Eötvös Loránd Tudományegyetem

Természettudományi Kar

Számítógéptudományi tanszék

Virtuális magánhálózatok tervezése

Témavezető:
Király Zoltán
egyetemi docens

Szerző:
Kovács Alex
Matematika BSc

Budapest, 2025

Köszönetnyilvánítás

Mindenekelőtt szeretném kifejezni őszinte hálámat témavezetőmnek, Király Zoltánnak az egész félév során nyújtott türelméért és támogatásáért.

Rendkívül hálás vagyok továbbá a családomnak a sokéves kitartó támogatásért, és végül, de nem utolsósorban a barátaimnak is, akik mindig mellettem álltak, és segítettek átvészelni a nehéz pillanatokat.

Tartalomjegyzék

1. fejezet

Bevezetés, definíciók[1]

1. Definíció (Virtual private network design (VPND) feladat)**.** Adott:

- $G = (V, E)$ összefüggő egyszerű gráf
- $c : E \to \mathbb{R}_{\geq 0}$ nemnegatív élköltségfüggvény
- $T \subseteq V$ fontos csúcsok halmaza, amelyeket **terminálok**nak nevezünk
- $b_{out}, b_{in} : T \to \mathbb{Z}_{\geq 0}$ a terminálok fel-, illetve letöltési korlátai

A feladat találni egy $u : E \to \mathbb{R}_{\geq 0}$ kapacitásfüggvényt, valamint minden $(i, j) \in T \times T$ rendezett terminálpárra kijelölni egy i és j közötti P_{ij} utat úgy, hogy:

- minden $D \in \mathbb{Q}_{\geq 0}^{T \times T}$ **érvényes forgalmi mátrix**ra teljesül, hogy az összes igény egyszerre kielégíthető a kijelölt utakon keresztül, azaz $D(i, j)$ egységnyi forgalmat hozzáadva P_{ij} éleihez $(\forall (i, j) \in T \times T)$ nem lépjük túl az élkapacitásokat
- $\sum_{e \in E} u(e)\ c(e)$ minimális

2. Definíció. $D \in \mathbb{Q}_{\geq 0}^{T \times T}$ **érvényes** forgalmi mátrixnak nevezzük (a b_{in} és b_{out} korlátokra nézve), ha

$$\forall i \in T : \sum_{j \in T, j \neq i} D(i, j) \leq b_{out}(i), \sum_{j \in T, j \neq i} D(j, i) \leq b_{in}(i),$$

azaz tiszteletben tartja a terminálok fel- és letöltési korlátait.

1. Észrevétel. *A VPND probléma NP-nehéz. [11]*

[1] Ez a fejezet a [6] cikk alapján készült

Bizonyítás. Könnyű látni, hogy a közismerten NP-nehéz minimális költségű Steiner-fa feladat visszavezethető a VPND-re a következőképpen:

Legyen a Steiner-fa feladat egy tetszőleges v terminájára $b_{in}(v) = 0$ és $b_{out}(v) = 1$, valamint $\forall u \neq v$ terminálra $b_{in}(u) = 1$ és $b_{out}(u) = 0$. Ekkor a minimális költségű Steiner-fa éleihez 1 kapacitást, többihez 0 kapacitást rendelve optimális megoldást kapunk a VPND probléma ezen példányára.

□

1.1. Eddigi eredmények

A VPND probléma számos különböző változatát tanulmányozták már, ezekből néhányban jelentős eredményeket értek el.

3. Definíció. A továbbiakban nevezzük **fa megoldás**nak az $u : E \to \mathbb{R}_{\geq 0}$ kapacitásfüggvényt, ha a pozitív kapacitású élek fát alkotnak. Az általános esetet nevezzük **gráf megoldás**nak.

4. Definíció (**szimmetrikus korlátok**). $\forall v \in T$ terminálra $b_{in}(v)$ és $b_{out}(v)$ értékek helyett egy $b(v) \in \mathbb{Z}_{\geq 0}$ érték adott, ami a terminál összforgalmára (bemenő + kimenő) felső korlát.

A [11] cikkben a probléma következő verzióit tanulmányozzák:

SymT (symmetric tree): szimmetrikus korlátok, opt. fa megoldást keresünk
Megmutatják, hogy polinomiális futási időben megoldható.

SymG (symmetric graph): szimmetrikus korlátok, opt. gráf megoldást keresünk
Belátják, hogy SymT optimuma $2(1 - \frac{1}{|T|})$-közelítést ad erre a verzióra.
Innen következik, hogy SymG polinomiális futási időben 2-közelíthető.

AsymT (asymmetric tree): b_{in} és b_{out} korlátok, opt. fa megoldást keresünk
Belátják, hogy NP-nehéz, emellett adnak rá egy 9.002-közelítő algoritmust.

AsymG (asymmetric graph): b_{in} és b_{out} korlátok, opt. gráf megoldást keresünk
Belátják, hogy NP-nehéz (1. észrevétel), a közelítő algoritmus kérdését viszont nyitva hagyják.

Ezek mellett még a "balanced" verzió is említésre méltó:

BalT (balanced tree): $\forall v \in T : b_{in}(v) = b_{out}(v)$, opt. fa megoldást keresünk
Polinomiális futási időben megoldható [14].

BalG (balanced graph): $\forall v \in T : b_{in}(v) = b_{out}(v)$, opt. gráf megoldást keresünk
AsymG speciális esete. A 3.1 alfejezetben mutatunk rá 2-közelítő algoritmust.

Az úgynevezett VPN-sejtés (eredetileg VPN conjecture), miszerint SymG-re mindig létezik optimális fa megoldás, hosszú ideig nyitott kérdés volt. Ezzel együtt SymG NP-nehézsége is ismeretlen volt. A sejtést végül a [10] cikkben igazolták. Ebből következik hogy a [11] cikkben bemutatott algoritmus valójában pontos megoldást ad SymG-re, nem csak közelítést. Így ma már tudjuk, hogy SymG polinomiális futási időben megoldható.

A [12] cikkben többek között megfogalmaznak egy randomizált algoritmust az AsymG verzióra, $5,55$-ös várható közelítési aránnyal, ami jelentős előrelépés a korábbi $O(\log n \log\log n)$-es eredményekhez képest. A [6] cikkben ezt az arányt $3,55$-re javítják.

A továbbiakban a fejezet elején megfogalmazott verzióra, azaz az AsymG-re fogunk VPND problémaként hivatkozni, és ezt tárgyaljuk részletesebben ebben a szakdolgozatban. Megpróbálunk minél jobb közelítési arányú, polinomiális futási idejű algoritmusokat adni, majd ezeket a gyakorlatban is leimplementáljuk és teszteljük különféle inputokon.

1.2. Észrevételek

A [12] cikket követve tehetünk néhány feltételezést, amelyek jelentősen leegyszerűsítik a problémát.

Minden v terminált képzeletben felbonthatunk $b_{out}(v)$ darab **küldő** (*sender*), és $b_{in}(v)$ darab **vevő** (*receiver*) terminálra. j terminált akkor nevezzük küldőnek, ha $b_{in}(j) = 0$ és $b_{out}(j) = 1$, vevőnek pedig akkor, ha $b_{in}(j) = 0$ és $b_{out}(j) = 1$.

Így az általánosság megszorítása nélkül feltételezhetjük, hogy minden terminál vagy küldő, vagy vevő. Ehhez elegendő azt kikötnünk, hogy az eredeti gráfbeli u és v terminálok másolatai között ugyanazokat az utakat jelöljük ki ($\forall (u, v) \in T \times T$).

Legyen mostantól $\mathcal{S}$ a küldők, $\mathcal{R}$ a vevők halmaza, továbbá legyen $R = |\mathcal{R}|$ és $S = |\mathcal{S}|$. A szakdolgozat további részében feltételezzük, hogy $R \geq S$. Szimmetria miatt ezt megtehetjük az általánosság megszorítása nélkül.

Legyen $B = (\mathcal{S} \cup \mathcal{R}, E^B)$ teljes páros gráf. Ekkor az eredeti gráfbeli élekhez kell kapacitásokat rendelnünk, és a terminálpárok közötti utak $\mathcal{P}$ halmazát kijelölnünk úgy, hogy $\mathcal{P}$-ben B minden sr éléhez legyen egy P_{sr} út, és B minden párosítását ezeken az utakon irányítva, minden élen legyen elég kapacitás.

Más szóval, az eredeti gráf minden e élére teljesül a következő:

$$|\{P_{rs} \in \mathcal{P} |\ e \in P_{rs}, rs \in M\}| \leq u(e), B\text{-nek minden } M \text{ párosítására.} \tag{1.1}$$

Megjegyzés. Vegyük észre, hogy a $\mathcal{P}$ utak egyértelműen meghatározzák a hozzájuk tartozó minimális u-t. Tehát elegendő megkeresnünk azokat az utakat, amelyek a minimális költségű kapacitásfüggvényt adják.

5. Definíció (Élsúlyozott egyszerű gráf metrikus lezártja)**.** Az eredeti gráf csúcshalmazán értelmezett teljes gráf, amelynek minden éléhez a két végpontjának eredeti gráfbeli távolságát rendeljük költségként. Ha két csúcs között nincs út az eredeti gráfban, akkor a távolságukat $+\infty$-nek definiáljuk (a mi esetünkben ez nem fog előfordulni, mivel összefüggő gráfokkal dolgozunk).

Könnyű látni, hogy G metrikus lezártjában a küldők és a vevők közötti bármely párosítás összköltsége alsó korlát az optimumra. Jelölje $\ell(u, v)$ az u és v csúcsok távolságát G-ben. B éleihez $\ell(r, s)$ költségeket rendelve azt kapjuk, hogy bármely B-beli párosítás összköltsége alsó korlát.

1. Lemma ([12])**.** *Legyen $B(\mathcal{S} \cup \mathcal{R}, E_B)$ a küldőkön és vevőkön értelmezett teljes páros gráf, az élköltségek ($\ell : E_B \to \mathbb{R}_{\geq 0}$) legyenek a G-beli távolságok a c élköltségfüggvénnyel. Ekkor bármely B-beli párosítás összköltsége alsó korlát a VPND feladat optimumára.*

2. fejezet

Alsó korlát az optimumra[1]

Ebben a fejezetben mutatunk egy alsó becslést a VPND feladat optimumértékére, javítva az 1. lemmából kapott eredményen. Bebizonyítjuk, hogy az $\mathcal{S} \cup \mathcal{R}'$-n ℓ költségfüggvénnyel értelmezett **teljes** (nem feltétlenül páros) gráf bármely párosításának összköltsége alsó becslést ad, ahol $\mathcal{R}'$ egy tetszőleges S elemű részhalmaza $\mathcal{R}$-nek. Mivel ebbe beletartoznak az $\mathcal{S} \cup \mathcal{R}$ páros gráf párosításai is, ezzel javítunk az előző fejezetben kapott eredményen. A bizonyításhoz Hu 2-termékes folyam tételét [13] fogjuk felhasználni.

1. Tétel (Hu 2-termékes folyam tétele [13])**.** *Adott $G = (V, E)$ egyszerű gráf, $u : E \to \mathbb{R}_{\geq 0}$ kapacitásfüggvény, $(s_1, r_1), (s_2, r_2)$ csúcspárok, valamint $d_1, d_2 \in \mathbb{R}_{\geq 0}$ igények. Pontosan akkor létezik (tört) 2-termékes folyam d_1 és d_2 értékekkel, ha minden $U \subseteq V$ részhalmazra teljesül a **vágási feltétel**, azaz $u(\delta(U)) \geq d_1\chi_1(U) + d_2\chi_2(U)$. Itt $\delta(U)$ az U által meghatározott vágás éleit jelöli, és $i \in 1, 2$-re $\chi_i(U) = 1$, ha a $\delta(U)$ vágás elválasztja s_i-t és r_i-t (vagyis a vágás különböző oldalán vannak), különben $\chi_i(U) = 0$.*

1. Következmény. *Adott $G = (V, E)$ egyszerű gráf, $u : E \to \mathbb{R}_{\geq 0}$ kapacitásfüggvény és $(s_1, s_2), (r_1, r_2) \in V$ csúcspárok. Legyen minden igény 1. Ha létezik 2-termékes folyam az $(s_1, r_1), (s_2, r_2)$ és az $(s_1, r_2), (s_2, r_1)$ párokra, akkor létezik 2-termékes folyam az $(s_1, s_2), (r_1, r_2)$ párokra is.*

Bizonyítás. Ha minden igény 1, a vágási feltétel ekvivalens azzal, hogy $u(\delta(U))$ legalább akkora, mint a $\delta(U)$ vágás által szétválasztott párok száma.

[1] Ez a fejezet részben a [6] cikk alapján készült

Hu tétele miatt elegendő megmutatni, hogy ha a vágási feltétel teljesül az $(s_1, r_1), (s_2, r_2)$ és az $(s_1, r_2), (s_2, r_1)$ párokra, akkor teljesül az $(s_1, s_2), (r_1, r_2)$ párokra is. Vegyünk ehhez egy tetszőleges $U \subseteq V$ halmazt.

1. Ha a $\delta(U)$ vágás nem választja szét sem az (s_1, s_2), sem az (r_1, r_2) párt:
   Az egyenlőtlenség jobb oldala 0, így a feltétel triviálisan teljesül.

2. Ha pontosan az egyik párt választja szét:
   Az általánosság megszorítása nélkül feltételezhetjük, hogy (s_1, s_2)-t választja szét, (r_1, r_2) eset hasonló.
   Ekkor a vágás szétválasztja vagy az (s_1, r_1), vagy az (s_2, r_1) párt (különben s_1 és s_2 a vágásnak ugyanazon az oldalán lennének). Tehát ebben az esetben $u(\delta(U)) \geq 1$.

3. Ha mindkét párt szétválasztja: A vágás szétválasztja vagy az (s_1, r_1) és (s_2, r_2), vagy az (s_1, r_2) és (s_2, r_1) párokat, így $u(\delta(U)) \geq 2$. Tehát ebben az esetben is teljesül a feltételünk.

Ezzel minden esetre beláttuk, hogy teljesül a vágási feltétel.

□

Megjegyzés. Ha megkövetelünk egész értékű folyamokat, ez már nem feltétlenül igaz. Tudjuk, hogy egész értékű 2-termékes folyam keresése NP-nehéz [8]. Hu eredményéből viszont következik, hogy ilyen esetben mindig létezik félegész (egész szám fele) értékű folyam.

A következő lemma azt mondja ki, hogy az $\mathcal{S}$ és $\mathcal{R}$ k darab részhalmazra való particionálása, és az így kapott k részfeladat optimumértékeinek összege alsó korlátot ad az eredeti feladat optimumára.

2. Lemma. *Legyen $S_1, S_2, \ldots, S_k$ $\mathcal{S}$-nek egy partíciója, és $R_1, R_2, \ldots, R_k$ $\mathcal{R}$-nek egy partíciója. Legyen I_i a VPND feladat példánya a G gráfon S_i küldőkkel és R_i vevőkkel, valamint legyen OPT_i az I_i példány optimumértéke. Ekkor:*

$$\sum_{i=1}^{k} OPT_i \leq OPT.$$

Bizonyítás. Legyen $\mathcal{P}$ az eredeti feladat optimális úthalmaza, u pedig a hozzá tartozó kapacitásfüggvény. Legyen $P_i \subseteq \mathcal{P}$ azon utak halmaza, amelyeknek mindkét

végpontja $S_i \cup R_i$-ben van. Könnyű látni, hogy P_i optimális megoldást ad az I_i részfeladatra (mert különben $\mathcal{P}$ sem lenne optimális). Legyen $u_i : E \to \mathbb{R}_{\geq 0}$ a P_i-hez tartozó kapacitásfüggvény. Elegendő megmutatni, hogy $u(e) \geq \sum_{i=1}^{k} u_i(e)$ minden $e \in E$ élre. (1.1)-ből következik, hogy minden $i \in \{1, 2, \ldots, k\}$-ra:

$$u_i(e) = \max_{M_i} |\{P_{rs} \in P_i \mid e \in P_{rs} \text{ és } rs \in M_i\}|,$$

ahol M_i végigmegy az összes S_i és R_i közötti párosításon.

Jelölje $\tilde{M}_i$ azt a párosítást, amelyen a maximum felvétetik. Ekkor az $\tilde{M} := \bigcup_{i=1}^{k} \tilde{M}_i$ diszjunkt unió egy párosítás az $\mathcal{S} \cup \mathcal{R}$ páros gráfon.

Így az (1.1)-ből következik, hogy:

$$u(e) \geq |\{P_{rs} \in \mathcal{P} \mid e \in P_{rs} \text{ és } rs \in \tilde{M}\}| = \sum_{i=1}^{k} |\{P_{rs} \in P_i \mid e \in P_{rs} \text{ és } rs \in \tilde{M}_i\}| = \sum_{i=1}^{k} u_i(e)$$

minden $e \in E$-re. Ezzel az állítást beláttuk. □

Most már készen állunk a következő tétel bizonyítására:

2. Tétel. *Vegyünk egy S elemű $\mathcal{R}' \subseteq \mathcal{R}$ részhalmazt (létezik ilyen, mivel feltételeztük, hogy $R \geq S$). Legyen M egy párosítás az $\mathcal{S} \cup \mathcal{R}'$-n értelmezett teljes gráfon. Ekkor:*

$$\sum_{vw \in M} \ell(v, w) \leq OPT.$$

Bizonyítás. Legyen $\mathcal{S} = \{s_1, s_2, \ldots, s_S\}$ és $\mathcal{R}' = \{r_1, r_2, \ldots, r_S\}$. Triviális, hogy elegendő csak teljes párosításokra bizonyítani a tételt. Tegyük fel (az általánosság megszorítása nélkül), hogy a párosítás a következő élekből áll:

$$s_1 s_2, s_3 s_4, \ldots, s_{2k-1} s_{2k}$$

$$r_1 r_2, r_3 r_4, \ldots, r_{2k-1} r_{2k}$$

$$s_{2k+1} r_{2k+1}, s_{2k+2} r_{2k+2}, \ldots, s_S r_S.$$

Vegyük $\mathcal{S}$ és $\mathcal{R}'$ következő particionálását:

$$S_i = \{s_{2i-1}, s_{2i}\}, R'_i = \{r_{2i-1}, r_{2i}\}, \quad 1 \leq i \leq k,$$

$$S_i = \{s_i\}, R'_i = \{r_i\}, \quad 2k+1 \leq i \leq S.$$

Legyen I_i a VPND feladat példánya S_i küldőkkel és R'_i vevőkkel, valamint legyen OPT_i az I_i optimumértéke. A 2. lemma miatt elegendő azt bizonyítani, hogy $\ell(s_{2i-1}, s_{2i}) + \ell(r_{2i-1}, r_{2i}) \leq \mathrm{OPT}_i, \forall i \in \{1, 2, \ldots, k\}$.

Az I_i részfeladat optimális megoldása egy olyan kapacitásfüggvény, amely megenged 2-termékes folyamot az $(s_{2i-1}, r_{2i-1}), (s_{2i}, r_{2i})$ és az $(s_{2i-1}, r_{2i}), (s_{2i}, r_{2i-1})$ párokra. Az 1. következmény szerint ez a kapacitásfüggvény az $(s_{2i-1}, s_{2i}), (r_{2i-1}, r_{2i})$ párokra is megenged 2-termékes folyamot. Tehát OPT_i legalább $\ell(s_{2i-1}, s_{2i}) + \ell(r_{2i-1}, r_{2i})$, és ezzel készen vagyunk a bizonyítással. □

2.1. Algoritmus az alsó korlát kiszámítására

A gyakorlatban nem éri meg a maximális összsúlyú $\mathcal{S} \cup \mathcal{R}'$ párosítást megkeresni, mivel ehhez $\mathcal{R}$-nek $\binom{R}{S}$ darab részhalmazát kellene végignéznünk. Helyette megfelel az is, ha először $\mathcal{R} \cup \mathcal{S}$-ben keressük meg a maximális súlyú párosítást, majd ebből töröljük a legkisebb $\mathcal{R}$-$\mathcal{R}$ éleket, pontosan annyit darabot, hogy ugyanannyi $\mathcal{R}$-$\mathcal{R}$ él legyen, mint $\mathcal{S}$-$\mathcal{S}$ él (mivel $R \geq S$, $\mathcal{R}$-$\mathcal{R}$ élből legalább annyit választottunk ki, mint $\mathcal{S}$-$\mathcal{S}$ élből). Így kapunk egy teljes párosítást $\mathcal{S} \cup \mathcal{R}'$-n, ahol $\mathcal{R}'$ egy S elemű részhalmaza $\mathcal{R}$-nek.

Ezzel a következő problémába ütközünk: ha $\mathcal{R} \cup \mathcal{S}$-en futtatunk egy maximális súlyú párosítás algoritmust (például Edmonds [5]), a kapott futási idő nem az input méretében, hanem $R + S$-ben lesz polinomiális. Mi viszont **erősen** polinomiális futási idejű algoritmust szeretnénk. Ehhez mindenképp el kell kerülnünk a terminálok szétbontását küldőkre és vevőkre.

6. Definíció (Max b-matching feladat)**.** Adott $G(V, E)$ egyszerű gráf a $c : E \to \mathbb{R}_{\geq 0}$ élsúlyozással, továbbá adott minden v csúcsra egy $b(v)$ korlát, $b : V \to \mathbb{Z}_{\geq 0}$. b-matchingnek nevezzük azt az $x : E \to \mathbb{Z}_{\geq 0}$ nemnegatív egész élsúlyozást, amelyre $\forall v \in V : \sum_{uv \in E} x(uv) \leq b(v)$. Keressük azt az x b-matchinget, amelyre $\sum_{e \in E} c(e)x(e)$ maximális. Azaz úgy szeretnénk kiválasztani éleket (ugyanazt az élt többször is kiválaszthatjuk) maximális összsúllyal, hogy minden v csúcsnak legfeljebb $b(v)$ éle legyen.

Vegyük észre, hogy az $\mathcal{R} \cup \mathcal{S}$ teljes gráfbeli maximális súlyú párosítás feladat visszavezethető max b-matchingre a metrikus lezárt T által feszített részgráfján, $b(v) = b_{in}(v) + b_{out}(v)$ korlátokkal.

Így sajnos nem tudjuk biztosan megmondani, hogy a kapott párosításban melyek voltak az $\mathcal{R}$-$\mathcal{R}$, illetve az $\mathcal{S}$-$\mathcal{S}$ élek, hogy törölhessük a megfelelő számú $\mathcal{R}$-$\mathcal{R}$ élt. Ezt a problémát úgy tudjuk megoldani, hogy minden terminált megduplázunk: legyenek v másolatai v_{in} és v_{out}. Az új gráfon legyenek a korlátok $b(v_{in}) = b_{in}(v)$, illetve $b(v_{out}) = b_{out}(v)$, valamint legyen az új élsúlyozás $\ell'(u^*v^*) = \ell(uv), \forall u^* \in \{u_{in}, u_{out}\}, v^* \in \{v_{in}, v_{out}\}$. Ezen a gráfon maximális b-matchinget keresve egy párosításbeli él mindkét végpontjáról biztosan meg tudjuk mondani, hogy küldő vagy vevő volt.

A max b-matching egy jól ismert, részletesen kutatott probléma, amelyre ismerünk polinomiális futási idejű algoritmusokat, például a [1] cikkben bemutatott algoritmus $O(|T|^4)$-ben adja meg ennek a $2|T|$ csúcsú teljes gráfnak a maximális b-matchingjét.

Algoritmus az alsó korlát kiszámítására: (pszeudokód)

$\forall i \in T$:
    Dijkstra(i) // a kapott $\ell(ij)$ távolságokat eltároljuk
T' és ℓ' megkonstruálása
$\forall i \in T$:
    $b(i_{in}) := b_{in}(i)$
    $b(i_{out}) := b_{out}(i)$
$M :=$ MaxBMatching(T', ℓ', b)
output: $\ell'(M)$

Aszimptotikus futási idő: $O(|T||E| \log |V| + |T|^4) = O(|T|(|E| \log |V| + |T|^3))$, a Dijkstra-algoritmus klasszikus bináris kupacos implementációját használva.

Megjegyzés. A Dijkstrát d-edfokú vagy Fibonacci-kupaccal implementálva ennél jobb aszimptotikus futási időt is el tudunk érni. Ezek azonban csak elméletben adnak jobb futási időt, a gyakorlatban sokkal lassabbak szoktak lenni a nagy konstans szorzók miatt [2]. Ezért a továbbiakban a bináris kupacos implementációt fogjuk használni.

3. fejezet

Közelítő algoritmusok

3.1. Determinisztikus algoritmus

Ötlet (1. algoritmus) [6]:

(1) Keressük meg a $v \in V$ csúcsot, amely minimalizálja a $\sum_{s\in\mathcal{S}} \ell(s,v) + \sum_{r\in\mathcal{R}} \ell(r,v)$ összeget.

(2) Minden $u \in \mathcal{R}\cup\mathcal{S}$-re az u és v közötti legrövidebb út mentén adjunk hozzá 1 kapacitást az élekhez.

Megjegyzés. Könnyű látni, hogy az algoritmus fa megoldást ad, mivel v-ből Dijkstra-algoritmust indítva kapunk egy v gyökerű fenyőt a legrövidebb utakból, és pontosan a fenyő élei fognak pozitív kapacitást kapni. $P_{i,j}$-nek a fabeli (egyértelmű) $i-j$ utat megválasztva látszik, hogy ez egy megengedett megoldás lesz.

Ismert, hogy ez az algoritmus optimális megoldást ad a SymT [11], illetve a BalT [14] verzióra. Továbbá azt is megmutatták, hogy SymG-re 2-közelítő [11].

Megjegyzés. Az 1. algoritmus "nem tesz különbséget" küldők és vevők között, csak a $b_{in}+b_{out}$ értékeket nézi. Innen látszik, hogy ha R sokkal nagyobb S-nél, nem mindig ad jó közelítést.

A következőkben az optimumértékre kapott alsó korlátot felhasználva belátjuk, hogy az 1. algoritmus $1+\frac{R}{S}$-közelítő.

Megjegyzés. Innen triviálisan következni fog, hogy BalG-re is 2-közelítő, mivel ebben a verzióban $R=S$.

A bizonyításhoz előbb az alábbi lemmára lesz szükségünk:

3. Lemma. [6]

$$\sum_{s\in\mathcal{S}}\sum_{r\in\mathcal{R}}\ell(s,r)\leq R\cdot OPT, \tag{2}$$

ahol OPT pedig a feladat optimumértéke.

Bizonyítás. Legyen $B(\mathcal{S}\cup\mathcal{R}, E^B)$ teljes páros gráf, $\ell : E^B \to \mathbb{R}_{\geq 0}$ élköltségfüggvénnyel. B éleit fel tudjuk osztani R db diszjunkt párosításra, legyen ezen párosítások halmaza $\mathcal{M}$. Ekkor az 1. lemma miatt $\forall M \in \mathcal{M}$ párosításra $\ell(M) \leq \mathrm{OPT}$, így:

$$\sum_{s\in\mathcal{S}}\sum_{r\in\mathcal{R}}\ell(s,r) = \sum_{M\in\mathcal{M}}\ell(M)\leq R\cdot\mathrm{OPT}$$

Ezzel a lemmát beláttuk. Készen állunk a közelítési arány bizonyítására. □

3. Tétel. [6] *Az 1. algoritmus* $1+\frac{R}{S}$*-közelítő.*

Bizonyítás. Legyen $G^m(\mathcal{R}\cup\mathcal{S}, E^m)$ $\mathcal{R}\cup\mathcal{S}$ metrikus lezártja, azaz a teljes gráf az $\mathcal{R}\cup\mathcal{S}$ csúcshalmazon, ℓ élköltségfüggvénnyel. Megmutatjuk, hogy létezik olyan $u\in\mathcal{R}\cup\mathcal{S}$, amelyre:

$$\sum_{v\in R\cup S}\ell(u,v)\leq(1+\frac{R}{S})\mathrm{OPT},$$

vagyis az u középpontú teljes csillaggráf összköltsége legfeljebb $(1+\frac{R}{S})\mathrm{OPT}$.

Ha $R=S$, akkor E^m éleit fel tudjuk osztani $2S-1$ diszjunkt teljes párosításra. A 2. tétel szerint minden párosítás összköltsége alsó korlátja OPT-nak. Mivel minden élt pontosan két csillaggráf tartalmaz:

$$\sum_{u\in\mathcal{R}\cup\mathcal{S}}\sum_{v\in\mathcal{R}\cup\mathcal{S}}\ell(u,v)=2\ell(E^m)\leq 2(2S-1)\mathrm{OPT}.$$

Így kell léteznie legalább egy csillaggráfnak, amelynek összköltsége legfeljebb $\frac{2(2S-1)\mathrm{OPT}}{2S} < 2\mathrm{OPT} = (1+\frac{R}{S})\mathrm{OPT}$.

A továbbiakban tegyük fel, hogy $R>S$. Jelölje $\mathcal{M}_S$ és $\mathcal{M}_R$ a legfeljebb $\lfloor\frac{S}{2}\rfloor$ elemű párosítások halmazát, amelyek csak küldőket, illetve csak vevőket tartalmaznak. A 2. tételből következik, hogy:

$$\ell(M_\mathcal{S})+\ell(M_\mathcal{R})\leq\mathrm{OPT}\quad\forall M_\mathcal{S}\in\mathcal{M}_\mathcal{S}, M_\mathcal{R}\in\mathcal{M}_\mathcal{R} \tag{3}$$

Innentől két esetet különböztetünk meg:

(A) $\ell(M_\mathcal{S})\leq\frac{\mathrm{OPT}}{2}, \forall M_\mathcal{S}\in\mathcal{M}_\mathcal{S}$

Legyen $G^m_\mathcal{S}$ G^m $\mathcal{S}$ által feszített részgráfja. Mivel $G^m_\mathcal{S}$ éleit fel tudjuk osztani S db

diszjunkt párosításra:

$$\sum_{s' \in \mathcal{S}} \sum_{s \in \mathcal{S}} \ell(s', s) = 2\ell(E^m_{\mathcal{S}}) \leq 2S\frac{\mathrm{OPT}}{2} = S \cdot \mathrm{OPT}. \tag{4}$$

(4)-ből és (2)-ből a következőt kapjuk:

$$\sum_{s' \in \mathcal{S}} \left(\sum_{s \in \mathcal{S}} \ell(s', s) + \sum_{r \in \mathcal{R}} \ell(s', r) \right) \leq (S + R)\mathrm{OPT}.$$

Ez azt jelenti, hogy léteznie kell legalább egy s^* küldőnek, amelyre:

$$\sum_{v \in \mathcal{S} \cup \mathcal{R}} \ell(s^*, v) \leq (1 + \frac{R}{S})\mathrm{OPT}.$$

(B) $\ell(M_{\mathcal{S}}) > \frac{\mathrm{OPT}}{2}$ (valamelyik) maximális összköltségű $M_{\mathcal{S}} \in \mathcal{M}_{\mathcal{S}}$-re.
Legyen a megszokott jelöléssel $G^m_{\mathcal{R}}$ a metrikus lezárt $\mathcal{R}$ által feszített részgráfja. Megmutatjuk, hogy bármely $G^m_{\mathcal{R}}$-beli maximális elemszámú $\tilde{M}$ párosításra:

$$\ell(\tilde{M}) \leq \frac{R}{S}\frac{\mathrm{OPT}}{2} \tag{5}$$

Mivel $G^m_{\mathcal{R}}$ éleit legfeljebb R db maximális párosításra tudjuk particionálni:

$$\sum_{r' \in \mathcal{R}} \sum_{r \in \mathcal{R}} \ell(r, r') \leq 2R\frac{R}{S}\frac{\mathrm{OPT}}{2} = \frac{R^2}{S}\mathrm{OPT}.$$

(2)-t hozzáadva kapjuk:

$$\sum_{r' \in R} \left(\sum_{s \in S} \ell(s, r') + \sum_{r \in R} \ell(r, r') \right) \leq (R + \frac{R^2}{S})\mathrm{OPT},$$

Ebből következik, hogy létezik olyan r^* vevő, amelyre:

$$\sum_{v \in \mathcal{S} \cup \mathcal{R}} \ell(v, r^*) \leq (1 + \frac{R}{S})\mathrm{OPT}.$$

(5) bizonyítását két esetre bontjuk, S paritásától függően:

(B.1) S páros.

A 2. tételből és az $\ell(M_{\mathcal{S}}) > \frac{\mathrm{OPT}}{2}$ feltételből következik, hogy $\ell(M_{\mathcal{R}}) \leq \frac{\mathrm{OPT}}{2}$ minden $M_{\mathcal{R}} \in \mathcal{M}_{\mathcal{R}}$ párosításra. Mivel $\tilde{M}$ egy maximális elemszámú párosítás,

$|\tilde{M}| = \lfloor \frac{R}{2} \rfloor \geq \frac{S}{2}$. Vegyük $\tilde{M}$-nek az $\frac{S}{2}$ legnagyobb költségű élét, jelölje ezt a párosítást M'. Mivel $M' \in \mathcal{M}_\mathcal{R}$, $\ell(M') \leq \frac{OPT}{2}$. Ebből következik, hogy $\tilde{M}$ éleinek átlagos költsége legfeljebb $\frac{\mathrm{OPT}/2}{S/2} = \frac{OPT}{S}$, amiből:

$$\ell(\tilde{M}) \leq |\tilde{M}| \frac{\mathrm{OPT}}{S} \leq \frac{R}{2} \frac{\mathrm{OPT}}{S} = \frac{R}{S} \frac{OPT}{2}.$$

(B.2) S páratlan.

Legyen s^* a küldő, amit az $M_\mathcal{S}$ párosítás nem tartalmaz. Minden $M_\mathcal{R} \in \mathcal{M}_\mathcal{R}$ párosításra és $r_1^* \in R$-re, amit M_R nem tartalmaz, a 2. tétel szerint:

$$\ell(M_\mathcal{S}) + \ell(M_\mathcal{R}) + \ell(s^*, r_1^*) \leq \mathrm{OPT},$$

tehát:

$$\ell(M_R) + \ell(s^*, r_1^*) \leq \frac{OPT}{2}.$$

Vegyünk egy másik r_2^* vevőt, ami nincs párosítva. Ilyen létezik, mivel $R > S$. ℓ-re triviálisan teljesül a háromszög-egyenlőtlenség: $\ell(r_1^*, r_2^*) \leq \ell(s^*, r_1^*) + \ell(s^*, r_2^*)$. Így:

$$\ell(M_R) + \frac{1}{2}\ell(r_1^*, r_2^*) \leq \frac{OPT}{2} \tag{6}$$

minden $M_\mathcal{R} \in \mathcal{M}_\mathcal{R}$-re és r_1^*, r_2^* párosítatlan vevőkre.

Legyen $\tilde{M}$-nek az $\frac{S-1}{2}$ legnagyobb költségű éle M' (párosítás), és legyen e' az $\frac{S+1}{2}$-edik. Mivel $M' \in \mathcal{M}_\mathcal{R}$, (6) miatt

$$\ell(M') + \frac{1}{2}\ell(e') \leq \frac{OPT}{2}.$$

Ebből következik, hogy $\tilde{M}$ éleinek átlagos költsége legfeljebb $2\frac{\frac{OPT}{2}}{S-1+1} = \frac{OPT}{S}$, amiből:

$$\ell(\tilde{M}) \leq |\tilde{M}| \frac{\mathrm{OPT}}{S} \leq \frac{R}{2} \frac{\mathrm{OPT}}{S} = \frac{R}{S} \frac{OPT}{2}.$$

□

3.2. Az 1. algoritmus implementációja

Könnyű észrevenni, hogy az implementációban nem szükséges a terminálokat felbontani küldőkre és vevőkre, mivel a távolságösszeg, amit minimalizálni szeretnénk, felírható a következő formában:

$$\sum_{i \in T} (b_{in}(i) + b_{out}(i))\ell(vi),$$

$\forall v \in V$ csúcsra. Így elérhető az erősen polinomiális futási idő.

Megjegyzés. Ahhoz, hogy minden csúcsra megkapjuk a fenti távolságösszeget, nem kell minden csúcsból Dijkstrát futtatnunk. Elegendő csak a terminálokból indítani, majd minden csúcsra összeadni a terminálóktól vett távolságokat (b_{in}+b_{out} értékekkel szorozva). $|T| \ll |V|$ esetén ez jelentősen jobb futási időt ad.

1. algoritmus: (pszeudokód)

$\forall i \in V$: távolságösszeg(i) := 0
$\forall i \in T$:
    Dijkstra(i) // a kapott $\ell(ij)$ távolságokat eltároljuk
    $\forall j \in V$:
        távolságösszeg(j) := távolságösszeg(j) + $(b_{in}(i) + b_{out}(i))\ell(ij)$
v := argmin$_{i \in V}$távolságösszeg(i)
Dijkstra(v) // minden csúcsnak eltároljuk a szülőjét a kapott fenyőben
// így rekonstruálni tudjuk a legrövidebb utakat
$\forall i \in T$:
    $j := i$
    while $j \neq$ v:
        $u(j, \text{szülő}(j)) := u(j, \text{szülő}(j)) + b_{in}(i) + b_{out}(i)$
        $j :=$ szülő(j)

Aszimptotikus futási idő: $O(|T||E| \log |V| + |T||V|) = O(|T||E| \log |V|)$

3.3. Randomizált algoritmus

Ötlet (2. algoritmus) [6]:

(1) Rakjunk minden vevőt véletlenszerűen S darab (kezdetben üres) részhalmaz egyikébe, egyenletesen, egymástól függetlenül. Így $\mathcal{R}$-nek egy partícióját kapjuk. A partíció nemüres részhalmazai közül válasszuk ki az egyik $\mathcal{R}'$ részhalmazt véletlenszerűen, egyenletesen.

(2) Minden $s \in \mathcal{S}$ küldőre számítsunk ki egy 2-közelítő Steiner-fát[1] az $\{s\} \cup \mathcal{R}'$ terminálhalmazon (jelöljük ezt $T(s)$-sel), majd ennek minden éléhez adjunk hozzá 1 kapacitást.

(3) Minden $r \in \mathcal{R}$ vevőre az r és $\mathcal{R}'$ közötti legrövidebb út mentén adjunk hozzá 1 kapacitást az élekhez.

Minden (s, r) küldő-vevő párra a következő utat jelöljük ki:

Legyen r^* $\mathcal{R}'$-nek az r-hez legközelebbi eleme (ha több is van, akkor a 3. fázisban kiválasztott). Vegyük a $T(s)$-beli (egyértelmű) $s - r^*$ út és az $r^* - r$ közötti (egyik) legrövidebb út konkatenációját. Mivel nem biztos, hogy ez út (csak séta), töröljük belőle a köröket addig, amíg nem kapunk utat. Könnyű látni, hogy a kapott élkapacitások már a körök törlése nélkül is megengednék a sétákat. Így tehát megengedett megoldást kapunk.

Megjegyzés. Ellentétben az 1. algoritmussal, ez nem mindig ad fa megoldást.

A 2. algoritmus közelítési arányának bizonyításához előbb szükségünk lesz pár lemmára. A továbbiakban jelöljük $st(V')$-vel a V' terminálhalmazon vett optimális Steiner-fa költségét.

2. Következmény. [6] *Vegyük $\mathcal{R}$-nek egy $\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_S$ partícióját, valamint egy tetszőleges teljes párosítást $\mathcal{S}$ és a részhalmazok között. Jelölje $\mathcal{R}(s)$ az s-hez párosított részhalmazt. Az $\{s\} \cup \mathcal{R}(s)$ terminálhalmazokon vett optimális Steiner-fák költségeinek összege alsó korlátot ad a VPND optimumértékére:*

$$\sum_{s \in \mathcal{S}} st(\{s\} \cup \mathcal{R}(s)) \leq OPT$$

[1] Léteznek ennél jobb közelítési arányú algoritmusok is a Steiner-fa feladatra, jelenleg (2025 május) 1,39 a legjobb [3]. Mivel ez kívül esik a szakdolgozatom keretein, az egyszerűség (és a jobb futási idő) kedvéért megelégszünk egy közismert 2-közelítő algoritmussal.

Bizonyítás. Ez következik a 2. lemmából, mivel az $\{s\} \cup \mathcal{R}(s)$ terminálhalmazon vett VPND feladat megoldásának tartalmaznia kell egy Steiner-fát ugyanezen a terminálhalmazon. □

4. Lemma. [6] *Egy véletlenszerűen, egyenletes eloszlással választott $s' \in \mathcal{S}$ esetén:*

$$E\left[st(\{s'\} \cup \mathcal{R}')\right] \leq \frac{OPT}{S\left(1 - e^{-\frac{R}{S}}\right)}$$

Bizonyítás. Vegyük a következő véletlen folyamatot: minden vevőt hozzárendelünk egy egyenletesen véletlenszerűen választott küldőhöz. Jelölje $\mathcal{R}(s)$ az s-hez rendelt vevők részhalmazát. A $\mathcal{R}(s)$ részhalmazok $\mathcal{R}$-t S db (esetleg üres) részhalmazra particionálják. Így az 1. következmény szerint:

$$\sum_{s \in \mathcal{S}} st(\{s\} \cup \mathcal{R}(s)) \leq OPT$$

Ez azt jelenti, hogy egy egyenletesen véletlenszerűen kiválasztott s' küldőre:

$$E\left[st(\{s'\} \cup \mathcal{R}(s'))\right] \leq \frac{OPT}{S}$$

Jelölje A azt az eseményt, hogy $\mathcal{R}(s')$ üres. Felírva a teljes valószínűség tételét:

$$E\left[st(\{s'\} \cup \mathcal{R}(s'))\right] = P(A)E\left[st(\{s'\} \cup \mathcal{R}(s')) \mid A\right] + P(\bar{A})E\left[st(\{s'\} \cup \mathcal{R}(s')) \mid \bar{A}\right]$$

Észrevehetjük, hogy:

$$P(\bar{A}) = 1 - \left(1 - \frac{1}{S}\right)^R \geq 1 - e^{-\frac{R}{S}},$$

Valamint azt, hogy:

$$E\left[st(\{s'\} \cup \mathcal{R}(s')) \mid A\right] = E\left[st(\{s'\})\right] = 0$$

Így:

$$E\left[st(\{s'\} \cup \mathcal{R}(s')) \mid \bar{A}\right] = \frac{E\left[st(\{s'\} \cup \mathcal{R}(s'))\right]}{P(\bar{A})} \leq \frac{OPT}{S\left(1 - e^{-\frac{R}{S}}\right)}$$

Mivel $\mathcal{R}(s')$ feltételes eloszlása $\bar{A}$-t feltéve megegyezik $\mathcal{R}'$ eloszlásával:

$$E\left[st(\{s'\} \cup \mathcal{R}')\right] = E\left[st(\{s'\} \cup \mathcal{R}(s')) \mid \bar{A}\right] \leq \frac{OPT}{S\left(1 - e^{-\frac{R}{S}}\right)}$$

□

5. Lemma. [6] *A 2. algoritmus 2. fázisában a hozzáadott kapacitások várható összköltsége legfeljebb:*

$$\frac{2OPT}{1-e^{-\frac{R}{S}}}$$

Bizonyítás. A várható összköltség:

$$E\left[\sum_{s\in\mathcal{S}} c(T(s))\right],$$

ahol $c(T(s))$ a $T(s)$ Steiner-fa költsége. Mivel 2-közelítő:

$$c(T(s)) \leq 2st(\{s\}\cup\mathcal{R}')$$

Legyen s' egy egyenletesen véletlenszerűen választott küldő. A 4. lemma szerint:

$$E\left[\sum_{s\in\mathcal{S}} c(T(s))\right] \leq 2E\left[\sum_{s\in\mathcal{S}} st(\{s\}\cup\mathcal{R}')\right] = 2SE\left[st(\{s'\}\cup\mathcal{R}')\right] \leq \frac{2OPT}{1-e^{-\frac{R}{S}}}$$

□

6. Lemma. [6] *A 3. fázisban a hozzáadott kapacitások várható összköltsége szintén legfeljebb:*

$$\frac{2OPT}{1-e^{-\frac{R}{S}}}$$

Bizonyítás. Először vegyük észre, hogy bármely két különböző $r' \neq r''$ vevőre $\frac{1}{S}$ annak a feltételes valószínűsége, hogy $r'' \in \mathcal{R}'$, $r' \in \mathcal{R}'$-t feltéve.

Ha $\mathcal{R}^*$-gal jelölünk egy egyenletesen véletlenszerűen kiválasztott (esetleg üres) részhalmazt a partícióból:

$$P(r'\in\mathcal{R}') = P(r'\in\mathcal{R}^* \mid \mathcal{R}^*\neq\emptyset) = \frac{P(r'\in\mathcal{R}^*\cap\mathcal{R}^*\neq\emptyset)}{P(\mathcal{R}^*\neq\emptyset)} = \frac{P(r'\in\mathcal{R}^*)}{P(\mathcal{R}^*\neq\emptyset)} = \frac{\frac{1}{S}}{1-(1-\frac{1}{S})^R}.$$

Hasonlóan:

$$P(r'\in\mathcal{R}'\cap r''\in\mathcal{R}') = \frac{(\frac{1}{S})^2}{1-(1-\frac{1}{S})^R}$$

Így valóban:

$$P(r'' \in \mathcal{R}' \mid r' \in \mathcal{R}') = \frac{P(r'' \in \mathcal{R}' \cap r' \in \mathcal{R}')}{P(r' \in \mathcal{R}')} = \frac{(\frac{1}{S})^2/(1-(1-\frac{1}{S})^R)}{(\frac{1}{S})/(1-(1-\frac{1}{S})^R)} = \frac{1}{S}$$

Most pedig tekintsük a következő véletlen folyamatot:

Kezdetben legyen $\mathcal{A} = \mathcal{B} = \{r'\}$, ahol r' egy egyenletesen véletlenszerűen kiválasztott vevő. Majd minden iterációban:

- vegyük azt az (egyik) $r \in \mathcal{R} \setminus \mathcal{A}$ vevőt, amely a legközelebb van $\mathcal{B}$-hez
- $\mathcal{A} := \mathcal{A} \cup \{r\}$
- $\frac{1}{S}$ valószínűséggel:
  - r és $\mathcal{B}$ közötti legrövidebb út éleihez 1 kapacitást adunk
  - $\mathcal{B} := \mathcal{B} \cup \{r\}$
- különben r és $\mathcal{B}$ közötti legrövidebb út éleihez 1 kapacitást adunk

Ezt addig iteráljuk, amíg $\mathcal{A} \neq \mathcal{R}$.

Könnyű látni, hogy ennek a véletlen folyamatnak a végén $\mathcal{B}$ eloszlása ugyanaz, mint $\mathcal{R}'$-nek. Jelölje a továbbiakban $\ell(v, V')$ egy v csúcs és egy V' csúcshalmaz közötti távolságot, r_t a t-edik iterációban kiválasztott vevőt, $\mathcal{B}_t$ pedig a $\mathcal{B}$ halmazt a t-edik iterációban.

$$E\left[\sum_{r \in \mathcal{R} \setminus \mathcal{R}'} \ell(r, \mathcal{R}')\right] = E\left[\sum_{r \in \mathcal{R} \setminus \mathcal{B}} \ell(r, \mathcal{B})\right] \leq E\left[\sum_{r_t \in \mathcal{R} \setminus \mathcal{B}} \ell(r_t, \mathcal{B}_t)\right] =$$

$$= E\left[\sum_{r_t \in \mathcal{R}} \left(1 - \frac{1}{S}\right) \ell(r_t, \mathcal{B}_t)\right] \leq E\left[\sum_{r_t \in \mathcal{R}} \left(\frac{1}{S}\right) S\ell(r_t, \mathcal{B}_t)\right] = E\left[\sum_{r_t \in \mathcal{B}} S\ell(r_t, \mathcal{B}_t)\right],$$

ahol az egyenlőtlenség onnan adódik, hogy $\mathcal{B}_t \subseteq \mathcal{B}, \forall t$.

Észrevehetjük, hogy ez utóbbi pontosan S-szerese annak a várható költségnek, amit $\mathcal{B}$ metrikus lezártján az r'-ből indított Prim-algoritmus ad. Ismert, hogy ez 2-közelítést ad a Steiner-fa feladatra [15], így tehát:

$$E\left[\sum_{r_t \in \mathcal{B}} S\ell(r_t, \mathcal{B}_t)\right] \leq 2SE\left[st(\mathcal{B})\right] = 2SE\left[st(\mathcal{R}')\right].$$

4. lemma szerint egy egyenletesen véletlenszerűen választott s' küldőre:

$$E\left[st(\mathcal{R}')\right] \leq E\left[st(\{s'\} \cup \mathcal{R}')\right] \leq \frac{OPT}{S\left(1 - e^{-\frac{R}{S}}\right)}.$$

Tehát összességében:

$$E\left[\sum_{r \in \mathcal{R}} \ell(r, \mathcal{R}')\right] \leq 2S\frac{OPT}{S\left(1 - e^{-\frac{R}{S}}\right)} = \frac{2OPT}{1 - e^{-\frac{R}{S}}}.$$

□

4. Tétel. *A 2. algoritmus* $\frac{4}{1-e^{-\frac{R}{S}}}$*-közelítő*

Bizonyítás. Triviális következménye a 3. és 4. lemmáknak.

□

3.4. A 2. algoritmus implementációja

1. fázis: Ahogy azt a 6. lemma bizonyításában láthattuk, $\mathcal{R}'$ kiválasztása szimulálható a következőképpen:

- Válasszunk ki egy $r' \in \mathcal{R}$-t a egyenletesen véletlenszerűen, és adjuk hozzá a halmazhoz.
- A többi vevő mindegyikét $\frac{1}{S}$ valószínűséggel adjuk hozzá a halmazhoz, egymástól függetlenül.

Megjegyzés. Ezt a véletlen folyamatot ebben a formában leszimulálni $O(R)$ lépést igényelne. Ha erősen polinomiális futási időt szeretnénk, akkor ennél okosabbnak kell lennünk. Vegyük észre, hogy az algoritmus hátralevő részében mindegy, hogy egy adott terminálhoz tartozó vevők közül hányat vettünk be $\mathcal{R}'$-be. Csak az számít, hogy bevettünk-e legalább egyet, tehát elegendő csak ezt szimulálnunk. Annak a valószínűsége, hogy egy v terminálnak egy vevőjét sem vettük be, $p_v = (1-\frac{1}{S})^{b_{in}(v)}$ (kivéve a kezdetben kiválasztott r'-t). Ezek különböző terminálokra független valószínűségi események. Minden v-re a p_v valószínűség kiszámítható $O(1)$ futási időben lebegőpontos műveletekkel (ha a bitek számát konstansnak vesszük), valamint szintén $O(1)$-ben szimulálható egy valószínűségi változó, amely p_v valószínűséggel hamis, különben igaz. A kezdeti r' kiválasztása megoldható egy véletlen egész szám generálásával $[0, R-1]$-ből egyenletesen.

2. fázis: Közismert, hogy a következő algoritmus 2-közelítést ad a minimális költségű Steiner-fa feladatra [15]:

- Készítsük el a gráf metrikus lezártjának a terminálok által feszített részgráfját.
- Ebben keressünk minimális feszítőfát (például Prim-, Kruskal- vagy egyéb algoritmussal)
- A minimális feszítőfa minden élére az annak megfelelő eredeti gráfbeli éleket, azaz a két végpontja közti eredeti gráfbeli (bármelyik) legrövidebb út éleit adjuk hozzá a megoldáshoz
- Mivel nem biztos, hogy az így kapott részgráf fa, vegyük a(z egyik) minimális feszítőfáját, a többi élt pedig töröljük belőle

Megjegyzés. Legyen F a minimális feszítőfa a metrikus lezárt $\mathcal{R}'$ által feszített részgráfján. Vegyük észre, hogy ehhez egy új s csúcsot és a hozzá tartozó éleket hozzáadva, az (egyik) új minimális feszítőfa élei csak $F \cup \delta(s)$-ből kerülhetnek ki (ahol $\delta(s)$ az s éleinek halmaza a metrikus lezártban). Ez azt jelenti, hogy elegendő csak ezekre az élekre futtatni a minimális feszítőfa algoritmust. Ha F-et már kiszámítottuk, az új minimális feszítőfát $O(|T| \log |T|)$-ben meg tudjuk kapni. Ezzel az optimalizációval jelentősen jobb futási időt kapunk, mint ha minden $\mathcal{R}' \cup \{s\}$-re az elejétől kezdve számítanánk ki újra a minimális feszítőfát.

Megjegyzés. A metrikus lezárt T által feszített részgráfját elég csak egyszer elkészíteni, és amúgy is, az összes többi algoritmusunknak is része az, hogy Dijkstrákat futtatunk a terminálokból. Így ennek az algoritmusnak a legköltségesebb része az utolsó két lépés marad: a minimális feszítőfából elkészíteni a Steiner-fát az eredeti gráfon. Mivel már a minimális feszítőfa összköltsége is 2-közelítés az optimális Steiner-fa költségére [15], sokszor ez is megfelel, és az utolsó két lépést kihagyhatjuk teljesen. Ezt a 4. fejezetben bővebben kifejtem. Egyelőre implementáljuk le az algoritmust a fent leírt formában.

3. fázis: a legrövidebb utakat megkaphatjuk egyetlen, több forrású Dijkstra futtatásával (a Dijkstrát egy képzeletbeli csúcsból indítjuk, amelyből 0 költségű éleket húzunk $\mathcal{R}'$ csúcsaiba).

Az alábbi implementáció aszimptotikus futási ideje:
$O(|T||E| \log |V| + |E| \log |E| + |T|^2 \log |T| + |T|(|T| \log |T| + |T||V| \log |V| + |E|)) =$
$O(|T|(|E| + |T||V|) \log |V|)$
(Dijkstrák + élek rendezése + Kruskalok + utak rekonstruálása)

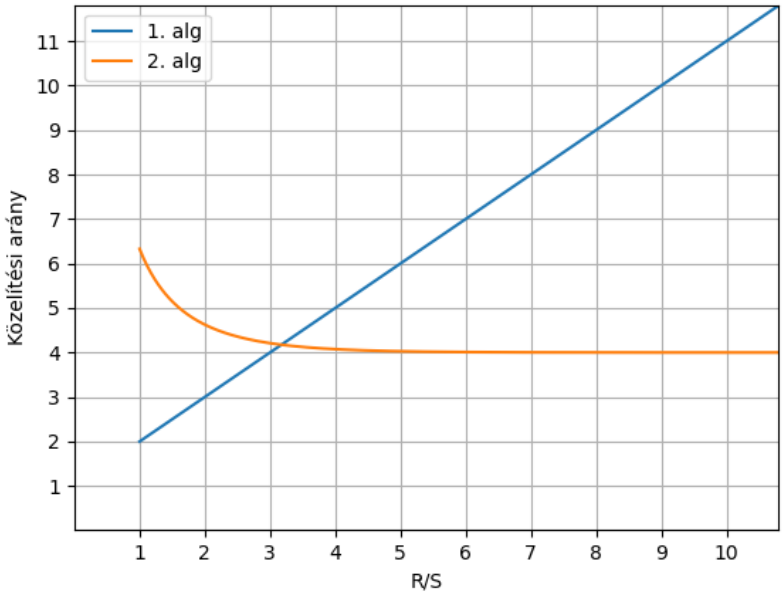
2. algoritmus (pszeudokód):

// 1. fázis
$\mathcal{R}' := \emptyset$
$r' :=$ RandomEgész$[0, R-1]$
$\forall v \in T, b_{in}(v) > 0$:
    $r' := r' - b_{in}(v)$
    if $r' < 0$:
        $\mathcal{R}' := \mathcal{R}' \cup \{v\}$
        break
$\forall v \in T, b_{in}(v) > 0, v \notin \mathcal{R}'$:
    $p := (1 - \frac{1}{S})^{b_{in}(v)}$
    if RandomValós$[0, 1) \geq p$:
        $\mathcal{R}' := \mathcal{R}' \cup \{v\}$
// 2. fázis
$\forall i \in T$:
    Dijkstra(i) // a távolságokat és a szülőket eltároljuk
// az eredeti gráf éleit előre rendezzük c szerint, hogy gyorsítsunk a
// Kruskal-algoritmuson
$F :=$ Kruskal$(\mathcal{R}', \ell)$ // Kruskal-algoritmus a metrikus lezárton
F számláló := 0
$\forall s \in T, b_{out}(s) > 0$:
    if $s \in \mathcal{R}'$:
        F számláló := F számláló $+ b_{out}(s)$
    else:
        $F' :=$ Kruskal$(F \cup \delta(s), \ell)$
        // ahol $\delta(s)$ s éleinek halmaza a metrikus lezártban
        Steiner := $\emptyset$
        $\forall uv \in F'$:
            // visszakövetjük az $u - v$ utat az u gyökerű Dijkstra-fenyőben
            $\forall e \in$ út: Steiner := Steiner $\cup \{e\}$
        Steiner := Kruskal(Steiner, c) // a rendezett éllistán végigiterálva,
        // minden élre megnézzük, hogy bevettük-e
        $\forall e \in$ Steiner: $u(e) := u(e) + b_{out}(s)$
Steiner := $\emptyset$
$\forall uv \in F$:
    // visszakövetjük az $u - v$ utat az u gyökerű Dijkstra-fenyőben
    $\forall e \in$ út: Steiner := Steiner $\cup \{e\}$
Steiner := Kruskal(Steiner, c) // a rendezett éllistán végigiterálva,
// minden élre megnézzük, hogy bevettük-e
$\forall e \in$ Steiner: $u(e) := u(e) +$ F számláló
// 3. fázis
Dijkstra($\mathcal{R}'$) // több forrású Dijkstra az eredeti gráfon
$\forall r \in T, b_{in}(r) > 0$:
    // visszakövetjük a legrövidebb utat r-ből $\mathcal{R}'$-be
    $\forall e \in$ út: $u(e) := u(e) + b_{in}(r)$

3.5. Összegzés

Az NP-nehéz VPND feladatra két közelítő algoritmust mutattunk be: az egyik determinisztikus, $1 + \frac{R}{S}$ közelítési aránnyal, a másik pedig randomizált, várhatóan $\frac{4}{1-e^{-\frac{R}{S}}}$ közelítési aránnyal. Láttuk, hogy mindkettő megvalósítható polinomiális futási időben.

Vegyük észre, hogy $1 + \frac{R}{S}$ nő, $\frac{4}{1-e^{-\frac{R}{S}}}$ viszont csökken $\frac{R}{S}$ függvényében.



3.1. ábra. A két algoritmus közelítési aránya $\frac{R}{S}$ függvényében

Ahogy az a fenti ábrán is látható, egy inputra mindkét algoritmust lefuttatva és az eredmények minimumát véve $\approx 4,17$-es várható közelítési arányt tudunk garantálni. Ezen javíthatunk, ha a 2. algoritmus több futásának vesszük a minimumát. A következő fejezetben többek között ezzel is fogunk kísérletezni.

Megjegyzés. Ha a 2. algoritmus 2. fázisában jobb közelítési arányú Steiner-fa algoritmust használunk, tovább tudunk javítani a végső közelítési arányon. Például ha $1,39$-közelítőt [3] használunk, a $4,17$-et le tudjuk csökkenteni $\approx 3,65$-re. Ez túlmutat a szakdolgozatom keretein, de az elméleti lehetőséget fontosnak tartom megemlíteni.

4. fejezet

Tesztelés

Ebben a fejezetben az eddigiekben bemutatott algoritmusok C++ implementációit fogjuk tesztelni különböző inputokon, lemérve a futási időket és összehasonlítva a kapott eredményeket. Továbbá, az optimumértékre alsó korlátot kiszámítva felülről becsüljük az algoritmusaink közelítési arányait.

Az implementációhoz a LEMON nevű C++ könyvtárat használtam. Azért erre a könyvtárra esett a választásom, mert a nekem szükséges algoritmusok (pl. Dijkstra, Kruskal) és adatszerkezetek (pl. gráfok, csúcs/él mapek) nagy részét hatékonyan megvalósították benne. A LEMON, illetve annak dokumentációja és telepítési útmutatója elérhető a `https://lemon.cs.elte.hu` címen. Az általam használt verziószám: 1.3.1.

A C++ kódok fordításához a `g++` fordítóprogram 15.1.1-es verzióját használtam `-O2` optimalizációs szinttel és `-std=c++17` szabvány megadásával. A programokat Arch Linux környezetben futtattam, az alábbi hardverkonfiguráción:

- Lenovo ThinkPad T480s (20L7001NGE) laptop
- Intel(R) Core(TM) i7-8550U (8) @ 4.00 GHz
- 16GB DDR4 RAM, 2400 MHz

A tesztesetek generálásához a NetworkX nevű Python könyvtárat használtam, a grafikonok elkészítéséhez pedig a Matplotlib nevű könyvtárat. Az általam használt összes C++ és Python kód, valamint az inputok és a kapott eredmények elérhetők a `https://github.com/kalex3/vpnd` címen.

4.1. Észrevételek, optimalizációk

Mint már említettem, ugyanazon a gráfon elegendő csak egyszer lefuttatni a $|T|$ darab Dijkstrát. Ez $O(|T||E|\log|V|$. A távolságokat eltárolva megkapjuk a gráf metrikus lezártjának T által feszített részgráfját, amire a 2. algoritmushoz szükség van. Ha a fenyőket is eltároljuk, rekonstruálni is tudjuk a legrövidebb utakat.

Ezután az 1. algoritmus csak annyiból áll, hogy kiválasztjuk a legkisebb távolságösszegű csúcsot, majd mindegyik terminálra visszakövetjük az oda vezető legrövidebb utat. Ez $O(|T||V|)$.

A 2. algoritmus legköltségesebb része az marad, hogy a metrikus lezártbeli minimális feszítőfák éleiből elkészítsük a Steiner-fákat az eredeti gráfon. Ez a legrosszabb esetben $O(T)$ darab feszítőfára összesen $O(|T|(|T||V|\log|V|+|E|))$ lépést igényel. Többszöri futtatás esetén viszont erre nincs mindig szükség: megtehetjük azt, hogy a generált $\mathcal{R}'$-kre először csak a minimális feszítőfa költségét számítjuk ki (ez $O(|T|^2\log|T|$ lépésben megoldható függetlenül az eredeti gráf méretétől), majd ehhez hozzáadjuk a 3. fázis költségét (ez $O(|T|^2)$, minden terminálra meghatározzuk a hozzá legközelebbi R'-beli terminál távolságát, és ezeket a b_{in} értékekkel megszorozva összeadjuk). Miután megkaptuk ezt a felső becslést minden $\mathcal{R}'$-re, kiválasztjuk a legjobbakat és csak ezekre készítjük el a Steiner-fákat (majd a kapott eredmények minimumát vesszük). Ez sokat javít a futási időn.

Szükségünk van továbbá egy alsó korlát kiszámítására is. A 2. fejezetben kapott eredmménnyel az a probléma, hogy csak kis $\frac{R}{S}$-ekre ad jó becslést. Viszont észrevehetjük, a 2. algoritmus közelítési arányának bizonyításából az 1. következmény szintén ad egy alsó korlátot:

$$\sum_{s\in\mathcal{S}} st(\{s\}\cup\mathcal{R}(s)) \leq OPT,$$

ahol $\mathcal{R}(s), \forall s\in\mathcal{S}$ a vevőknek egy tetszőleges partíciója.

Az optimális Steiner-fák költségét alulról tudjuk becsülni 2-közelítések felével. Hátravan T darab minimális feszítőfa kiszámítása úgy, hogy a végső futási idő ne függjön T-től. Ezt megoldhatjuk például a következő algoritmussal:

Új alsó korlát kiszámítása (pszeudokód):

sum := 0
while $R > 0, S > 0$:
    generáljunk $\mathcal{R}'$-t a szokásos módon
    válasszunk ki egy s-t véletlenszerűen a nemnulla b_{out} értékű terminálok közül
    $min := +\infty$
    $\forall r \in \mathcal{R}'$:
        if $b_{in}(r) < min$:
            $min := b_{in}(r)$
    if $b_{out}(s) < min$:
        $min := b_{out}(s)$
    $sum := sum + \ell(\text{Kruskal}(s \cup \mathcal{R}'))/2$
    for $r \in \mathcal{R}'$:
        $b_{in}(r) := b_{in}(r) - min$
        $R := R - min$
    $b_{out}(s) := b_{out}(s) - min$
    $S := S - min$
return *sum*

Mivel minden iterációban lenullázunk legalább egy új b_{in} vagy b_{out} értéket, az algoritmus legfeljebb $2|T|$ iterációban véget ér. Így tehát az algoritmus össz futási ideje: $O(2|T| \cdot |T|^2 \log |T|) = O(|T|^3 \log |T|)$.

Megjegyzés. Előfordulhat, hogy maradnak olyan vevők, amiket egyik küldőhöz sem rendeltünk hozzá. Ezeket utólag hozzárendelve valamelyikhez, mivel egy terminálhalmaz bővítésével az optimális Steiner-fa költsége nem csökkenhet, az algoritmus végén kapott érték továbbra is alsó korlát.

Mivel ez nagy $\frac{R}{S}$ esetén jelentősen jobb közelítési arányt garantál, mint a 2. fejezetben kapott alsó korlát, a továbbiakban ezt fogjuk használni.

4.2. A tesztelés menete

Az algorimusainkat különböző csúcsszámú és élsűrűségű gráfokon futtatjuk. Ugyanazon a gráfon többféle terminálszámot és $\frac{R}{S}$ arányt is kipróbálunk. Először véletlen gráfokon teszteljük őket, majd pedig egy valódi gráfon is.

Az eredményeket az $\frac{R}{S}$ arány függvényében fogjuk vizsgálni (mivel elsősorban ettől függ, hogy melyik algoritmus teljesít jobban), továbbá egy alsó korlátot kiszámítva felülről becsüljük a közelítési arányokat. Mint azt látni fogjuk, a gyakorlatban az eddig ismert $3,55$-ös elméleti aránynál [6] jeletősen jobb eredményeket kapunk.

A 2. algoritmus 1000 futásából a felső becslés szerinti legjobb 10 minimumát vettem. A felső becsléseket összeátlagoltam, ezzel közelítettem a 2. algoritmus várható eredményét. Alsó korlátot a 4.1-ben bemutatott algoritmus 1000-szeri futtatásával kaptam, az eredmények maximumát véve.

4.3. Tesztelés véletlen gráfokon

Először az Erdős–Rényi modell [7] szerint generáltam véletlen gráfokat. Ebben a modellben két paraméterünk van: n és p. n a csúcsszám, és minden csúcspár közé p valószínűséggel húzunk élt, egymástól függetlenül. A p paraméterrel lehet állítani a gráf (várható) élszámát.

Mivel az Erdős-Rényi modell nem garantál összefüggő gráfot, a kapott gráf legnagyobb összefüggő komponensét vettem. 3-féle élsűrűséggel kísérleteztem:

- ritka gráfok: $m \approx n \log n$ él (egy legalább ennyi élű Erdős-Rényi véletlen gráf nagy valószínűséggel összefüggő [7])
- középsűrű gráfok: $m \approx n\sqrt{n}$ él
- sűrű gráfok: $m \approx \frac{1}{4}n^2$ él

3 különböző terminálszám közül választottam:

- $|T| \approx \log n$
- $|T| \approx \sqrt[3]{n}$
- $|T| \approx \sqrt{n}$

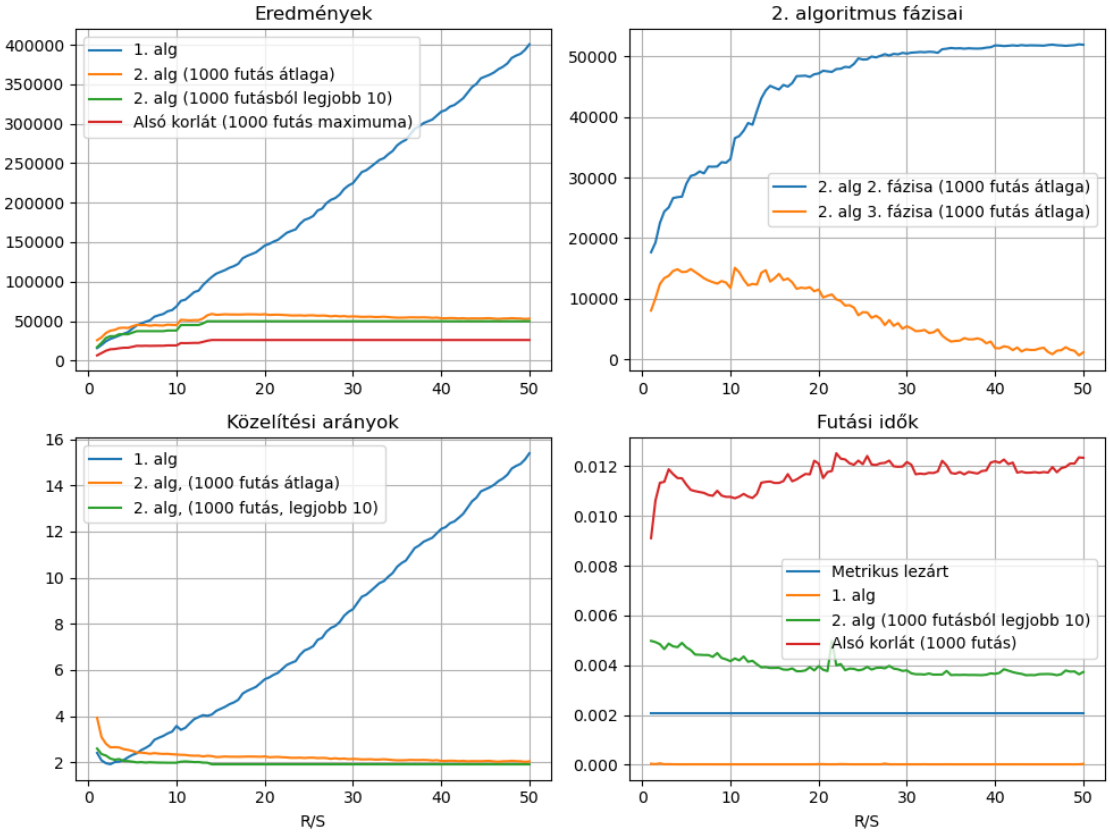
Az élköltségeket egyenletes eloszlással vettem a $[0, 100)$ intervallumból.

Először meghatározott számú küldőt és vevőt osztottam szét egyenletesen véletlenszerűen a terminálok között, kezdetben $R = S = 200$.

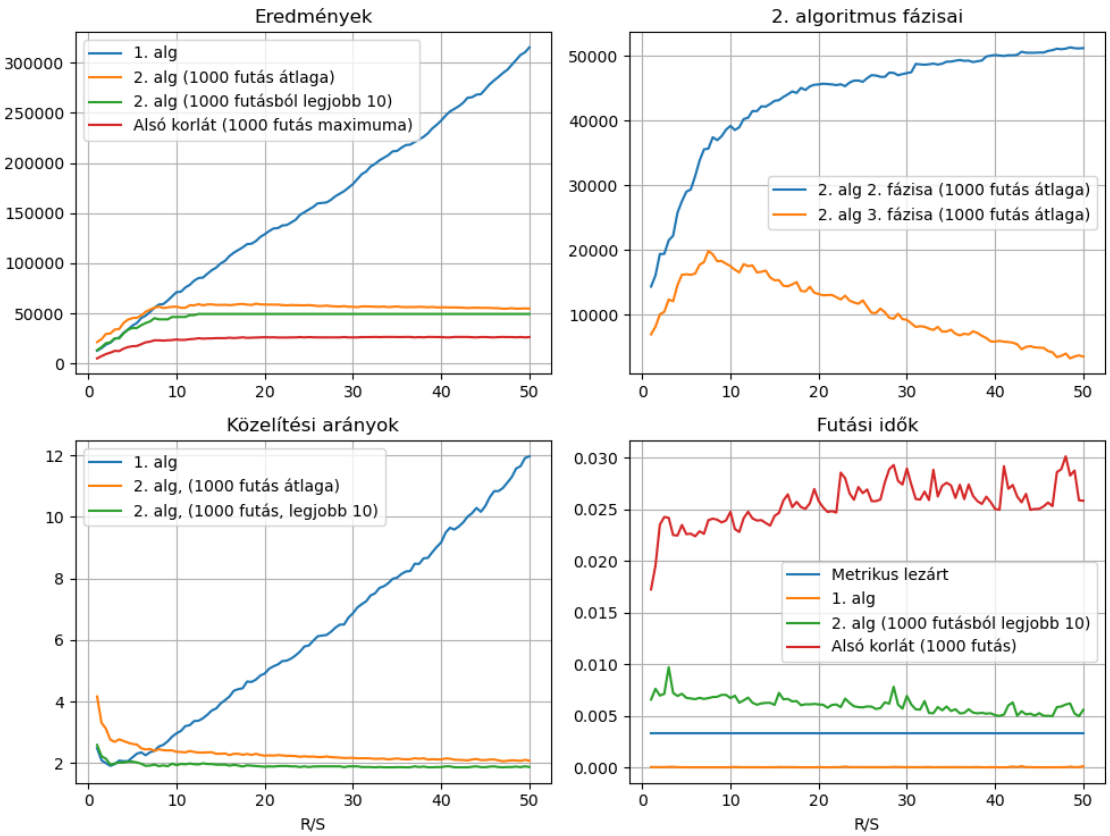
Ezután egyenletesen véletlenszerűen kiválasztott terminálóknak növeltem a b_{in} értékét (összesen 100-at hozzáadva minden iterációban), ezzel változtatva az $\frac{R}{S}$ arányt. Az eredményekről, közelítési arányokról, illetve futási időkről grafikonokat készítettem $\frac{R}{S}$ függvényében. Külön vizsgáltam a 2. algoritmus 2. és 3. fázisában hozzáadott költséget.

1000 csúcsú gráfokon az alábbi eredmények születtek:

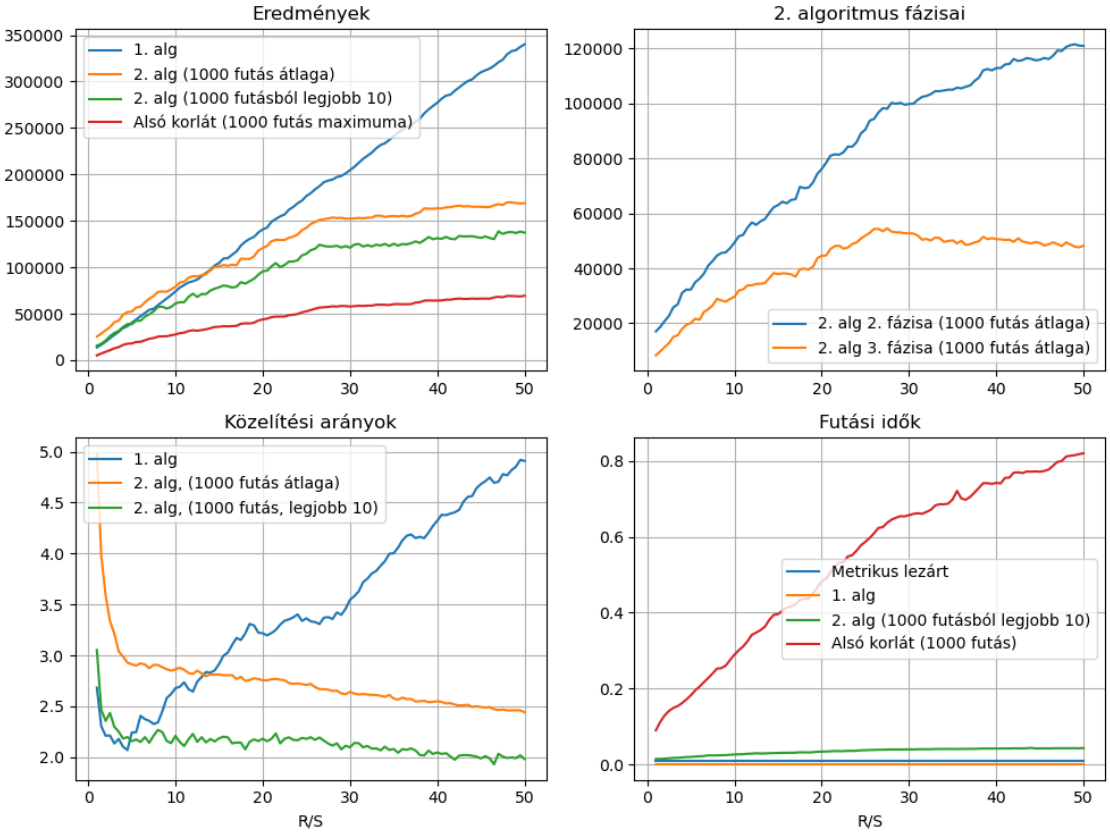
1 000 csúcs, 7 000 él, 7 terminál



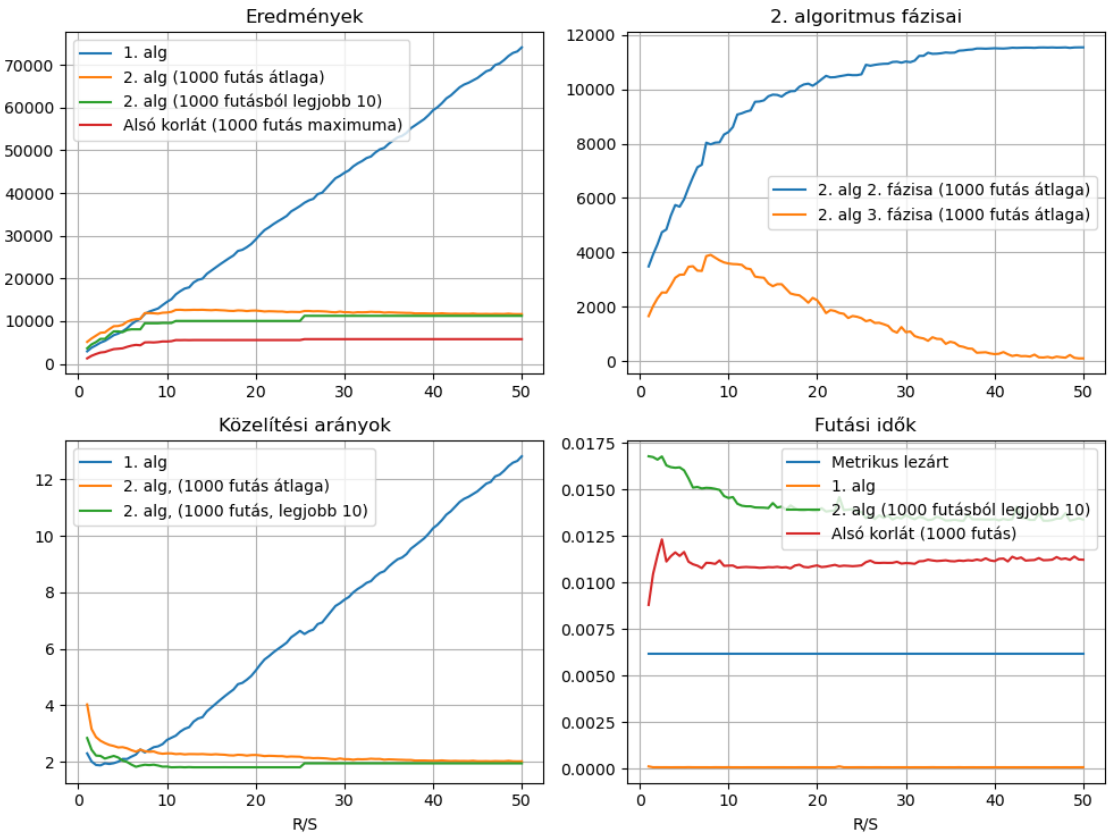
1 000 csúcs, 7 000 él, 10 terminál



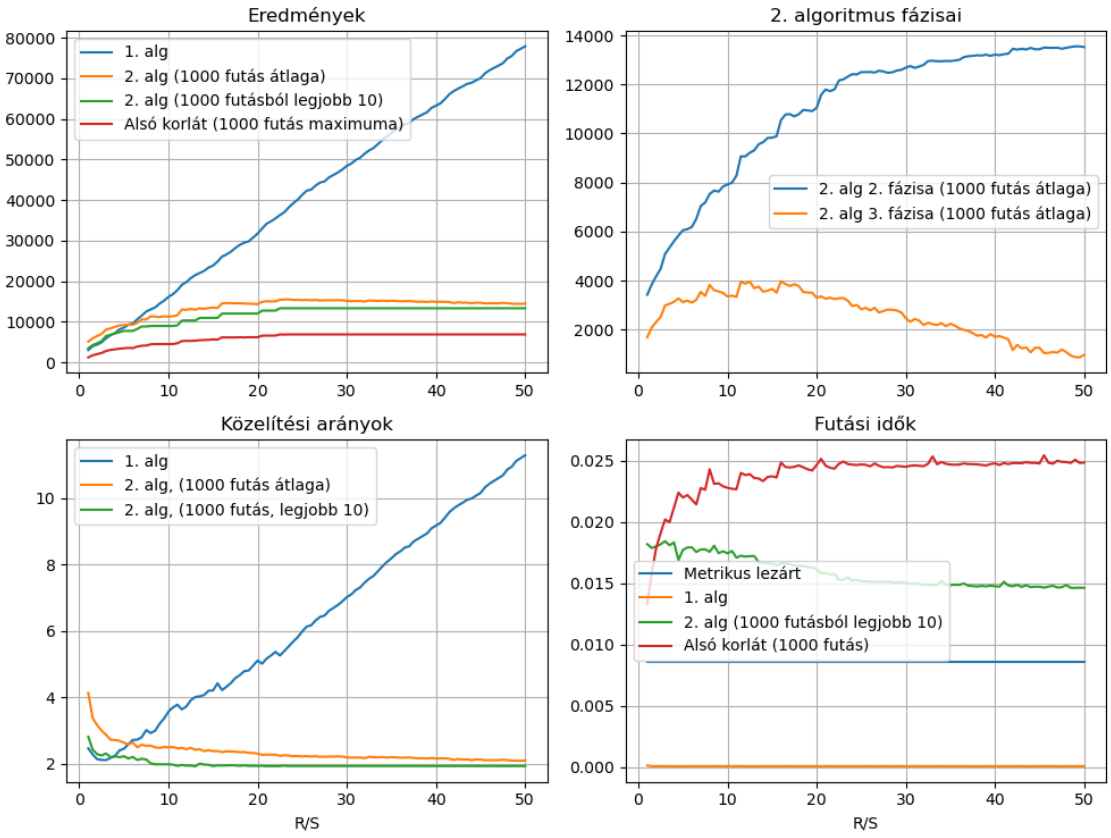
1 000 csúcs, 7 000 él, 32 terminál



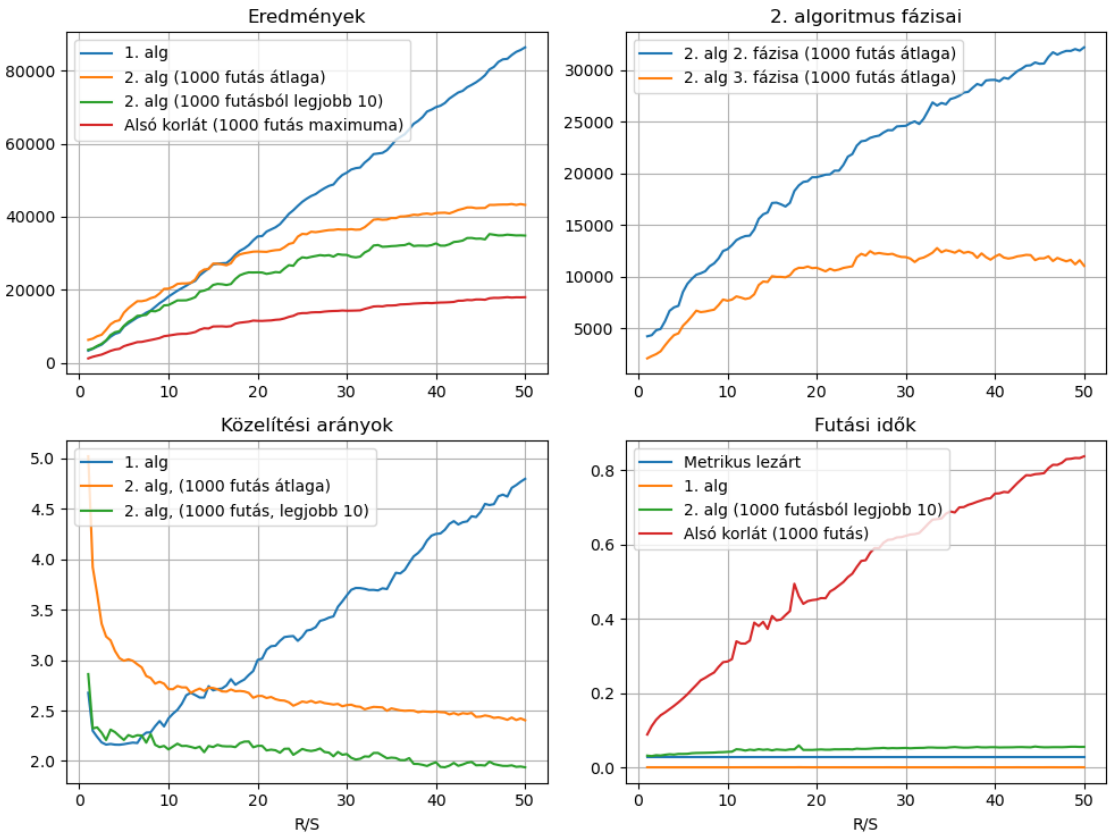
1 000 csúcs, 32 000 él, 7 terminál

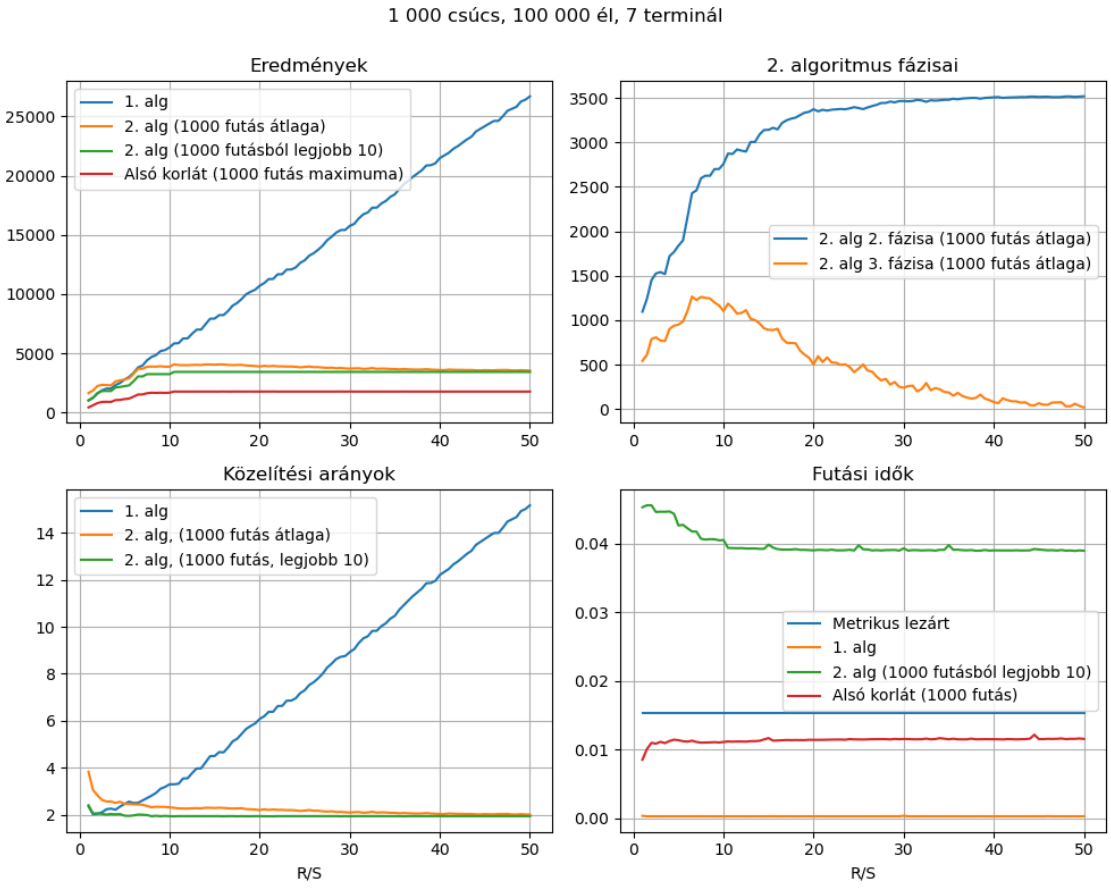


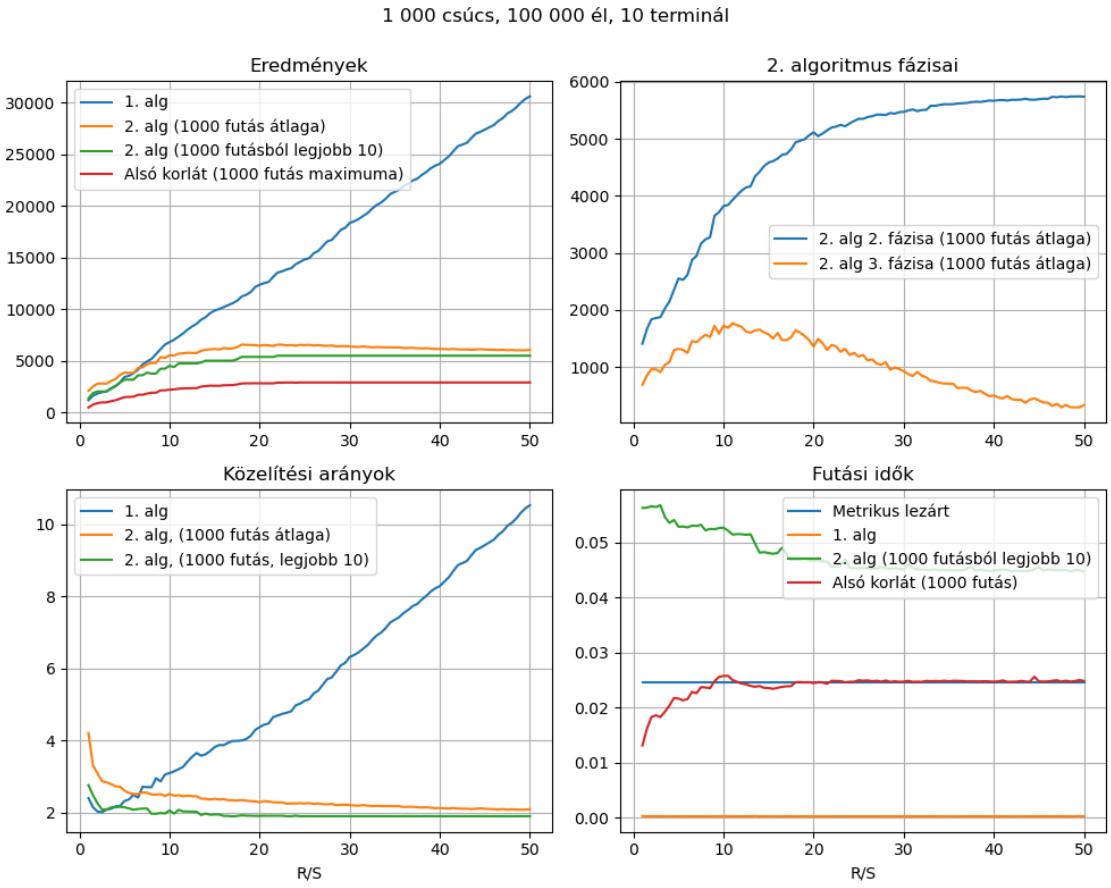
1 000 csúcs, 32 000 él, 10 terminál



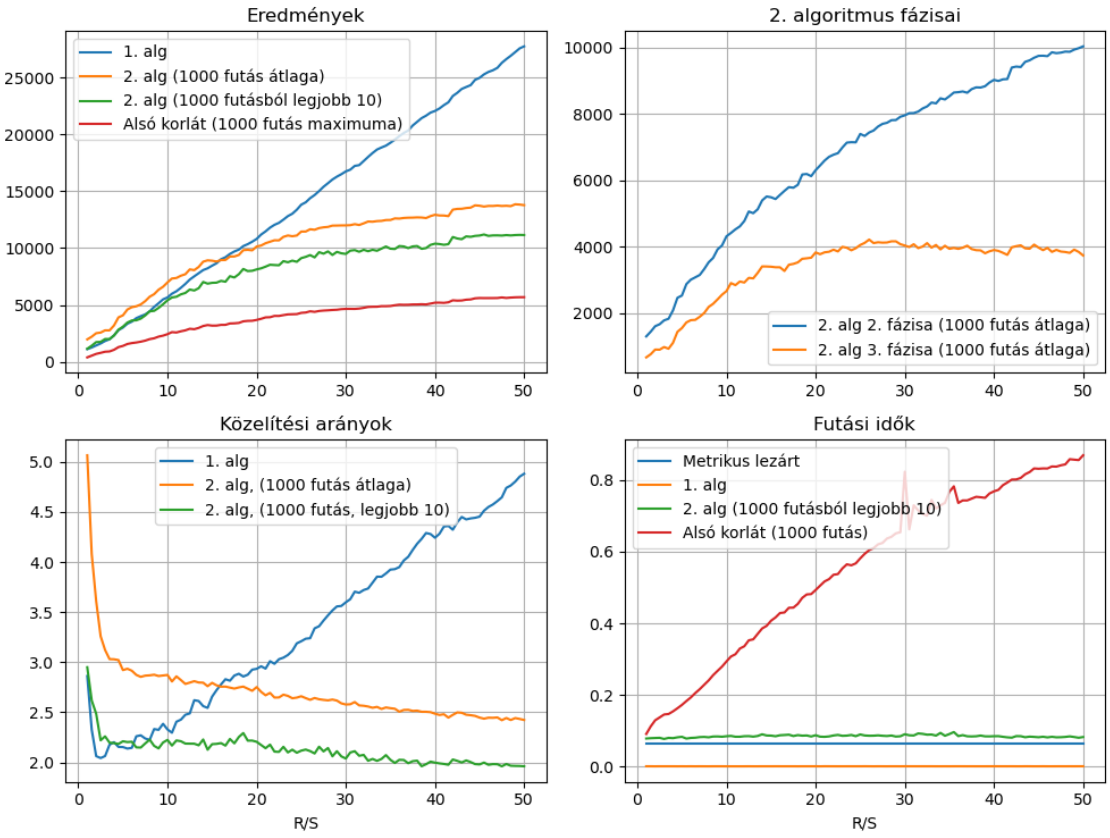
1 000 csúcs, 32 000 él, 32 terminál



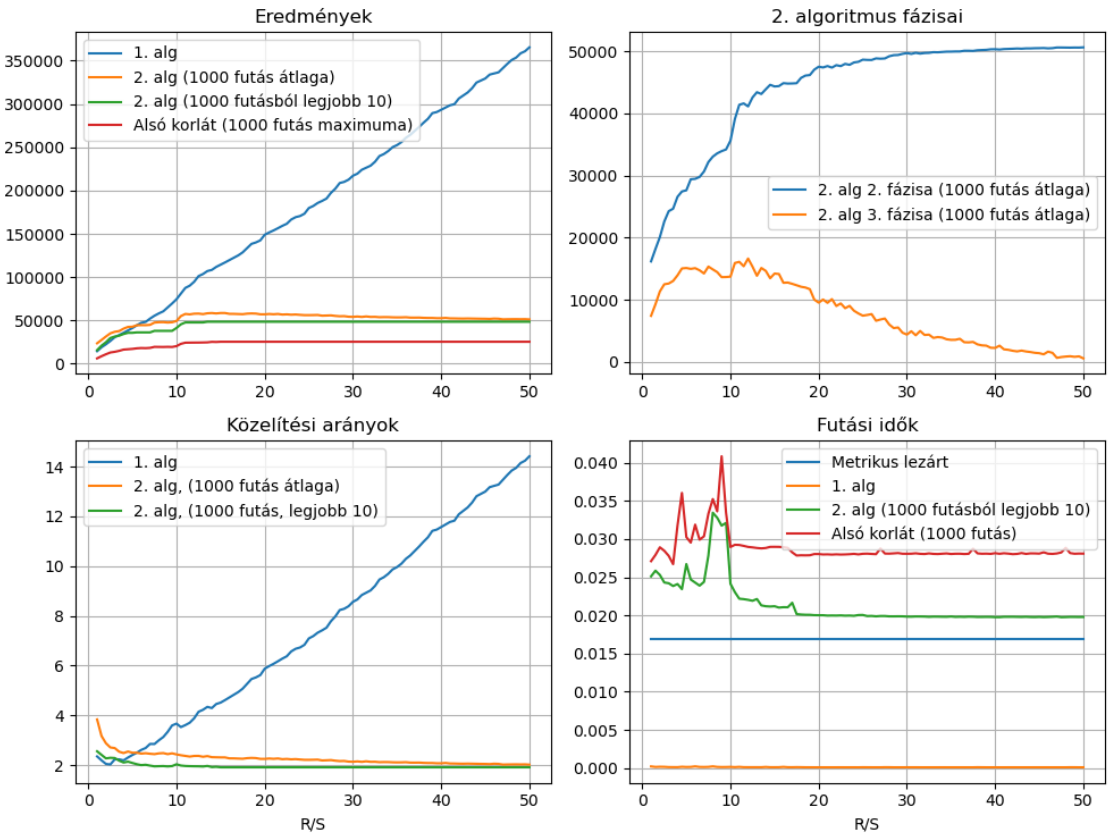

1 000 csúcs, 100 000 él, 7 terminál
Eredmények
1. alg
2. alg (1000 futás átlaga)
2. alg (1000 futásból legjobb 10)
Alsó korlát (1000 futás maximuma)
2. algoritmus fázisai
2. alg 2. fázisa (1000 futás átlaga)
2. alg 3. fázisa (1000 futás átlaga)
Közelítési arányok
1. alg
2. alg, (1000 futás átlaga)
2. alg, (1000 futás, legjobb 10)
Futási idők
Metrikus lezárt
1. alg
2. alg (1000 futásból legjobb 10)
Alsó korlát (1000 futás)
R/S
R/S


1 000 csúcs, 100 000 él, 10 terminál
Eredmények
1. alg
2. alg (1000 futás átlaga)
2. alg (1000 futásból legjobb 10)
Alsó korlát (1000 futás maximuma)
2. algoritmus fázisai
2. alg 2. fázisa (1000 futás átlaga)
2. alg 3. fázisa (1000 futás átlaga)
Közelítési arányok
1. alg
2. alg, (1000 futás átlaga)
2. alg, (1000 futás, legjobb 10)
Futási idők
Metrikus lezárt
1. alg
2. alg (1000 futásból legjobb 10)
Alsó korlát (1000 futás)
R/S
R/S

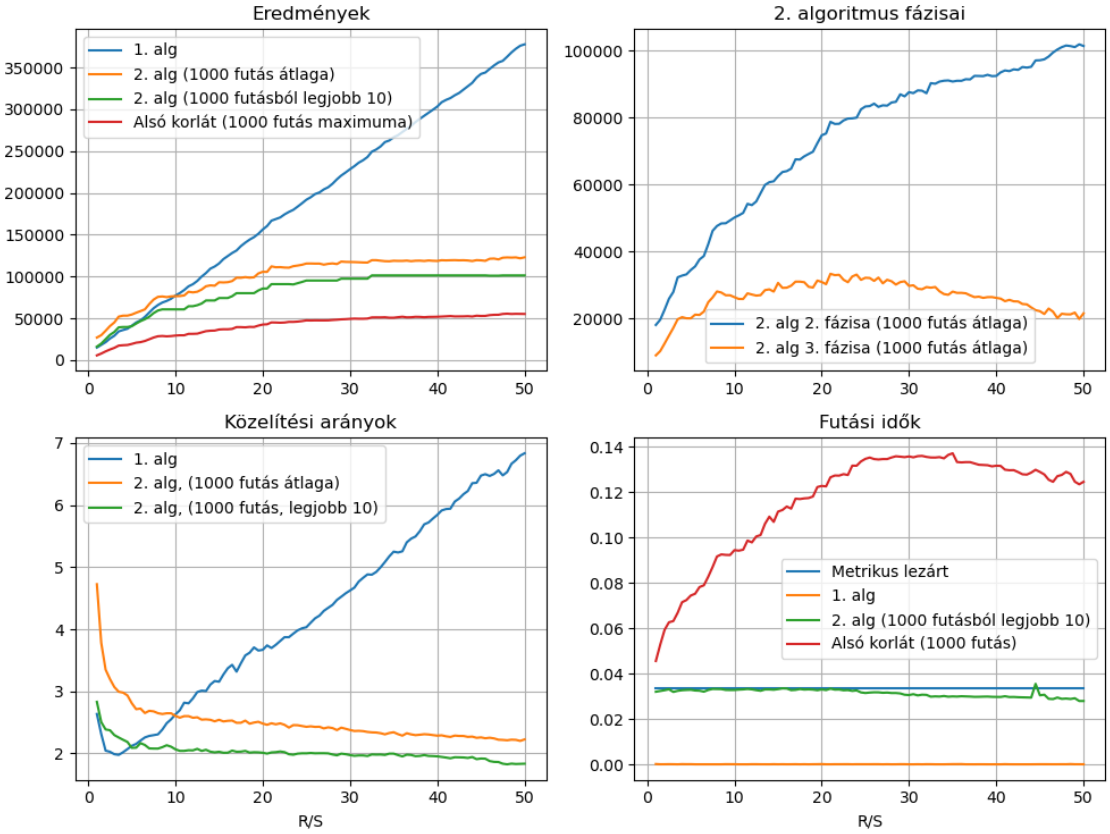
1 000 csúcs, 100 000 él, 32 terminál



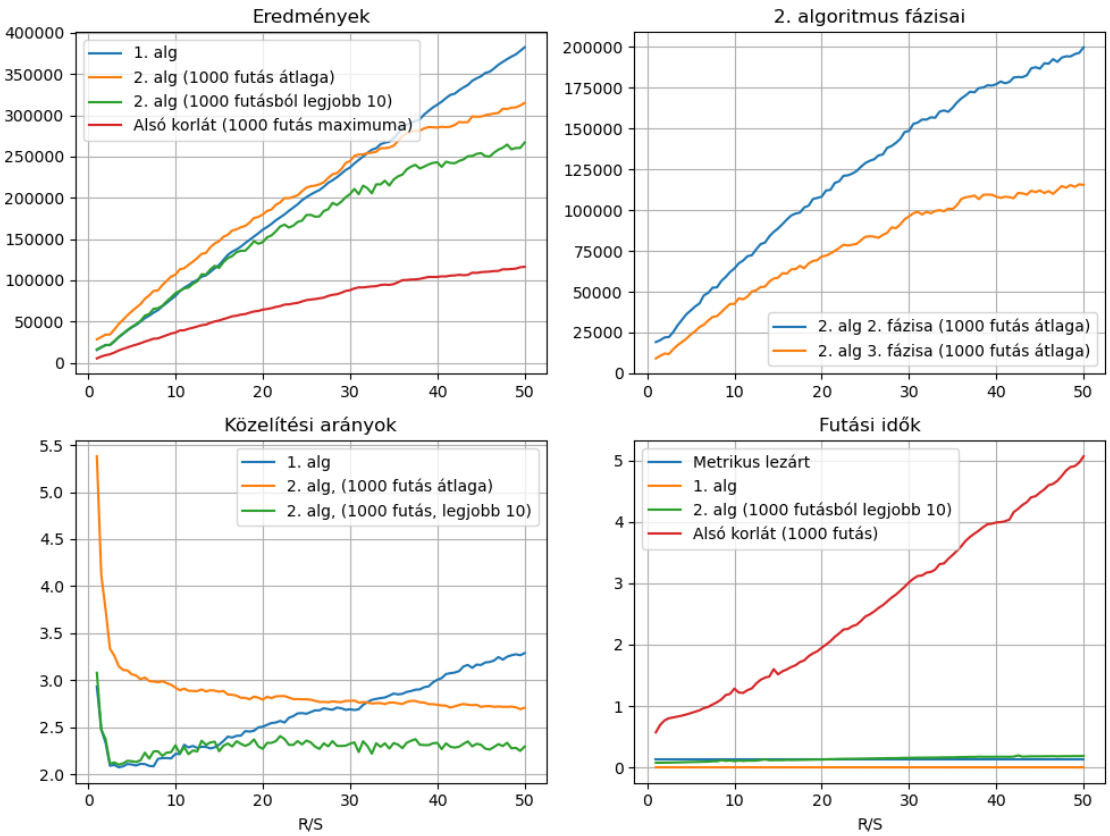
5 000 csúcs, 42 000 él, 8 terminál

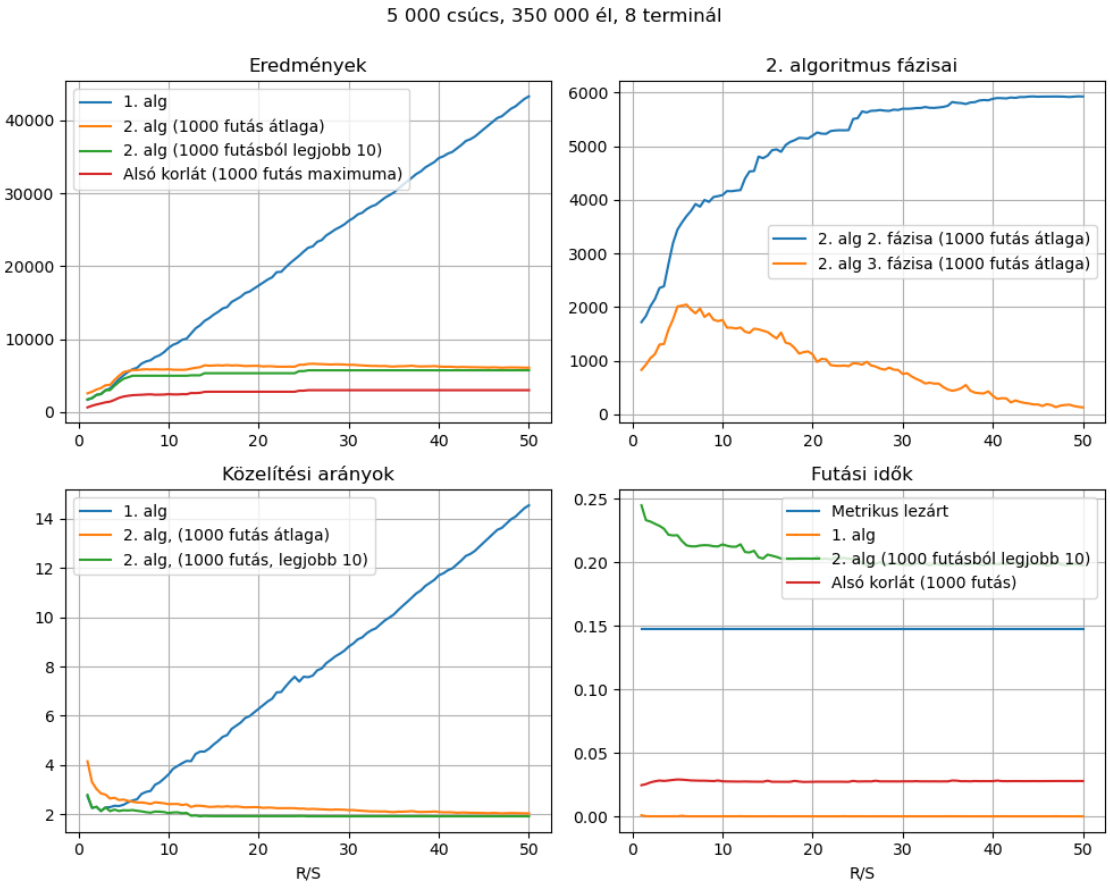


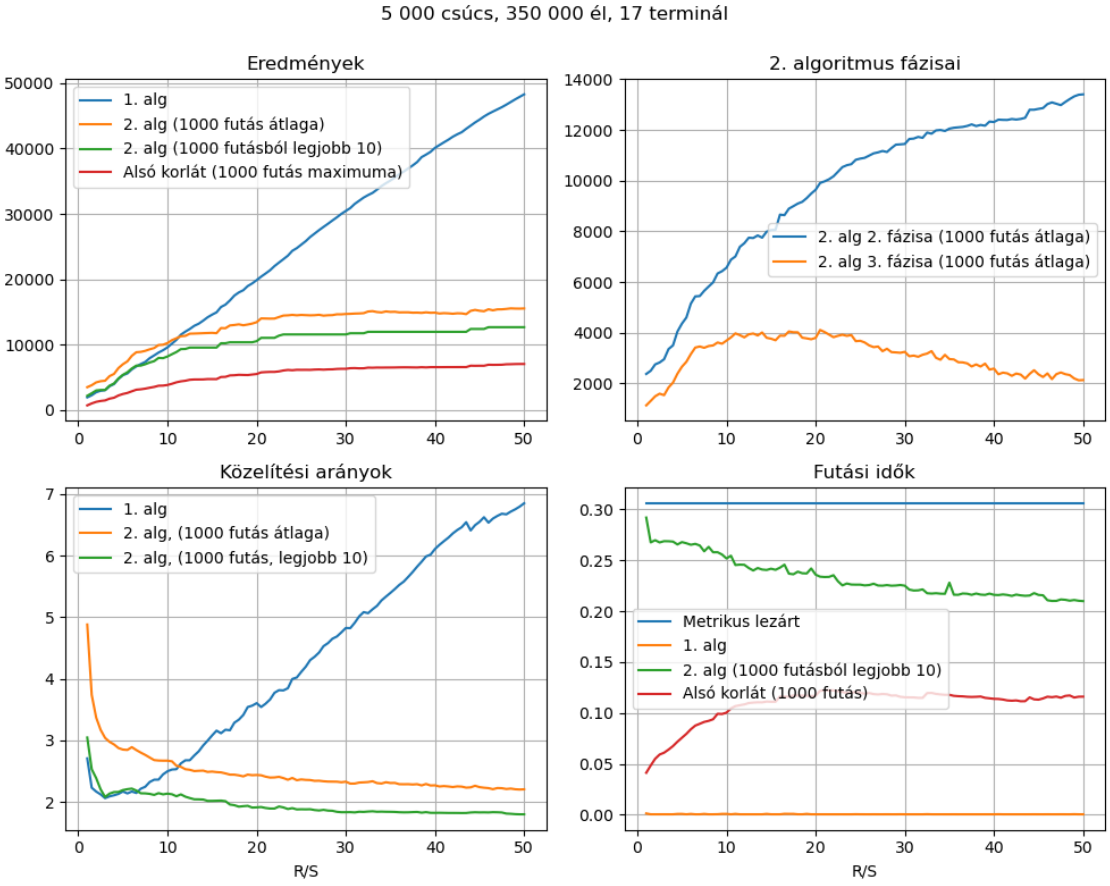
5 000 csúcs, 42 000 él, 17 terminál



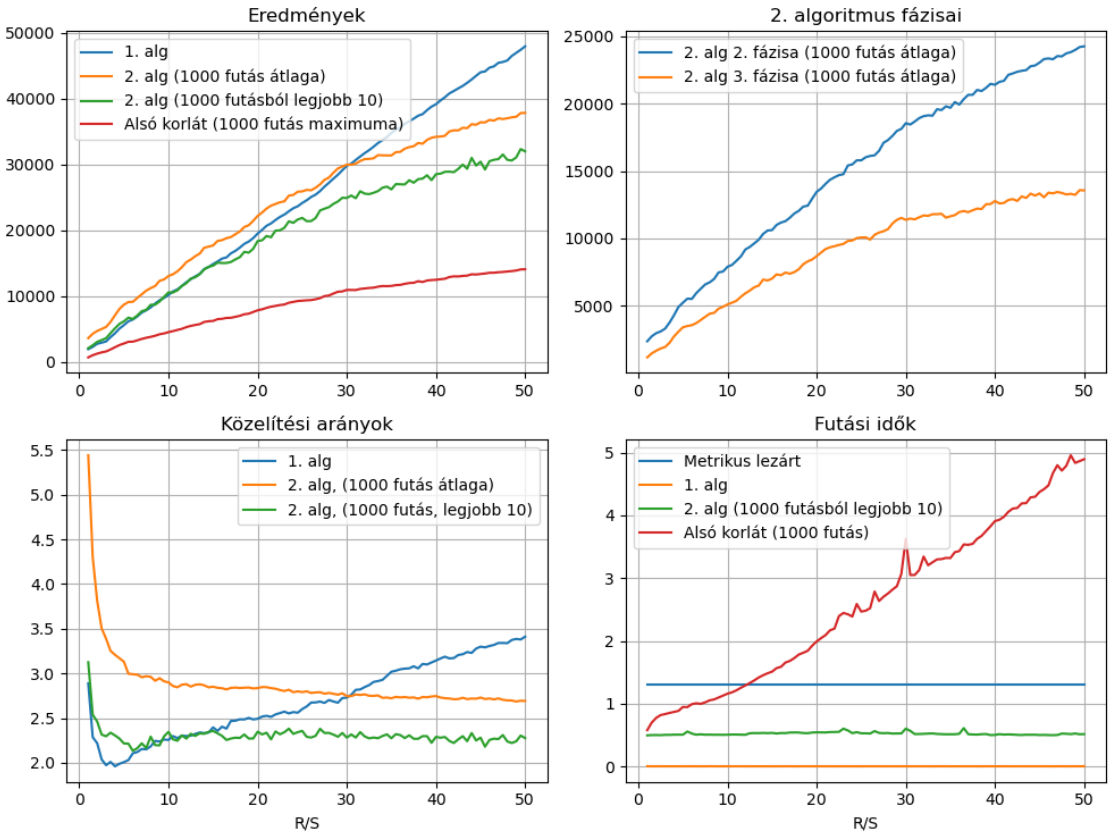
5 000 csúcs, 42 000 él, 70 terminál



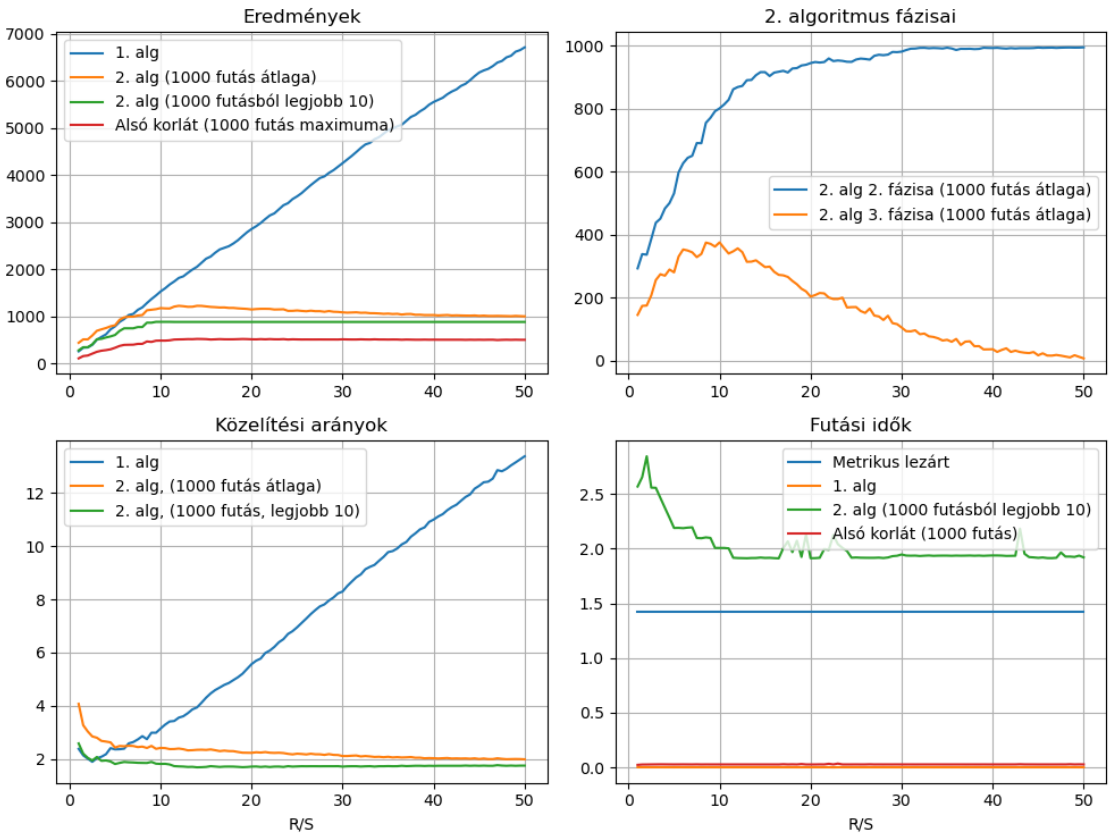

5 000 csúcs, 350 000 él, 8 terminál
Eredmények
1. alg
2. alg (1000 futás átlaga)
2. alg (1000 futásból legjobb 10)
Alsó korlát (1000 futás maximuma)
2. algoritmus fázisai
2. alg 2. fázisa (1000 futás átlaga)
2. alg 3. fázisa (1000 futás átlaga)
Közelítési arányok
1. alg
2. alg, (1000 futás átlaga)
2. alg, (1000 futás, legjobb 10)
Futási idők
Metrikus lezárt
1. alg
2. alg (1000 futásból legjobb 10)
Alsó korlát (1000 futás)
R/S
R/S


5 000 csúcs, 350 000 él, 17 terminál
Eredmények
1. alg
2. alg (1000 futás átlaga)
2. alg (1000 futásból legjobb 10)
Alsó korlát (1000 futás maximuma)
2. algoritmus fázisai
2. alg 2. fázisa (1000 futás átlaga)
2. alg 3. fázisa (1000 futás átlaga)
Közelítési arányok
1. alg
2. alg, (1000 futás átlaga)
2. alg, (1000 futás, legjobb 10)
Futási idők
Metrikus lezárt
1. alg
2. alg (1000 futásból legjobb 10)
Alsó korlát (1000 futás)
R/S
R/S

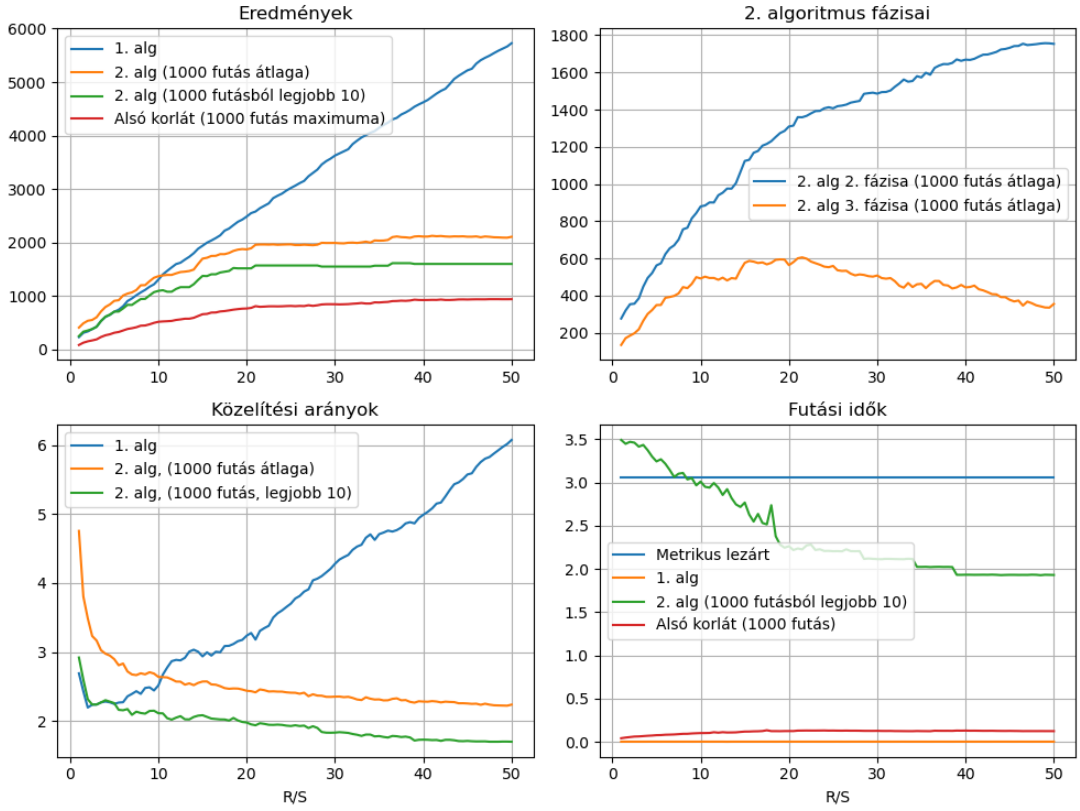
5 000 csúcs, 350 000 él, 70 terminál



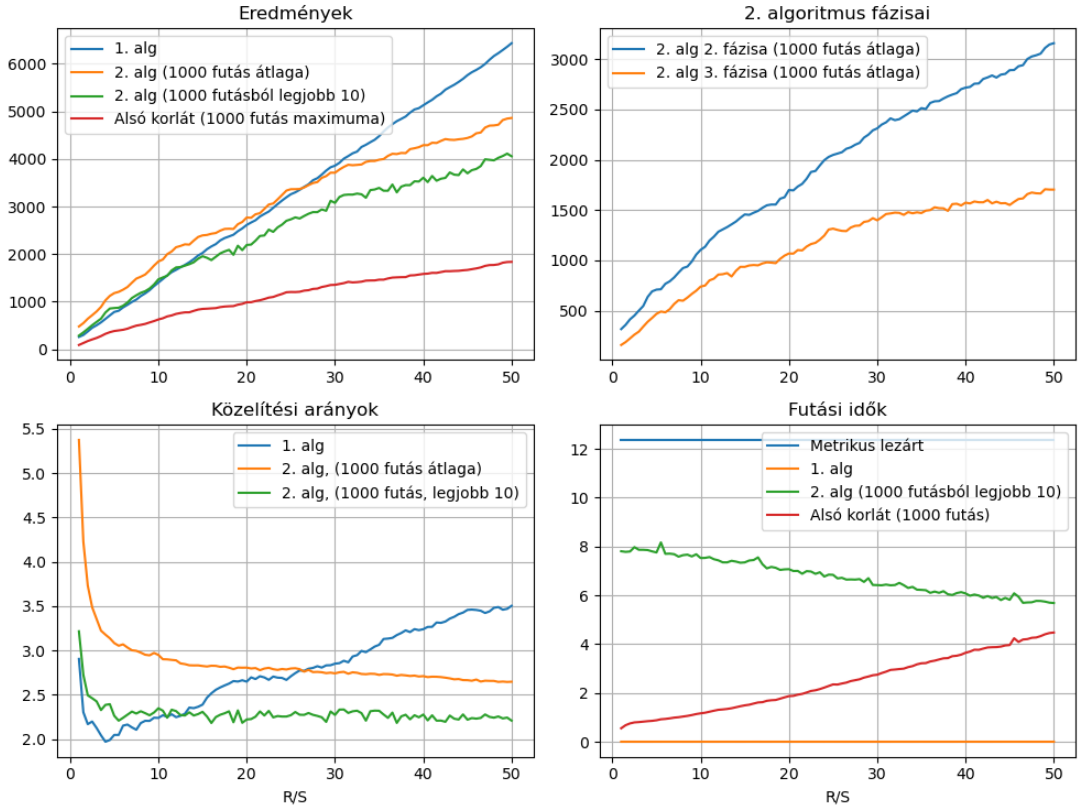
5 000 csúcs, 2 500 000 él, 8 terminál



5 000 csúcs, 2 500 000 él, 17 terminál



5 000 csúcs, 2 500 000 él, 70 terminál



Észrevehetjük, hogy a 2. algoritmus kevés terminál esetén teljesít jobban. Minél több a terminál, annál nagyobb $\frac{R}{S}$ aránynál "előzi be" az 1. algoritmust.

Láthatjuk, hogy a 2. algoritmus 3. fázisának várható költsége 0-hoz közelít $\frac{R}{S}$ növekedésével. Ez azzal magyarázható, hogy nagyobb $\frac{R}{S}$ esetén várhatóan több terminált veszünk be $\mathcal{R}'$-ve, így csökken a vevők $\mathcal{R}'$-től vett távolságainak várható összege.

Nagy terminálszám esetén az alsó korlátot kiszámító algoritmus futási idejének növekedése szintén azzal magyarázható, hogy várhatóan nagyobb $\mathcal{R}'$-re kell minimális feszítőfákat számolnunk. Bizonyos $\frac{R}{S}$ arány felett (amikor már nagy valószínűséggel az összes terminál $\mathcal{R}'$-ben lesz) láthatjuk, hogy a futási idő eléri a maximumát.

Az 1. és 2. algoritmus minimumát véve a várható közelítési arány egyik esetben sem haladta meg a 3-at. Ha pedig a 2. algoritmus 1000 futásából vesszük a legjobb 10 felső becslésű minimumát, mindegyik esetben $2,5$ alatt volt az arány.

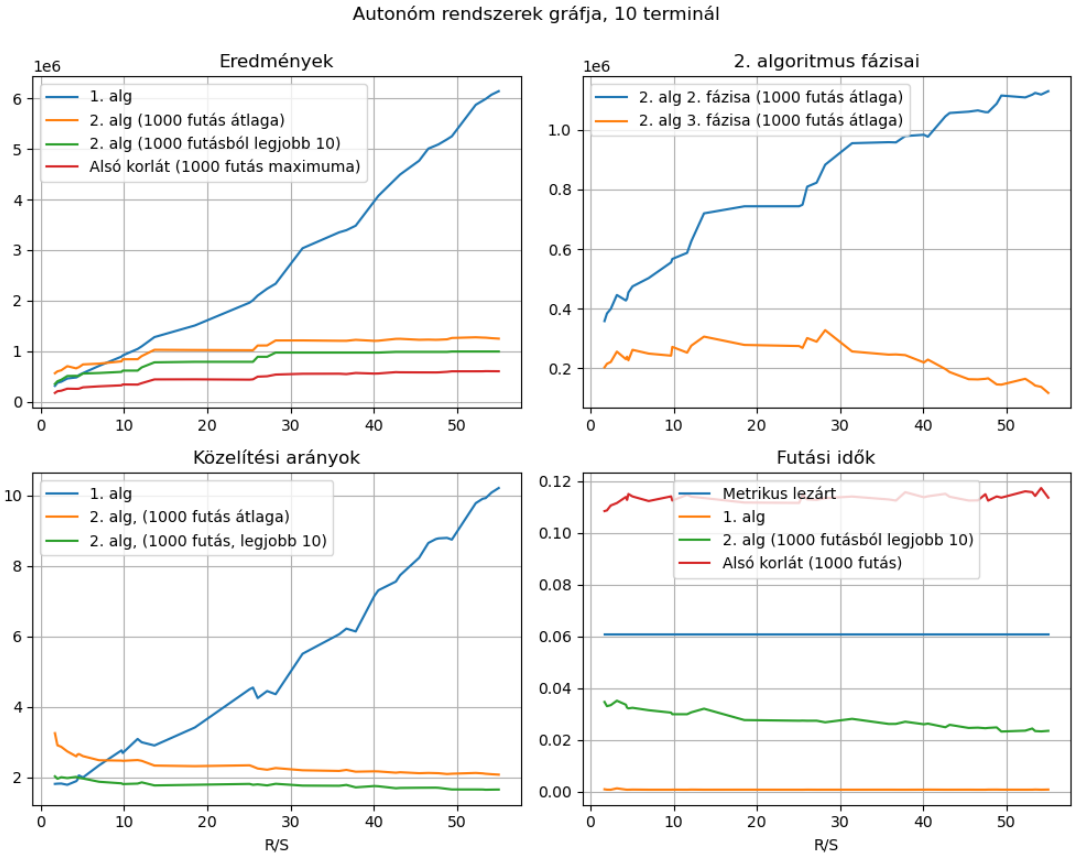
4.4. Tesztelés valódi gráfon

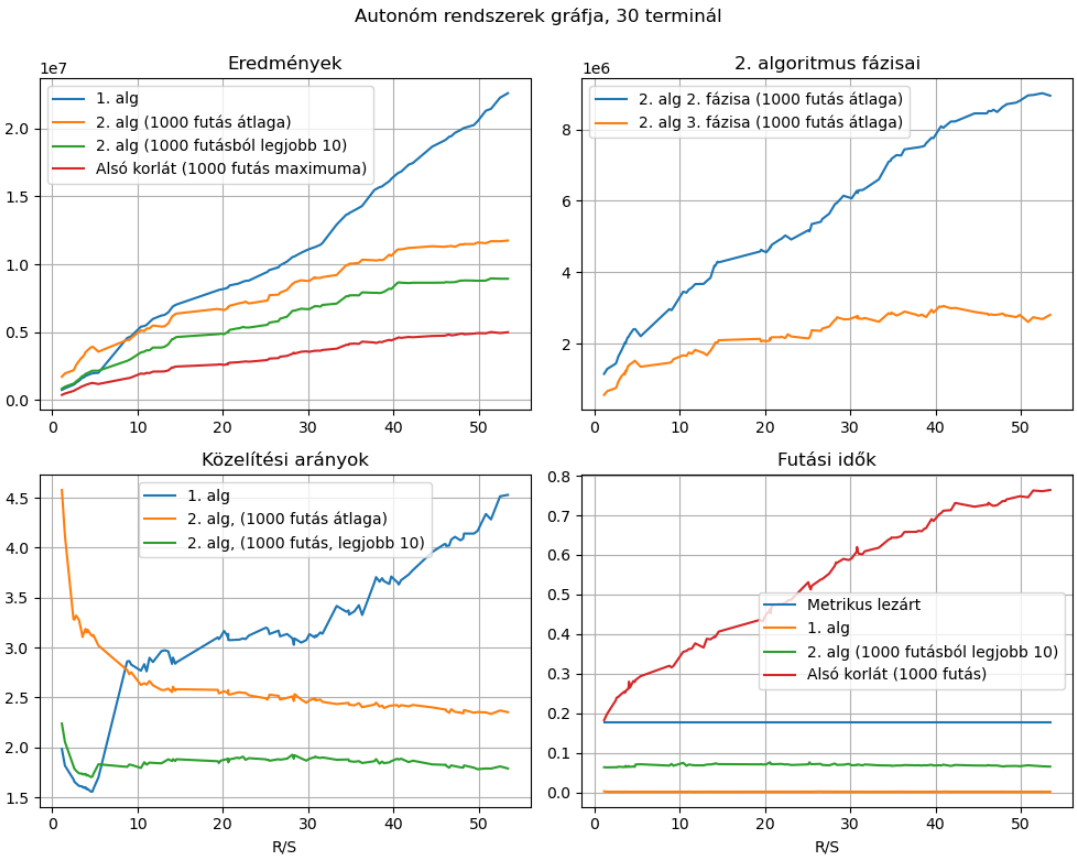
Mivel az Erdős-Rényi véletlen gráfok nem állnak közel a valósághoz, érdemes letesztelnünk az algoritmusokat valós adatokon alapuló gráfokon is.

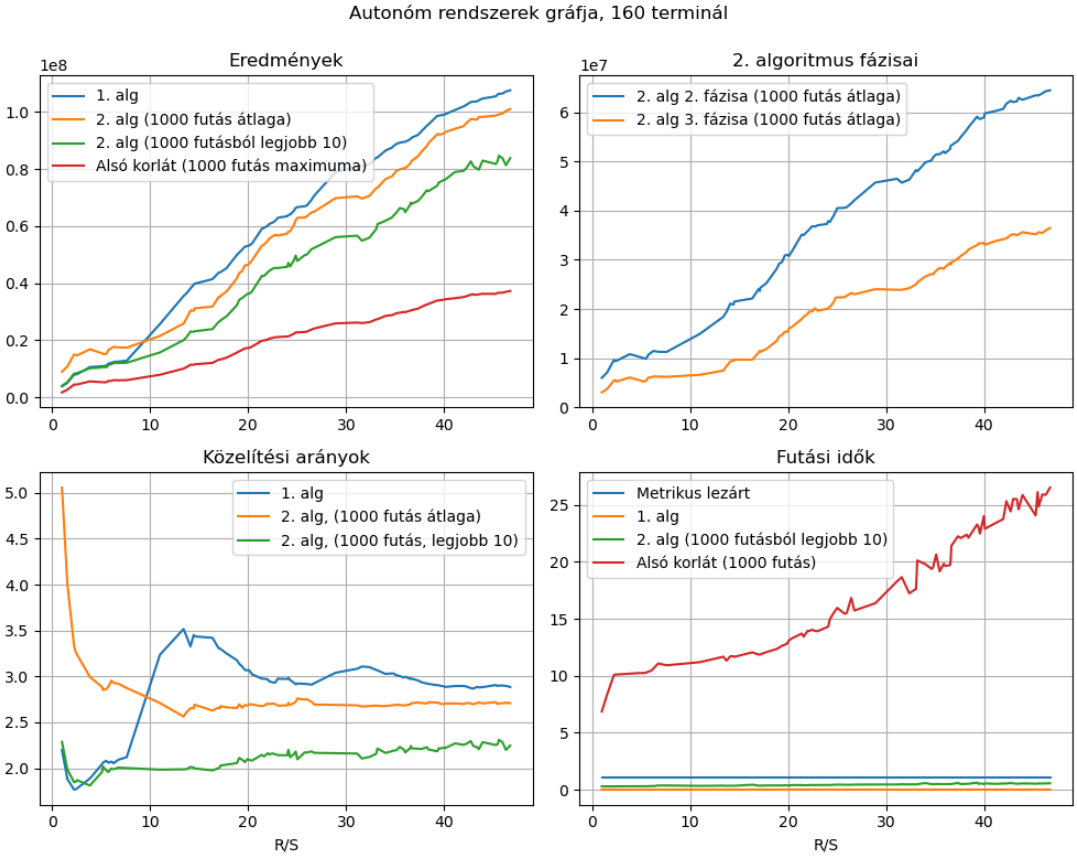
Vegyük például az internet autonóm rendszereinek gráfját 2007-ből, a `https://snap.stanford.edu/data/as-Caida.html` címről. Ez egy 26475 csúcsú, 106762 élű gráf.

Az élköltségeket továbbra is a $[0, 100)$ intervallumon vett egyenletes eloszlásból generáltam. A csúcsok közül egyenletesen véletlenszerűen kiválasztottam 10, 30 illetve 160 terminált. A b_{in} és b_{out} értékeket lognormális eloszlásból választottam, mivel ez jobban tükrözi a valóságot, majd egyenletesen véletlenszerűen választott terminálok b_{in} értékeihez adtam hozzá szintén lognormális eloszlásból vett értékeket.

Láthatjuk, hogy valódi gráfon sem kapunk rosszabb közelítési arányokat.




Autonóm rendszerek gráfja, 30 terminál
Eredmények
1e7
1. alg
2. alg (1000 futás átlaga)
2. alg (1000 futásból legjobb 10)
Alsó korlát (1000 futás maximuma)
2. algoritmus fázisai
1e6
2. alg 2. fázisa (1000 futás átlaga)
2. alg 3. fázisa (1000 futás átlaga)
Közelítési arányok
1. alg
2. alg, (1000 futás átlaga)
2. alg, (1000 futás, legjobb 10)
R/S
Futási idők
Metrikus lezárt
1. alg
2. alg (1000 futásból legjobb 10)
Alsó korlát (1000 futás)
R/S


Autonóm rendszerek gráfja, 160 terminál
Eredmények
1e8
1. alg
2. alg (1000 futás átlaga)
2. alg (1000 futásból legjobb 10)
Alsó korlát (1000 futás maximuma)
2. algoritmus fázisai
1e7
2. alg 2. fázisa (1000 futás átlaga)
2. alg 3. fázisa (1000 futás átlaga)
Közelítési arányok
1. alg
2. alg, (1000 futás átlaga)
2. alg, (1000 futás, legjobb 10)
R/S
Futási idők
Metrikus lezárt
1. alg
2. alg (1000 futásból legjobb 10)
Alsó korlát (1000 futás)
R/S

4.5. Összegzés, további lehetőségek

Összességében tehát $2,5$-ös közelítési arányt tudtunk elérni a gyakorlatban. Ezt ugyan elméletben nem bizonyítottuk, de sokféle különböző gráfon tesztelve ezek ígéretes eredmények, melyek jelentős előrelépést hozhatnak az eddigi $3,55$-ös elméleti arányhoz képest.

Már említettem annak a lehetőségét, hogy a 2. algoritmus 2. fázisában egy jobb közelítési arányú Steiner-fa algoritmust is használhatunk. Ez a végső közelítési arányon javítana, a futási időn viszont rontana, így ez egy "trade-off" amit mérlegelnünk kell.

A [6] cikkben megfogalmazott 3. algoritmus is érdekes lehet. Ez annyiban tér el a 2. algoritmustól, hogy $|\mathcal{R}'| \leq \log n$ esetén közelítés helyett pontos Steiner-fát számít ki a Dreyfus-Wagner algoritmussal [4]. Ezzel javítottak az elméleti közelítési arányon: $3,79$-ről $3,55$-re sikerült lecsökkenteniük. Mivel a gyakorlatban nem javít sokat, a szakdolgozatom keretein belül nem foglalkoztam ezzel, viszont a jövőbeli kutatások során érdemes lehet ezt a lehetőséget is figyelembe venni.

És végül, de nem utolsósorban, érdemes megjegyezni, hogy az algoritmusaink jól párhuzamosíthatók. Az egymástól független Dijkstrákat és Kruskalokat, valamint a Steiner-fák elkészítését szét tudjuk osztani különböző szálak között. Így párhuzamosítással a futási idők akár a töredékükre is csökkenthetők.

Irodalomjegyzék

[1] Richard P. Anstee. „A polynomial algorithm for b-matchings: An alternative approach”. *Information Processing Letters* 24.3 (1987), 153–157. old. DOI: 10.1016/0020-0190(87)90178-5.

[2] Birszki Levente. „Adatstruktúrák összehasonlító elemzése”. BSc szakdolgozat. Eötvös Loránd Tudományegyetem, 2023. URL: https://www.math.elte.hu/thesisupload/thesisfiles/2023bsc_alkmat3y-u3j6yk.pdf.

[3] Jaroslaw Byrka, Fabrizio Grandoni, Thomas Rothvoß és Laura Sanità. „An improved LP-based approximation for steiner tree”. *Proceedings of the Forty-Second ACM Symposium on Theory of Computing*. 2010, 583–592. old. DOI: 10.1145/1806689.1806769.

[4] Stuart E. Dreyfus és Robert A. Wagner. „The Steiner problem in graphs”. *Networks* 1.3 (1971), 195–207. old. DOI: 10.1002/net.3230010302.

[5] Jack Edmonds. „Paths, Trees, and Flowers”. *Canadian Journal of Mathematics* 17 (1965), 449–467. old. DOI: 10.4153/CJM-1965-045-4.

[6] Friedrich Eisenbrand, Fabrizio Grandoni, Gianpaolo Oriolo és Martin Skutella. „New Approaches for Virtual Private Network Design”. *SIAM Journal on Computing* 37.3 (2007), 706–721. old. DOI: 10.1137/060654827.

[7] Erdős Pál és Rényi Alfréd. „On random graphs I.” *Publicationes Mathematicae Debrecen* 6 (1959), 290–297. old. DOI: 10.5486/PMD.1959.6.3-4.12.

[8] Shimon Even, Alon Itai és Adi Shamir. „On the complexity of time table and multi-commodity flow problems”. *16th annual symposium on foundations of computer science (sfcs 1975)*. IEEE. 1975, 184–193. old. DOI: 10.1109/SFCS.1975.21.

[9] J. Andrew Fingerhut, Subhash Suri és Jonathan S. Turner. „Designing Least-Cost Nonblocking Broadband Networks". *Journal of Algorithms* 24.2 (1997), 287–309. old. ISSN: 0196-6774. DOI: 10.1006/jagm.1997.0866.

[10] Navin Goyal, Neil Olver és F. Bruce Shepherd. „The VPN conjecture is true". *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*. STOC '08. 2008, 443–450. old. ISBN: 9781605580470. DOI: 10.1145/1374376.1374440.

[11] Anupam Gupta, Jon Kleinberg, Amit Kumar, Rajeev Rastogi és Bulent Yener. „Provisioning a virtual private network: a network design problem for multi-commodity flow". *Proceedings of the thirty-third annual ACM symposium on Theory of computing*. 2001, 389–398. old. DOI: 10.1145/380752.380830.

[12] Anupam Gupta, Amit Kumar és Tim Roughgarden. „Simpler and better approximation algorithms for network design". *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*. 2003, 365–372. old. DOI: 10.1145/780542.780597.

[13] Te C. Hu. „Multi-commodity network flows". *Operations research* 11.3 (1963), 344–360. old. DOI: 10.1287/opre.11.3.344.

[14] Giuseppe Italiano, Stefano Leonardi és Gianpaolo Oriolo. „Design of trees in the hose model: The balanced case". 34. köt. 6. 2006, 601–606. old. DOI: 10.1016/j.orl.2005.09.005.

[15] Lawrence Kou, George Markowsky és Leonard Berman. „A fast algorithm for Steiner trees". *Acta informatica* 15.2 (1981), 141–145. old. DOI: 10.1007/BF00288961.

MI alapú eszközök használatról szóló nyilatkozat

Alulírott Kovács Alex nyilatkozom, hogy a szakdolgozatom elkészítése során az alább felsorolt feladatok elvégzésére a megadott MI alapú eszközöket alkalmaztam:

Feladat	**Felhasznált eszköz**	**Felhasználás helye**
Python kód generálása	GPT-4.1	4. fejezet
Nyelvhelyesség ellenőrzése	Writefull	Teljes dolgozat

A felsoroltakon túl más MI alapú eszközt nem használtam.