cvičenie

V toruse rozmerov $n \times n$ (t.j. zacyklenej mriežke) sú na začiatku zobudené dva vrcholy. Napíšte algoritmus, pomocou ktorého sa každý z nich dozvie identifikátor druhého s použitím $O(n)$ správ.
Ako sa úloha zmení, ak je komunikačná sieť hyperkocka?

Nájdite asymptoticky optimálny (čo do počtu správ) algoritmus na voľbu šéfa v úplnom bipartitnom grafe $K_{n,n}$. Dokážte jeho zložitosť a optimalitu.

Nájdite algoritmus na voľbu šéfa v k-chordálnom kruhu s použitím $O(n + \frac{n}{k} \log \frac{n}{k})$ správ.

$\star$

broadcasting a voľba šéfa na (ne?)orientovanej hyperkocke s lineárnym počtom správ

routovanie

cieľ: doručovať srávy medzi ľubovoľnou dvojicou

modely

- destination based
- splittable
- connections (wormhole)
- buffering
- selfish

ciele

- statické váhy (najkratšie cesty)
- dynamické váhy (hot potato)
- deadlock

najkratšie cesty

```
begin (* Initialize S to ∅ and D to ∅-distance *)
        S := ∅ ;
        forall u, v do
                if u = v then D[u, v] := 0
                else if uv ∈ E then D[u, v] := ω_uv
                else D[u, v] := ∞ ;
        (* Expand S by pivoting *)
        while S ≠ V do
           (* Loop invariant: ∀u, v : D[u, v] = d^S(u, v) *)
           begin pick w from V \ S ;
                   (* Execute a global w-pivot *)
                   forall u ∈ V do
                           (* Execute a local w-pivot at u *)
                           forall v ∈ V do
                                   D[u, v] := min ( D[u, v], D[u, w] + D[w, v] ) ;
                   S := S ∪ {w}
           end     (* ∀u, v : D[u, v] = d(u, v) *)
end
```

najkratšie cesty

```
var S_u   : set of nodes ;
    D_u   : array of weights ;
    Nb_u  : array of nodes ;

begin S_u := ∅ ;
      forall v ∈ V do
            if v = u
              then begin D_u[v] := 0 ; Nb_u[v] := udef end
            else if v ∈ Neigh_u
              then begin D_u[v] := ω_uv ; Nb_u[v] := v end
            else begin D_u[v] := ∞ ; Nb_u[v] := udef end ;
      while S_u ≠ V do
        begin pick w from V \ S_u ;
              (* All nodes must pick the same node w here *)
              if u = w
                then "broadcast the table D_w"
                else "receive the table D_w"
              forall v ∈ V do
                    if D_u[w] + D_w[v] < D_u[v] then
                      begin D_u[v] := D_u[w] + D_w[v] ;
                            Nb_u[v] := Nb_u[w]
                      end ;
              S_u := S_u ∪ {w}
        end
end
```

najkratšie cesty

```
var S_u   : set of nodes ;
    D_u   : array of weights ;
    Nb_u  : array of nodes ;

begin S_u := ∅ ;
      forall v ∈ V do
            if v = u
              then begin D_u[v] := 0 ; Nb_u[v] := udef end
            else if v ∈ Neigh_u
              then begin D_u[v] := ω_uv ; Nb_u[v] := v end
            else begin D_u[v] := ∞ ; Nb_u[v] := udef end ;
      while S_u ≠ V do
         begin pick w from V \ S_u ;
               (* Construct the tree T_w *)
               forall x ∈ Neigh_u do
                     if Nb_u[w] = x then send ⟨ ys, w ⟩ to x
                                    else send ⟨ nys, w ⟩ to x ;
               num_rec_u := 0 ; (* u must receive |Neigh_u| messages *)
               while num_rec_u < |Neigh_u| do
                     begin receive ⟨ ys, w ⟩ or ⟨ nys, w ⟩ message ;
                           num_rec_u := num_rec_u + 1
                     end ;
               if D_u[w] < ∞ then (* participate in pivot round *)
                 begin if u ≠ w
                         then receive ⟨ dtab, w, D ⟩ fromthis Nb_u[w] ;
                       forall x ∈ Neigh_u do
                             if ⟨ ys, w ⟩ was received from x
                               then send ⟨ dtab, w, D ⟩ to x ;
                       forall v ∈ V do (* local w-pivot *)
                             if D_u[w] + D[v] < D_u[v] then
                               begin D_u[v] := D_u[w] + D[v] ;
                                     Nb_u[v] := Nb_u[w]
                               end
                 end ;
               S_u := S_u ∪ {w}
         end
end
```

Netchange

```
var Neigh_u      : set of nodes ;     (* The neighbors of u *)
    D_u          : array of 0.. N ;   (* D_u[v] estimates d(u, v) *)
    Nb_u         : array of nodes ;   (* Nb_u[v] is preferred neighbor for v *)
    ndis_u       : array of 0.. N ;   (* ndis_u[w, v] estimates d(w, v) *)

Initialization:
    begin forall w ∈ Neigh_u, v ∈ V do ndis_u[w, v] := N ;
          forall v ∈ V do
                begin D_u[v] := N ; Nb_u[v] := udef end ;
          D_u[u] := 0 ; Nb_u[u] := local ;
          forall w ∈ Neigh_u do send ⟨ mydist, u, 0 ⟩ to w
    end

Procedure Recompute (v):
    begin if v = u
            then begin D_u[v] := 0 ; Nb_u[v] := local end
            else begin (* Estimate distance to v *)
                       d := 1 + min{ndis_u[w, v] : w ∈ Neigh_u} ;
                       if d < N then
                         begin D_u[v] := d ;
                               Nb_u[v] := w with 1 + ndis_u[w, v] = d
                         end
                       else begin D_u[v] := N ; Nb_u[v] := udef end
                 end ;
          if D_u[v] has changed then
            forall x ∈ Neigh_u do send ⟨ mydist, v, D_u[v] ⟩ to x
    end
```

```
Processing a ⟨ mydist, v, d ⟩ message from neighbor w:
    { A ⟨ mydist, v, d ⟩ is at the head of Q_wv }
    begin receive ⟨ mydist, v, d ⟩ from w ;
          ndis_u[w, v] := d ; Recompute (v)
    end

Upon failure of channel uw:
    begin receive ⟨ fail, w ⟩ ; Neigh_u := Neigh_u \ {w} ;
          forall v ∈ V do Recompute (v)
    end

Upon repair of channel uw:
    begin receive ⟨ repair, w ⟩ ; Neigh_u := Neigh_u ∪ {w} ;
          forall v ∈ V do
                begin ndis_u[w, v] := N ;
                      send ⟨ mydist, v, D_u[v] ⟩ to w
                end
    end
```

Netchange

```
var Neigh_u   : set of nodes ;   (* The neighbors of u *)
    D_u       : array of 0.. N ; (* D_u[v] estimates d(u, v) *)
    Nb_u      : array of nodes ; (* Nb_u[v] is preferred neighbor for v *)
    ndis_u    : array of 0.. N ; (* ndis_u[w, v] estimates d(w, v) *)

Initialization:
  begin forall w ∈ Neigh_u, v ∈ V do ndis_u[w, v] := N ;
        forall v ∈ V do
              begin D_u[v] := N ; Nb_u[v] := udef end ;
        D_u[u] := 0 ; Nb_u[u] := local ;
        forall w ∈ Neigh_u do send ⟨ mydist, u, 0 ⟩ to w
  end

Procedure Recompute (v):
  begin if v = u
          then begin D_u[v] := 0 ; Nb_u[v] := local end
          else begin (* Estimate distance to v *)
                     d := 1 + min{ndis_u[w, v] : w ∈ Neigh_u} ;
                     if d < N then
                       begin D_u[v] := d ;
                             Nb_u[v] := w with 1 + ndis_u[w, v] = d
                       end
                     else begin D_u[v] := N ; Nb_u[v] := udef end
               end ;
        if D_u[v] has changed then
          forall x ∈ Neigh_u do send ⟨ mydist, v, D_u[v] ⟩ to x
  end
```

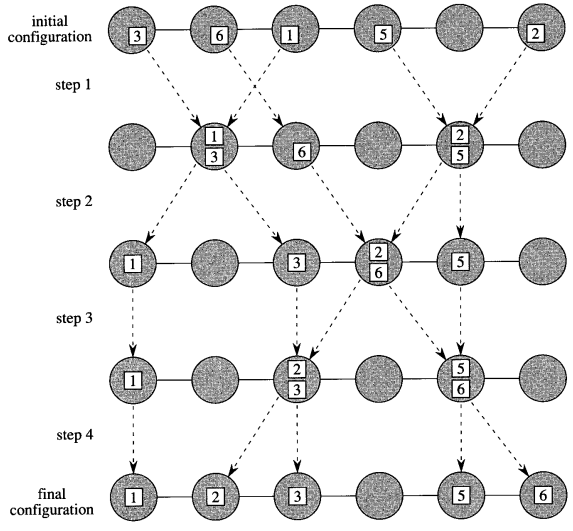
```
Processing a ⟨ mydist, v, d ⟩ message from neighbor w:
    { A ⟨ mydist, v, d ⟩ is at the head of Q_wv }
    begin receive ⟨ mydist, v, d ⟩ from w ;
          ndis_u[w, v] := d ; Recompute (v)
    end

Upon failure of channel uw:
    begin receive ⟨ fail, w ⟩ ; Neigh_u := Neigh_u \ {w} ;
          forall v ∈ V do Recompute (v)
    end

Upon repair of channel uw:
    begin receive ⟨ repair, w ⟩ ; Neigh_u := Neigh_u ∪ {w} ;
          forall v ∈ V do
                begin ndis_u[w, v] := N ;
                      send ⟨ mydist, v, D_u[v] ⟩ to w
                end
    end
```

korektnosť

lexikograficky klesá hodnota $[t_0, t_1, \ldots, t_N]$
kde t_i je počet správ $\langle \textbf{mydist}, i \rangle +$ počet dvojíc u, v kde $D_u[v] = i$

packet routing

- synchrónny režim
- vrcholy majú pakety (uložené v bufferoch)
- v jednom kroku po jednej linke ide max. jeden paket
- algoritmus = odchádzajúce linky + priorita bufferov
- celkový čas



packet routing na mriežke $\sqrt{N} \times \sqrt{N}$

Každý vrchol má 1 paket, do každého smeruje 1 paket (permutation routing)

Najprv riadok, potom stĺpec. Prednosť má ten s najdlhšou cestou.

analýza: stačí $2\sqrt{N} - 2$ krokov

- po $\sqrt{N} - 1$ krokoch je každý v správnom stĺpci (nebrzdia sa)
- routovanie v stĺpci ide v $\sqrt{N} - 1$ krokoch
  - pre každé i platí: po $N - 1$ krokoch sú koncové pakety na koncových miestach
  - dôvod: zdržujú sa iba navzájom

veľkosť buffra v najhoršom prípade: $2/3\sqrt{N} - 3$

veľkosť buffra: priemerný prípad I

setting

Každý vrchol má jeden paket s **náhodným cieľom**

max. veľkosť buffra $\approx$ počet zahnutí vo vrchole

psť, že aspoň r zahne $\leq \binom{\sqrt{N}}{r} \left(\frac{1}{\sqrt{N}}\right)^r < \left(\frac{e}{r}\right)^r$

pre $r = \frac{e \log N}{\log \log N}$ je psť $o(N^{-2})$

veľkosť buffra: priemerný prípad II

wide-channel: nepredbiehajú sa

lema

psť, že vo wch prejde aspoň $\alpha\Delta/2$ paketov cez hranu e počas $t+1, t+2, \ldots, t+\Delta$ je najviac $e^{(\alpha-1-\alpha \ln \alpha)\Delta/2}$

očakávaný počet paketov na hrane $(i,j) \mapsto (i+1,j)$ je

$$\frac{2i(\sqrt{N}-i)\Delta}{N} \leq \frac{\Delta}{2}$$

chceme ukázať, že s veľkou psťou ich neprejde príliš viac

Černovov odhad

lema

Majme n nezávislých Bernoulihho náh. prem. $X_1, \ldots, X_n$, pričom $Pr[X_k = 1] \leq P_k$. Potom

$$Pr[X \geq \beta P] \leq e^{(1-\frac{1}{\beta}-\ln \beta)\beta P}$$

kde $X = \sum X_i$, $P = \sum P_i$

$$E[e^{\lambda X_k}] \leq 1 + P_k(e^{\lambda} - 1) \leq e^{P_k(e^{\lambda}-1)}$$

$$E[e^{\lambda X}] \leq e^{P(e^{\lambda}-1)}$$

$$P[e^{\lambda X} \geq e^{\lambda \beta P}] \leq \frac{E[e^{\lambda X}]}{e^{\lambda \beta P}} \leq e^{P(e^{\lambda}-1)-\lambda \beta P}$$

veľkosť buffra: priemerný prípad II

lema

Majme n nezávislých Bernoulihho náh. prem. $X_1, \ldots, X_n$, pričom $Pr[X_k = 1] \leq P_k$. Potom

$$Pr[X \geq \beta P] \leq e^{(1-\frac{1}{\beta}-\ln \beta)\beta P}$$

kde $X = \sum X_i$, $P = \sum P_i$

lema

psť, že vo wch prejde aspoň $\alpha\Delta/2$ paketov cez hranu e počas $t+1, t+2, \ldots, t+\Delta$ je najviac $e^{(\alpha-1-\alpha \ln \alpha)\Delta/2}$

očakávaný počet paketov na hrane $(i,j) \mapsto (i+1,j)$ je

$$\frac{2i(\sqrt{N}-i)\Delta}{N} \leq \frac{\Delta}{2}$$

chceme ukázať, že s veľkou psťou ich neprejde príliš viac

veľkosť buffra: priemerný prípad II

lema

ak je paket vo vzd. d od hrany e v čase T, a p prejde cez e v čase $T + d + \delta$, potom v každom kroku $[T + d, T + d + \delta]$ prejde paket cez e

dosledok

ak paket prejde cez e v čase T vo wch, a prejde cez e v čase $T + \delta$ v št. , tak v každom kroku $[T, T + \delta]$ prejde paket

lema

ak počas $[T + 1, T + \Delta]$ prejde cez e x paketov v št., tak pre nejaké t prejde $x + t$ paketov cez e v čase $[T + 1 = t, T + \Delta]$ vo wch.

lema

psť, že cez e prejde viac ako $\alpha\Delta/2$ paketov počas konkrétneho okna Δ krokov je najviac $O(e^{(\alpha-1-\alpha \ln \alpha)\Delta/2})$

dynamické routovanie

model

V každom kroku sa v každom vrchole s psťou λ narodí paket s náhodným cieľom.

stabilita

Pre $\lambda \geq 4/\sqrt{N}$ je systém nestabilný

veta

Ak je $\lambda \leq 0.99\frac{4}{\sqrt{N}}$, tak psť zdržania konkrétneho paketu o Δ krokov je $e^{-O(\Delta)}$.
W.h.p. stačí buffer $O(1 + \frac{\log T}{\log N})$.

hierarchické routovanie

cieľ: minimalizovať počet rozhodnutí

veta

Pre sieť s N vrcholmi stačí $O(\sqrt{N})$ rozhodnutí pri použití 3 farieb.

s-klastre:

- každý je súvislý, pokrývajú všetky vrcholy
- každý obsahuje aspoň s vrcholov a má polomer najviac $2s$

kostra spájajúca centrá klastrov: m listov $\Rightarrow$ $m-2$ vetvení

veta

Pre sieť s N a pre $f \leq \log N$ stačí $O(f \cdot N^{1/f})$ rozhodnutí a $2f+1$ farieb

po i klastrovaniach s parametrom s: m_i listov, max. $m_i - 2$ vetvení $\Rightarrow$ $m_{i+1} = m_i(2/s)$

kompaktné routovanie

intervalové routovanie

- vrcholy majú čísla $1 \ldots n$
- každý port má priradený interval

s lineárnymi intervalmi problém $\Rightarrow$ pavúk

s cyklickými intervalmi ide v stromoch $\Rightarrow$ vo všetkých grafoch (kostra)

najkratšie cesty

nie vždy sa dá (glóbus)

kompaktné routovanie

viac intervalov?

Majme graf s max. stupňom Δ a optimálnym k-IRS. Nech $Q = \{q_1, \ldots, q_l\}$ a $W = \{w_1, \ldots, l_m\}$ sú dizjunktné množiny vrcholov také, že $\forall w_i, w_j \in W, w_i \neq w_j \exists q \in Q$ také, že pre žiadnu hranu (q, q') neplatí, že do w_i aj do w_j sa routuje po q'. Potom

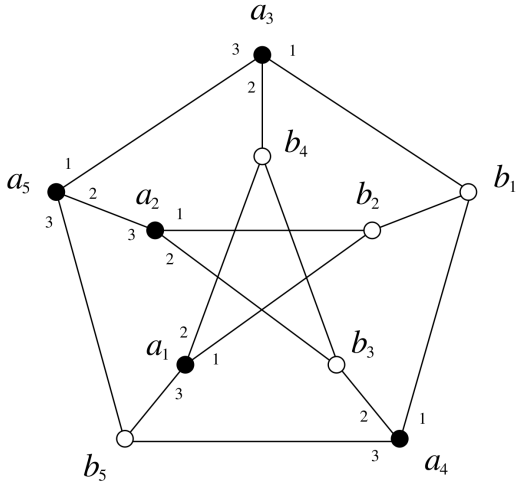
$$k \geq \frac{m}{l\Delta}$$

kompaktné routovanie – dolný odhad

matrix of constraints

existujú dve množiny vrcholov A, B, že pre každú dvojicu a_i, b_j používa port $m_{i,j}$

Γ	b_1	b_2	b_3	b_4	b_5
a_1	1	1	2	2	3
a_2	1	1	2	2	3
a_3	1	1	2	2	3
a_4	1	1	2	2	3
a_5	1	2	2	1	3



kompaktné routovanie – dolný odhad

veta

pre každú maticu existuje graf, ktorého je "m.o.c."

veta

každá kompaktná schéma vyžaduje aspoň $\Omega(n \log n)$ bitov v $\Omega(n^{\varepsilon})$ vrcholoch.

cvičenia

mriežka $\sqrt{n} \times \sqrt{n}$, prednosť má hocikto; ukážte, že v najhoršom prípade treba viac ako $2\sqrt{n}$ krokov, ale stačí $O(\sqrt{n})$

mriežka $\sqrt{n} \times \sqrt{n}$, v každom vrchole správa do náhodného. Ukážte, že w.h.p. do žiadnoho vrchola nesmeruje viac ako $3 \log n / \log \log n$ správ.

majme cestu z n procesorov, každý chce routovať práve dva pakety (červený a modrý), pričom červené aj modré tvoria permutáciu. ukážte, že stačí n krokov

Nájdite IRS s jedným intervalom po najkratších cestách pre hyperkocku

odolnosť voči chybám – strata správ

Problém dohody

- synchrónny systém
- známe identifikátory
- každý má na vstupe 0/1
- správy sa môžu strácať
- každý proces sa musí rozhodnúť
- treba zaručiť
  - **Dohoda:** všetky procesy sa rozhodnú na tú istú hodnotu
  - **Terminácia:** každý proces sa rozhodne v konečnom čase
  - **Netrivialita:**
    1. Ak všetci začnú s hodnotou 0, musia sa dohodnúť na 0.
    2. Ak všetci začnú s hodnotou 1 a správy sa nestrácajú, musia sa dohodnúť na 1.

neexistuje deterministické riešenie

- 2 vrcholy, 1 linka
- sporom, nech existuje a trvá r kôl
- výpočet, kde začnú obaja s hodnotou 1 a nestrácajú sa správy
- dohodnú sa na 1
- stratí sa posledná správa, jeden z nich to nezistí
- výpočet, kde neprejde ani jedna správa a dohodnú sa na 1
- jeden z nich dostane na vstup 0
- aj druhý

randomizované riešenie (úplný graf)

komunikačný pattern

zoznam trojíc (i, j, t): v čase t sa nestratí správa z $i \mapsto j$

(fixný) adversary = vstup a komunikačný pattern
Dohoda: $Pr[$ nejaké dva procesy sa rozhodnú na rôznu hodnotu$] \leq \varepsilon$

algoritmus s $\varepsilon = 1/r$

daný adversary γ: dvojice (i, t), kde i-procesor, t-čas majme usporiadanie:

1. $(i, t) \leq_\gamma (i, t')$, kde $t \leq t'$
2. ak $(i, j, t) \in \gamma$, tak $(i, t-1) \leq_\gamma (j, t)$
3. tranzitivita

úroveň informovanosti

1. $level_\gamma(i, 0) = 0$
2. ak $t > 0$ a existuje $j \neq i$ také, že $(j, 0) \not\leq_\gamma (i, t)$, tak $level_\gamma(i, t) = 0$
3. nech l_j je $\max\{level_\gamma(j, t') \mid (j, t') \leq_\gamma (i, t)\}$
   potom $level_\gamma(i, t) = 1 + \min\{l_j \mid j \neq i\}$

algoritmus

- prvý proces vygeneruje náhodný kľúč
- procesy si počítajú level
- rozhodnutie 1, ak všetci majú 1 a môj level je aspoň kľúč

$rounds := rounds + 1$
let (L_j, V_j, k_j) be the message from j, for each j from which a message arrives
if some $k_j \neq undefined$ then $key := k_j$
for all $j \neq i$ do
    if some $V_{i'}(j) \neq undefined$ then $val(j) := V_{i'}(j)$
    if some $L_{i'}(j) > level(j)$ then $level(j) := \max\{L_{i'}(j)\}$
$level(i) := 1 + \min\{level(j) : j \neq i\}$
if $rounds = r$ then
    if $key \neq undefined$ and $level(i) \geq key$ and $val(j) = 1$ for all j then
        $decision := 1$
    else $decision := 0$

dôkaz

- **Dohoda:** $Pr[$ nejaké dva procesy sa rozhodnú na rôznu hodnotu$] \leq \varepsilon$
- **Terminácia:** každý proces sa rozhodne v konečnom čase
- **Netrivialita:**
  1. Ak všetci začnú s hodnotou 0, musia sa dohodnúť na 0.
  2. Ak všetci začnú s hodnotou 1 a správy sa nestrácajú, musia sa dohodnúť na 1.

terminácia a netrivialita sú zrejmé
pre fixný pattern, aká je pravdepodobnosť nezhody?
levely sa líšia max o 1, preto jediný problém je ak $key = \max\{l_i\}$

dolný odhad

ľubovoľný r-kolový algoritmus má pravdepodobnosť nezhody aspoň $\frac{1}{r+1}$

orez

pre adversary B s patternom γ a proces i, $B' = \mathit{prune}(B, i)$

1. ak $(j, 0) \leq_\gamma (i, r)$ tak sa vstup j zachová, inak znuluje
2. trojica (j, j', t) je v kom. patterne B', akk je v γ a $(j', t) \leq_\gamma (i, r)$

$P^B[i \text{ sa rozhodne 1}] = P^{\mathit{prune}(B,i)}[i \text{ sa rozhodne 1}]$

lema

Ak majú na vstupe všetci 1, $P[i \text{ sa rozhodne 1}] \leq \varepsilon(\mathit{level}(i, r) + 1)$

lema

Ak majú na vstupe všetci 1, $P[i \text{ sa rozhodne 1}] \leq \varepsilon(level(i,r)+1)$

- indukcia na $level(i,r)$: nech $level(i,r) = 0$:
- $B' = prune(B,i) = prune(B',i)$
- $P^{B}[i \text{ sa rozhodne 1}] = P^{B'}[i \text{ sa rozhodne 1}]$
- od j-čka neprišla správa, $B'' = prune(B',j) = prune(B'',j)$ je triviálny adversary
- $P^{B'}[j \text{ sa rozhodne 1}] = P^{B''}[j \text{ sa rozhodne 1}]$
- lenže $P^{B''}[j \text{ sa rozhodne 1}] = 0$, takže $P^{B'}[j \text{ sa rozhodne 1}] = 0$
- psť nezhody je ε, $\Rightarrow |P^{B'}[i \text{ sa rozhodne 1}] - P^{B'}[j \text{ sa rozhodne 1}]| \leq \varepsilon$
- preto $P^{B'}[i \text{ sa rozhodne 1}] \leq \varepsilon$ a $P^{B}[i \text{ sa rozhodne 1}] \leq \varepsilon$
- nech $level(i,r) > 0$
- $B' = prune(B,i) = prune(B',i)$