knotenüberdeckung**  $U$  von  $G$ :  $\forall$  Kanten  $uv$  gilt  $u \in U$  OR  $v \in U$ . —  $U \subset V$  s.d. jede Kante hat ein Ende in  $U$

**König-Egervary**:  $G$  bipartit  $\Rightarrow$  |maximale Paarung| == |minimale Knotenüberdeckung|

**Flussnetze**:  $N = (D, \kappa, s, q)$ ,  $D$  Digraph,  $\kappa: E \rightarrow \mathbb{R}_0^+$  Kostenfunktion

**Schnitt eines Flussnetzes**: Teilmenge  $S$ , die die Quelle aber nicht die Senke enthält.

**Kapazität** eines Schnittes:  $\kappa(S)$  = Kapazität der Endknoten des Schnittes. **Minimaler Schnitt**  $S = \forall S' \kappa(S) \leq \kappa(S')$   
maximaler Fluss == min Schnitt

## Planarität

für ebene Darstellungen gelten:  $n - m + f = 2$ ,  $n$ =Knoten,  $m$ =Kanten,  $f$ =Flächen

if  $n \geq 3$   $3n - 6$  Kanten höchstens

if  $n \geq 3$  und  $g \geq 3$  höchstens  $\max\{g(n-2)/(g-2), n-1\}$  Kanten ( $g$  = Länge eines kürzesten Kreises)

ein Graph ist planar  $\Leftrightarrow$  kein Subgraph von  $G$  ist homöomorph zu  $K_5, K_{3,3}$

## Datenstrukturen

### Adjazenzmatrix

$n \cdot n$ , immer symmetrisch

$a_{ij} = 1$  falls  $v_i v_j \in E$ , 0 sonst

### Inzidenzmatrix

$n \cdot m$ ,  $e_{ij} = 1$  wenn  $v_i$  mit  $e_j$  inzidiert, 0 sonst

Spaltensumme = 2, Zeilensumme = Grad des Knoten

## Netzwerke

Floyd-Warshall (S.288)

Kürzeste Abstände für alle Knoten  $O(|V|^3)$

for  $k=1$  to  $n$  do:  $d(u, w) = \min(d^{k-1}(u, w), d^{k-1}(u, v_k) + d^{k-1}(v_k, w))$

Mit jeder Iteration gucken ob es einen kürzeren Weg über den Knoten  $v_k$  gibt

Dijkstra (S.289)

Kürzeste Wege für einzelnen Knoten  $O(|V|^2)$  oder  $O(|E| + |V| \log |V|)$

Nachbarkanten untersuchen nach kürzeren Wegen

Kruskal (S.291)

min. Spannbäume

Durchlaufe Kanten nach wachsendem Gewicht, füge hinzu, wenn Komponente noch nicht im Spannbaum

Ford-Fulkerson (S.293)

bestimmt maximalen Fluss in  $N$

erst alle Knoten markieren, dann Fluss vergrößern und erneut markieren.

## Optimierung

Entscheidungsprobleme: NP-Vollständig

Optimierungsprobleme: NP-Hart

Eine Maximierung von  $f$  entspricht einer Minimierung von  $-f$  [ $\max\{f(x) | x \in X\} == -\min\{-f(x) | x \in X\}$ ]

## Backtracking – Kombinatorische Optimierung (S. 307)

Exhaustives durchsuchen des gesamten Suchraumes.

Abschneiden von Teilbäumen durch Bonding-Funktionen: (S. 310)

$x = (x_1, \dots, x_k)$  Teillösung und  $P(x)$  der zugehörige Maximalwert aller Lösungen von  $x$

Bonding Funktion  $B(x)$  s.d.  $\forall x B(x) \geq P(x)$ . So kann abgeschnitten werden wenn  $B(x) \leq$  dem aktuellen Höchstwert

## Heuristiken (S. 313)

Ordnet jeder Lösung eine Nachbarschaft von anderen Lösungen zu.  
Nachbarschaftsfunktion  $N(x)$

### Bergauf-Methode

Als Nachbarschaft von  $x$  wird ein Wert  $y$  mit  $f(y) > f(x)$  gesucht.  
Sobald  $f(y)$  nicht mehr grösser wird aufgehört zu suchen.

## Simulated Annealing (S. 316)

Falls  $f(y) < f(x)$  benutzt  $random \in [0, 1] < e^{\frac{f(y)-f(x)}{T}}$  um zu entscheiden ob  $x$  (doch) durch  $y$  ersetzt wird.  
Abkühlungsplan  $T$ .  $T_0$  wird hoch gewählt und nach jeder Iteration um einen Prozentsatz gesenkt bis Endtemperatur  $T_f$  erreicht ist  
Dies soll das verlassen von lokalen Optima am Anfang des Prozesses ermöglichen.

## Genetische Algorithmen

Initialisiert Population  $P$  mit  $N$  Individuen  
Iteration: Selektion, Mutation, Kreuzung

Selektion: Wähle die besten  $N$  Individuen aus  $P$   
Mutation: Ersetze Individuen durch Benachbarte  
Kreuzung: Kreuze Paare aus der Population

## Lineare Programmierung S. 327

### LP und Dualität S.328

(primales) LP  
$$\max c^T x$$
$$\text{s.d. } Ax \leq b$$
$$x \geq 0$$

duales LP  
$$\min b^T y$$
$$\text{s.d. } A^T y \geq c$$
$$y \geq 0$$

### ILP S.336

Total Unimoular, wenn die Determinanten aller quadratischen Untermatrizen  $+1$ ,  $0$  oder  $-1$  ist.

Inzidenzmatrix von  $G$  ist total Unimoular wenn  $G$  bipartit.  
Inzidenzmatrix von  $G$  ist total Unimoular wenn  $G$  digraph.

Wenn  $A$  total Unimoular und  $b \in \mathbb{Z}^m$ , dann sind alle Ecken ganzzahlig.

### Graphische Lösungen

$\alpha x_1 + \beta x_2 = c$   
intersection form:  $\frac{x}{a} + \frac{y}{b} = 1$   
 $m = -\frac{\alpha}{\beta}$ ;  $b = \frac{c}{\beta}$

## Bergauf

```

wähle zulässige Lösung  $x \in X$  //Startpunkt
 $x^* \leftarrow x$  //beste Lösung
 $searching \leftarrow \mathbf{true}$ 
while ( $searching$ ) {
     $y \leftarrow H(x)$ 
    if ( $y == fail$ )
         $searching \leftarrow \mathbf{false}$ 
    else {
         $x \leftarrow y$ 
        if ( $f(x) > f(x^*)$ )
             $x^* \leftarrow x$ 
    }
}

wähle  $H(x)$  Problemspezifisch

```

## Simulated Annealing

```

 $T \leftarrow T_0$ 
wähle zulässige Lösung  $x \in X$  //Startpunkt
 $x^* \leftarrow x$  //beste Lösung
while ( $T \geq T_f$ ) {
     $y \leftarrow H(x)$ 
    if ( $y == fail$ )
        return  $x^*$ 
    if ( $f(y) > f(x)$ ) {
         $x \leftarrow y$  //Aufwärtsbewegung
        if ( $f(x) > f(x^*)$ )
             $x^* \leftarrow x$ 
    } else {
         $r \leftarrow \mathbf{random}(0, 1)$  //Abwärtsbewegung
        if ( $r < e^{\frac{f(y)-f(x)}{T}}$ )
             $x \leftarrow y$ 
         $T \leftarrow \alpha \cdot T$  //  $\alpha = .99$ 
    }
}

```

## Backtracking

```

function_name(...,  $k$ ) {
    if ( $k = 0$ )
        setup...
    if ( $k = n$ ) {
        if ( check_valid( $x_1, \dots, x_k$ ) and  $k > optSize$ )
             $optSize \leftarrow k$ 
    } else {
         $x_k \leftarrow 0$ 
        function_name ( $x_1, \dots, x_k, k + 1$ )
         $x_k \leftarrow 1$ 
        [if (bonding ( $x_1, \dots, x_k$ )) ]
        function_name ( $x_1, \dots, x_k, k + 1$ )
    }
}

main() {
     $optSize \leftarrow 0$ 
     $n \leftarrow ?$ 
    function_name(0)
    print problem
}

Backtracking variations:
use sets instead of bitvector
bonding function also applies to case  $x_k = 0$ 

```