BIRKÓZÁS A BONYOLULTSÁGGAL: HATÉKONY ALGORITMUSOK[1]

Rónyai Lajos
az MTA levelező tagja, tudományos tanácsadó, MTA SZTAKI, BME – ronyai@sztaki.hu

Most jön a végleges egyensúly meg a kibontakozás, a bonyolultságok hirtelen elszégyellik magukat, és elenyésznek.
Füst Milán[2]

A ***bonyolultság*** fontos szakszó a számításelméletben. Rendszerint valamilyen számítási erőforrással (idő, tár, kommunikáció) kapcsolatban használatos, és egy feladat/algoritmus erőforrásigényére, illetve felhasználására utal. Így beszélhetünk egy számítási feladat időbonyolultságáról vagy éppen egy algoritmus (számítási eljárás, módszer) tárbonyolultságáról. Az előbbi esetben a feladat megoldásához minimálisan szükséges számítási időre gondolunk, míg a másodikban az algoritmus által felhasznált tárterület nagyságára. A számítástudományban vannak a bonyolultságnak a köznapi alapjelentéshez (az egyszerűség hiánya) közelebb álló technikai megfelelői is: ilyen a programok, programszerű struktúrák ún. leírási bonyolultsága és a Kolmogorov-bonyolultság.

Az elnevezés főként az első két értelemben, az idő- és tárigénnyel összefüggésben vált fontossá. Rendszeres vizsgálata a hatvanas évek közepén kezdődött, és ma már köteteket töltenek meg az ilyen tárgyú eredmények (például Papadimitriou, 1999). A számításelmélet legismertebb, leginkább reflektorfényben levő nyitott problémája, a P=NP? kérdés is az első értelmezés szerinti bonyolultság természetét tudakolja.

A hatékony, vagyis minél alacsonyabb bonyolultságú algorimusok tervezése, kutatása elméleti és gyakorlati szempontból egyaránt nagy jelentőséggel bíró irányzat. Innen választottam néhány – szerintem fontos és érdekes – példát. Nevezetes algoritmusokra szeretném felhívni a figyelmet, és ennek kapcsán érzékeltetni valamit a terület jellegéből, szépségéből és interdiszciplináris kapcsolataiból. Olyan gondolatokról, módszerekről lesz szó, amelyek már széles körben bizonyították erejüket, és amelyek a szülőhelyüktől távolabbi területek művelői számára is hordozhatnak tanulságokat.

Terjedelmi korlátok és a folyóirat profilja miatt mellőznöm kell a technikai részleteket. Kérem az Olvasót, nézze el az ebből az egyszerűsítésből eredő fogyatékosságokat. Cserébe talán különösebb bonyodalmak nélkül vethet pillantást az algoritmusok világára. A téma iránt behatóbban érdeklődőknek a következő munkákat ajánlom (Cormen, 1999; Lovász, 1987; Papadimitriou, 1999 és Rónyai, 2000).

Kell egy jó feladat – az LLL-algoritmus

A nyolcvanas évek elején igazi tudományos szenzációként robbant a hír: Arjen K. Lenstra, Hendrik W. Lenstra Jr. és Lovász László megtalálták az algebrai számítások világának Szent Grálját: hatékony (pontosabban mondva

[1] Köszönet illeti Krámli Andrást és Vámos Tibort a kézirathoz fűzött megjegyzéseikért.
[2] *A feleségem története,* Magvető, 1968. 33. oldal.

polinom idejű[3]) algoritmust adtak racionális együtthatós polinomok felbontására (Lenstra, 1982). Módszerükkel lehetővé vált az $a_0+a_1x+\ldots+a_kx^k$ alakú kifejezések gyors felbontása hasonló formájú egyszerűbb kifejezések szorzatára (már amennyiben ilyen felbontás egyáltalán létezik). Itt az $a_0, a_1,\ldots,a_k$ racionális számok, az x pedig egy változó. A felfedezés jelentőségét aligha lehet túlbecsülni. Az algebrai kifejezések szorzattá alakításának hasznossága nyilvánvaló a matematikával foglalkozó ember számára, legyen szó házi feladatot megoldó kisdiákról vagy terebélyes számításokat végző mérnökről, kutatóról. A problémával már Sir Isaac Newton is foglalkozott az *Arithmetica Universalis*-ban (1707). Az első véges lépésszámú módszert a csillagász Friedrich von Schubert dolgozta ki (1793). Ezt Leopold Kronecker fedezte újra fel (1882), és az eljárás az ő nevével került be a szakmai köztudatba. Igazán komoly előrelépésre 1969-ig kellett várni. Hans Zassenhaus eljárása sokkal jobb ugyan az elődeinél, de a legrosszabb esetekben még mindig exponenciális ideig zakatol.

Az út az áttöréshez egy geometriai természetű számítási feladaton keresztül vezetett: tegyük fel, hogy van néhány vektorunk, $\mathbf{v}_1, \ldots, \mathbf{v}_m$ valamely többdimenziós valós térből. A $\mathbf{v}_i$ vektorok által meghatározott *rács* az $a_1\mathbf{v}_1+a_2\mathbf{v}_2+\ldots+a_m\mathbf{v}_m$ alakú vektorok összessége, ahol az $a_1,\ldots,a_k$ számok tetszőleges egészek. Ilyen rácsnak tekinthetők a négyzetháló csúcspontjai.

Ha adott egy rács, akkor próbálkozhatunk minél rövidebb nem nulla vektort találni benne. Jól ismert – vagy inkább hírhedt – ilyen természetű feladat a legrövidebb nem nulla vektor megkeresése. Eddig senki sem talált hatékony módszert a feladat megoldására, és a szakértők körében uralkodó vélekedés szerint nem is létezik ilyen.

[3] Egy algoritmus polinom idejű, ha van olyan c>0, hogy n bitből álló bemeneten legfeljebb n^c ideig dolgozik.

Ami a hatékony felbontást illeti, kiderült egyfelől, hogy kevesebb is elég. A felbontási feladathoz kapcsolódó rácsban nem kell feltétlenül a legrövidebb vektort megtalálni, megteszi egy ennél esetleg jóval hosszabb, de még mindig elég rövid vektor is. Másfelől Lovász László ragyogó módszert adott, amivel gyorsan lehetséges ilyen nem túl hosszú vektort lelni. A rácsredukció, illetve LLL-algoritmus néven azóta széles körben ismertté vált eljárása a $\mathbf{v}_i$ vektorok helyett egy másik, viszonylag rövid vektorokból álló rendszert ad, amelyik ugyanazt a rácsot határozza meg, mint kiinduló vektoraink. Az új rendszer legrövidebb vektora pedig legfeljebb $2^{(m-1)/2}$-szer lehet hosszabb, mint a rács legrövidebb vektora, ahol m a befoglaló tér dimenziója. Az eredmény meglepő és értékes vonása, hogy a tér dimenzióján túl a legrövidebb vektor közelítésének jósága nem függ a rácstól magától.

Példaként nézzük a

$$\mathbf{v}_1=(1,0,2002),$$
$$\mathbf{v}_2=(0,1,2001),$$
$$\mathbf{v}_3=(1,2,2500)$$

vektorok által kifeszített rácsot a térben. A Mathematica szimbolikus számítási rendszer LLL-eljárása erre a bemenetre a

$$\mathbf{w}_1=(-1,1,-1),$$
$$\mathbf{w}_2=(-5,-2,4),$$
$$\mathbf{w}_3=(53,236,182)$$

vektorokat adja eredményül. A $\mathbf{v}_i$ vektorok mind hosszabbak 2 000-nél, a $\mathbf{w}_1$ hossza pedig mindössze $\sqrt{3}$. A módszer a közel párhuzamos és hosszú $\mathbf{v}_i$ vektorokat a sokkal rövidebb és jóval kevésbé hegyes szöget bezáró $\mathbf{w}_i$ vektorokkal helyettesítette. Mintha egy virág szirmait nyitotta volna ki a napsugár.

A felfedezésből azt szeretném hangsúlyozni, hogy *új feladatot fogalmaztak meg* (redukált vektorrendszer számítása), *ami egyszerre bizonyult kezelhetőnek és hasz-*

nosnak is. Új, addig nem létezett feladat született – mindjárt a megoldással együtt. A rácsredukció algoritmikus építőkővé vált, ami sok helyen alkalmazható. Ennek megfelelően több szimbolikus számításokra szakosodott programcsomag kínál rácsredukciós eljárást.

A módszert polinomok felbontása mellett sikerrel alkalmazzák egy sor más területen, így például a kriptográfiában, az illetéktelen hozzáféréssel szemben biztonságos kommunikáció tudományában, és diofantikus egyenletek megoldására is. Fontos alkalmazási terepet jelent az ***egész relációk*** keresésére. Itt a feladat a következő: adottak az $r_1,\ldots,r_k$ valós számok; keressünk közöttük fennálló egész lineáris összefüggést. Pontosabban mondva olyan nem csupa nulla $a_1, \ldots,a_k$ egészeket, melyekkel

$$a_1r_1+a_2r_2+\ldots+a_kr_k=0$$

teljesül. Az egész relációk számítógépes keresése egy új, és egyre inkább teret nyerő irányzat, a ***kísérleti matematika*** jeles eszköze (Bailey, 2000). Segítségével több matematikai azonosságot sikerült megsejteni, majd azután szigorúan bizonyítani. Vannak alkalmazásai a kvantumtérelmélet Feynman-diagramjainak vizsgálatában is.

A tudományos haladás természetének megfelelően a rácsredukció módszereit az évek során többen is fejlesztették, finomították, illetve általánosították. Ezek közül – némiképpen hazabeszélve – egyet említek. Ivanyos Gábor és Szántó Ágnes, az MTA SZTAKI munkatársai kiterjesztették a módszert az olyan esetekre is, amikor a vektorainkat tartalmazó térben a távolság felvehet pozitív és negatív értékeket egyaránt (ún. indefinit eset) (Ivanyos, 1996).

Tudás több területről – többtestrendszerek gyors szimulációja

Az itt következő módszerek arra mutatnak példát, hogy különböző ismeretkörökből származó egyszerű gondolatok együttesen milyen hatékony és erőteljes eredményt adhatnak.

Tegyük fel, hogy a térben mozgó n test $p_1,p_2,\ldots, p_n$ (a továbbiakban csak részecskéknek nevezem őket) pályáját szeretnénk követni az időben, ahogy az egymásra gyakorolt erőhatások függvényében mozognak. A hatás lehet gravitáció (Newton-tér), elektrosztatikus erő (Coulomb-tér), kémiai kötési erő és más is. Az ilyen értelemben vett többtestrendszerek mozgására már három részecske esetén sincs pontos matematikai megoldás. Egy sor alkalmazási területen azonban ez a nehézség mit sem csökkenti azok kíváncsiságát, akik igen nagy számú részecske mozgásáról szeretnének képet kapni. Ilyen területek a csillagászat és az égi mechanika (Newton-tér), molekuláris dinamika (elektrosztatikus erők, Lennard-Jones-potenciál stb.), és a folyadékok dinamikája.

Használható matematikai megoldás híján a kutatók gyakran fordulnak a ***számítógépes szimuláció*** eszközéhez. Gépeiken mintegy lejátsszák a kérdéses folyamatot, és közben vizsgálják jellemzőinek alakulását. Ennek a jellegzetesen számítógépes megközelítésnek a fizikai alkalmazásaival foglalkozik Vicsek Tamás írása (Vicsek, 1990).

A többtestrendszerek szimulációjában a következő helyzet meghatározásakor, a következő pillanatfelvétel elkészítésekor az erők számítása az időigényes részfeladat. A p_1 részecskére ható erő a többi részecske erőhatásának (vektor)összege. Például ha gravitációs térben vagyunk, akkor a p_i részecskének a p_1-re gyakorolt hatása ($i > 1$)

$$F_i=G\cdot m_1\cdot m_i\left(\frac{x_i-x_1}{r_i^3},\frac{y_i-y_1}{r_i^3},\frac{z_i-z_1}{r_i^3}\right),$$

ahol (x_j,y_j,z_j) a p_j helyvektora, m_j a p_j részecske tömege, G a gravitációs állandó, végül $r_i=\sqrt{(x_1-x_i)^2+(y_1-y_i)^2+(z_1-z_i)^2}$ a p_1 és p_i közötti távolság. A p_1-re ható teljes erő pedig $\mathbf{F}_2+\mathbf{F}_3+\ldots+\mathbf{F}_n$. Innen látható, hogy a

részecskék helyének és tömegének ismeretében az egy részecskére ható erő n-nel arányos számú művelettel (összeadás, kivonás, szorzás, osztás, négyzetgyökvonás) megkapható. Ennek megfelelően az n darab erő kiszámítása legfeljebb n^2 nagyságrendű műveletet jelent. Hasonló a helyzet más (nem gravitációs) erők esetén is. Egyszerű négyzetes futási idejű algoritmusunk van tehát. Látszólag ideális a helyzet, hiszen alacsony időbonyolultságú módszerrel rendelkezünk. Ez az időigény azonban túl soknak bizonyul, amikor milliós nagyságrendű részecskéből álló rendszereket szeretnénk vizsgálni. Egy, a galaxisok keletkezésével, eloszlásával kapcsolatos szimulációban 17 000 000 részecske (csillag) mozgását szerették volna követni hatszáz időlépésen keresztül. Az imént vázolt módszerrel a szimuláció több mint egy esztendőt igényelt volna egy $5 \cdot 10^9$ művelet/másodperc sebességű gépen (512 processzoros Intel Delta, ideális párhuzamosítással). Az alább bemutatandó módszerrel a feladatot röpke huszonnégy óra futási idővel meg lehetett oldani.

A négyzetes bonyolultság csökkentésére irányuló törekvések igen szép és gyors (közelítő) algoritmusokhoz vezettek. Ezek egyike a Joshua Barnes és Piet Hut által 1986-ban javasolt ***Barnes–Hut-algoritmus*** (Barnes, 1986). Ennek a számításigénye $O(n \log n)$ művelet – az n részecske eloszlására vonatkozó valószerű egyenletességi feltételek mellett.

A Barnes–Hut-algoritmus a következő fizikai jellegű egyszerűsítő ötletet használja: ha a részecskék valamely S részhalmaza a p_i-től távoli kis kockában elfér, akkor az S együttes hatását p_i-re úgy számíthatjuk, hogy S-et helyettesítjük a tömegközéppontjába tett össztömegével. Ezt az egyszerűsítő-közelítő gondolatot már Newton is alkalmazta.

A számítások gyors szervezését a számítógépes grafika/geometria területén népszerű ***nyolcfa*** (angolul ***octtree***) adatszerkezet szolgálja. A nyolcfa gyökere egy C kocka, ami tartalmazza az összes, a szimulációban részt vevő részecskét. A gyökérnek nyolc fia van. Ezek a $C_1, \ldots, C_8$ kockák, amiket úgy kapunk, hogy a C-t a középpontján átmenő, a lapjaival párhuzamos síkok mentén szétvágjuk. A C_i kockákat azután ugyanígy felvágjuk kisebb darabokra. Ezzel a finomítással akkor állunk meg, amikor a keletkező kis dobozokban (kockákban) a részecskék száma legfeljebb 1.

A Barnes–Hut-algoritmus első fázisában nyolcfát építünk a $p_1, \ldots, p_n$ részecskékhez. Az egyenletességi feltevés miatt a fa szintjeinek számára logn-nel arányos felső korlát adható, és emiatt a fa konstrukciójának időigénye $O(n \log n)$ lesz. A következő menetben a fát alkotó kisebb kockáktól a nagyobbak felé haladva sorban kiszámoljuk az egyes kockák (pontosabban a bennük levő részecskék) tömegközéppontját és össztömegét.

Az utolsó, a harmadik fázis a tulajdonképpeni erőszámítás. Minden egyes i-re meghatározzuk (közelítőleg) a p_i részecskére ható $\mathbf{G}_i$ erőt. Ezt a nagyobb dobozoktól a kisebbek felé haladva tesszük. A munkát a gyökérrel, a teljes rendszert befoglaló C kockával kezdjük. Tegyük fel, hogy éppen az N doboznál tartunk. Ha N-ben csak egy részecske van, akkor ennek a hatását hozzáadjuk $\mathbf{G}_i$-hez. Nézzük most azt az esetet, amikor N-ben több részecske található! Legyen az N kocka élhossza d, a távolsága p_i-től pedig D. Itt szerepet játszik még egy $0 < \Theta < {}^1/_2$ paraméter, ami a pontosságot szabályozza. Először megvizsgáljuk, hogy ${}^d/_D < \Theta$ teljesül-e. Ha igen, akkor az egyszerűsítő ötletet alkalmazzuk: az N-beli részecskék a p_i-től való távolságukhoz képest kicsi csomóban helyezkednek el, így hatásukat helyettesítjük a tömegközéppontjukban elhelyezkedő össztömegükkel. Az első fázis eredményeként ezeket a jellemzőket itt már ismerjük. Ha a tesztnél a válasz nemleges, akkor N helyett a gyermekeit vesszük. Ekkor az N járuléka $\mathbf{G}_i$-hez a nyolc fia járulékának

összege lesz. Megmutatható, hogy a fa egy szintjén legfeljebb konstans (ami Θ-tól függ, de N-től nem) számú csúcsra lehet ***nem*** a válasz. Ebből következik a kedvező korlát a futási időre.

A módszer három egyszerű gondolat meghökkentően hatékony ötvözete. Az első a ***fizikai*** egyszerűsítő ötlet, amiről már esett szó. A második a súlypontszámítás szerencsés ***geometriája***: az N dobozba eső részecskék tömegközéppontja és össztömege gyorsan megkapható pusztán a 8 gyermekének tömegközéppontjából és össztömegéből. Végül pedig a számításokat okos, hatékony rendbe szervező ***algoritmikus*** konstrukciót a nyolcfa adatszerkezetet említhetjük.

Leslie Greengard és Vladimir Rokhlin nevéhez fűződik a nyolcvanas évek közepén kidolgozott ***gyors multipólus-módszer*** (szokásos rövidítéssel: FMM) (Greengard, 1987; Board, 2000). Ez (Newton- és Coulomb-terekben) az erő helyett a potenciál becslésére ad közvetlen módszert, ami elegendő, hiszen a potenciál elég jó minőségű közelítéséből az erő (a potenciál negatív gradiense) jól becsülhető. Greengard és Rokhlin Taylor-típusú sorfejtéseket használnak a potenciál közelítésére. A számítások szervezését itt is a nyolcfa adatszerkezet segíti. A gyakorlatban az FMM valamivel lassúbb, mint a Barnes–Hut-algoritmus, ám cserébe jóval pontosabb eredményeket ad.

A rangos ***Computing in Science and Engineering*** című IEEE-folyóirat 2000. januári/februári számában egy tízes toplistát ad közre, amelyet – a szerkesztők véleménye szerint – a XX. század legfontosabb tíz algoritmusából állítottak össze. Ezek közül az időrendben utolsó a gyors multipólus-módszer.

A fák között az erdõ – mögöttes szemantikájú indexelés

Az információ korát éljük, aminek egyik megnyilvánulása, hogy igen sok az információforrás, és egyre nagyobb gondot jelent a közöttük, bennük való a tájékozódás. Az adatok roppant rengetegében való eligazodást szeretnék megkönnyíteni a számítógépes keresőrendszerek, amelyeknek egyik családját alkotják a szövegek kereshető tárolását támogató ***vektorteres indexelők***. A módszercsalád a szövegekből álló adathalmazt jókora valós mátrixként ábrázolja (legalábbis logikai szinten). A mátrix sorvektorai felelnek meg a szövegeknek, az oszlopai pedig a szavaknak (a szótár elemeinek). Ha tehát N szövegünk van, és ezeket egy M szóból álló szótárral ábrázoljuk, akkor az adathalmaz leírása egy N-szer M-es mátrix lesz. A mátrixban az s szövegnek és a t szónak megfelelő pozícióban a $v_{s,t}$ nemnegatív valós szám (súly) szerepel. A súly 0, ha a t szó nem fordul elő s-ben, különben pedig $v_{s,t}$ pozitív. A súlyok meghatározására többféle módszer használatos. Az értéke általában függ attól, hogy a t hányszor fordul elő s-ben (lokális tényező), és attól is, hogy a t mennyire fontos szó az összes szöveghez viszonyítva (globális tényező). Például egy színikritikákból álló szöveggyűjtemény esetén a ***színész*** szó a keresések szempontjából csekély jelentőséggel bír, hiszen vélhetően majdnem minden dokumentumban előfordul. Ennek a szónak tehát célszerű alacsony globális súlyt tulajdonítani.

A vektorteres indexelők a keresőkérdést magát is szövegnek tekintik, így abból ugyanúgy kaphatunk egy M komponensből álló vektort, mint bármely más szövegből. Ezáltal a keresés problematikája geometriai síkra terelődik. A kérdésre a választ azok a szövegek adják, amelyek vektorai leginkább hasonlóak a kérdés vektorához. A hasonlóságot többnyire a vektorok által bezárt szög koszinuszával mérik. Ekkor a nagyjából egy irányba mutató vektorok minősülnek hasonlónak.

A vektorteres indexelők – árnyaltabb, analóg adatábrázolásuk miatt – finomabb eredményekre képesek, mint a korábbi kereső eljárások, amelyek egy szó és egy szöveg között csak kétféle viszonyt (a szó szere-

pel a szövegben avagy nem) jegyeznek fel. A szinonimák és más asszociatív kapcsolatok azonban rajtuk is kifognak. Ha mondjuk az *étkezés* szót tartalmazó szövegekre kérdezünk, akkor a kereső nem fogja esetleg visszaadni azokat a szövegeket, amelyekben a *vacsora*, *fûszer*, *elõétel* szavak szerepelnek, de az *étkezés* éppen nem.

A 90-es évek elején Susan T. Dumais, Scott Deerwester, Michael W. Berry, Thomas K. Landauer és munkatársaik átütő erejű új gondolatokat vittek a vektorteres indexelők világába (Deerwester, 1990; Berry, 1995; Berry, 1999). Pontosabban szólva, az alapötletet, a rang-redukciót (lásd később) korábban már gyümölcsözően alkalmazták a statisztikában (főkomponens-analízis), a képfeldolgozásban és más területeken. Az indexelés területén új és váratlan megközelítésüknek a *mögöttes szemantikájú indexelés* (*latent semantic indexing*, röviden LSI) nevet adták. Módszerük igen hatásosnak bizonyult a szinonimák kezelésében, ami azért különös, mert semmiféle erre kihegyezett nyelvi eszközt (tezauruszok, taxonómiák és hasonlók) nem használnak. Szemléletük lényege – itt a szép szövegek kedvelői vegyenek mély lélegzetet –, hogy a keresés szempontjából *zajnak*, *bizonytalanságnak* tekinthető az a sokféleség, amit a nyelv lehetővé tesz ugyanannak a tartalomnak a kifejezésére. Emiatt a szövegek és szavak viszonyát leíró A mátrixban sokkal több szabadsági fok van, mint amennyit a tárolt szövegek jelentése indokolna. Itt a *szabadsági fok* az A mátrix rangja, a lineárisan független oszlopainak maximális száma. Kísérleteik szerint N=70 000 szöveg és M=90 000 alapszó esetén A rangja közel maximális (azaz 70 000) lehet, míg a ténylegesen ,,értelmes" szabadsági fokok száma kb. 200-300-ra tehető. Az általános recept ennek megfelelően az, hogy A helyett vegyük a hozzá legközelebbi[4] legfeljebb 200 rangú B mátrixot. A B mátrixot az A Lánczos-felbontása (szinguláris értékek szerinti felbontása) segítségével kaphatjuk meg. Az A Lánczos-felbontása egy A=UDV alakú előállítás, melyre (egyebek között) igaz, hogy a D egy r-szer r-es diagonális mátrix, ahol r az A rangja, aminek a főátlóbeli értékei nemnegatív valós számok (az A ún. szinguláris értékei). Ebben a D-ben írjunk nullát a kétszáz legnagyobb elemet kivéve minden más helyre, és legyen H az így kapott mátrix. Ekkor igaz lesz, hogy B=UHV. Lánczos Kornél – a lineáris algebrai számítások zsenije – még egy fontos ponton kapcsolódik a történethez. Ha a szövegek nem túl nagyok, akkor az A mátrix ritka abban az értelemben, hogy kevés nem nulla eleme van. Ilyenkor a felbontás számításakor alkalmazhatók a ritka mátrixok kezelésére kihegyezett, Lánczostól származó technikák.

Az A helyett a B mátrixot használhatjuk a szövegek és szavak viszonyának ábrázolására. B-nek azt az értelmet tulajdoníthatjuk, hogy a szövegeket kétszáz fontos mesterséges fogalom súlyozott kombinációjaként kevertük ki. Ezzel a kétszáz mögöttes fogalommal írhatók le a keresőkérdések, sőt maguk a szavak is. A tapasztalatok szerint a rangredukció jelentős mértékben csökkenti a már érintett szemantikus zajt, és tényleg megtalálja a fák között az erdőt. Az eredeti reprezentáció bonyolultságának (jelen esetben a mátrix rangjának) csökkentése, egyszerűsítése révén hasznos összefüggések válnak láthatóvá. Például a szinonim szavak egymáshoz hasonlónak mutatkoznak a redukált térben akkor is, ha nem, vagy csak ritkán fordulnak elő egyazon szövegben. Ezt – úgy tűnik – a számottevő közös kontextusuk felismerésével éri el a módszer. A B mátrix egyfajta lényegtükörnek tekinthető. Használatával jelentős mértékű, de termékeny egyszerűsítés megy végbe. Eltekintünk a szövegek finom részleteitől, gazdagságától. Mindezekért cserébe a sokszínűség takarásá-

[4] A közelség itt az N×M-dimenziós térbeli euklideszi távolság szerint értendő.

ból kiemelkednek, láthatóvá válnak a keresésben hasznos asszociatív kapcsolatok. Landauer és Dumais elképzelhetőnek tartják, hogy az emberi emlékezet működésében is vannak hasonló egyszerűsítő-kiemelő mozzanatok.

Az LSI-módszernek a keresésen túl más érdekes alkalmazásai is vannak. A legkülönösebb ezek közül talán a szinonimák felismerésével kapcsolatos Landauer–Dumais-kísérlet. A *Grolier's Academic American Encyclopaedia* mintegy 30 ezer szócikkét mint szövegeket feldolgozták egy kb. 61 ezer elemű szótár felett. Az A mátrixnak ekkor nagyjából N=30 000 sora és M=61 000 oszlopa van. Az A rangját háromszázra redukálták, majd az így kapott B mátrix szinonímafelismerő képességét vizsgálták. Erre a célra a jól ismert angol nyelvvizsga, a TOEFL 80-kérdéses szinoníma-tesztjeit használták. Kérdésenként négy lehetséges szó (például *imposed, believed, requested, correlated*) közül kell kiválasztani, hogy melyik jelentése hasonlít leginkább a megadott ötödikére (*levied*). A vizsgálatot úgy végezték, hogy a 300-dimenziós LSI-térben kiszámították az ötödik szó hasonlóságát (koszinusz-mérték) az első négyhez, és azt a szót választották, amelyiknél a legnagyobb érték adódott. Az előbbi példánál a hasonlósági értékek rendre 0,70, 0,09, 0,05, ?0,03 voltak, vagyis a program magabiztosan megtalálta a helyes választ *(imposed)*. Összességében a kérdések 64%-ára válaszolt helyesen, ami döbbenetesen jó teljesítmény, figyelembe véve, hogy ugyanennyi a diákok átlagos eredményessége a hivatalos TOEFL-vizsgákon. A program egy magas szintű humán képesség dolgában tud versenyezni magával az emberrel.

A bonyolultságok persze nem enyésznek el, ahogy Störr kapitány történetében sem, de tanulmányozásuk sok érdekes és hasznos eredményt hozhat. Ennek a folyamatnak fontos jellegzetessége a különböző területekről származó gondolatok egymásra találása, összhangja. Különösen gyümölcsöző lehet az alkalmazási terület és az algoritmika együttműködése, és szép szerep juthat a messzebbről jövő, szokatlan analógiáknak is.

Kulcsszavak: *algoritmus, számítási bonyolultság, keresés, szimuláció, számításelmélet*

IRODALOM

Bailey, David H. (2000): Integer Relation Detection. Computing in Science and Engineering. 2(**1**), 24–28.

Barnes, Joshua E. – Hut, Piet (1986): A Hierarchical O(NlogN) Force-Calculation Algorithm. Nature, 324(Dec. 4. 1986), 446–449.

Berry, Michael W. – Dumais, Susan T. – O'Brien, Gavin W. (1995): Using Linear Algebra for Intelligent Information Retrieval. SIAM Review, 37(**4**), 573–595.

Berry, Michael W. – Drmac, Zlatko – Jessup, Elizabeth R. (1999): Matrices, Vector Spaces, and Information Retrieval. SIAM Review, 41(**2**), 335–362.

Board, John – Schulten, Klaus (2000): The Fast Multipole Algorithm. Computing in Science and Engineering. 2(**1**), 76–79.

Cormen, Thomas H. – Leiserson, Charles E. – Rivest, Ronald L. (1999): *Algoritmusok*. Műszaki, Budapest

Deerwester Scott – Dumais Susan T. – Furnas, George W. – Landauer, Thomas K. – Harshman, Richard (1990): Indexing by Latent Semantic Analysis. Journal of American Society for Information Science. 41(**6**), 391–407.

Greengard, Leslie – Rokhlin, Vladimir (1987): A Fast Algorithm for Particle Simulation. Journal of Computational Physics. 73, 325–348.

Ivanyos Gábor – Szántó Ágnes (1996): Lattice Basis Reduction for Indefinite Forms and an Application. Discrete Mathematics. 153, 177–188.

Lenstra, Arjen K. – Lenstra, Hendrik W. – Lovász László (1982): Factoring Polynomials with Rational Coefficients. Mathematische Annalen. 261, 515–534.

Lovász László – Gács Péter (1987): *Algoritmusok*. Tankönyvkiadó, Budapest

Papadimitriou, Christos H. (1999): *Számítási bonyolultság.* Novadat Bt., Győr

Rónyai Lajos – Ivanyos Gábor – Szabó Réka (2000): *Algoritmusok.* Typotex, Budapest

Vicsek Tamás (1990): Számítógépes szimuláció: a fizikai jelenségek megértésének új módszere. Magyar Tudomány. **9**, 1048–1054.