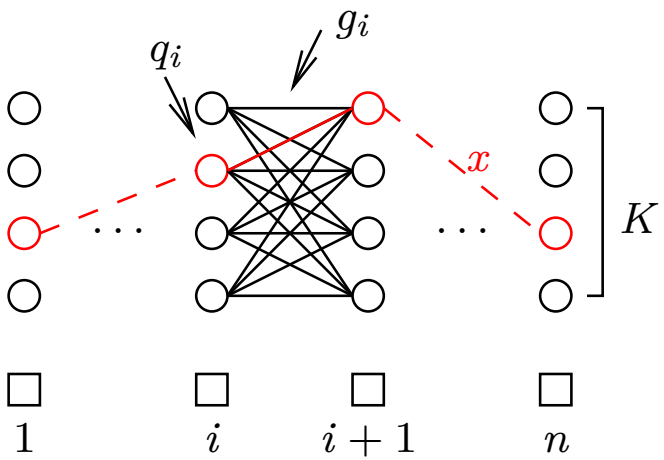
Mustererkennung: Dynamische Optimierung

D. Schlesinger – TUD/INF/KI/IS

Dynamische Optimierung für Ketten (Aufgabe)

Gegeben:

- Die **Menge der Knoten** R – nummeriert mit $i = 1 \dots n$
- Eine endliche **Zustandsmenge** K, Elemente sind $k \in K$
- **Knotenfunktionen** $q_i : K \to \mathbb{R}$ für alle Knoten,
  $q_i(k)$ bedeutet: „die Bewertung des k-ten Zustands im i-ten Knoten“.
- **Kantenfunktionen** $g_i : K \times K \to \mathbb{R}$ für alle Paare benachbarter Knoten,
  $g_i(k, k')$ bedeutet: „die Bewertung des Zustandspaares (k, k') auf der Kante $(i, i+1)$“.
- Eine **Zustandsfolge** x ist eine Abbildung $x : R \to K$, die jedem Knoten einen Zustand zuordnet,
  x_i bedeutet: „der durch die Abbildung x im Knoten i ausgewählte Zustand“,
  $q_i(x_i)$ bedeutet: „die Bewertung des ausgewählten Zustands im Knoten i“



Gesucht:

$$x^* = \arg\min_x \left[\sum_{i=1}^{n} q_i(x_i) + \sum_{i=1}^{n-1} g_i(x_i, x_{i+1}) \right]$$

Dynamische Optimierung für Ketten (Ableitung)

$$\min_x \left[\sum_{i=1}^{n} q_i(x_i) + \sum_{i=1}^{n-1} g_i(x_i, x_{i+1}) \right] =$$

$$\min_{x_1} \min_{x_2} \dots \min_{x_n} \left[\sum_{i=1}^{n} q_i(x_i) + \sum_{i=1}^{n-1} g_i(x_i, x_{i+1}) \right] =$$

$$\min_{x_2} \dots \min_{x_n} \left[\sum_{i=2}^{n} q_i(x_i) + \sum_{i=2}^{n-1} g_i(x_i, x_{i+1}) + \min_{x_1} \Big(q_1(x_1) + g_1(x_1, x_2) \Big) \right] =$$

$$\min_{x_2} \dots \min_{x_n} \left[\sum_{i=2}^{n} q_i(x_i) + \sum_{i=2}^{n-1} g_i(x_i, x_{i+1}) + F(x_2) \right] =$$

$$\min_{x_2} \dots \min_{x_n} \left[\sum_{i=2}^{n} \tilde{q}_i(x_i) + \sum_{i=2}^{n-1} g_i(x_i, x_{i+1}) \right]$$

mit $\tilde{q}_2(k) = q_2(k) + F(k)$, sonst unverändert, d.h. $\tilde{q}_i(k) = q_i(k)$ für $i = 3 \dots n$.

Funktionen F nennt man Bellmansche Funktionen, sie repräsentieren die Kosten der besten Fortsetzungen auf den bereits bearbeiteten Teil des Problems.

Dynamische Optimierung für Ketten (Algorithmus)

```
// Forward pass
for i = 2 bis n
    for k = 1 bis K
        best = ∞
        for k' = 1 bis K
            if q_{i-1}(k') + g_{i-1}(k', k) < best
                best = q_{i-1}(k') + g_{i-1}(k', k),  pointer_i(k) = k'
        q_i(k) = q_i(k) + best

// Backward pass
best = ∞
for k = 1 bis K
    if q_n(k) < best
        best = q_n(k),  x_n = k
for i = n − 1 bis 1
    x_i = pointer_{i+1}(x_{i+1})
```

$pointer_i(k)$ ist der beste Vorgänger für den Zustand k im i-ten Knoten.

Dynamische Optimierung für Ketten (Parallelisierung)

Zeitkomplexität: $O(nK^2)$

K Prozessoren:
die Schleife über k kann parallelisiert werden $\rightarrow O(nK)$

Weitere Möglichkeit – mittlere Knoten Eliminieren.

$$\min_{x_1}\min_{x_2}\min_{x_3}\Big[q_1(x_1)+g_1(x_1,x_2)+q_2(x_2)+g_2(x_2,x_3)+q_3(x_3)\Big]=$$

$$\min_{x_1}\min_{x_3}\Big[q_1(x_1)+q_3(x_3)+\min_{x_2}\Big(g_1(x_1,x_2)+q_2(x_2)+g_2(x_2,x_3)\Big)\Big]=$$

$$\min_{x_1}\min_{x_3}\Big[q_1(x_1)+q_3(x_3)+g(x_1,x_3)\Big]$$

$n/2$ Prozessoren:
die Eliminierungen können (fast) parallel ausgeführt werden $\rightarrow O(\log n \cdot K^3)$

$n/2 \cdot K^2$ Prozessoren $\rightarrow O(\log n \cdot K)$

Dynamische Optimierung für allgemeine Graphen

Man betrachte den folgenden Prozess der Erzeugung eines Graphen:

Die Knoten werden nach und nach in den Graphen eingefügt (ein Knoten am Anfang).

Der neu eingefügte Knoten wird mit einem vollverbundenen Teilgraphen durch die Kanten verbunden. Dieser vollverbundene Teilgraph besteht aus maximal w Knoten.

Nachdem alle Knoten eingefügt sind, werden manche Kanten entfernt.

Gegeben sei ein Graph. Seine **Breite** (treewidth) ist die kleinste Zahl w so, dass der Graph durch den wie oben beschriebenen Prozess erzeugt werden kann – partieller w-Baum.

Beispiele:
Ketten, Bäume: $w = 1$
Zyklen, Simple Netze: $w = 2$
Gitter: $n \times m$: $w = \min(n, m)$

Bei einem fixierten w kann in polynomieller Zeit beantwortet werden, ob ein gegebener Graph die Breite w hat – polynomiell in n, allerdings exponentiell in w
$\rightarrow$ die Aufgabe der Bestimmung von w ist NP-vollständig.

Dynamische Optimierung für allgemeine Graphen

Die Idee der Dynamischen Optimierung: wenn die Reihenfolge der Knoten Bekannt ist, kann man die Knoten in der umgekehrten Reihenfolge eliminieren. Die Bellmansche Funktionen haben dabei die Ordnung maximal w, d.h. $F : K^w \to \mathbb{R}$.

Die Dynamische Optimierung hat die Zeitkomplexität $O(nK^{w+1})$

Beispiele:
Kette: eliminiert wird der „erste“ Knoten, $w = 1$, $F : K \to \mathbb{R}$, $O(nK^2)$
Baum: eliminiert wird immer ein Blatt, alles andere – dasselbe
Zyklus: eliminiert wird ein beliebiger Knoten, $w = 2$, $F : K \times K \to \mathbb{R}$, $O(nK^3)$