# Hľadanie najkratšieho spoločného nadreťazca použitím neurónových sietí

G.Andrejková
Ústav informatiky
Prírodovedecká fakulta UPJŠ

## Obsah

- Problém najkratšieho spoločného nadreťazca (Shortest Common SuperString) – SCSS Problém
- Motivácia, príklady použitia
- Zaradenie problému do triedy APX
- Hopfieldove neurónové siete (HNN)
- HNN pri riešení problému najkratšieho spoločného nadreťazca

## Motivácia, použitie

- v dátovej kompresii a
- pri analýze DNA molekúl
- DNA je reprezentovaná ako reťazec nad množinou nukleotidov {A, C, G, T} - približne $3 * 10^9$ symbolov pre človeka
- aktuálne laboratórne metódy - utriedenie iba malých fragmentov s najviac približne 500 bázami
- z fragmentov sa budujú nadreťazce DNA molekúl



## SCSS Problém

- $S = \{s_1,\ldots,s_n\}$ – množina reťazcov
- $s_{ij}$ – dĺžka prekrytia $s_i$ a $s_j$, pričom $s_{ii} = |s_i|$
- nájsť permutáciu reťazcov $\pi$ takú, že $|S_\pi|$ je minimálna
- $|S_\pi| = \sum_{i=1}^{n} |s_i| - \sum_{i=1}^{n-1} s_{\pi(i),\pi(i+1)}$

# Príklad

- S = {napad, batoh, ohnostroj, ostroha, trojuholnik, koliba, rozkol, volant}
- |S| = 5+5+9+7+11+6+6+6=26+29=55

Optimálny nadreťazec (superstring):

napadostro|harozkolib|atohnostro|juholnikvo|lant

Dĺžka je 44



# Matica dĺžok prekrytí v danom príklade

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| napad | 5 | | | | | | | |
| batoh | | 5 | 2 | | | | | |
| ohnostroj | | | 9 | | 4 | | | |
| ostroha | | | | 5 | | | | |
| trojuholnik | | | | | 11 | 1 | | |
| koliba | | 2 | | | | 6 | | |
| rozkol | | | | | | 3 | 6 | |
| volant | | | | | 1 | | | 6 |

# Greedy algoritmus

- V množine reťazcov vyberie dvojicu rôznych reťazcov s najvyšším prekrytím, tú spojí a nový reťazec vloží naspäť do množiny.
- Vstup: množina S obsahujúca n reťazcov
- while |S|>1
  - vyber *a, b* $\in$ S, a$\neq$b s maximálnym prekrytím; nech *c* je čiastočný superstring, ktorý vznikne zlepením *a* a sufixu *b*
  - potom S = (S\{a, b}) $\cup$ {c}
- Výstup: Posledný reťazec v S



Príklad: S = {napad, batoh, ohnostroj, ostroha, trojuholnik, koliba, rozkol, volant}

- $S_1$={ ohnostrojuholnik, napad, batoh, ostroha, koliba, rozkol, volant}
- $S_2$={ohnostrojuholnik, napad, batoh, ostroha, rozkoliba, volant}
- $S_3$={ rozkolibatoh, ohnostrojuholnik, napad, ostroha, volant }
- $S_5$={ rozkolibatohnostrojuholnik, napad, ostroha, volant }
- $S_6$ = {rozkolibatohnostrojuholniknapadostrohavolant}, $|S_6|$=44

## Príklad:

- $S = \{x(ab)^k, (ab)^k y, (ba)^k\}$
- V tomto prípade je ľahko vidieť, že najkratší nadreťazec je $x(ab)^{k+1}y$ s dĺžkou 2k+4.
- Greedy algoritmus nám dá ako výsledok reťazec $x(ab)^k y(ba)^k$ s dĺžkou 4k+2.



## TGreedy algoritmus

- Vstup: množina S obsahujúca n reťazcov
- while |S|>1
  - vyber $a, b \in S$ s nenulovým prekrytím
    - ak $a \neq b$
      - nech *c* je superstring, ktorý vznikne zlepením *a* a sufixu *b*
      - potom $S = (S \setminus \{a, b\}) \cup \{c\}$
    - ak $a = b$
      - $S = S \setminus \{a\}$
      - $T = T \cup \{a\}$
- Na množinu T uplatniť Greedy algoritmus
- Výstup: Posledný reťazec v S

## TGreedy algoritmus - príklad

- $S=\{s_1, s_2, s_3, s_4\}$
- $s_1$ = abba, $s_2$ = banabana, $s_3$ = our, $s_4$ = urban
- $s_5$ = spojenie($s_2$,$s_2$) = banabana $\Rightarrow$
  $S_1 = \{s_1, s_3, s_4\}$, $T_1=\{s_5\}$
- $s_6$ = spojenie($s_3$,$s_4$) = ourban $\Rightarrow$
  $S_2 = \{s_1, s_6\}$, $T_2=\{s_5\}$
- $s_7$ = spojenie($s_1$,$s_1$) = abba $\Rightarrow$
  $S_3 = \{s_6\}$, $T_3=\{s_5, s_7\}$
- $s_8$ = spojenie($s_6$,$s_6$) = ourban $\Rightarrow$
  $S_4 = \varnothing$, $T_4=\{s_5, s_7, s_8\}$={banabana, abba, ourban}
- teraz aplikujeme metódu greedy a dostaneme ourbanabanabba



## Kruhové pokrytie (Cycle cover)

- Uvažujme orientovaný graf G.
- Kruhové pokrytie definujeme ako množinu cyklov v grafe G a to takých, že každý vrchol sa nachádza práve v jednom cykle.
- Cena (váha) kruhového pokrytia je suma všetkých váh obsiahnutých v cykloch.
- Minimálne kruhové pokrytie je kruhové pokrytie s najmenšou cenou, teda je to najmenšia suma všetkých váh obsiahnutých v cykloch.

# Kruhové pokrytie (Cycle cover)

- Ako vytvoriť vhodný graf k množine reťazcov?

- Definujme $d(S_i, S_j)$ ako vzdialenosť reťazca $S_j$ od reťazca $S_i$, t. j. pre reťazce

  $S_3$ = your, $S_4$ = urban platí

  $d(S_3, S_4)= 3$, $ov(S_3, S_4)= 2$

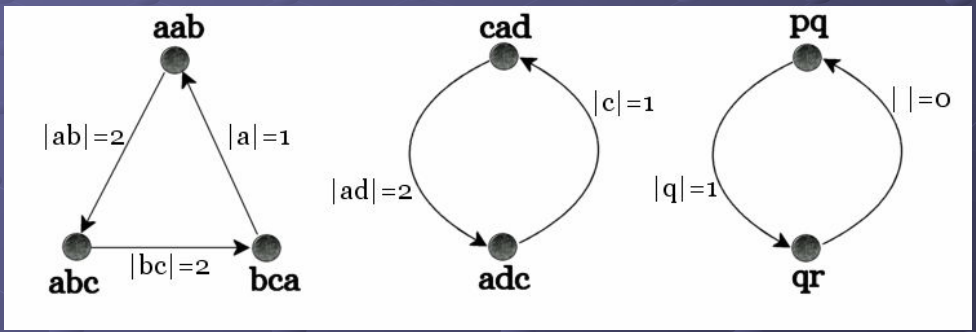
  $d(S_4, S_3)= 4$, $ov(S_4, S_3)= 0$



# Kruhové pokrytie (Cycle cover)

S={aab, abc, bca, adc, cad, pq, qr}

Kruhové pokrytie:




Dĺžka pokrytia je 3+3+3-2-2 + 3+3-2 +2+2-1= 5+4+3=12

# Kruhové pokrytie (Cycle cover)

- Nech σ je poradie vrcholov v kruhovom pokrytí, počet reťazcov je n
- Dĺžka kruhového pokrytia v prípade reťazcov je

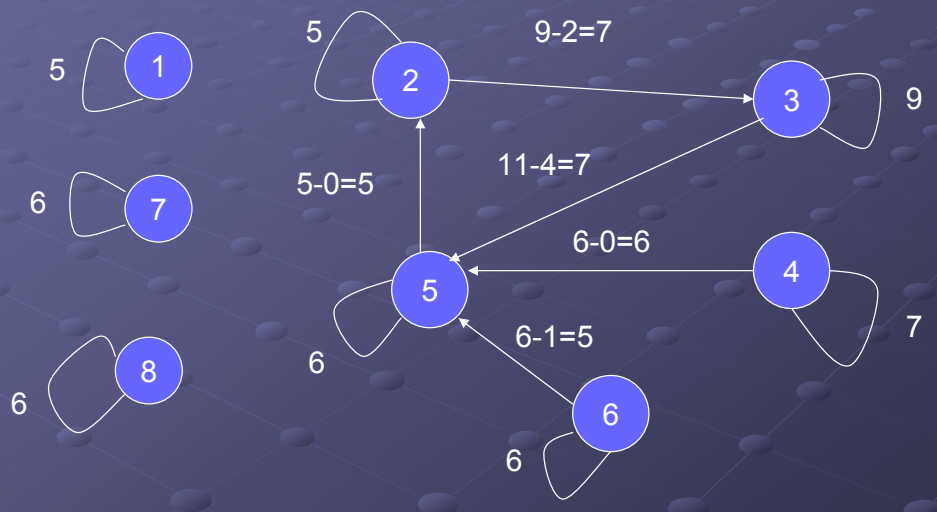
$$|C_\sigma| = \sum_{i=1}^{n} |s_i| - \sum_{i=1}^{n-1} |s_{\sigma(i)}, s_{\sigma(i+1)}| - |s_{\sigma(n)}, s_{\sigma(1)}|$$

Dĺžka nadreťazca pre reťazce v kruhu v tomto prípade je

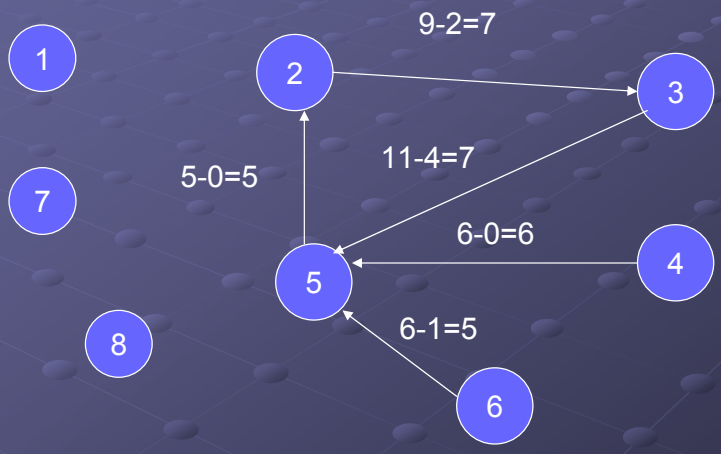
$$|C_\sigma| = \sum_{i=1}^{n} |s_i| - \sum_{i=1}^{n-1} |s_{\sigma(i)}, s_{\sigma(i+1)}|$$



# Pomocný graf pre S = {napad, batoh, ohnostroj, ostroha, trojuholnik, koliba, rozkol, volant}

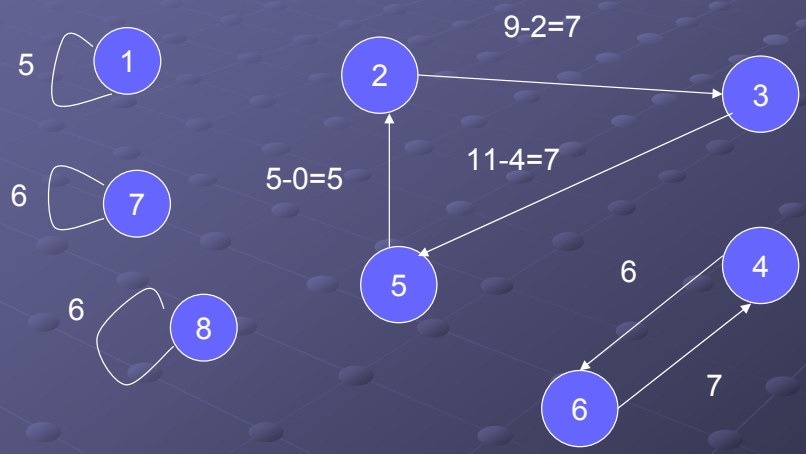


## Graf vzdialeností pre S = {napad, batoh, ohnostroj, ostroha, trojuholnik, koliba, rozkol, volant}






## Kruhové pokrytie pre S = {napad, batoh, ohnostroj, ostroha, trojuholnik, koliba, rozkol, volant}




Nech cyklus je v poradí 2-3-5

Dĺžka kruhového pokrytia je 5+6+6+ 6+7 + 7+7+5= 30 + 19= 49

## Pozorovania

- Váha minimálneho pokrytia cyklami je najviac rovná dĺžke najkratšieho nadreťazca.
- Ak máme dva cykly C a C' v minimálnom pokrytí pomocou cyklov v grafe $G_s$ k množine reťazcov S, ak s a s' sú reťazce z C a C', tak platí

$$ov(s,s') \leq w(C) + w(C').$$



## MGreedy algoritmus – 4 aproximačný

- Vstup: S - množina n reťazcov
- Výstup: Superstring pre S
- Algoritmus:

1. Vytvoriť graf vzdialeností $G_s$ k množine S
2. Nájsť minimálne kruhové pokrytie $C=\{C_1,...,C_r\}$ pre graf $G_s$
3. Nech $\{S_{i1},...,S_{in(i)}\}$ sú reťazce cyklu $C_i$. Nech SUSi je superstring získaný z $S_{ij}$, j=1, 2, …n(i) prekrývaním v tom poradí ako sú v cykle.
4. Pospájaj všetky SUSi dokopy.

# Analýza dĺžky nadreťazca

- Nech $\{S_{i1},...,S_{in(i)}\}$ sú reťazce cyklu $C_i$.
- Každý SUSi je superstringom pre $\{S_{i1},...,S_{in(i)}\}$.
- # je superstring pre všetky SUSi. Keďže $S_{ij}$ sú reťazce z S tak # je superstringom pre reťazce v množine S.
- Nech OPT(S)=n
- Nech najdlhší reťazec každom cykle $C_i$ je $l_i$, jeho dĺžka je $|l_i|$
- Teda

$$|SUSi| = d(i1,i2)+...+d(in(i)-1,in(i))+|s_{in(i)}|$$



# Analýza dĺžky nadreťazca

- $|SUSi| = d(i1,i2)+...+d(in(i)-1,in(i))+|s_{in(i)}|$
- $\leq w(C_i)+|s_{in(i)}|$
- $\leq w(C_i)+|l_i|$

Teda

$$|\#| \leq \sum_{i=1}^{r} ( |l_i| + w(C_i) )$$

Tiež platí

$$n \geq <l_1, \ldots, l_r> = d(l_1,l_2) + ... + d(l_{r-1},l_r) + |l_r|$$

## Analýza dĺžky nadreťazca

- $n \geq d(l_1,l_2) + \ldots + d(l_{r-1},l_r) + |l_r|$
- $\geq d(l_1,l_2) + \ldots + d(l_{r-1},lr) + d(l_r,l_1)$
- $= (|l_1|-ov(l_1,l_2)) + \ldots + (|l_r|-ov(l_r,l_1))$
- $= \sum |l_i| - \sum ov(l_i,l_{i+1})$
- $\geq \sum |l_i| - \sum [(w(C_i) + w(C_{i+1})]$
- $= \sum |l_i| - 2*\sum w(C_i)$

Taktiež platí, že $CYC(G_s) \leq OPT(S)=n$.



## Analýza dĺžky nadreťazca

- $\sum w(C_i) \leq n$

- $|\#| \leq \sum (|l_i| + w(C_i))$
- $= 3 \sum w(C_i) + \sum |l_i| - 2 \sum w(C_i)$
- $\leq 3n + n = 4n$

Tým sme ukázali, že aproximujúci superstring má dĺžku maximálne 4x tak veľkú ako optimálny.

## Vylepšenie algoritmu

- Cykly v grafe pomáhajú rozdeliť množinu reťazcov do skupín
- Ak aplikujeme Greedy algoritmus na reťazce v cykloch, tak dostaneme vylepšenie, namiesto 4 bude odhadnutá dĺžka 3.



## Test č. 1: S = {aristoteles, prometeus, usa, kopaniciar, teleskop, sadra}.

- MGreedy aristoteleskopaniciarprometeusadra 34
- TGreedy prometeusadraristoteleskopaniciar 33
- Greedy prometeusadraristoteleskopaniciar 33
- Optimálny prometeusadraristoteleskopaniciar
- 33

## NP optimalizačný problém *A* je štvorica *(I, sol, m, goal)*

- I je množina inštancií problému *A* a je rozpoznateľná v polynomiálnom čase.
- K danej inštancii *x* z *I* nech *sol(x)* označuje množinu všetkých dosiahnutých riešení pre *x*. Tieto riešenia sú krátke, teda existuje polynóm *p* taký, že pre ľubovoľné *y* zo *sol(x)* platí $|y|<=p(|x|)$.

  Navyše, je v polynomiálnom čase rozhodnuteľné, či pre ľubovoľné *y* také, že $|y|<=p(|x|)$ platí *y* je v *sol(x)*.



## NP optimalizačný problém *A* je štvorica *(I, sol, m, goal)*

- K danej inštancii *x* a dosiahnutému riešeniu *y* nech *m(x, y)* označuje kladnú, celočíselnú mieru pre *y* (tiež sa často nazýva hodnotou *y*).

  Funkcia *m* je vypočítateľná v polynomiálnom čase a tiež sa nazýva objektívna funkcia.
- *goal* patrí do *{min, max}*.

## Zaradenie problému do triedy APX

- Množina všetkých NP optimalizačných problémov vytvára triedu **NPO**.
- *Cieľom* v NPO probléme je vzhľadom na inštanciu *x* nájsť optimálne riešenie, teda dosiahnuté riešenie *y* je také, že

  $m(x, y) = goal\{m(x, y'): y' \text{ patrí do } sol(x)\}$.



## Polynomiálne ohraničený problém

- NPO problém sa nazýva polynomiálne ohraničený, ak existuje polynóm ***q*** taký, že pre ľubovoľnú inštanciu *x* a pre ľubovoľné riešenie *y* pre *x*, platí $m(x,y) \leq q(|x|)$.
- Trieda NPO PB je trieda všetkých polynomiálne ohraničených problémov.

## Výkonnostný pomer riešenia y vzhľadom k inštancii x

- Nech A je NPO problém. K danej inštancii x a dosiahnutému riešeniu y pre x definujeme výkonnostný pomer pre y vzhľadom k x takto:

$$R(x, y)= \max \{m(x,y)/opt(x), opt(x)/m(x,y)\}$$

- Výkonnostný pomer je vždy číslo väčšie ako 1 a je veľmi blízke 1, keď dosiahnuté riešenie je blízke optimálnemu riešeniu.



## r(n) - aproximujúci algoritmus pre A

- Nech A je NPO problém a nech T je algoritmus, ktorý pre inštanciu x problému A poskytne dosiahnuté riešenie T(x).
- Hovoríme, že T je r(n) - aproximujúci algoritmus pre A, pre danú funkciu r: N --> (1, ∞), ak pre ľubovoľnú inštanciu x výkonnostný pomer dosiahnutého riešenia T(x) vzhľadom na x splňuje nasledujúcu nerovnosť

$$R(x, T(x)) <= r(|x|).$$

## Trieda APX

- Pre danú triedu funkcií F hovoríme, že NPO problém A patrí do triedy F-APX, ak pre nejakú funkciu r z F pre A existuje r(n)-aproximujúci časovo polynomiálny algoritmus T.
- Ak F je rovná množine konštantných funkcií, triedu nazývame APX.



## Doteraz známe heuristiky

| Autori | Ratio | Odkaz |
|---|---|---|
| Blum (1991) | 3 | [1] |
| S. Teng, F.Yao (1993) | 2 8/9 | [2] |
| A. Czumaj (1994) | 2 5/6 | [3] |
| R.Kosaraju, J.Park, C.Stein (1994) | 2 50/63 | [4] |
| C. Armen, C. Stein (1995) | 2 ¾ | [5] |
| C. Armen, C. Stein (1996) | 2 2/3 | [6] |
| D. Breslauer, T. Jiang, Z. Jiang (1997) | 2.596 | [7] |
| Z. Sweedyk (1999) | 2 ½ | [8] |
| MGreedy algoritmus | 4 | ukázané |

# Graf prekrytí – overlap graf

- Orientovaný, úplný, hranovo ohodnotený graf prekrytiami reťazcov
- Hamiltonova cesta určí superstring z reťazcov reprezentovaných vrcholmi grafu,
- Graf je úplný, Hamiltonova cesta existuje,
- Hamiltonova cesta určuje superstring, ak žiadny reťazec nie je podreťazcom iného.



Graf prekrytí pre S = {napad, batoh, ohnostroj, ostroha, trojuholnik, koliba, rozkol, volant}

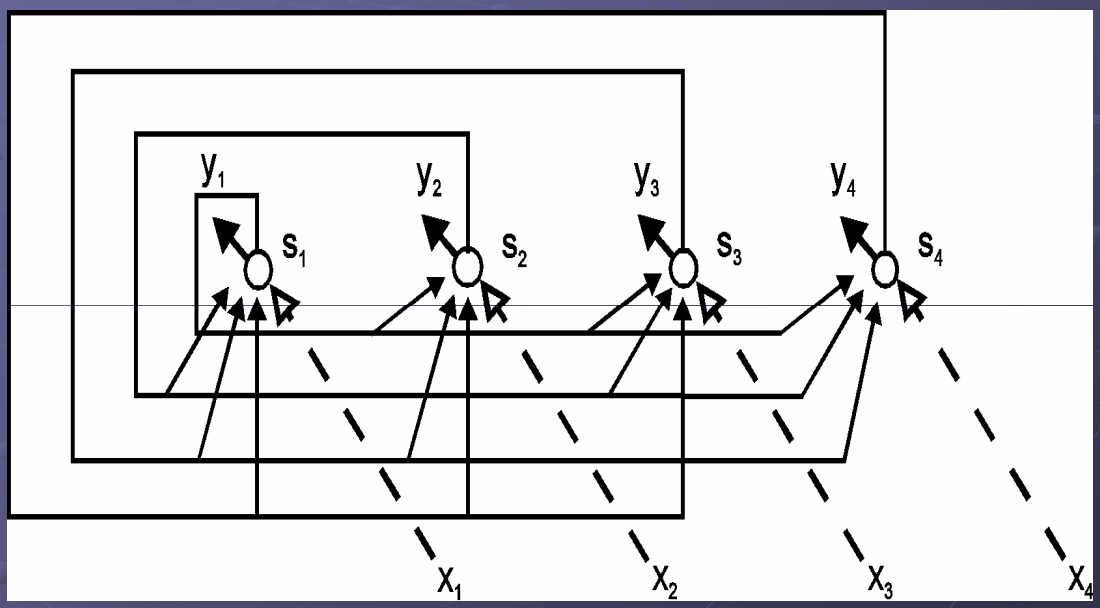
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| napad | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| batoh | 0 | 5 | 2 | 0 | 0 | 0 | 0 | 0 |
| ohnostroj | 0 | 0 | 9 | 0 | 4 | 0 | 0 | 0 |
| ostroha | 0 | 0 | 0 | 5 | 0 | 0 | 0 | 0 |
| trojuholnik | 0 | 0 | 0 | 0 | 11 | 1 | 0 | 0 |
| koliba | 0 | 2 | 0 | 0 | 0 | 6 | 0 | 0 |
| rozkol | 0 | 0 | 0 | 0 | 0 | 3 | 6 | 0 |
| volant | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 6 |

# Hopfieldove neurónové siete

- J. J. Hopfield, 1982
- patria do triedy plne rekurentných sietí s binárnym vstupom,
- je trénovaná bez dozoru,
- **Topológia siete.** Každý neurón v sieti je prepojený so všetkými ostatnými neurónmi, pričom prepojenia sú symetrické, to znamená, že $w_{ij} = w_{ji}$ pre $i, j = 1,..., n$.

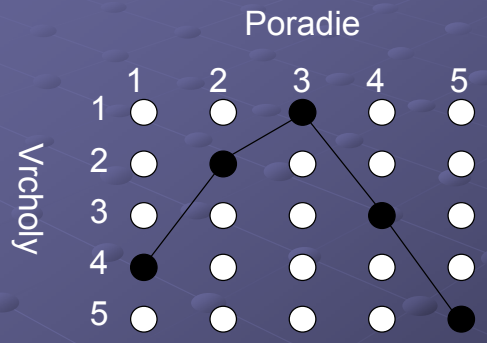


# Hopfieldove neurónové siete



# HNS pri riešení SCSS problému

Je možné predstaviť si iné usporiadanie siete






# HNS pri riešení SCSS problému

- $V_i(t+1)=\sigma(h_i(t)) = \sigma\left(\sum_{j=1}^{n} w_{ij}V_j(t)+\theta i\right)$

- $E(t)=-\frac{1}{2}\sum\sum_{i\neq j}^{n} V_i(t)V_j(t)w_{ij}-\sum_i \theta_i V_i(t)+\sum_{i=1}^{n}\int_0^{V_i(t)}\sigma^{-1}(v)dV$

- Energetická funkcia vo vzťahu k modelu

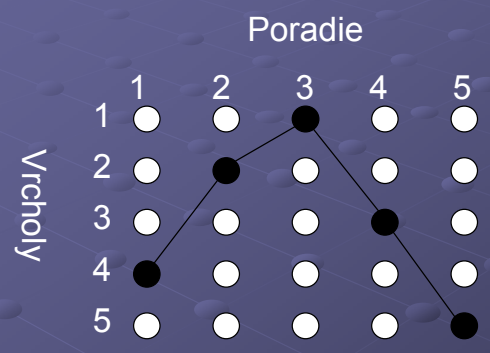
$$E_{hop}=-\frac{1}{2}\sum_i^n\sum_j^n\sum_k^n\sum_l^n T_{ij,kl}V_{ij}V_{kl}-\sum_i^n\sum_j^n \Theta_{ij}V_{ij}$$

# HNS pri riešení SCSS problému

- $n \times n$ neurónov
- Podobná idea ako pri riešení problému TSP
  - Najviac jeden neurón v každom riadku
  - Najviac jeden neurón v každom stĺpci
  - V celej sieti je aktivovaných najviac n neurónov
  - Nadreťazec má minimálnu dĺžku



# HNS pri riešení SCSS problému




| Reťazce | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| dbbbaa | 6 | 1 | 2 | 1 | 0 |
| abcdbb | 0 | 6 | 0 | 0 | 0 |
| aaabbccd | 1 | 0 | 8 | 0 | 0 |
| adddbbabc | 0 | 3 | 0 | 9 | 1 |
| ccdaa | 0 | 1 | 2 | 1 | 5 |

# HNS pri riešení SCSS problému

$$E_a = \frac{A}{2}\sum_{x}^{n}\sum_{j}^{n}\sum_{l\neq j}^{n} V_{xj}V_{xl} \qquad E_b = \frac{B}{2}\sum_{i}^{n}\sum_{x}^{n}\sum_{k\neq i}^{n} V_{ix}V_{kx}$$

$$E_c = \frac{C}{2}\left(\sum_{i}^{n}\sum_{j}^{n} V_{ij} - n\right)^2$$

$$E_d = \frac{D}{2}\left(\sum_{i=1}^{n} V_{i1}s_{ii} + \sum_{y=2}^{n}\sum_{i=1}^{n}\sum_{k=1,k\neq i}^{n} V_{iy}V_{k,y-1}\left(s_{ii} - s_{ki}\right)\right)$$



# HNS pri riešení SCSS problému

- Krok 1
  - Priradiť váhy prepojeniam neurónov a prahy neurónom

$$T_{ij,kl} = -4A\delta_{ik}\left(1-\delta_{jl}\right) - 4B\delta_{jl}\left(1-\delta_{ik}\right)$$
$$-4C\left(1-\delta_{ik}\delta_{jl}\right)\delta_{j,l-1}\left(1-\delta_{ik}\right)\left(s_{ii}-s_{ki}\right)$$

$$\Theta_{ij} = Cn - \frac{D}{2}\delta_{1j}s_{ii}$$

# HNS pri riešení SCSS problému

- Krok 2 – Inicializácia neurónov
  - nastaviť $V_{ij}(0)$ na náhodné hodnoty z intervalu (0,1)
- Krok 3 – Konvergencia

$$\Delta V_{ij}(t) = \eta\left(-V_{ij}(t) + \sigma\left(\sum_{k=1}^{n}\sum_{l=1}^{n} T_{ij,kl} V_{kl}(t) - \Theta_{ij}\right)\right)$$

$$V_{ij}(t+1) = V_{ij}(t) + \Delta V_{ij}(t)$$



# HNS pri riešení SCSS problému

- Krok 4
  - Ak sieť neskonvergovala, prejsť na Krok 1 a nastaviť nové hodnoty parametrov A,B,C,D
  - Inak skontrolovať dosiahnuté riešenie

## HNS pri riešení SCSS problému

- testovanie 4,10,12,50 reťazcov
- len špeciálne prípady reťazcov
  - reťazce majú tú istú dĺžku
  - váhy po inicializácii boli symetrické
  - energia bola klesajúca a sieť skonvergovala do stabilného stavu
- boli prípady, keď riešenia neboli nájdené



## Budúce plány

- modifikovať algoritmus pre všeobecné prípady reťazcov
- idea
  - transformovať orientovaný graf na neorientovaný
    - každý vrchol je nahradený troma novými vrcholmi
  - zaručiť "korektnosť" Hamiltonovej cesty v neorientovanom grafe

## Problém konzistentných nadreťazcov

- Nech máme množinu P pozitívnych reťazcov a množinu N negatívnych.
- Treba nájsť nadreťazec, ktorý obsahuje všetky reťazce z množiny P ako svoje podreťazce a z množiny N neobsahuje ani jeden podreťazec.



## Problém konzistentných nadreťazcov

- Ak by množina N bola prázdna, jedná sa o úlohu najkratšieho spoločného nadreťazca.
- Je jasné, že úloha má riešenie iba v prípade $P \cap N = \emptyset$.
- Tiež predpokladáme, že zjednotenie množín P a N je také, že žiaden reťazec nie je podreťazcom druhého.

# Problém konzistentných nadreťazcov

- Bez ujmy na všeobecnosti predpokladáme, že nie je žiaden reťazec, nazvime ho *a*, o dĺžke jedna, ktorý sa nachádza v množine N. Ak by taký bol, tak vzhľadom na podmienku na voľnú inklúziu neexistuje reťazec v $P \cup N$, ktorý by obsahoval *a*, teda by sme mohli *a* vymazať z abecedy a množiny N.
- V prípade, že $P = \emptyset$, úloha sa stáva triviálnou.
- Keďže problém je odvodený od problému najkratšieho spoločného nadreťazca, ktorý je NP - ťažký, tak aj problém najkratšieho konzistentného nadreťazca ostáva NP - ťažký.
- Čo aproximačné algoritmy?



# Literatúra


1. Blum, Jiang, Li, Tromp, Yannakakis: *Linear approximation of shortest superstrings.* Journal of the ACM 41(4) (1994)
2. Teng, Yao: *Approximating a shortest superstring*, Proceedings of the 34th Annual IEEE Symposium on Foundations of Computer Science(1993)
3. Czumaj, Gasianec, Piotrow a Rytter: *Parallel and sequential approximations of shortest superstrings*. Proceedings on the Scandinavian Workshop on Algorithm Theory, Lecture Notes in Computer Science 824 (1994)
4. Kosaraju, Park, Stein: *Long tours and shortest superstrings*, Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science(1994)

# Literatúra


5. Armen, Stein: *Improved length bounds for the shortest superstring problem*, proceedings of the 5th International Workshop on Algorithms and Data Structures, Lecture Notes in Computer Science 955 (1995)
6. Armen, Stein: *A 2 2/3-approximation algorithm for the shortest superstring problem*, Proceedings Combinatorial Pattern Matching, Lecture Notes in Computer Science 1075 (1996)
7. Breslauer, Jiang D., Jiang Z.: *Rotations of periodic strings and short superstrings*, Journal of ALgorithms 24 (1997)
8. Sweedyk: *A 21/5-approximation algorithm for shortest superstring*, SIAM J.Comput. 29(3) (1999)




# Ďakujem za pozornosť

→ KOŠICE