Önálló laboratórium beszámoló

Távközlési és Médiainformatikai Tanszék

Készítette:	**Kerékfy Miklós**
Neptun-kód:	**XFQ3OX**
Ágazat:	**Hálózatok és szolgáltatások**
E-mail cím:	**miklos@kerekfy.hu**
Konzulens:	**Dr. Heszberger Zalán**
E-mail címe:	**heszberger@tmit.bme.hu**

Téma címe: Útvonal-választási algoritmusok megismerése és kidolgozása

Feladat

Az Internet robbanásszerű növekedése napról-napra próbára teszi az eredetileg kis hálózat méretekre kitalált útvonal-választó („routing”) algoritmusokat. Egyre kevésbé működik a régi elgondolások foltozásszerű javítgatása, alapvetően új módszerek kitalálása a cél.

A laborfeladat célja új útvonal-választó algoritmusok kitalálása volt. Az elérendő célok a kis számítási komplexitás, alacsony memória igény, ill. az ezen elvárások tekintetében esetlegesen elosztottan megvalósítható működés.

A félév elején a feladat a bővebb téma egyéni megismerése volt, ajánlott cikkek mentén: legrövidebb utas útvonal-választók, gráf feszítők, policy alapú útvonal-választás. A félév hátralévő részében közös munkával dolgoztuk fel a már ismert megoldásokat, majd egyéni munkám volt ezek tudományosan kielégítő összegzése, és bemutatása, valamint új irányok és lehetőségek feltárása.

2011/2012. 1. félév

Kerékfy Miklós (XFQ3OX) 2012.05.04.

A laboratóriumi munka környezetének ismertetése, a munka előzményei és kiindulási állapota

Bevezető/elméleti összefoglaló

Ma az internetes útvonal-választás az alapvetően statikusan konfigurált BGP (Border Gateway Protocol - határátjáró protokoll) útvonalválasztókon alapszik, melyek az internetet alkotó AS-ek (Autonomous System - autonóm rendszerek) között működnek. Az útvonalakat a megváltozott linkek állapota alapján különböző metrikákra támaszkodva tárolják le, majd osztják meg a szomszédjaikkal. Ennek a rendszernek két fő hiányossága van: a statikussága miatt a rendszerekben kézileg kell bekonfigurálni a BGP útvonal-választókat, valamint a másik probléma az, hogy a RT-k (Routing Table - útvonal-választó táblázat) nagyságának növekedése közelít az exponenciálishoz az útvonal-választók számához képest [1]. Míg ez utóbbi problémát különböző módszerek (pl. [2],[3],[4]) próbálták csökkenteni, a megoldáshoz nem sikerült közelebb kerülni, ugyanis egy megfelelően skálázódó algoritmus lineárisnál lassabban kellene, hogy növekedjen. Azt azonban Gavoille és Pérennès [5]-ben bebizonyítja, hogy minden algoritmushoz, mely a legrövidebb utakon alapú útvonal-választást valósít meg, lehet olyan n pontú d fokú gráfot adni, melyen az útvonal-választáshoz szükséges memóriaigény lokálisan $\Theta(n \cdot log\, d)$, és így globálisan $\Theta(n^2 \cdot log\, d)$. Mivel ez az alsó korlát a hagyományos RT-k tárolásának memóriaigényével egyezik meg, azt jelenti, hogy azoknál jobb legrövidebb utas útvonal-választó algoritmus nem létezhet.

A legrövidebb utas algoritmusok helyett tehát azok az algoritmusok rendelkezhetnek megfelelően skálázódó memóriaigénnyel, melyek nem ragaszkodnak szigorúan a legrövidebb út megtalálásához. A szakirodalom az ilyen útvonal-választást kompaktnak nevezi, utalva arra, hogy az eredményezett RT-k kisebb méretűek lesznek. Ezek az algoritmusok k „stretch”-csel rendelkeznek, ami azt jelenti, hogy az általuk megtalált út hossza legfeljebb k-szorosa a legrövidebb útnak ($k \geq 1$). (A stretch fogalmát a gráfelméletben a feszítő részfák jellemzésénél használják, ahol egy G gráfnak a k stretch-ű G' feszítő részfájában minden A',B' csúcspár közötti távolság legfeljebb k-szorosa az eredeti gráfban levő A,B csúcspár távolságának [6].) Erdős Pál 1963-ban felállított [7], azóta meg nem cáfolt a gráfok girth paraméterével (legrövidebb kör hossza) kapcsolatos sejtéséből következik az a megállapítás, hogy általános gráfok esetén, a legrosszabb esetben legalább $\Omega(n^{1+1/k})$ memória szükséges $2k+1$-nél alacsonyabb stretch eléréséhez. Mivel ennek a korlátnak az átlépése tehát elég valószínűtlen, az algoritmusok egyéb paraméterei, nevesen az előszámítási igény, valamint a lekérdezési idő lesznek azok, melyeket a memória minimumon tartása mellett érdemes optimalizálni.

Az útvonalválasztás témakörében a kutatások másik jelentős témája a beágyazás, mely azt jelenti, hogy a gráf pontjaihoz egyértelműen hozzárendelünk koordinátákat egy metrikus térből. Egy $G(V,E)$ gráf (X,p) metrikus térbe (ahol X a koordinátáit, p pedig a metrikát jelöli) való mohó beágyazása $f: V \rightarrow X$, úgy, hogy minden s-t forrás-cél pontpárra igaz, hogy s-nek van egy olyan G-beli v szomszédja amelyre $p(f(v), f(t)) < p(f(s), f(t))$ teljesül.

A téma nagyon mélyen taglalt, az egyik alapkutatás a témában Kleinberg [8], és dimenzióbeli valamint méretbeli megkötéseket Maymounkov [9] is mélyen megvizsgált. A kutatások jelentős része azonban „no-stretch”, azaz k=1 stretch-ű beágyazást vizsgál, ami alapvetően nagyobb tárolási kapacitást igényel. Egynél nagyobb stretch-ű beágyazást vizsgálták

már Flury és társai [10], majd idén márciusban elhangzott előadásukban Cvetovski és Crovella [11], azonban érdekes módon vizsgálataik során nem használják fel a következő összefüggéseket. Zwick és Thorup eredményei [12][13], valamint mások erre épülő munkái (például Pătraşcu és Roditty [14]) rendkívül jó kiindulást adnak arra, hogy a két általam tanulmányozott témát összekössük.

A munka eredményeinek továbbviteléhez a policy alapú kompakt útvonal-választás elméletét használtam, melyet behatóbban Rétvári és társai [15] cikke alapján ismertem meg. Ott az általános policy alapú útvonalválasztó algebrát (melynek forrása többek közt [16]) használva mutatják be általánosságban és néhány konkrét esetet kiemelve azt, hogy a legrövidebb út mellett milyen útvonalválasztási módszerek létez(het)nek. Egy útvonalválasztó algebrát az ott bevezetett jelölésekkel $\mathcal{A}$-nak nevezve megadható a következő definíció: $\mathcal{A} = (W, \varphi, \oplus, \preccurlyeq)$ egy teljesen rendezett félcsoport, melyben W az élekhez rendelhető lehetséges súlyok halmaza, φ egy végtelen-elem, melyet W nem tartalmaz, $\oplus$ a súlyokon végezhető művelet, $\preccurlyeq$ pedig a súlyok közti reláció. Ekkor például a legrövidebb út algebrát $\mathcal{S} = (\mathbb{R}+, \infty, +, \leq)$ módon lehet leírni.

A munka állapota, készültségi foka a félév elején

A munkát ebben a félévben kezdtem el, ezért csupán a konzulensemtől kapott ajánlott irodalom, valamint a konzultációk során kapott többletinformációk voltak azok, amire építeni tudtam.

Az elvégzett munka és eredmények ismertetése

Az általam konkrétan elvégzett munka bemutatása

Kompakt útvonalválasztás (compact routing)

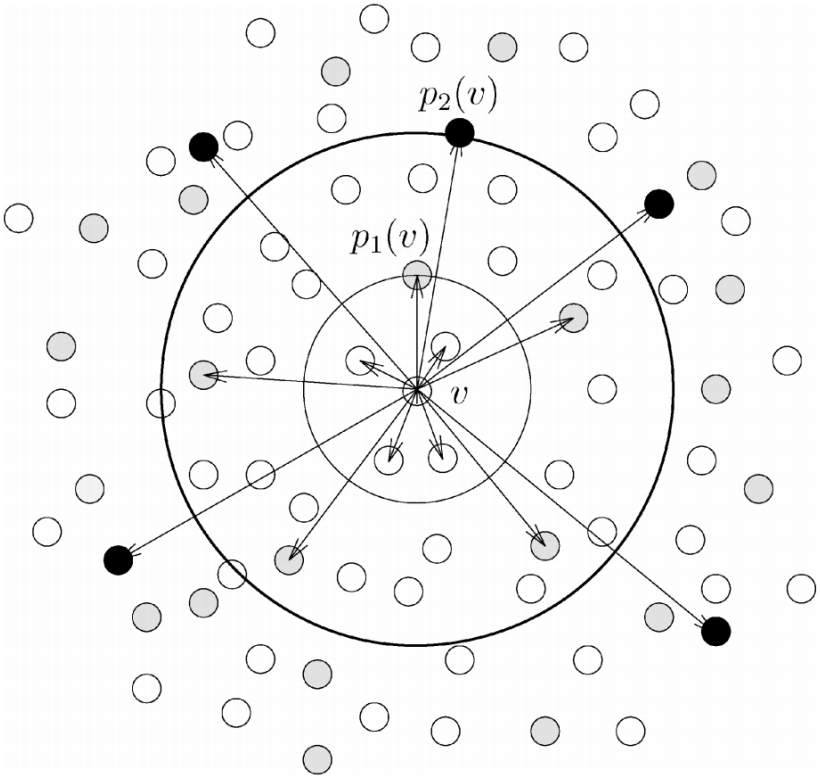
Ismerve az Erdős-sejtés által megadott alsó korlátot, több algoritmust is lehet találni, melyek ebben a paraméterben optimálishoz közelinek számítanak. Dor és társai [17] cikkében például egy 3 stretch-ű, $\tilde{O}(n^{5/3})$ memóriában $\tilde{O}(m^{2/3}\ n)$ feldolgozási idővel egy $O(1)$, azaz konstans időben lekérdezhető algoritmust írnak le. (Az $\tilde{O}(n)$ jelölés jelentése megegyezik az $O(n \cdot \log n)$ jelölésével.) Awerbuch és társai [18] eredménye általános strech-re egy $64k$ strech-ű, $\tilde{O}(kn^{1+1/k})$ memóriaigényű, $\tilde{O}(mn^{1/k})$ feldolgozási idejű és $\tilde{O}(kn^{1/k})$ lekérdezési idejű algoritmus. Cohen [19] ezt az eredményt jelentősen javítja úgy, hogy a strech $2k+\varepsilon$-ra csökkenti, de Matoušek [20] ezt egy teljesen eltérő módszerrel tovább csökkentette $2k-1$-re. Ez az eredmény már eléri memóriaigényben az Erdős által megjósolt alsó határt. Thorup és Zwick [12] eredménye egy olyan módszer, mellyel a lekérdezési időt konstanssá lehet tenni, azaz $\tilde{O}(kn^{1/k})$ helyett $O(k)$ időben lehet az adatszerkezetből kinyerni az utakat. Ezen eredményeket foglalja össze az 1. táblázat.

Stretch	Memória	Feldolgozás	Lekérdezés	Forrás
3	$\tilde{O}(n^{5/3})$	$\tilde{O}(m^{2/3}\ n)$	$O(1)$	Dor és társai [17]
$64k$	$\tilde{O}(kn^{1+1/k})$	$\tilde{O}(mn^{1/k})$	$\tilde{O}(kn^{1/k})$	Awerbuch és társai [18]
$2k+\varepsilon$	$\tilde{O}(kn^{1+1/k})$	$\tilde{O}(mn^{1/k})$	$\tilde{O}(kn^{1/k})$	Cohen [19]
$2k-1$	$\tilde{O}(kn^{1+1/k})$	$\tilde{O}(mn^{1/k})$	$\tilde{O}(kn^{1/k})$	Matoušek [20]
$2k-1$	$O(kn^{1+1/k})$	$O(kmn^{1/k})$	$O(k)$	Thorup és Zwick [12]

1. táblázat: Néhány algoritmus és adatmodell paraméterei

Thorup és Zwick [12] eredményének megismerése során több olyan módszert lehet megismerni, melyek használata a későbbiekben további eredmények felderítéséhez használható.

Az algoritmus bemenetén egy tetszőleges $G(V,E)$ gráf áll, ahol V a csúcsok halmaza és $|V| = n$, E pedig az élek halmaza és $|E| = m$. Célunk egy $2k-1$ stretch-ű útvonalválasztást lehetővé tevő adatszerkezet, ahol k pozitív természetes szám. Először az algoritmus létrehoz $k+1$ db halmazt, melyeket $A_0, A_1, \ldots A_k$-val jelölünk és középpontoknak vagy Cowen [21] elnevezése alapján „landmark”-oknak (≈mérföldkő) nevezzük. $A_0 = V$, A_k pedig üres halmaz. A_i $(0 < i < k$ esetén) tartalmazza A_{i-1} összes elemét $n^{-1/k}$ valószínűséggel, egymástól függetlenül. A_i -k várható mérete így exponenciálisan csökken (például $n = 100$ és $k = 2$ esetén $|A0| = 100, |A1| \cong 10, |A2| = 0$). Egy tetszőleges v,w pontpár távolságát jelölje $d(v,w)$, és a gráfban közöttük vett legrövidebb út hosszát adja meg. Ekkor legyen $d(A_i,v)$ az a legkisebb távolság, mely v és egy A_i-beli pont között van. A jelölésből következik, hogy $d(A_0, v) = 0$ és $d(A_k, v) = \infty$ tetszőleges v-re. Jelölje továbbá $p_i(v)$ A_i halmaz azon pontját, melyre igaz, hogy $d(p_i(v), v) = d(A_i, v)$ és nevezzük ezt a v pont i-edik tanújának. Az így felépített A_i landmark halmazokon végigmenve megállapíthatjuk azon pontok halmazát, melyek A_{i-1}-ben benne vannak, de A_i-ben már nem $(0 < i < k)$. Ezen kiesett pontokat jelölje w, a gráf összes pontját v. Ekkor minden v-nek egy $B(v)$ „bunch”-át képezzük úgy, hogy abban minden olyan w benne lesz, melyre igaz, hogy $d(w, v) < d(A_i, v)$.



1. ábra: Egy v pont bunch-ának felépítése k=3 esetén [12]

Az 1. ábra bemutatja *k=3* esetén *v* pont bunch-ának képzését, minden nyíl egy bunch-beli pontot jelöl. A fehér pontok azok, melyek az $A_0 \rightarrow A_1$váltásnál esnek ki, a szürkék a $A_1 \rightarrow A_2$, a feketék pedig az $A_2 \rightarrow A_3$ váltások kieső pontjai. Először bekerülnek a fehér pontok közül azok, melyek közelebb vannak *v*-hez mint $p_1(v)$, majd a szürke pontok közül azok, melyek közelebb vannak *v*-hez, mint $p_2(v)$, végül az összes fekete pont (hiszen $A_3 = A_k$ üres halmaz).

A bunch-ok felépítésével párhuzamosan minden *w*-ben eltárolunk egy *C(w)* halmazt (cluster), melynek minden olyan *v* eleme, amire igaz, hogy *w* benne van *B(v)*-ben, azaz a bunch és a cluster egymás ellentettje. Minden *w*-ben létrehozunk egy *T(w)* feszítőfát *C(w)* elemei fölött. Ekkor az útvonalválasztás a következő módon zajlik: mindkét pontból felváltva ellenőrizzük, hogy nincs-e a másik pont a bunch-ában, majd ha nincs, elkezdjük a tanúikkal ugyanezt az ellenőrzést végrehajtani. Konstans (*k*-ban) lépés után biztosan találunk egy olyan *w* pontot, mely benne lesz a forrás és a cél bunchában is. Ekkor az ebben a pontban található *T(w)* fa segítségével fogunk tudni csomagokat küldeni a forrás és a cél között. Az algoritmus vázlatos megvalósítását a függelékben (10.oldal) lehet megtalálni.

A valódi útvonal-választás algoritmusát Thorup és Zwick egy külön cikkben [13] részletezi. Itt bemutatnak egy hatékony módszert arra, hogy hogyan lehet fákban a pontok megfelelő címkézésével hatékony útvonalválasztást megvalósítani, majd bemutatják azt is, hogy hogyan lehet a korábban ismertetett fákban alacsony stretch-ű utat találni. A [13] függelékében szereplő bizonyítás megmutatja, hogy a konstrukcióban minden pontnak rendelkezésére áll annyi információ lokálisan, hogy bármely másik pontba találjon egy olyan fát, melyen keresztül *4k-5* stretch-ű úton el tudja azt érni. A kívánt *2k-1* stretch-ű fa megtaláláshoz olyan információk is szükségesek, melyek a másik pontban állnak csak rendelkezésre, ezért egy ún. kézfogási algoritmusra van szükség. Ezen algoritmus során legyen *u* a forrás, *v* pedig a cél. Ekkor *u*-ban először megtaláljuk a *4k-5* stretch-ű fát, és ennek segítségével elküldünk egy üzenetet *v*-nek, mely tartalmazza azokat az információkat, melyeket *u* lokálisan ki tudott számolni. Ezek segítségével *v* ki tudja számolni a *2k-1* stretch-ű fát

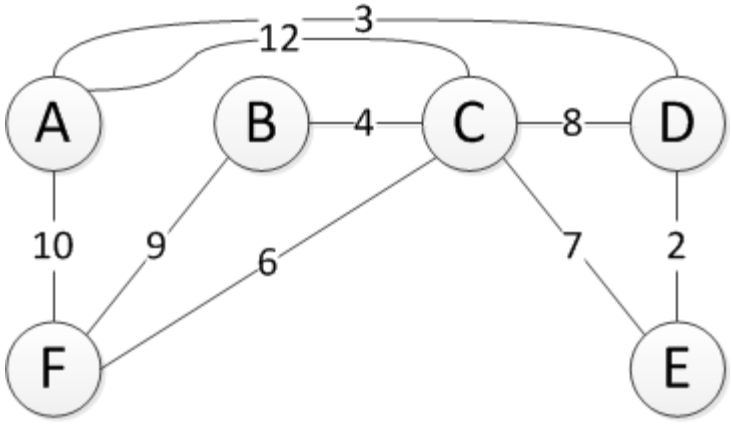
tartalmazó pontot, és ezt az információt visszaküldi *u*-nak. Ezután *u*-ban rendelkezésre áll minden adat ahhoz, hogy a további kommunikációra már a *2k-1* stretch-ű utat használja. Mivel a kézfogás algoritmusa konstans 2 lépést ad hozzá a kommunikációhoz, az átlagos kommunikációs feladatok mellett elhanyagolható pluszköltséget jelent, így a módszert valóban megfelelőnek tekinthetjük *2k-1* stretch-ű útvonalválasztásra.

Mohó beágyazás (greedy embedding)

Mind az euklideszi, mind a Bolyai-Lobacsevszkij hiperbolikus geometria terén sok eredmény született hatékony beágyazási algoritmusokról. Ezek azonban elsősorban a no-stretch beágyazást vizsgálják, az egynél nagyobb stretch csak kevés tanulmányban merül fel. A kompakt útvonalválasztás elmélete azonban annyira rokon a beágyazással, hogy csak néhány feltevéssel a két témát össze lehet kötni, és az egyikben folyó kutatásokat a másikban hasznosítani.

Maymounkov [9]-ben megmutatja, hogy bármilyen fát be lehet ágyazni mohó módon egy $O(log\ n)$ dimenziós euklideszi térbe úgy, hogy ehhez $O(log^2\ n)$ bites koordinátákat kell megadni. Mivel az előző fejezetben bemutatott konstrukció fákból áll, ezért egy elméletet felállítva hatékony *2k-1* stretch mohó beágyazást javasolhatunk. Az elmélet az, hogy az előző fejezetben felállított ritka feszítő részgráf beágyazható $o(n)$ dimenzióban, ami a korábbi eredmények alapján valószínű. Ekkor legyen egy G gráf mohó beágyazása a következő: állítsuk fel a Thorup-Zwick féle feszítő részgráfot, majd ágyazzuk azt be no-stretch mohó módon. Az így kapott adatszerkezetünk $o(n)$ dimenzión ábrázolt n db pont, és a rajta mohó útvonalválasztással elérhető utak teljesítik a *2k-1*-es felső korlátot. Mivel $o(n)$ dimenziósak a pontjaink, azoknak csupán ennyi lokális adatot kell tárolniuk, azaz globálisan $o(n^2)$ helyigényt fog jelenteni. Ez kevesebb, mint a hagyományos RT-s módszer, ám több, mint a legtöbb kompakt útvonalválasztásnál elérhető. Amit azonban egy ilyen technikával nyerni lehet, hogy az útvonalválasztókban a helyi döntéseket meg lehet hozni O(1) időben, hiszen csupán a szomszédok koordinátáit kell ismerni ahhoz, hogy eldönthesse, merre kell a csomagot továbbítani. Az így nyert számítási kapacitás az útvonalválasztókban kifejezetten jelentős lehet.

A következő példa rávilágít arra, hogy hogyan lehet megvalósítani az általam javasolt módszert. Legyen G egy súlyozott irányítatlan gráf a 2. ábra alapján.



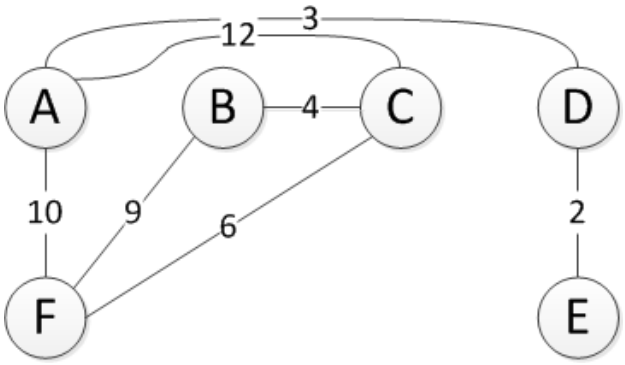
2. ábra: G gráf pontjai és súlyozott élei

A gráf alapján létrehozott legrövidebb út metrika a 2. táblázatban látható.

	A	B	C	D	E	F
A	0	15	11	3	5	10
B		0	4	12	11	9
C			0	8	7	6
D				0	2	13
E					0	13
F						0

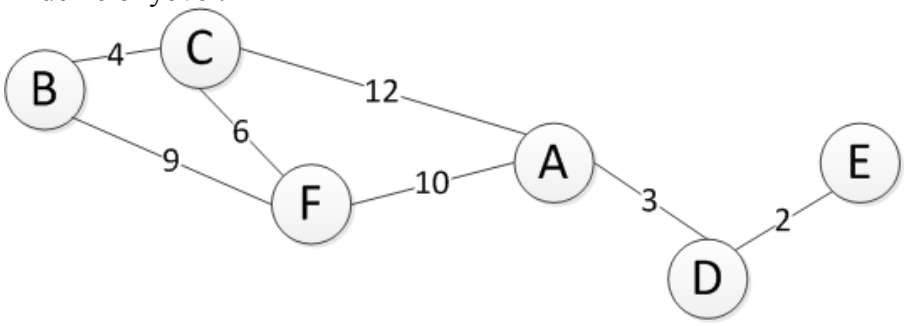
2. táblázat: G gráf legrövidebb út metrikája

Legyenek $k = 2$ mellett a landmark halmazok $A_0 = \{A, B, C, D, E, F\}$, $A_1 = \{A, F\}, A_2 = \emptyset$. Ekkor a bunch-ok: $B(A) = \{A, F\}, B(B) = \{A, C, F\}, B(C) = \{A, B, F\}, B(D) = \{A, E, F\}, B(E) = \{A, D, F\}, B(F) = \{A, F\}$. Az ezek alapján létrejövő G' feszítő részgráf a 3. ábra szerint néz ki.



3. ábra: G gráf G' feszítő részgráfja

A G' gráfot például a 4. ábrán szereplő módon lehet 2 dimenzióba beágyazni. Az így kapott beágyazás *2k-1* stretch-ű az eredeti G gráfra nézve, és rendelkezik Thorup és Zwick konstrukciójának minden előnyével.



4. ábra: G' beágyazása 2 dimenzióba

Policy alapú útvonal-választás (policy routing)

Az eddigi munkám továbbgondolására a policy alapú kompakt útvonal-választás irányát választottam. Egy $\mathcal{A}$ útvonalválasztó algebrát az elméleti bevezető alapján az $\mathcal{A} = (W, \varphi, \oplus, \preccurlyeq)$ jelöléssel lehet megadni. Az eddig ismertetett algoritmusok tehát kizárólag a $\mathcal{S} = (\mathbb{R}+, \infty, +, \leq)$ legrövidebb út algebrára teljesülnek. Érdekes és több helyen előforduló további algebrák például a legszélesebb út, vagy a legmegbízhatóbb út algebrái. Előbbit a $\mathcal{W} = (\mathbb{R}+, 0, min, \geq)$ definícióval, utóbbit pedig a $\mathcal{R} = ((0,1], 0, *, \geq)$ definícióval lehet megadni. Összetett algebrák közül az érdekes tulajdonságokkal bíró „shortest-widest path", azaz a legszélesebb utak közüli legrövidebb utak algebráját érdemes vizsgálni, mely a $\mathcal{SW} = (\mathbb{R}^+, \varphi, \oplus, \preccurlyeq)$ leírással adható meg. Itt az élekhez rendelhető súlyok halmaza $\mathbb{R}^+$, melyből (w, s) módon lehet súlyt képezni, $\varphi = (0, \infty)$, $(w_1, s_1) \oplus (w_2, s_2) = (min(w_1, w_2), s_1 + s_2)$, valamint $(w_1, s_1) \preccurlyeq (w_2, s_2) = \begin{cases} s_1 \leq s_2, ha\ w_1 = w_2 \\ w_1 \geq w_2, egyébként \end{cases}$.

A legszélesebb út algebráról belátható, hogy jól tömöríthető, mivel mindig reprezentálható egy fában, melynek tárolási költsége $O(log\ n)$ [12] Itt tehát olyan multiplikatív jellegű stretch-ről nincs értelme beszélni, mint a legrövidebb útnál, ugyanis a további tömöríthetőség már nem valószínű. Rétváriék [15] azt is belátják, hogy az általánosított stretch fogalmát a legszélesebb utakon csak úgy lehet értelmezni, hogy minden k-stretch út végül 1-stretch út lesz, mivel a min művelet k-szor egymás utáni végrehajtása megegyezik a min művelettel.

Legmegbízhatóbb út (most reliable path)

A legmegbízhatóbb út Rétváriék vizsgálatai [15] alapján nagyon hasonló a legrövidebb út algebrához, mivel ugyanazokkal a szigorúan monoton és izotón tulajdonságokkal rendelkezik. A monotonitás itt azt jelenti, hogy nincs negatív élsúly, azaz új éleknek az úthoz vétele nem javíthatja annak jóságát, szigorúan monoton ennek megfelelően az, ha csak ronthatja az új él az út jóságát. Az izotón tulajdonság azt jelenti, hogy ha egy útvonal lokálisan preferált egy másik útvonalhoz képest, akkor ugyanazon további élek hozzá vétele mindkét útvonalhoz ezt a relációt nem változtatja meg.

Tekintsük az 1. ábra absztrakcióját a következő módon: legyenek most a *v*-hez „közeli" pontok a *v*-vel legmegbízhatóbb éleken összekötött pontok. Ekkor *v* bunch-ába bekerülnek a korábbihoz hasonló módon néhányan a legmegbízhatóbb pontok közül, illetve néhányan a „távolabbi" körökből. A legrövidebb utaknál ismertetett lekérdező algoritmussal analóg módon lehet itt is megtalálni azt a pontot, mely a forrás és a cél bunch-ában is benne van.

$algorithm\ dist_R(u, v)$
$w := u;\ i := 0$
$while\ w \notin B(v)$
  $i := i + 1$
  $(u, v) := (v, u)$
  $w := p_i(u)$
$return\ \delta_R(w, u) * \delta_R(w, v)$

A *2k-1*-es multiplikatív stretch a $*$ műveletetre értelmezve azt jelenti, hogy egy $\delta(u, v)$ legmegbízhatóbb út esetén a mi $dist_k$ algoritmusunk legrosszabb esetben $\delta(u, v)^{2k-1}$ megbízhatóságú utat ad. Ezt bizonyítani a Thorup, Zwick [12] 10. oldalának alján található bizonyítással analóg módon lehet.

1. TÉTEL: $dist_R(u, v) \geq \delta_R(u, v)^{2k-1}$, ha δ_R az $\mathcal{R} = ((0,1], 0, *, \geq)$ algebra felett értelmezett legmegbízhatóbb út két pont között, $dist_R$ pedig a Thorup-Zwick módszer $\mathcal{R}$ -re való alkalmazása

BIZONYÍTÁS: Az algoritmus során a két pont cserélgetése nem változtatja az azok közötti út megbízhatóságát, melyet jelöljön $\Delta = \delta_R(u, v)$. Azt szeretnénk belátni, hogy minden iteráció során legrosszabb esetben Δ-szorosára csökken $\delta_R(w, u)$. Mivel $k-1$ iteráció lesz, az algoritmus végén $\delta_R(w, u) \geq \Delta^{k-1}$ teljesül. A monotonitás miatt $\delta_R(w, v) \geq \delta_R(w, u) * \delta_R(u, v) \geq \Delta^{k-1} * \Delta \geq \Delta^k$, tehát az algoritmus végén $\delta_R(w, u) * \delta_R(w, v) \geq \Delta^{k-1} * \Delta^k \geq \Delta^{2k-1}$.

Jelölje u_i, v_i, w_i az algoritmus *u,v,w* változóinak adott *i* melletti értékét. A ciklus kezdetén $w_0 = u_0$, így $\delta_R(w_0, u_0) = 1$. Lássuk be, hogy $\delta_R(w_i, u_i) \geq \delta_R(w_{i-1}, u_{i-1}) * \Delta$, amikor a ciklus *i*-edik iterációjánál tartunk. Az *i*-edik iterációból következik, hogy $w_{i-1} \notin B(v_{i-1})$, ami akkor és csak akkor lehet, ha $\delta_R(w_{i-1}, v_{i-1}) \leq \delta_R(A_i, v_{i-1}) = \delta_R(p_i(v_{i-1}), v_{i-1})$. A cserélgetés miatt $v_{i-1} = u_i$ és $w_i = p_i(u_i)$. Az előbbiekből levezethető, hogy:

$\delta_R(w_i, u_i) = \delta_R(p_i(u_i), u_i) = \delta_R(p_i(v_{i-1}), v_{i-1}) \geq \delta_R(w_{i-1}, v_{i-1}) \geq \delta_R(w_{i-1}, u_{i-1}) * \Delta$, amivel beláttuk a tételt. □

Tehát felhasználva Thorup és Zwick módszereit meg lehet adni egy olyan adatszerkezetet, melyen *2k-1*-es stretch-ű utak adhatók meg a legmegbízhatóbb útra nézve. Tulajdonságait tekintve megegyezik az eredetivel, tehát $O(kn^{1+1/k})$ memória szükséges hozzá, a feldolgozás $O(kmn^{1/k})$ ideig tart és a lekérdezés *O(k)* időben megvalósítható.

A legrövidebb legszélesebb utak (shortest-widest paths)

A komplex útvonalválasztó algebrák közül az egyik legérdekesebb ez, mivel az izotón tulajdonság nem teljesül rá [15]. Ez azt jelenti, hogy hiába van egy optimális út A-ból B-be és egy optimális B-ből C-be, van olyan gráf, hogy A-ból C-be a két út egymásutánja nem optimális. Rétvári és társai [15] 4. tétele kimondja és megmutatja, hogy vannak olyan nem izotón gráfok, melyeken nem lehet lineáris alatti memóriaigényt elérni semmilyen k-stretch-re. Ezért általános esetre a gráfok ezen osztályára hatékony algoritmust adni nem tudtam.

Összefoglalás

- Megismertem az internet útvonal-választás alapvető hiányosságait
- Elolvastam azokat az alap eredményeket, melyek alsó korlátokat adnak a különböző útvonal-választások memóriaigényére
- A kompakt útvonal-választásban elért eredményeket megismertem
- Részletesen tanulmányoztam Zwick és Thorup módszereit
- Megismertem a mohó beágyazással kapcsolatosan általánosan felmerült problémákat
- Felvetettem a két témakör összekapcsolását
- Megismertem a policy routing témakörét
- Átalakítottam Zwick és Thorup módszerét legmegbízhatóbb út algebrára
- Megismertem a shortest-widest útvonalválasztáson keresztül a stretch-csel elérhető memóriacsökkenés korlátait

Irodalomjegyzék

[1] **Geoff Huston**: BGP Table Data, http://bgp.potaroo.net/ [Letöltve: 2012.02.29]

[2] **R. Gummadi, R. Govindan, N. Kothari, B. Karp, Y.-J. Kim, S. Shenker**: „Reduced state routing in the Internet", In: *Proceedings of the Third ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets-III)* San Diego, CA, USA, 2004.

[3] **L. Subramanian, M. Caesar, C. T. Ee, M. Handley, M. Mao, S. Shenker, I. Stoica**: „HLP: A next generation inter-domain routing protocol", In: *ACM SIGCOMM 2005*, pp. 13-24, Philadelphia, PA, USA, 2005.

[4] **M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, I. Stoica, S. Shenker**: „ROFL: Routing on flat labels", In: *ACM SIGCOMM 2006*, pp. 363-374, Pisa, Italy, 2006.

[5] **C. Gavoille, S. Pérennès**: „Memory requirement for routing in distributed networks", In: *Proceedings of the 15th Annual ACM Symposium on Principles of Distributed Computing*, pp. 125-133, Philadelphia, PA, USA, 1996.

[6] **D. Peleg, A. Schäffer**: „Graph spanners", In: *Journal of Graph Theory 13*, pp. 99-116, 1989.

[7] **P. Erdős**: „Extremal problems in graph theory", In: *Theory of Graphs and Its Applications,* Published: House Czechoslovak Academy of Science, Prague, pp. 29-36, 1963.

[8] **R. Kleinberg**: „Geographic routing using hyperbolic space", In: *Proceedings of the 32nd IEEE INFOCOM (INFOCOM 2007)*, pp. 1902-1909, 2007.

[9] **P. Maymounkov**: „Greedy Embeddings, Trees, and Euclidean vs. Lobachevsky Geometry", Technical Report, 2006 [Letöltve: 2012.03.21, http://pdos.csail.mit.edu/~petar/pubs.html]

[10] **R. Flury, S.V. Pemmaraju, R. Wattenhofer**: „Greedy Routing with Bounded Stretch", In: *Proceedings of the 34th IEEE INFOCOM (INFOCOM 2009)*, pp. 1737-1745, 2009.

[11] **A. Cvetkovski, M. Crovella**: „Low-Stretch Greedy Embedding Heuristics", *Fourth International Workshop on Network Science for Communication Networks (NetSciCom 2012)*, 2012.

[12] **M. Thorup, U. Zwick**: „Approximate distance oracles", In: *Proceedings of the 33th Annual ACM Symposium on Theory of Computing*, pp. 183-192, Crete, Greece, 2001.

[13] **M. Thorup, U. Zwick**: „Compact routing schemes", In: *Proceedings of the 13th annual ACM symposium on Parallel algorithms and architectures*, pp. 1-10, Crete, Greece, 2001.

[14] **M. Pătraşcu, L. Roditty:** „Distance Oracles Beyond the Thorup--Zwick Bound", In: *51st IEEE Symposium on Foundations of Computer Science (FOCS 2010)*, 2010.

[15] **G. Rétvári, A. Gulyás, Z. Heszberger, M. Csernai, J. J. Bíró**: „Compact policy routing", In: *Proceedings of the 30th Annual ACM Symposium on Principles of Distributed Computing*, pp. 149-158, San José, CA, USA, 2011.

[16] **J. Sobrinho**: „Algebra and algorithms for QoS path computation and hop-by-hop routing in the Internet".IEEE/ACM Transactions on Networking, Vol. 10, pp. 541-550, 2002.

[17] **D. Dor, S. Halperin, U. Zwick**: „All pairs almost shortest paths” In: *SIAM Journal on Computing, Vol. 29*, pp. 1740-1759, 2000.
[18] **B. Awerbuch, B. Berger, L. Cowen, D. Peleg**: „Near-linear time construction of sparse neighborhood covers”, In: *SIAM Journal on Computing, Vol. 28*, pp. 263-277, 1999.
[19] **E. Cohen**: „Fast algorithms for constructing t -spanners and paths with stretch t” In: *SIAM Journal on Computing, Vol. 28*, pp. 210-236, 1999.
[20] **J. Matoušek**: „On the distortion required for embedding finite metric spaces into normed spaces”, In: *Israel Journal of Mathematics, Vol. 93*, pp. 333-344, 1996.
[21] **L. Cowen**: „Compact routing with minimum stretch”, In: *Journal of Algorithms, Special issue for SODA’99*, pp. 170-183, 2001.

Függelék

A következőkben Thorup és Zwick [12]-ben bemutatott algoritmusainak általam implementált C++ kódú változatának vázlatát közlöm.

```
class Vertex {
      char* label; // pont egyedi elnevezése
      Edge* list; // éllista
};
class Edge {
      Vertex* a; // pontjaira mutatók
      Vertex* b; //
      double weight; // súly
};
class Graph {
      Vertex* V; // ponttömb
      Edge* E; // éltömb
      int n; // pontok száma
};
class DistanceOracle {
      int k; // stretch paramétere
      int n; // gráf pontjainak száma
      double INF=99999999; // "végtelen"
      Vertex** A; // landmark halmazok
      double** dist; // távolságok
      Vertex** p; // tanúk
      Vertex* select(Vertex* A) { // valószínűségi kiválasztás
            Vertex* selected;
            int i=0, j=0;
            double prob = pow(n,-1/k);
            while(A[i]!=NULL) {
                  srand(time(NULL));
                  if((rand()%10000)<prob*10000) {
                        selected[j]=A[i];
                        j++;
                  }
                  i++;
            }
            return selected;
      }
      double d(Vertex u, Vertex v) {
            // Dijsktra algoritmusával számolt legrövidebb út hossza
```

```
	}
	double d(Vertex* A, Vertex v, Vertex& p) {
		double d_ret=INF;
		p=v;
		if(A != NULL) {
			int i=0;
			while(A[i]!=NULL && d_ret>0) {
				double d_tmp=d(A[i],v);
				if(d_tmp<d_ret) {
					d_ret=d_tmp;
					p=A[i];
				}
				i++;
			}
		}
		return d_ret;
	}
	void prePro(Graph G) { // előfeldolgozás
		A[0]=G.V;
		A[k]=NULL;
		n = G.n;
		for(int i=1;i<k;i++) {
			A[i]=select(A[i-1]); // "let Ai contain each element
			// of Ai-1 independently, with probability n^-1/k"
		}
		for(int i=k-1;i>=0;i--) {
			for(int j=0;j<n;j++) { // "for every v e V"
				dist[i][j] = d(A[i],G.V[j],p[i][j]);
				if(dist[i][j] = dist[i+1][j]) {
					p[i][j]=p[i+1][j];
				}
			}
			int j=0,k=0;
			while(A[i][j]!=NULL) { // "for every w e Ai - Ai+1"
				int flag=0;
				while(A[i+1][k]!=NULL) {
					if((A[i][j]).label==(A[i+1][k]).label) {
						flag=1;
					}
					k++;
				}
				if(flag==1) {
					growtree(); // fa felépítése
				}
				j++;
			}
		}
	}
};
```